Better Sparsifiers for Directed Eulerian Graphs

Sushant Sachdeva
University of Toronto, Canada

Anvith Thudi
University of Toronto, Canada

Yibin Zhao
University of Toronto, Canada

Abstract

Spectral sparsification for directed Eulerian graphs is a key component in the design of fast algorithms for solving directed Laplacian linear systems. Directed Laplacian linear system solvers are crucial algorithmic primitives to fast computation of fundamental problems on random walks, such as computing stationary distributions, hitting and commute times, and personalized PageRank vectors. While spectral sparsification is well understood for undirected graphs and it is known that for every graph G, $(1+\varepsilon)$-sparsifiers with $O(n\varepsilon^{-2})$ edges exist [Batson-Spielman-Srivastava, STOC '09] (which is optimal), the best known constructions of Eulerian sparsifiers require $\Omega(n\varepsilon^{-2}\log^4 n)$ edges and are based on short-cycle decompositions [Chu et al., FOCS '18].

In this paper, we give improved constructions of Eulerian sparsifiers, specifically:

1. We show that for every directed Eulerian graph $\vec{G}$, there exists an Eulerian sparsifier with $O(n\varepsilon^{-2}\log^2 n\log^2\log n + n\varepsilon^{-4/3}\log^{8/3} n)$ edges. This result is based on combining short-cycle decompositions [Chu-Gao-Peng-Sachdeva-Sawlani-Wang, FOCS '18, SICOMP] and [Parter-Yogev, ICALP '19], with recent progress on the matrix Spencer conjecture [Bansal-Meka-Jiang, STOC '23].
2. We give an improved analysis of the constructions based on short-cycle decompositions, giving an $m^{1+\delta}$-time algorithm for any constant $\delta > 0$ for constructing Eulerian sparsifiers with $O(n\varepsilon^{-2}\log^3 n)$ edges.

2012 ACM Subject Classification Computing methodologies → Linear algebra algorithms; Theory of computation → Graph algorithms analysis; Mathematics of computing → Computations on matrices

Keywords and phrases Graph algorithms, Linear algebra and computation, Discrepancy theory

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.119

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2311.06232

Funding *Sushant Sachdeva*: NSERC Discovery Grant RGPIN-2018-06398, an Ontario Early Researcher Award (ERA) ER21-16-284, and a Sloan Research Fellowship.
Anvith Thudi: NSERC Vanier Fellowship.
Yibin Zhao: NSERC Discovery Grant RGPIN-2018-06398 ands Ontario Early Researcher Award (ERA) ER21-16-284 awarded to SS.

Acknowledgements We thank Arun Jambulapati for notifying us of an issue in a previous version of this manuscript. SS and YZ would also like to thank the Simons Institute for the Theory of Computing Fall 2023 program for its support and where a significant part of this project evolved.

1 Introduction

Given a graph $G(V,E)$, a sparsifier of G is a graph H on the same set of vertices V, but hopefully supported on a subset of the edges $E' \subset E$ such that H approximately preserves certain properties of G. Several notions of graph sparsification have been well studied for undirected graphs, e.g. spanners (approximately preserving distances), cut sparsifiers, spectral sparsifiers, etc.

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).
Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;
Article No. 119; pp. 119:1–119:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Spectral sparsification is a particularly influential notion of undirected graph sparsification [44]. Spectral sparsifiers generalize cut-sparsifiers introduced by Benczur-Karger [10], which guarantees that the total weight of every vertex cut is preserved up to a multiplicative factor of $(1+\varepsilon)$ in the sparsifier. Efficient spectral sparsification was a core development that led to nearly-linear time solvers for Laplacian linear systems [44]. It further inspired the Laplacian paradigm, resulting in faster algorithms for many graph problems including sampling/counting random spanning trees [16, 17] and approximating edge centrality measures [31].

The first construction of spectral sparsifiers for undirected graphs by Spielman and Teng required $\Omega(n\varepsilon^{-2}\text{poly}(\log n))$ number of edges with a large, unspecified power of $\log n$. Subsequently, Spielman and Srivastava [42] constructed a spectral sparsifier with $O(n\varepsilon^{-2}\log n)$ edges probabilistically by independently sampling each edge with probability proportional to its leverage score. In a complete graph, sampling edges independently with probability p requires $p=\Omega(\varepsilon^{-2}\log n)$ to achieve $(1+\varepsilon)$-spectral sparsification; thus such construction requires $\Omega(n\varepsilon^{-2}\log n)$ edges. Batson-Spielman-Srivastava [7] further improved this to show that there exist spectral sparsifiers for undirected graphs with $O(n\varepsilon^{-2})$ edges and that this is tight even for the complete graph. Thus, they essentially settled the question of the optimal size of undirected spectral sparsifiers.

For directed graphs, sparsification has been trickier to define. It is immediate to see that in a complete bipartite graph with all edges directed from the left vertices to the right vertices, if one wishes to approximately preserve all directed cuts, one must preserve all the edges. This means that there is no non-trivial cut-sparsification (or its generalization) for arbitrary directed graphs. Such pathological cases can be avoided if one restricts to Eulerian directed graphs, i.e. a graph where each vertex has its total weighted in-degree equal to its total weighted out-degree, in which case cut sparsification becomes equivalent to cut sparsification of undirected graphs. Indeed, Cohen-Kelner-Peebles-Peng-Rao-Sidford-Vladu [13] defined a meaningful generalization of spectral sparsification (and hence cut sparsification) to Eulerian directed graphs. The standard notion of Eulerian approximation and (sparsification) requires exact preservation of the differences between in and out degrees while ensuring the difference in directed Eulerian Laplacians is small with respect to the Laplacian of the undirectification of the graph. That is, for $\epsilon \in (0,1)$,

$$\left\|\mathbf{L}_G^{\frac{\pm}{2}}(\mathbf{L}_{\vec{H}}-\mathbf{L}_{\vec{G}})\mathbf{L}_G^{\frac{\pm}{2}}\right\| \le \epsilon.$$

We call these sparsifiers Eulerian sparsifiers for brevity. In a manner similar to the original Spielman-Teng construction, [13] gives a nearly-linear time $\widetilde{O}(m)$-time algorithm to build an Eulerian sparsifier with $O(n\varepsilon^{-2}\text{poly}(\log n))$ edges, with a large unspecified power of $\log n$.

Since Eulerian sparsification generalizes undirected spectral sparsification, $\Omega(n\varepsilon^{-2})$ edges are necessary for constructing Eulerian sparsifiers. There has been progress in proving the existence of Eulerian sparsifiers with fewer edges: Chu-Gao-Peng-Sachdeva-Sawlani-Wang [11] introduced the short-cycle decomposition, a decomposition of an unweighted graph as a union of short edge-disjoint cycles, and a few extra edges. As a simple lemma, they showed that every undirected graph can be represented as a union of edge-disjoint cycles of length $2\log n$, with at most $2n$ extra edges. Using this short-cycle decomposition, [11] were able to prove Eulerian sparsifiers with $O(n\varepsilon^{-2}\log^4 n)$ edges exist. However, the following natural question remains unanswered:

What is the best possible sparsity guarantee for constructing Eulerian sparsifiers?

In this paper, we make progress on this question. First, we present an improved analysis of the short-cycle based Eulerian sparsification from [11].

▶ **Theorem 1.** *For every constant $\delta > 0$, there is an algorithm that takes as input a directed Eulerian graph $\vec{G}$ and returns an ε-Eulerian sparsifier of $\vec{G}$ with $O(n\varepsilon^{-2}\log^3 n)$ edges in $m^{1+\delta}$ time.*

The above algorithm relies on independently toggling short cycles: with probability $\frac{1}{2}$ all clockwise edges are deleted and counter-clockwise edges are doubled, otherwise vice-versa. Given that the edges in each $O(\log n)$ length short-cycle are toggled in a completely correlated manner, and the cycles are toggled independently, this approach cannot lead to a sparsity better than $O(n\varepsilon^{-2}\log^3 n)$ (see Remark 9). To go past the above result, we leverage discrepancy theory, specifically recent progress on the matrix Spencer conjecture by Bansal, Jiang, and Meka [6]. (See Section 1.1 for a description of the matrix Spencer conjecture.) While the matrix Spencer conjecture is not directly useful for our application, we utilize the underlying machinery from [6] and the short-cycle decomposition to prove the following:

▶ **Theorem 2** (Informal). *There is an algorithm that given an Eulerian graph $\vec{G}$, can compute in poly-time an ε-Eulerian sparsifier of $\vec{G}$ with $n\varepsilon^{-2}\log^2 n + n\varepsilon^{-3/4}\log^{8/3} n$ edges (up to $\log\log n$ factors).*

For small ε, e.g. $\varepsilon^{-1} = \Omega(\log n)$, the above theorem gives an $n\varepsilon^{-2}\log^2 n$ bound, only a $\log^2 n$ factor away from the lower bound. In Section 4.1 we show that assuming the matrix partial colouring conjecture, one can improve this result to prove the existence of ε-Eulerian sparsifiers with $n\varepsilon^{-2}\log^2 n$ edges for all ε (up to $\log\log n$ factors).

1.1 Related works

Sparsification

There are four major approaches for undirected spectral sparsification: expander decomposition [45, 4, 20], spanners [21, 23, 26], importance sampling [43, 22], and potential function based sparsification [8, 3, 28, 29]. More closely related to Eulerian sparsification is undirected degree preserving sparsification, introduced by Chu-Gao-Peng-Sachdeva-Sawlani-Wang [11]. Degree preserving sparsification is useful for constructing spectral sketches. More importantly for us, techniques for degree preserving sparsification can generally be extended to work for directed Eulerian sparsification.

Cohen-Kelner-Peebles-Peng-Rao-Sidford-Vladu [13] showed the first degree preserving (implicitly) and Eulerian sparsifier using expander decomposition. The algorithm performs random sampling of the directed edges with probability related to the degrees within each expander. Recent work by Ahmadinejad-Peebles-Pyne-Sidford-Vadhan [2] establishes an "equivalence", albeit with significantly stronger requirements than spectral approximations, between degree preserving and Eulerian sparsification under the notion of singular value approximation. They established the first Eulerian sparsifier with both nearly-linear sparsity and nearly-linear runtime, albeit with a large poly$(\log n)$ factor in both. However, the expander approach bottlenecks at $\Omega(n\varepsilon^2\log^3 n)$ due to a lowerbound on the optimal tradeoff between the expansion factor and the number of expanders [41].

The technique of using short cycles for sparsification [11] also applies to degree preserving and Eulerian sparsifications with sparsity $O(n\varepsilon^{-2}\log^2 n)$ and $O(n\varepsilon^{-2}\log^4 n)$ respectively. Improved short cycle decompositions were subsequently designed in [32, 35] to facilitate faster construction of sparsifiers. Our first result Theorem 1 follows closely to [11] and reduces the gap between degree-preserving and Eulerian sparsification under this technique.

Recently Jambulapati-Reis-Tian [19] constructed new degree preserving sparsifiers using discrepancy theory. They showed operator norm discrepancy bodies are well conditioned[1] for the symmetric and PSD matrices that arise from undirected sparsification and used an approximate version of the framework from Reis-Rothvoss [39] to give a colouring of the edges (corresponding to adding and deleting edges) under the linear constraint needed for degree preservation. However, the underlying discrepancy bodies studied by Jambulapati-Reis-Tian [19] do not align with Eulerian sparsification where matrices are no longer positive semidefinite and the primary statistic one has control over is matrix variance (see Section 4).

Directed Laplacian solvers

Cohen-Kelner-Peebles-Peng-Sidford-Vladu [14] initiated the line of work that studies solving directed Laplacian linear systems. They established a reduction from solving general directed Laplacian systems to Eulerian Laplacian systems.Cohen-Kelner-Peebles-Peng-Rao-Sidford-Vladu [13] gave an almost linear time algorithm for solving Eulerian Laplacians using the squaring identities from Peng-Spielman [37]. Subsequently, Cohen-Kelner-Kyng-Peebles-Peng-Rao-Sidford [12] gave the first nearly linear time solver using the standard approximate LU factorization techniques that enjoyed great success in undirected Laplacian solvers [27, 24, 40]. Ahmadinejad-Jambulapati-Saberi-Sidford [1] further established a reduction from solving systems of (asymmetric) M-matrices to Eulerian Laplacian systems, giving fast computation of several problems closely associated with the Perron-Frobenius theorem. Peng-Song [36] extended the approach from [12] and gave an approach for extending an algorithm for building Eulerian sparsifiers to a fast solver for Eulerian Laplacian linear systems. Combined with Theorem 1, they give an $O(n\log^4 n\log(\frac{n}{\varepsilon}))$ time solver with $m^{1+\delta}$ preprocessing time for any constant $\delta > 0$. Kyng-Meierhans-Probst-Gutenberg [25] established the first derandomized directed Laplacian solver in almost linear time.

Discrepancy theory

The Matrix Spencer Conjecture [47, 34] is a major open problem in discrepancy theory:

▶ **Conjecture 3** (Matrix Spencer Conjecture). *Given $n \times n$ symmetric matrices $\mathbf{A}_1, \ldots, \mathbf{A}_m \in \mathbb{R}^{n\times n}$ with $\|\mathbf{A}_i\| \le 1$, there exist signs $x \in \{\pm 1\}^m$ such that $\|\sum_{i=1}^m x_i \mathbf{A}_i\| \le O(\sqrt{m} \cdot \max\{1, \sqrt{\min\{1, \log(\frac{n}{m})\}}\})$.*

As a natural comparison, for a uniform random colouring $x \in \{\pm 1\}^m$, the matrix Chernoff bound [46] gives the following bound which has a gap of $\sqrt{\log n}$ to Conjecture 3 when $m \ge n$:

$$\mathbb{E}\left[\left\|\sum_i x_i \mathbf{A}_i\right\|\right] = O\left(\sqrt{\log n}\right) \cdot \left\|\sum_i \mathbf{A}_i^2\right\|^{\frac{1}{2}} \le O(\sqrt{m\log n}).$$

We refer readers to [30, 18, 15] for recent progress toward solving this conjecture.

Many natural problems in studying the spectra of matrices can be viewed as discrepancy theory problems, e.g., graph sparsification [8, 38] and the Kadison-Singer problem [33]. Reis-Rothvoss [38] studies the geometry of operator norm balls for a collection of matrices where, $\|\sum_i |\mathbf{A}_i|\|$ is small. This was subsequently used in Jambulapati-Reis-Tian [19] to show optimal degree preserving sparsification. As previously mentioned, this line of work does not apply to Eulerian sparsification since matrices that emerge from our setting do not satisfy

[1] I.e., satisfy certain Gaussian measure lowerbound

that $\| \sum_i |\mathbf{A}_i| \|$ is small. Bansal-Jiang-Meka [6] resolved the Matrix Spencer Conjecture for matrices of rank $n/(\log^{O(1)} n)$ using a recent advancement in matrix concentration bounds due to Bandeira-Boedihardjo-van-Handel [5]. The partial colouring result for controlling operator norm used in [6] serves as the main machinery in our existential results (see Lemma 4). Specifically, the matrices we study naturally satisfy $\| \sum_i \mathbf{A}_i^2 \|$ is small.

1.2 Technical overview

Our approach to constructing Eulerian sparsifiers builds on the framework of Chu-Gao-Peng-Sachdeva-Sawlani-Wang [11]. The sparsification algorithm in [11] combines importance sampling of edges with a short cycle decomposition. At each iteration, the algorithm restricts its attention to edges with small "importance" in the undirected graph (edges with leverage score $w_e \mathbf{b}_e^\top \mathbf{L}_G^+ \mathbf{b}_e$ at most constant times the average leverage score, $O(\frac{n}{m})$). The algorithm then performs a short cycle decomposition on these edges – expressing the graph as a union of uniformly weighted edge-disjoint short cycles and a few extra edges. For each short cycle, the algorithm independently keeps either the clockwise edges or the counter-clockwise edges with probability $\frac{1}{2}$ each. The number of edges reduces by a constant fraction in expectation at each iteration. After doubling the weights of the cycle edges retained, the algorithm guarantees that the Eulerianess of each cycle is preserved and, hence, the entire graph. Moreover, when combined with the undirected leverage score condition above, changes in directed short cycles also have a small variance overall. The matrix Bernstein inequality for asymmetric matrices guarantees a small approximation error for this randomized step. We repeat this process until the desired approximation error is met.

Our improved result of this algorithm is due to th improved variance bounds in Lemma 8 for random matrices corresponding to short cycles. Rather than bounding the variance through complete graphs as in [11], we bound it with respect to the undirected cycle. This improved variance also serves a critical role in our partial colouring approach in Section 4.

In the rest of our paper, we present our existential result which uses the partial colouring lemma, Lemma 4, from [6] to choose how to sparsify the short cycles. The algorithm follows the same high-level approach as the random sampling construction above. For each directed short cycle, instead of independently sampling cycle edges, we will use the partial colouring given by Lemma 4. In each iteration, Lemma 4 gives a partial colouring with sufficiently many fully coloured entries (i.e., ± 1 entries) on all cycles. It then allows us to remove a constant fraction of the cycle edges with less error than random sampling.

▶ **Lemma 4** ([6] Lemma 3.1)**.** *There exists constants $c, c' > 0$ such that given symmetric matrices $\mathbf{A}_1, \ldots, \mathbf{A}_m \in \mathbb{R}^{n \times n}$ satisfying $\| \sum_{i=1}^m \mathbf{A}_i^2 \| \leq \sigma^2$ and $\sum_{i=1}^m \|\mathbf{A}_i\|_F^2 \leq mf^2$ and a point $\mathbf{y} \in (-1,1)^m$, there is an algorithm* PartialColour *that returns in polynomial time a point $\mathbf{x} \in [-1,1]^m$ such that $|\{i : x_i \in \{\pm 1\}\}| > c'm$ and*

$$\left\| \sum_{i=1}^m (x_i - y_i) \mathbf{A}_i \right\| \leq c(\sigma + (\log^{\frac{3}{4}} n)\sqrt{\sigma f}). \tag{1}$$

There are two major challenges in applying Lemma 4. Firstly, within each iteration, we cannot afford to fully colour all the cycles by recursively applying Lemma 4, since we might have to perform the partial colouring $O(\log n)$ times, resulting in an additional log factor in the sparsity. Thus, we are always left with fractionally coloured cycles (i.e., entries with magnitude < 1). Such cycles must still be incorporated into the sparsified graph to guarantee the error given by Lemma 4. However, we cannot explicitly modify the graph to include edges corresponding to these cycles, as we would lose the integral and polynomially bounded weight conditions and the short cycle decomposition could no longer be applied to this new graph. The second challenge also comes from incorporating fractionally coloured cycles in the next iteration. Unlike the undirected case, the two parts of a directed cycle do not necessarily have the same number of edges. For example, a directed cycle with all edges in the same direction has all the edges in one part and none in the other. If we start our colouring process from a non-zero initial partial colouring (i.e., a non-zero $\mathbf{y}$ to Lemma 4), we could end up at a colouring where almost no edges are removed.

To deal with these problems, our algorithm handles the integral weighted portion $\vec{G}$ of the graph $\vec{G}'$ and the fractionally coloured cycles $\overline{S}$ separately (see Algorithm 4). For the integral weighted portion, we perform the partial colouring to guarantee at least a constant fraction of edges are removed. We then add the fractionally coloured cycles into the set $\overline{S}$. For the set of fractionally coloured cycles $\overline{S}$, we adjust their colouring by considering the difference between the partial colours and ± 1 to ensure that a good portion of cycles in $\overline{S}$ are fully coloured after the procedure to guarantee the size of $\overline{S}$ does not blow up. In both cases, the error incurred by the partial colouring operation is controlled to guarantee our desired final error (Theorem 2).

2 Preliminaries

We use $\widetilde{O}(\cdot)$ to suppress polylog factors in n, m. We say "with high probability in n" for an event occuring with probability $1 - n^{-\Omega(1)}$. For graphs, n is assumed to be the number of vertices and is often omitted. All logarithms in the paper are base 2.

Linear Algebra

We use boldface to denote vectors, and use $\mathbf{0}$ and $\mathbf{1}$ for the all-zeros and all-ones vectors. We let $\mathbf{e}_u$ denote the vector that is 1 in the uth coordinate and 0 elsewhere. We denote $\mathbf{b}_{uv} = \mathbf{e}_u - \mathbf{e}_v$ for any $u \neq v$. For vectors $\mathbf{u}, \mathbf{v}$ of equal dimension, $\mathbf{u} \circ \mathbf{v}$ is the entrywise product. For a linear subspace $\mathcal{W}$ of a vector space $\mathcal{V}$, we denote $\mathcal{W}^\perp$ as the orthogonal complement of $\mathcal{W}$ in $\mathcal{V}$.

Matrices are denoted in boldface capticals. We use $\ker(\mathbf{A}), \mathrm{im}(\mathbf{A})$ to denote the kernel and image of $\mathbf{A}$. For any u, we let $(\mathbf{A})_u$ denote the uth column of $\mathbf{A}$. The Kronecker product of matrices $\mathbf{A}$ and $\mathbf{B}$ is denoted $\mathbf{A} \otimes \mathbf{B}$. A symmetric matrix $\mathbf{A}$ is positive semidefinite (PSD) (resp. positive definite (PD)) if, for any vector $\mathbf{x}$ of compatible dimension, $\mathbf{x}^\top \mathbf{A} \mathbf{x} \geq 0$ (resp. $\mathbf{x}^\top \mathbf{A} \mathbf{x} > 0$). Let $\mathbf{A}$ and $\mathbf{B}$ be two symmetric matrices of the same dimension, then we write $\mathbf{B} \preccurlyeq \mathbf{A}$ or $\mathbf{A} \succcurlyeq \mathbf{B}$ if $\mathbf{A} - \mathbf{B}$ is PSD. The ordering given by $\preccurlyeq$ is called Loewner partial order.

▶ **Lemma 5.** *If* $\mathbf{A} \succcurlyeq \mathbf{B}$ *and* $\mathbf{C}$ *is any matrix of compatible dimension, then* $\mathbf{C}\mathbf{A}\mathbf{C}^\top \succcurlyeq \mathbf{C}\mathbf{B}\mathbf{C}^\top$.

Let $\|\mathbf{A}\|$ and $\|\mathbf{A}\|_F = \sqrt{\mathrm{Tr}\left(\mathbf{A}^* \mathbf{A}\right)}$ denote the operator norm and Frobenius norm of a matrix $\mathbf{A}$. The operator norm is equal to the largest singular value of $\mathbf{A}$. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, we define the Hermitian (symmetric) lift of $\mathbf{A}$ by

$$\mathsf{hlift}(\mathbf{A}) = \begin{bmatrix} & \mathbf{A} \\ \mathbf{A}^\top & \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$$

The norms of Hermitian lifts satisfy $\|\mathsf{hlift}(\mathbf{A})\| = \|\mathbf{A}\|$ and $\|\mathsf{hlift}(\mathbf{A})\|_F = 2\|\mathbf{A}\|_F$. Given a symmetric matrix with eigenvalue decomposition $\mathbf{A} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i^\top$, where $\{\mathbf{v}_i\}_i$ form an orthonormal basis, the pseudoinverse is defined as $\mathbf{A}^+ = \sum_{i:\lambda_i \neq 0} \frac{1}{\lambda_i} \mathbf{v}_i \mathbf{v}_i^\top$. The absolute value of $\mathbf{A}$ on eigenvalues is defined as $|\mathbf{A}| = \sum_{i:\lambda_i \neq 0} |\lambda_i| \mathbf{v}_i \mathbf{v}_i^\top$. Note that $|\mathbf{A}|$ is PSD. Similarly for symmetric PSD matrix $\mathbf{A}$ we have $\mathbf{A}^{1/2} = \sum_{i:\lambda_i \neq 0} \sqrt{\lambda_i} \mathbf{v}_i \mathbf{v}_i^\top$ and $\mathbf{A}^{+/2} = \sum_{i:\lambda_i \neq 0} \frac{1}{\sqrt{\lambda_i}} \mathbf{v}_i \mathbf{v}_i^\top$.

Graphs and Laplacians

$\vec{G} = (V, E, \mathbf{w})$ denotes a weighted directed graph (allowing multi-edges) with edge weights $\mathbf{w} : E \to \mathbb{R}_{\geq 0}$. G denotes the undirected graph of $\vec{G}$ where directed edgex $e \in E(\vec{G})$ correspond to undirected edges between the same vertices with half the weight. $\vec{G}$ is Eulerian if the weighted in degree equals the weighted out degree for each vertex $v \in V$.

We define the adjacency matrix of $\vec{G}$ as a non-negative matrix $\mathbf{A}_{\vec{G}}$ with $\mathbf{A}_{uv} = w_{uv}$ if $(u, v) \in E$ and 0 otherwise. The weighted degree matrix of $\vec{G}$ is a non-negative diagonal matrix $\mathbf{D}_{\vec{G}}$ corresponding to the weighted out-degrees of $\vec{G}$. We define the directed Laplacian of $\vec{G}$ as $\mathbf{L}_{\vec{G}} = \mathbf{D}_{\vec{G}} - \mathbf{A}_{\vec{G}}^\top$ and it satisfies $\mathbf{1}^\top \mathbf{L}_{\vec{G}} = \mathbf{0}^\top$, i.e. $(\mathbf{L}_{\vec{G}})_{uu} = -\sum_{v \neq u} \mathbf{L}_{vu}$ for all $u \in V$. For a weighted Eulerian directed graph $\vec{G}$, its graph Laplacian additionally satisfies $\mathbf{L}_{\vec{G}}\mathbf{1} = \mathbf{0}$. Assuming Eulerian graph $\vec{G}$, the associated undirected graph Laplacian matrix of G is $\mathbf{L}_G = \frac{1}{2}(\mathbf{L}_{\vec{G}} + \mathbf{L}_{\vec{G}}^\top)$. $\mathbf{L}_G$ is symmetric and PSD. For an undirected Laplacian $\mathbf{L}_G$, the effective resistance and leverage score of an edge $e \in E(G)$ are defined by $\mathrm{Reff}_G(e) = \mathbf{b}_e^\top \mathbf{L}_G^+ \mathbf{b}_e$ and $\boldsymbol{\tau}_G(e) = w_e \,\mathrm{Reff}_G(e)$ where we fixed an arbitrary orientation for the undirected edge e. We use n and m for the number of vertices and edges in G. As is standard, we study strongly connected Eulerian graphs with positive integral and polynomially bounded edge weights (i.e., weights bounded by $n^{O(1)}$).

3 Eulerian sparsification via short cycle decomposition

We first present an improved analysis of constructing Eulerian sparsifiers using short cycle decompositions analogous to [11]. In particular, we provide a better variance analysis of the error terms in sparsification than what was used by [11]; by Matrix Bernstein [46] this will allow us to use fewer edges to retain a desired error bound.

We first recall the definition of a short cycle decomposition of a graph G.

▶ **Definition 6.** *An $(\hat{m}, L)$-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 $\hat{m}$ edges are not in the union of the cycles.*

We let CycleDecomposition be an algorithm that takes as input an unweighted graph with n vertices and m edges and returns a $(\hat{m}, L)$-short cycle decomposition in time $\mathrm{T}_{\mathrm{CD}}(m, n)$. As in [11], we assume the *super-additivity* of T_{CD}, $\sum_i \mathrm{T}_{\mathrm{CD}}(m_i, n) \leq \mathrm{T}_{\mathrm{CD}}\left(\sum_i m_i, n\right)$, for all $m_i \geq n$. Relevant to us is a construction of short cycle decompositions which for every constant $\delta > 0$, gives an $(O(n \log n), O(\log n))$-short cycle decomposition in time $m^{1+\delta}$.

▶ **Lemma 7** ([35] Theorem 2)**.** *For any $\delta > 0$, there is an algorithm that computes an $(O(n \log n), O(2^{\frac{1}{\delta}} \log n))$-short cycle decomposition of an undirected unweighted graph in $2^{O(\frac{1}{\delta})} m n^\delta$ time.*

Our random sampling based sparsification algorithm is the same as [11]. We repeatedly sparsify an Eulerian graph by keeping only the “clockwise” or “counter-clockwise” edges of each cycle in a short cycle decomposition of the graph, see CycleSparsify in Algorithm 2 and CycleSparsifyOnce in Algorithm 3.

Stated in other words, we will sparsify a cycle by partitioning it into two sets and removing one randomly. For a directed cycle $\vec{C}$, we take $\vec{F}, S$ to be the outputs of CorrectOrientation($\vec{C}$). In particular, $\vec{F}$ is the cycle $\vec{C}$ corrected so that every vertex has an incoming edge and an outgoing edge, and S is the undirected graph coming from the set of edges in $\vec{C}$ whose direction we reversed (where the edge weight in S are the same as the original edge

Algorithm 1 CorrectOrientation($\vec{C}$).

Pick an arbitrary edge e_1 in $\vec{C}$ and let v_1 be its tail vertex. Define $V_{\vec{C}}$ as the vertex set of $\vec{C}$.
Initialize $E_{\vec{S}} \leftarrow \emptyset$, $E_{\vec{F}} \rightarrow \{e_1\}$, $V_{\vec{F}} = \{v_1\}$, $i = 1$
while $|V_{\vec{C}} \setminus V_{\vec{F}}| > 0$ **do**
  $i \leftarrow i + 1$
  Take e_{i+1} be the other edge incident on v_i.
  If e_{i+1} is outgoing from v_i, take v_{i+1} the head of e_{i+1} and update $E_{\vec{F}} \leftarrow E_{\vec{F}} \cup \{e_{i+1}\}$, $V_{\vec{F}} \leftarrow V_{\vec{F}} \cup \{v_{i+1}\}$.
  Else let v_{i+1} be the tail of e_{i+1} and update $E_{\vec{S}} \leftarrow E_{\vec{S}} \cup \{e_{i+1}\}$, $E_{\vec{F}} \leftarrow E_{\vec{F}} \cup \{\text{rev}(e_{i+1})\}$, $V_{\vec{F}} \leftarrow V_{\vec{F}} \cup \{v_{i+1}\}$.
return $\vec{F}$ *defined by* $E_{\vec{F}}$ *and* $V_{\vec{F}}$, *and* S *the undirected graph defined by* $E_{\vec{S}}$ *and the incident vertices of* $E_{\vec{S}}$.

Algorithm 2 CycleSparsify($\vec{G}, \varepsilon$, CycleDecomposition).

Decompose each edge by its binary representation.
Compute $\mathbf{r}$ a 1.5-approximate effective resistances in G.
while $|E(\vec{G})| \geq O(\hat{m}\log n + \varepsilon^{-2} n L^2 \log n)$ **do**
  $\vec{G} \leftarrow$ CycleSparsifyOnce($\vec{G}, \mathbf{r}$, CycleDecomposition).
return $\vec{G}$.

weights). We consider the direction of edges defined by $\vec{F}$ as clockwise. Then, the edges in S are all the counter-clockwise edges in $\vec{C}$. For a cycle C and its corresponding directed cycle $\vec{C}$, the directed graph Laplacian added at line 7 in CycleSparsifyOnce is the following:

$$\begin{cases} \mathbf{L}_{\vec{C}} + \mathbf{L}_{\vec{F}} - \mathbf{L}_S & \text{w.p. } \frac{1}{2} \\ \mathbf{L}_{\vec{C}} - \mathbf{L}_{\vec{F}} + \mathbf{L}_S & \text{w.p. } \frac{1}{2} \end{cases}$$

which means the changes incurred on the directed graph Laplacian is

$$\begin{cases} \widetilde{\mathbf{L}} & \text{w.p. } \frac{1}{2} \\ -\widetilde{\mathbf{L}} & \text{w.p. } \frac{1}{2} \end{cases}, \text{where } \widetilde{\mathbf{L}} = \mathbf{L}_{\vec{F}} - \mathbf{L}_S. \tag{2}$$

Note that this change preserves the difference between the in and out degrees of $\vec{C}$. Either a vertex had an incoming and outgoing edge (and so difference 0), in which case both edges are either in $\vec{F} \setminus S$ or in S and hence always added together with the same weights (so still difference 0). Alternatively a vertex had two incoming or outgoing edges, in which case only one is ever added with twice the weight, which then still preserves the difference between in and out degree.

To obtain the improved approximation error guarantees, we show Lemma 8 that bounds the effect of $\mathbf{L}_{\vec{F}}$. Compared to Lemma 5.6 in [11], our result improves the bound by a factor of L.

▶ **Lemma 8.** *If $\vec{C}$ is a equal weighted directed cycle of length L contained in a graph $\vec{G}$ where each edge $\vec{e} \in \vec{C}$ satisfies $\boldsymbol{\tau}_G(e) \leq \rho$. Then, $\mathbf{L}_{\vec{F}}^{\top}\mathbf{L}_G^{+}\mathbf{L}_{\vec{F}} \preceq O(L^2\rho)\mathbf{L}_C$.*

Algorithm 3 CycleSparsifyOnce($\vec{G}, \mathbf{r}$, CycleDecomposition).

Input: A directed Eulerian graph $\vec{G}$ where edge weights are integral powers of 2,a 2-approximate effective resistances $\mathbf{r}$ in G, a short cycle decomposition algorithm CycleDecomposition.

Output: A directed Eulerian graph $\vec{H}$ where edge weights are integral powers of 2.

$\vec{H} \leftarrow \vec{G}$ with only the edges which satisfies $w_e r_e > \frac{4n}{m}$ and remove these edges from $\vec{G}$.

Partition $\vec{G}$ into uniformly weighted graph $\vec{G}_1, \ldots, \vec{G}_s$ where $\vec{G}_i$ has all edge weights 2^i and $s = O(\log n)$.

for *each* $\vec{G}_i$ **do**

  $\{C_{i,1}, \ldots, C_{i,t}\} \leftarrow$ CycleDecomposition(G_i) and let $\vec{C}_{i,j}$ be the corresponding directed graph of $C_{i,j}$ in $\vec{G}_i$.

  $\vec{H} \leftarrow \vec{H} + \vec{G}_i \setminus \left(\bigcup_{j=1}^{t} \vec{C}_{i,j}\right)$.

  for *each cycle* $\vec{C}_{i,j}$ **do**

    With probability $\frac{1}{2}$, add all its clockwise edges with twice their weight to $\vec{H}$. Otherwise, add the counter-clockwise edges instead.

return $\vec{H}$.

Proof. Let $\mathbf{\Pi}_C = \mathbf{I}_C - \frac{1}{L}\mathbf{1}_C\mathbf{1}_C^\top$ be the projection matrix on the support of C except the all one vector on C. Notice that $\ker^\perp(\mathbf{L}_C) = \mathrm{im}(\mathbf{L}_C) = \mathrm{im}(\mathbf{\Pi}_C)$. Furthermore, we have $\mathrm{im}(\mathbf{L}_{\vec{F}}) \subset \mathrm{im}(\mathbf{\Pi}_C)$ (as $\mathbf{1}_C \in \mathrm{im}^\perp(\mathbf{L}_{\vec{F}})$) so $\mathbf{\Pi}_C \mathbf{L}_{\vec{F}} = \mathbf{L}_{\vec{F}}$, and also $\mathrm{im}^\perp(\mathbf{\Pi}_C) \subset \ker(\mathbf{L}_{\vec{F}}^\top)$ hence $\mathbf{L}_{\vec{F}}^\top \mathbf{\Pi}_C = \mathbf{L}_{\vec{F}}^\top$. Thus, $\mathbf{L}_{\vec{F}}^\top \mathbf{L}_G^+ \mathbf{L}_{\vec{F}} = \mathbf{L}_{\vec{F}}^\top \mathbf{\Pi}_C \mathbf{L}_G^+ \mathbf{\Pi}_C \mathbf{L}_{\vec{F}}$.

Let w be the weight of each edge in $\vec{C}$. Then, $\mathbf{L}_{\vec{F}} = w(\mathbf{I} - \mathbf{P})$ where $\mathbf{P}$ is a permutation matrix on the vertices of C corresponding to the transition matrix $\vec{F}$ and $\mathbf{L}_C = \frac{w}{2}(2\mathbf{I} - \mathbf{P} - \mathbf{P}^\top)$. Now, $\mathbf{L}_{\vec{F}}^\top \mathbf{\Pi}_C \mathbf{L}_{\vec{F}} = \mathbf{L}_{\vec{F}}^\top \mathbf{L}_{\vec{F}} = w^2(\mathbf{I} - \mathbf{P}^\top)(\mathbf{I} - \mathbf{P}) = w^2(2\mathbf{I} - \mathbf{P} - \mathbf{P}^\top) = 2w\mathbf{L}_C$. As $\ker(\mathbf{L}_G^+) \subseteq \ker(\mathbf{\Pi}_C)$, it suffices to show $\|\mathbf{\Pi}_C \mathbf{L}_G^+ \mathbf{\Pi}_C\| = O(\frac{L^2\rho}{w})$. We can write out each column of $\mathbf{\Pi}_C$ by $(\mathbf{\Pi}_C)_u = \frac{1}{L}\sum_{v \in C, v \neq u} \mathbf{b}_{uv}$ for $u \in C$ and $\mathbf{0}$ otherwise. As effective resistance is a metric, $w\mathbf{b}_{uv}^\top \mathbf{L}_G^+ \mathbf{b}_{uv} \leq (L-1)\rho$ for any distinct vertices $u, v \in C$. Note that this factor of L is an upperbound on the combinatorial distance from u to v in C. Then,

$$\begin{aligned}
\left|(\mathbf{\Pi}_C)_x^\top \mathbf{L}_G^+ (\mathbf{\Pi}_C)_u\right| &= \left|\left(\frac{1}{L}\sum_{y \in C, y \neq x} \mathbf{L}_G^{\frac{+}{2}} \mathbf{b}_{xy}\right)^\top \left(\frac{1}{L}\sum_{v \in C, v \neq u} \mathbf{L}_G^{\frac{+}{2}} \mathbf{b}_{uv}\right)\right| \\
&\leq \sum_{y \neq x, y \in C} \sum_{v \neq u, v \in C} \frac{1}{wL^2} \left\|w^{\frac{1}{2}} \mathbf{L}_G^{\frac{+}{2}} \mathbf{b}_{xy}\right\| \cdot \left\|w^{\frac{1}{2}} \mathbf{L}_G^{\frac{+}{2}} \mathbf{b}_{uv}\right\| \\
&\leq (L-1)^2 \times \frac{(L-1)\rho}{wL^2} \leq \frac{L\rho}{w}.
\end{aligned}$$

By Gershgorin circle theorem and the length of C, any eigenvalue of $\mathbf{\Pi}_C \mathbf{L}_G^+ \mathbf{\Pi}_C$ cannot exceed $\frac{L^2\rho}{w}$ as required. ◀

▶ Remark 9. There is still a gap of factor L when comparing Lemma 8 to the undirected case. It turns out Lemma 8 is tight. Consider the multi-graph $\vec{G}$ that consists of a directed cycle with edges of weight 1 in the same orientation $\vec{F}$ and a undirected cycle C on the same vertices of edge weight ρ^{-1} for $\rho < \frac{1}{2}$. Then, each edge of the directed graph has undirected leverage score $\Theta(\rho)$ while $\mathbf{L}_{\vec{F}}^\top \mathbf{L}_G^+ \mathbf{L}_{\vec{F}} = \Theta(\frac{\rho}{L})\mathbf{L}_K$ where $\mathbf{L}_K$ is the Laplacian of unit clique on the vertices of $\vec{G}$. Since $\mathbf{L}_K$ cannot be bounded by $o(L^3)\mathbf{L}_F$, this gives the lowerbound.

When combined with Lemma 5.5 of [11], we obtain the following spectral bounds on matrices which appear later in our variance analysis. We refer readers to [11] and the full version of our paper for the proofs of all subsequent claims in this section.

▶ **Lemma 10.** *Let $\vec{C}$ is an equal weighted directed cycle of length L contained in a graph $\vec{G}$ where each edge $\vec{e} \in \vec{C}$ satisfies $\boldsymbol{\tau}_G(e) \leq \rho$. Then* $\mathbf{L}_G^{\frac{+}{2}}(\widetilde{\mathbf{L}}^\top \mathbf{L}_G^+ \widetilde{\mathbf{L}})\mathbf{L}_G^{\frac{+}{2}} \preceq O(L^2\rho) \cdot \mathbf{L}_G^{\frac{+}{2}} \mathbf{L}_C \mathbf{L}_G^{\frac{+}{2}}$ *and* $\mathbf{L}_G^{\frac{+}{2}}(\widetilde{\mathbf{L}} \mathbf{L}_G^+ \widetilde{\mathbf{L}}^\top)\mathbf{L}_G^{\frac{+}{2}} \preceq O(L^2\rho) \cdot \mathbf{L}_G^{\frac{+}{2}} \mathbf{L}_C \mathbf{L}_G^{\frac{+}{2}}$.

The matrix Bernstein's inequality [46] then gives the sparsification and error guarantees of running CycleSparsifyOnce in Lemma 11.

▶ **Lemma 11.** *Given a directed Eulerian graph $\vec{G}$ whose edge weights are integral powers of 2, and additionally 2-approximate effective resistances $\mathbf{r}$ in G, the algorithm* CycleSparsifyOnce *returns in $O(m) + \mathrm{T}_{\mathrm{CD}}(m, n)$ time a directed Eulerian graph $\vec{H}$ with edge weights still being powers of 2 such that if the number of edges in G satisfy $m = \Omega(\hat{m} \log n + nL^2 \log n)$, then with high probability, the number of edges in $\vec{H}$ is at most $\frac{15}{16}m$ and*

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

We now provide the guarantees of CycleSparsify, which repeatedly calls CycleSparsifyOnce until a criterion on the number of edges is met.

▶ **Theorem 12.** *Given as input an Eulerian graph $\vec{G}$ with polynomial bounded integral edge weights and $\varepsilon \in (0, \frac{1}{2})$, the algorithm* CycleSparsify *returns in $O(m \log^2 n) + \mathrm{T}_{\mathrm{CD}}(O(m \log n), n)$ time a Eulerian graph $\vec{H}$ with $O(\hat{m} \log n + \varepsilon^{-2} nL^2 \log n)$ edges such that with high probability,*

$$\left\| \mathbf{L}_G^{\frac{+}{2}}(\mathbf{L}_{\vec{G}} - \mathbf{L}_{\vec{H}})\mathbf{L}_G^{\frac{+}{2}} \right\| \leq \varepsilon.$$

Plugging in Lemma 7, we obtain the improved results on constructing Eulerian Sparsifiers with short cycle decompositions, summarized in Theorem 1.

▶ **Theorem 1.** *For every constant $\delta > 0$, there is an algorithm that takes as input a directed Eulerian graph $\vec{G}$ and returns an ε-Eulerian sparsifier of $\vec{G}$ with $O(n\varepsilon^{-2} \log^3 n)$ edges in $m^{1+\delta}$ time.*

4 Sparsification via partial colouring

In the previous algorithm CycleSparsify, the approach to sparsifying was to randomly pick one part of each cycle (out of a partitioning of the cycle into two parts) to remove from the graph. The analysis then followed by observing on average this leads to a good approximation, and that furthermore the variance in this random construction is sufficiently small. In this section, we show, however, that by using recent partial colouring results on operator norm discrepancy bodies to pick what parts of a cycle to remove, we can obtain better sparsifiers. The main partial colouring result we use, relevant for picking a subset of matrices to keep with minimal error, is restated below.

Algorithm 4 ColourSparsify($\vec{G}, \varepsilon$).

Decompose each edge by its binary representation.
Compute $\mathbf{r}$ a 1.5-approximate effective resistances in G.
Let $\overline{S}$ be a set of cycles initialized to empty and let $\overline{x}$ be its corresponding partial colouring.
Set $\vec{G}' \leftarrow \vec{G}+$ ColourWeights($\overline{S}, \overline{x}$).
while $m' \geq O(n\varepsilon^{-2}\log^2 n(\log\log n)^2 + n\varepsilon^{-\frac{4}{3}}\log^{\frac{8}{3}} n)$ **do**
  if $4m \geq m'$ **then**
    $\vec{G}, \vec{G}', \overline{S}, \overline{x} \leftarrow$ ColourSparsifyGraph($\vec{G}, \vec{G}', \overline{S}, \overline{x}, \mathbf{r}$).
  else
    $\vec{G}, \vec{G}', \overline{S}, \overline{x} \leftarrow$ ColourSparsifyCycle($\vec{G}, \vec{G}', \overline{S}, \overline{x}, \mathbf{r}$).
return $\vec{G}'$.

Algorithm 5 ColourWeights(S, x).

Let $\vec{H}$ be an empty directed graph.
for *each cycle $C \in S$ and corresponding directed cycle $\vec{C}$* **do**
  Add all the clockwise (resp. counter-clockwise) edges in $\vec{C}$ with $1 + x_C$ (resp. $1 - x_C$) times their weight to $\vec{H}$. Note if $1 + x_C = 0$ (resp. $1 - x_C = 0$) the corresponding edge is not added.
return $\vec{H}$.

▶ **Lemma 4** ([6] Lemma 3.1)**.** *There exists constants $c, c' > 0$ such that given symmetric matrices $\mathbf{A}_1, \dots, \mathbf{A}_m \in \mathbb{R}^{n\times n}$ satisfying $\|\sum_{i=1}^m \mathbf{A}_i^2\| \leq \sigma^2$ and $\sum_{i=1}^m \|\mathbf{A}_i\|_F^2 \leq mf^2$ and a point $\mathbf{y} \in (-1,1)^m$, there is an algorithm* PartialColour *that returns in polynomial time a point $\mathbf{x} \in [-1,1]^m$ such that $|\{i : x_i \in \{\pm 1\}\}| > c'm$ and*

$$\left\|\sum_{i=1}^m (x_i - y_i)\mathbf{A}_i\right\| \leq c(\sigma + (\log^{\frac{3}{4}} n)\sqrt{\sigma f}). \tag{1}$$

For this section, we assume the short cycle decomposition guarantees by Lemma 7 with $\hat{m} = O(n\log n)$ and $L = O(\log n)$. For each cycle C with its corresponding directed cycle $\vec{C}$, we set $\mathbf{A}(C) = \mathsf{hlift}(\mathbf{L}_{G'}^{\frac{+}{2}}(\mathbf{L}_{\vec{F}_C} - \mathbf{L}_{S_C})\mathbf{L}_{G'}^{\frac{+}{2}})$ where $\vec{F}_C$ is the cycle $\vec{C}$ with all edges set in clockwise direction and S_C is undirected graph with the set of edges corresponding to the counter-clockwise edges in $\vec{C}$, same as in Section 3. Note that this orientation is set initialy by CorrectOrientation after a short cycle decomposition step and fixed through out the execution. Given a set of cycles S, we let $\mathcal{A}[S]$ be the collection $\{\mathbf{A}(C)\}_{C\in S}$.

ColourWeights is our partial colouring alternative of the random selection of edges in a cycle in CycleSparsifyOnce. It similarly does not change the difference between the in-degree and out-degree and preserves integral weights, stated in Lemma 13.

▶ **Lemma 13.** *Given a set of cycles S where each cycle is uniformly weighted, and any partial colouring $x \in [-1,1]^S$, the algorithm* ColourWeights *returns a directed graph $\vec{H}$ such that the difference in the in and out degrees are the same as in $\sum_{C\in S}\vec{C}$. If $x \in \{\pm 1\}^S$, $\vec{H}$ also has integral edge weights with the largest edge weight at most twice the largest edge weight in cycles in S.*

Algorithm 6 COLOURSPARSIFYGRAPH($\vec{G}, \vec{G}', \overline{S}, \overline{\mathbf{x}}, \mathbf{r}$).

Input: A directed Eulerian graph $\vec{G}$ where edge weights are integral powers of 2, a set of cycles $\overline{S}$ where each cycle is edge disjoint from G, a partial colouring $\overline{\mathbf{x}} \in (-1,1)^{\overline{S}}$, a graph $\vec{G}' = \vec{G} +$ COLOURWEIGHTS($\overline{S}, \overline{\mathbf{x}}$), a 2-approximate effective resistances $\mathbf{r}$ in G'.

Output: A directed Eulerian graph $\vec{H}$ where edge weights are integral powers of 2, a set of cycles $\overline{T}$ where each cycle is edge disjoint from H, a partial colouring $\overline{\mathbf{z}} \in (-1,1)^{\overline{T}}$, a graph $\vec{H}' = \vec{H} +$ COLOURWEIGHTS($\overline{T}, \overline{z}$).

1 Let $\vec{H} \leftarrow \vec{G}$ with only the edges which satisfy $w_e r_e > \frac{16n}{m'}$ and remove them from $\vec{G}$.
2 Partition $\vec{G}$ into uniformly weighted graph $\vec{G}_1, \ldots, \vec{G}_q$ where $\vec{G}_i$ has all edge weights 2^i and $q = O(\log n)$.
3 Let S be the set of all cycles after applying CYCLEDECOMPOSITION on $\vec{G}_1, \ldots, \vec{G}_s$ and set $\vec{H} \leftarrow \vec{H} + \sum_{i=1}^{s} \vec{G}_i \setminus \left(\bigcup_{j=1}^{t} \vec{C}_{i,j} \right)$.
4 $T', \overline{T}', \mathbf{y}, \overline{\mathbf{y}} \leftarrow$ COLOURTARGET($S, \mathbf{0}, \frac{1}{8}m$).
5 If COLOURWEIGHTS ($T', \mathbf{y}$) has more edges than COLOURWEIGHTS ($T', -\mathbf{y}$), we take $\mathbf{y} \leftarrow -\mathbf{y}$ and $\overline{\mathbf{y}} \leftarrow -\overline{\mathbf{y}}$.
6 $\vec{H} \leftarrow \vec{H} +$ COLOURWEIGHTS($T', \mathbf{y}$).
7 $\overline{T} \leftarrow \overline{T}' \cup \overline{S}$ and set $\overline{\mathbf{z}} \leftarrow \overline{\mathbf{y}} + \overline{\mathbf{x}}$.
8 $\vec{H}' \leftarrow \vec{H} +$ COLOURWEIGHTS($\overline{T}, \overline{\mathbf{z}}$).
9 **return** $\vec{H}, \vec{H}', \overline{T}, \overline{\mathbf{z}}$.

Proof. For the degree condition, it suffices to consider a single cycle C and show that the reweighted directed cycle, say $\vec{C}'$ in line 3 preserves the differences of the in and out degrees of $\vec{C}$. Recall the definition of $\vec{F}$ and S of C, see CORRECTORIENTATION,and the argument in Section 3 for showing degree differences preservation under the special case of $x \in \{\pm 1\}$. Note first that the edge weights are the same. Either a vertex has an incoming and outgoing edge (and so difference 0), in which case both edges are either in $\vec{F} \setminus S$ or in S and hence always added together with the same weights of (so still difference 0). Alternatively a vertex has two incoming or outgoing edges, in which case one edge gets a new weight of $1+x$ and the other gets $1-x$, which then still preserves the difference between in and out degree.

If $x \in \{\pm 1\}$ the edge weights of $\vec{C}'$ is exactly twice that of C unless $\vec{C}'$ is emtpy. Thus, $\vec{H}$ still has integral edge weights with largest weight at most doubled. ◀

For the rest of this section, we refer to a set of uniformly weighted cycles (two cycles can have different weights) as a set of cycles for simplicity. We write $m(S) = \sum_{C \in S} |E(C)|$ as the total number of edges in S. In COLOURSPARSIFY, COLOURSPARSIFYGRAPH and COLOURSPARSIFYCYCLE, by applying $'$ to a graph we mean $\vec{G}' = \vec{G} +$ COLOURWEIGHTS($\overline{S}, \overline{x}$). We denote m' as the number of edges in $\vec{G}'$. Note that this is the primary number of edges we consider rather than m.

Towards analyzing COLOURSPARSIFY, we first state the guarantees of the COLOURTARGET subroutine which guarantees a partial colouring of at least a specified size.

▶ **Lemma 14.** *The outputs of* COLOURTARGET(S, y, m_t) *satisfy that* $m(\overline{S}) \leq m_t$ *and the number of calls to* PARTIALCOLOUR *is* $O\left(\log(\frac{|S|L}{m_t})\right)$. *If additionally the set of cycles* S *satisfies* $\sum_{C \in S} \|\mathbf{A}(C)\| \leq \sigma^2$ *and* $\sum_{C \in S} \|\mathbf{A}(C)\|_F^2 \leq v$, *then the outputs also satisfy*

$$\left\| \sum_{C \in S} (x + \overline{x} - y) \mathbf{A}(C) \right\| \leq O\left(\sigma \cdot \log\left(\frac{|S|L}{m_t} \right) + (\log^{\frac{3}{4}} n) \sigma^{\frac{1}{2}} \left(\frac{vL}{m_t} \right)^{\frac{1}{4}} \right)$$

Algorithm 7 ColourSparsifyCycle($\vec{G}, \vec{G}', \overline{S}, \overline{\mathbf{x}}, \mathbf{r}$).

Input: A directed Eulerian graph $\vec{G}$ where edge weights are integral powers of 2, a set of cycles $\overline{S}$ where each cycle is edge disjoint from G, a partial colouring $\overline{\mathbf{x}} \in (-1,1)^{\overline{S}}$, a graph $\vec{G}' = \vec{G}+$ ColourWeights$(\overline{S}, \overline{\mathbf{x}})$, a 2-approximate effective resistances $\mathbf{r}$ in G'.

Output: A directed Eulerian graph $\vec{H}$ where edge weights are integral powers of 2, a set of cycles $\overline{T}$ where each cycle is edge disjoint from H, a partial colouring $\overline{\mathbf{z}} \in (-1,1)^{\overline{T}}$, a graph $\vec{H}' = \vec{H}+$ ColourWeights$(\overline{T}, \overline{z})$.

Set $\overline{S}'$ be an empty set of cycles initialy. For each $C \in \overline{S}$, let C' be C with its weight by $(1 - |\overline{x}_C|)$ and add C' to $\overline{S}'$.

$T', \overline{T}', \mathbf{y}, \overline{\mathbf{y}} \leftarrow$ ColourTarget$(\overline{S}', \mathbf{0}, \frac{1}{4}m')$

if $m(\{C' \in T' : |\overline{x}_C - (1 - |\overline{x}_C|)y_{C'}| = 1\}) > m(\{C' \in T' : |\overline{x}_C + (1 - |\overline{x}_C|)y_{C'}| = 1\})$ **then**

  $\mathbf{y} \leftarrow -\mathbf{y}, \overline{\mathbf{y}} \leftarrow -\overline{\mathbf{y}}$.

Set $\mathbf{z}, \overline{\mathbf{z}}$ to be the parts of $\overline{\mathbf{x}} + (1 - |\overline{\mathbf{x}}|) \circ (\mathbf{y} + \overline{\mathbf{y}})$ with magnitude 1 and < 1 respectively. Here we abused $\circ$ to let C and C' refering to the same index, Set the partition $T, \overline{T}$ of $\overline{S}$ accordingly.

$\vec{H} \leftarrow \vec{H} +$ ColourWeights$(T, \mathbf{z})$.

$\vec{H}' \leftarrow \vec{H}+$ ColourWeights$(\overline{T}, \overline{\mathbf{z}})$.

return $\vec{H}, \vec{H}', \overline{T}, \overline{\mathbf{z}}$.

Proof. Notice that each cycle has its number of edges bounded by L. We have $m(\overline{S}) \leq L|\overline{S}| \leq m_t$ by the terminating condition of the while loop in ColourTarget. Since the size of $\overline{S}$ decreases by a factor of $1 - c'$ by Lemma 4, by the ith round we have $|\overline{S}| \leq (1 - c')^i |S|$ and at termination this is $\leq \frac{m_t}{L}$. This then gives the claimed number of iterations.

Consider the error bound. Combine the number of iterations with the first term in (1) of Lemma 4, we get our desired first term. For the second term, recall from above that $|\overline{S}|$ decreases geometrically. Then $f = \left(\frac{v}{|\overline{S}|}\right)^{\frac{1}{2}}$ increases exponentially over the iterations. Hence the sum of the second terms in (1) is bounded by the last one with $f = O\left((\frac{vL}{m_t})^{\frac{1}{2}}\right)$, giving

$$O\left((\log^{\frac{3}{4}} n)\sigma^{\frac{1}{2}} f^{\frac{1}{2}}\right) = O\left((\log^{\frac{3}{4}} n)\sigma^{\frac{1}{2}} \left(\frac{vL}{m_t}\right)^{\frac{1}{4}}\right)$$

as required. ◀

Now, parallel to Lemma 11, we state the approximation guarantees of ColourSparsifyGraph and ColourSparsifyCycle in Lemmas 15 and 16. The proof of Lemma 15 follows closely to that of Lemma 16 and we refer reader to the full version of our paper.

▶ **Lemma 15.** *If the input graphs $\vec{G}, \vec{G}'$ satisfy $4m \geq m'$ and the input set of cycles $\overline{S}$ and it corresponding partial colours $\overline{x}$ satisfies that each cycle $C \in \overline{S}$ has $w_e r_e \leq \frac{4n}{m'}$ for each edge $e \in C$, the algorithm* ColourSparsifyGraph *returns $\vec{H}$ with edge weights still being powers of 2 and at most twice the largest weight in $\vec{G}$, a set of cycles $\overline{T}$ with its corresponding partial colours $\overline{\mathbf{z}}$ satisfying $\vec{H}' = \vec{H}+$* ColourWeights$(\overline{T}, \overline{\mathbf{z}})$ *is an Eulerian graph and each cycle $C \in \overline{T}$ also has $w_e r_e \leq \frac{4n}{m'_H}$ for each edge $e \in C$, where $m'_H = |E(\vec{H})|$. and,*

$$\left\|\mathbf{L}_{\vec{G}'}^{\frac{+}{2}}(\mathbf{L}_{\vec{G}'} - \mathbf{L}_{\vec{H}'})\mathbf{L}_{\vec{G}'}^{\frac{+}{2}}\right\| \leq O\left(\sqrt{\frac{n\log^2 n}{m'}}\log\log n + \left(\frac{n\log^{\frac{8}{3}} n}{m'}\right)^{\frac{3}{4}}\right).$$

Algorithm 8 ColourTarget(S, y, m_t).

Input: A set of cycles S of size $s = |S|$, a partial colouring $y \in (-1,1)^S$, and a target mass of m_t edges.

Output: A set of fully coloured cycles $S \backslash \overline{S}$ with colouring x, A set of partially coloured cycles $\overline{S}$ with colouring $\overline{x}$ satisfying $\overline{x} \in (-1,1)^{\overline{S}}$.

1 Initialize $x = 0$ be a empty colouring over S.
2 Define $\overline{S}$ to always be the set of fractionally coloured cycles in S and let $\overline{s} = |\overline{S}|$ always. Set $\overline{x}$ be the partial colour on $\overline{S}$ always.
3 **while** $\overline{s} > \frac{m_t}{L}$ **do**
4 $\quad x[\overline{S}] \leftarrow$ PartialColour$(\mathcal{A}[\overline{S}], \overline{x})$.
5 Let $\overline{x} \leftarrow x$ with entries of magnitude < 1 and set $x \leftarrow x - \overline{x}$.
6 **return** $S \backslash \overline{S}, \overline{S}, x, \overline{x}$.

▶ **Lemma 16.** *If the input set of cycles $\overline{S}$ and it corresponding partial colours $\overline{x}$ satisfies that each cycle $C \in \overline{S}$ has $w_e r_e \leq \frac{4n}{m'}$ for each edge $e \in C$, the algorithm* ColourSparsifyCycle *returns $\vec{H}$ with edge weights still being powers of 2 and at most twice the largest weight in $\vec{G}$, a set of cycles $\overline{T}$ with its corresponding partial colours $\overline{y}$ satisfying $\vec{H}' = \vec{H}+$* ColourWeights$(\overline{T}, \overline{y})$ *is an Eulerian graph and each cycle $C \in \overline{T}$ also has $w_e r_e \leq \frac{4n}{m'_H}$ for each edge $e \in C$, where $m'_H = |E(\vec{H})|$. and,*

$$\left\|\mathbf{L}_{\vec{G}'}^{\frac{\pm}{2}}(\mathbf{L}_{\vec{G}'} - \mathbf{L}_{\vec{H}'})\mathbf{L}_{\vec{G}'}^{\frac{\pm}{2}}\right\| \leq O\left(\sqrt{\frac{n\log^2 n}{m'}}\log\log n + \left(\frac{n\log^{\frac{8}{3}} n}{m'}\right)^{\frac{3}{4}}\right).$$

Before we prove Lemma 16, we need Lemma 17 regarding scaling matrices in the set of extra cycles $\overline{S}$.

▶ **Lemma 17.** *For directed Eulerian graph $\vec{G}$, a set of cycles $\overline{S}$ where each cycle $C \in \overline{S}$ satisfies that $\vec{G}$ and $\vec{C}$, the corresponding directed cycle of C, are edge-disjoint. Let $\overline{x} \in (-1,1)^S$ be a fractional colouring on $\overline{S}$. Then the Eulerian graph $\vec{G}' = \vec{G}+$* ColourWeights$(\overline{S}, \overline{x})$ *satisfies*

$$\mathbf{L}_G + \sum_{C \in \overline{S}} (1 - |\overline{x}_C|)\mathbf{L}_C \preceq \mathbf{L}_{G'}.$$

Proof. For any $C \in \overline{S}$, let $\vec{C}' =$ ColourWeights$(C, \overline{x}_C)$ where we abused the definition to take in a single cycle instead of a set of cycles. Note that the undirectification $\mathbf{L}_{G'} = \mathbf{L}_G + \sum_{C \in \overline{S}} \mathbf{L}_{C'}$. Since $|\overline{x}_C| < 1$, all edges in C must be present in C' and the minimum edge weight is at least $1 - |\overline{x}_C|$ times the original uniform edge weights of C. Then, $(1 - |\overline{x}_C|)\mathbf{L}_C \preceq \mathbf{L}_{C'}$. Summing over all C, we get

$$\mathbf{L}_G + \sum_{C \in \overline{S}} (1 - |\overline{x}_C|)\mathbf{L}_C \preceq \mathbf{L}_G + \sum_{C \in \overline{S}} \mathbf{L}_{C'} = \mathbf{L}_{G'}.$$ ◀

Proof of Lemma 16. The edge weights condition of $\vec{H}$ is guaranteed by Lemma 13. Also by Lemma 13, both $\vec{H}$ and $\vec{H}'$ are Eulerian. Observe that $m'_H \leq m$ always, and $\overline{T} \subset \overline{S}$. Then, the output cycles still satisfy the approximate leverage score condition. Now, by line 5, the output Eulerian graph $\vec{H}'$ satisfies

$$\mathsf{hlift}\left(\mathbf{L}_{\vec{G}'}^{\frac{\pm}{2}}(\mathbf{L}_{\vec{H}'} - \mathbf{L}_{\vec{G}'})\mathbf{L}_{\vec{G}'}^{\frac{\pm}{2}}\right) = \sum_{C \in \overline{S}} (z_C + \overline{z}_C - x_C)\mathbf{A}(C) = \sum_{C \in \overline{S}} (1 - |x_C|)(y_C + \overline{y}_C)\mathbf{A}(C)$$

where all vectors are taken as the final values. By definition $\mathbf{A}(C') = (1 - |x_C|)\mathbf{A}(C)$ and

$$\sum_{C \in \overline{S}} (1 - |x_C|)(y_C + \overline{y}_C)\mathbf{A}(C) = \sum_{C' \in \overline{S}'} (y_C + \overline{y}_C)\mathbf{A}(C')$$

By definition of hlift, each matrix $\mathbf{A}(C)^2$ is block diagonal with blocks $\mathbf{L}_{G'}^{\frac{+}{2}}\widetilde{\mathbf{L}}_{\vec{C}}^{\top}\mathbf{L}_{G'}^{+}\widetilde{\mathbf{L}}_{\vec{C}}\mathbf{L}_{G'}^{\frac{+}{2}}$ and $\mathbf{L}_{G'}^{\frac{+}{2}}\widetilde{\mathbf{L}}_{\vec{C}}\mathbf{L}_{G'}^{+}\widetilde{\mathbf{L}}_{\vec{C}}^{\top}\mathbf{L}_{G'}^{\frac{+}{2}}$. Here $\widetilde{\mathbf{L}}_{\vec{C}} = \mathbf{L}_{\vec{F}} - \mathbf{L}_S$ with fixed orientation (recall CorrectOrientation). Since every cycle $C \in \overline{S}$ satisfies $\boldsymbol{\tau}_{G'}(e) \leq \rho$ for each $e \in C$, by Lemma 10, both matrices are spectrally bounded by $O(L\rho) \cdot \mathbf{L}_{G'}^{\frac{+}{2}}\mathbf{L}_C\mathbf{L}_{G'}^{\frac{+}{2}}$. Thus, by the disjointness of G and $\overline{S}$,

$$\sum_{C' \in \overline{S}'} \mathbf{A}(C')^2 \preceq \sum_{C \in \overline{S}} (1 - |x_C|)\mathbf{A}(C)^2 \preceq O(L\rho) \cdot \mathbf{I}_2 \otimes \mathbf{L}_{G'}^{\frac{+}{2}} \left(\sum_{C \in \overline{S}} (1 - |x_C|)\mathbf{L}_C \right) \mathbf{L}_{G'}^{\frac{+}{2}}$$
$$\preceq O(L^2\rho) \cdot \mathbf{I}_{2n},$$

where we used the PSD property of $\mathbf{A}(C)^2$ and the fact $1 - |x_C| \leq 1$ for the first inequality and Lemma 17 for the second inequality. The sum of Frobenius norm squared is then

$$\sum_{C' \in \overline{S}'} \|\mathbf{A}(C')\|_F^2 \leq \sum_{C \in \overline{S}} (1 - |x_C|) \operatorname{Tr}\left(\mathbf{A}(C)^2\right) = \operatorname{Tr}\left(\sum_{C \in \overline{S}} (1 - |x_C|)\mathbf{A}(C)^2 \right) = O(nL^2\rho).$$

We can now apply Lemma 14 with $m_t = \frac{1}{4}m', \sigma^2 = O(L^2\rho)$ and $v = O(nL^2\rho)$ to get

$$\left\| \sum_{C' \in \overline{S}'} (y_C + \overline{y}_C - 0)\mathbf{A}(C') \right\| \leq O\left(\sqrt{L^2\rho} \cdot \log\left(\frac{4|\overline{S}|L}{m'} \right) + (\log^{\frac{3}{4}} n)(L^2\rho)^{\frac{1}{4}} \left(\frac{4nL^3\rho}{m'} \right)^{\frac{1}{4}} \right)$$
$$= O\left(\sqrt{\frac{nL^2}{m'}} \log L + \left(\frac{nL^{\frac{5}{3}} \log n}{m'} \right)^{\frac{3}{4}} \right)$$

where we used $|\overline{S}'| = |\overline{S}| \leq m'$. Finally, note that

$$\left\| \mathbf{L}_{G'}^{\frac{+}{2}} (\mathbf{L}_{\vec{G}'} - \mathbf{L}_{\vec{H}'}) \mathbf{L}_{G'}^{\frac{+}{2}} \right\| = \left\| \mathsf{hlift}\left(\mathbf{L}_{G'}^{\frac{+}{2}} (\mathbf{L}_{\vec{G}'} - \mathbf{L}_{\vec{H}'}) \mathbf{L}_{G'}^{\frac{+}{2}} \right) \right\| = \left\| \sum_{C' \in \overline{S}'} (y_C + \overline{y}_C - 0)\mathbf{A}(C) \right\|. \blacktriangleleft$$

The sparsification induced by ColourSparsifyGraph is conditional, and we state the condition and sparisification induced in Lemma 18. However, even when the condition is not met, we are guaranteed each ColourSparsifyCycle will geometrically make progress towards satisfying the condition needed for Lemma 18. This is stated in Lemma 19.

▶ **Lemma 18.** *For inputs $\vec{G}, \vec{G}', \overline{S}, \overline{x}, \mathbf{r}$ to* ColourSparsifyGraph *satisfying that $4m \geq m' \geq \Omega(n \log^2 n)$, the outputs satisfy that the number of edges in $\vec{H}'$ is upperbounded by $m'_H \leq \frac{63}{64}m'$.*

Proof. Since $\mathbf{r}$ is 2-approximate effective resistances, $\sum_e w_e r_e \leq 2(n-1)$, we have at most $\frac{1}{8}m' \leq \frac{1}{2}m$ edges are removed from $\vec{G}$ in line 1. Since $m \geq \frac{1}{4}m' = \Omega(n \log^2 n)$ and the number of edges not in any cycle is $\hat{m}q = O(n \log^2 n)$, by picking an appropriate constant in $\Omega(n \log^2 n)$, we can guarantee the total number of edges in all cycles satisfies $m(S) \geq \frac{1}{4}m$. Lemma 14 then guarantees $m(\overline{T}') \leq \frac{1}{8}m$ and that $m(T') \geq \frac{1}{8}m$.

Now, by ColourWeights, the total number of edges in ColourWeights$(T', \mathbf{y})$ and ColourWeights$(T', -\mathbf{y})$ is exactly $m(T')$. Thus, line 5 means at least $\frac{1}{2}m(T') \geq \frac{1}{16}m \geq \frac{1}{64}m'$ edges are removed in total as required. ◀

▶ **Lemma 19.** *If inputs $\vec{G}, \vec{G}', \overline{S}, \overline{x}, \mathbf{r}$ to* ColourSparsifyCycle *satisfies that $4m < m'$, then either the number of edges in $\vec{H}'$ decreases to $m'_H \le \frac{63}{64}m'$, or the number of edges in $\vec{H}$ satisfies $4m_H \ge m'_H$.*

Proof. Suppose $m'_H > \frac{63}{64}m'$. By Lemma 14, $m(\overline{T'}) \le \frac{1}{8}m'$. Since $\mathbf{y} \in \{\pm 1\}^{T'}$, we have $\{C' \in T' : |\overline{x}_C - (1 - |\overline{x}_C|)y_{C'}| = 1\} \cup \{C' \in T' : |\overline{x}_C + (1 - |\overline{x}_C|)y_{C'}| = 1\} = T'$. Let the two sets above be T'_1 and T'_2, Then, $m(T'_1) + m(T'_2) \ge m(T')$ [2]. This means, after re-adjusting the colouring in line 4,

$$m(\overline{T}) \le \frac{1}{2}m(T') + m(\overline{T'}) \le \frac{1}{2}m' + \frac{1}{8}m' = \frac{5}{8}m' \le \frac{40}{63}m'_H.$$

Then, we get the desired inequality, $m_H = m'_H - m(\overline{T}) \ge \frac{23}{63}m'_H \ge \frac{1}{4}m'_H$. ◀

With these analyses above, we can now formally state and prove Theorem 2.

▶ **Theorem 20** (Theorem 2 Formal)**.** *Given input a Eulerian graph $\vec{G}$ with polynomial bounded integral edge weights and $\varepsilon \in (0, \frac{1}{2})$, the algorithm* ColourSparsify *returns in polynomial time a Eulerian graph $\vec{H}$ with $O(n\varepsilon^{-2}\log^2 n(\log\log n)^2 + n\varepsilon^{-\frac{3}{4}}\log^{\frac{8}{3}} n)$ edges satisfying*

$$\left\| \mathbf{L}_G^{\frac{\pm}{2}} (\mathbf{L}_{\vec{G}} - \mathbf{L}_{\vec{H}}) \mathbf{L}_G^{\frac{\pm}{2}} \right\| \le \varepsilon.$$

Proof. By Lemmas 18 and 19, in every two iterations the number of edges must decreases by at least a constant fraction, as the condition $4m \ge m'$ must be satisfied at least once. Note that initialy $m = m' \ge \frac{1}{4}m'$ is satisfied. Thus, the total number of iterations is at most $O\left(\log(\frac{m \log n}{n})\right) = O(\log n)$ where the extra $\log n$ comes from the decomposition by weights.

By Lemmas 15 and 16, the largest edge weight doubles each iteration. Thus, the edge weights in each $\vec{G}$ are still integral and polynomially bounded over $O(\log n)$ iterations.

As the number of edges decreases geometrically every $O(1)$ iterations, the total error is asymptotically bounded by the error in the last round for both terms in Lemmas 15 and 16:

$$O\left(\sqrt{\frac{n\log^2 n}{m'}}\log\log n + \left(\frac{n\log^{\frac{8}{3}} n}{m'}\right)^{\frac{3}{4}}\right).$$

where m' is the number of edges in $\vec{G}'$ in the last round. Since the algorithm stops at $m' \ge \Omega(n\varepsilon^{-2}\log^2 n(\log\log n)^2)$ and $m' \ge \Omega(n\varepsilon^{-\frac{3}{4}}\log^{\frac{8}{3}} n)$ edges, the largest of both terms must be bounded by $\frac{1}{2}\varepsilon$ by picking appropriate constant for the stopping condition.

This small error also implies that our 1.5-approximate effective resistances $\mathbf{r}$ stays as 2-approximate throughout the algorithm. Then, by Lemma 15 and Lemma 16, the set of cycles $\overline{S}$ always satisfy $w_e r_e \le \frac{4n}{m'}$ where m' is the number of edges in $\vec{G}'$ throughout as required. Lemma 4 guarantees the polynomial running time of our algorithm. ◀

4.1 Conjectural improvements

In this section we consider an improvement on our existential results due to the partial colouring conjecture, Conjecture 21. Corollary 22 then follows by changing the termination condition of the while loop on line 5 to $m' \ge O(n\varepsilon^{-2}\log^2 n(\log\log n)^2)$.

[2] Contrary to the proof of Lemma 18, this is an inequality since magnitude of 1 can be achieved using both $y_{C'}$ and $-y_{C'}$ if $x_C = 0$.

▶ **Conjecture 21** (Matrix partial colouring conjecture). *There exists constants* $c_1, c_2 > 0$ *and* $c_3 > 1$ *such that the following holds. Given symmetric matrices* $\mathbf{A}_1, \ldots, \mathbf{A}_m \in \mathbb{R}^{n \times n}$ *that satisfy* $m \geq c_3 n$, $\|\sum_{i=1}^m \mathbf{A}_i^2\| \leq \sigma^2$, *and a point* $\mathbf{y} \in (-1, 1)^m$, *there exists a point* $\mathbf{x} \in [-1, 1]^m$ *such that* $|\{i : x_i \in \{\pm 1\}\}| > c_2 m$ *and*

$$\left\|\sum_{i=1}^{m}(x_i - y_i)\mathbf{A}_i\right\| \leq c_1 \sigma. \tag{3}$$

▶ **Corollary 22.** *Assume Conjecture 21. There is an algorithm that given a Eulerian graph* $\vec{G}$, *computes a* ε*-Eulerian sparsifier of* $\vec{G}$ *with* $n\varepsilon^{-2}\log^2 n$ *edges (up to* $\log\log n$ *factors).*

▶ Remark 23. While improvements on matrix concentration results for Gaussian random variables [5] naturally leads to improved matrix partial colouring through the Gaussian measure analysis of matrix discrepancy bodies, Conjecture 21 need not rely on this approach (e.g. [9, 15]). On the other hand, even if the matrix concentration guarantees of [5] hold for Rademacher random variables, it does not lead to an efficient algorithm for Theorem 2. This is due to the difficulties in controlling the matrix covariance factor in Theorem 1.2 of [5]. We refer reader to the proof of Lemma 3.1 in [6].

References

1 AmirMahdi Ahmadinejad, Arun Jambulapati, Amin Saberi, and Aaron Sidford. *Perron-Frobenius Theory in Nearly Linear Time: Positive Eigenvectors, M-matrices, Graph Kernels, and Other Applications*, pages 1387–1404. Society for Industrial and Applied Mathematics, 2019. doi:10.1137/1.9781611975482.85.

2 AmirMahdi Ahmadinejad, John Peebles, Edward Pyne, Aaron Sidford, and Salil Vadhan. Singular value approximation and sparsifying random walks on directed graphs. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 846–854, 2023. doi:10.1109/FOCS57990.2023.00054.

3 Zeyuan Allen-Zhu, Zhenyu Liao, and Lorenzo Orecchia. Spectral sparsification and regret minimization beyond matrix multiplicative updates. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 237–245, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2746539.2746610.

4 Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P. Woodruff, and Qin Zhang. On sketching quadratic forms. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, pages 311–319, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2840728.2840753.

5 Afonso S Bandeira, March T Boedihardjo, and Ramon van Handel. Matrix concentration inequalities and free probability. *Inventiones mathematicae*, pages 1–69, 2023.

6 Nikhil Bansal, Haotian Jiang, and Raghu Meka. Resolving matrix spencer conjecture up to poly-logarithmic rank. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pages 1814–1819, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3564246.3585103.

7 Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.

8 Joshua Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Rev.*, 56(2):315–334, January 2014. doi:10.1137/130949117.

9 József Beck. Roth's estimate of the discrepancy of integer sequences is nearly sharp. *Combinatorica*, 1(4):319–325, 1981.

10 András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in Õ(n2) time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 47–55, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/237814.237827.

11 Timothy Chu, Yu Gao, Richard Peng, Sushant Sachdeva, Saurabh Sawlani, and Junxing Wang. Graph sparsification, spectral sketches, and faster resistance computation, via short cycle decompositions. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 361–372, 2018. doi:10.1109/FOCS.2018.00042.

12 Michael B. Cohen, Jonathan Kelner, Rasmus Kyng, John Peebles, Richard Peng, Anup B. Rao, and Aaron Sidford. Solving directed laplacian systems in nearly-linear time through sparse lu factorizations. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 898–909, 2018. doi:10.1109/FOCS.2018.00089.

13 Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian 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*, STOC 2017, pages 410–419, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3055399.3055463.

14 Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. Faster algorithms for computing the stationary distribution, simulating random walks, and more. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 583–592, 2016. doi:10.1109/FOCS.2016.69.

15 Daniel Dadush, Haotian Jiang, and Victor Reis. A new framework for matrix discrepancy: Partial coloring bounds via mirror descent. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 649–658, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3519967.

16 David Durfee, Rasmus Kyng, John Peebles, Anup B Rao, and Sushant Sachdeva. Sampling random spanning trees faster than matrix multiplication. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 730–742. ACM, 2017. arXiv:1611.07451.

17 David Durfee, John Peebles, Richard Peng, and Anup B. Rao. Determinant-preserving sparsification of SDDM matrices with applications to counting and sampling spanning trees. In *FOCS*, pages 926–937. IEEE Computer Society, 2017. arXiv:1705.00985.

18 Samuel B. Hopkins, Prasad Raghavendra, and Abhishek Shetty. Matrix discrepancy from quantum communication. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 637–648, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3519954.

19 Arun Jambulapati, Victor Reis, and Kevin Tian. *Linear-Sized Sparsifiers via Near-Linear Time Discrepancy Theory*, pages 5169–5208. Society for Industrial and Applied Mathematics, 2023. doi:10.1137/1.9781611977912.186.

20 Arun Jambulapati and Aaron Sidford. Efficient $\tilde{O}(n/\epsilon)$ spectral sketches for the laplacian and its pseudoinverse. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 2487–2503, USA, 2018. Society for Industrial and Applied Mathematics.

21 Michael Kapralov and Rina Panigrahy. Spectral sparsification via random spanners. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 393–398, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2090236.2090267.

22 Ioannis Koutis, Alex Levin, and Richard Peng. Improved Spectral Sparsification and Numerical Algorithms for SDD Matrices. In Thomas Wilke Christoph Dürr, editor, *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 266–277, Paris, France, February 2012. LIPIcs. URL: https://hal.science/hal-00678205.

23 Ioannis Koutis and Shen Chen Xu. Simple parallel and distributed algorithms for spectral graph sparsification. *ACM Trans. Parallel Comput.*, 3(2), August 2016. doi:10.1145/2948062.

24 Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 842–850, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2897518.2897640.

25 Rasmus Kyng, Simon Meierhans, and Maximilian Probst Gutenberg. Derandomizing directed random walks in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 407–418, 2022. doi:10.1109/FOCS54457.2022.00046.

26 Rasmus Kyng, Jakub Pachocki, Richard Peng, and Sushant Sachdeva. *A Framework for Analyzing Resparsification Algorithms*, pages 2032–2043. Society for Industrial and Applied Mathematics, 2017. doi:10.1137/1.9781611974782.132.

27 Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians - fast, sparse, and simple. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 573–582, October 2016. doi:10.1109/FOCS.2016.68.

28 Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, pages 250–269, USA, 2015. IEEE Computer Society. doi:10.1109/FOCS.2015.24.

29 Yin Tat Lee and He Sun. An sdp-based algorithm for linear-sized spectral sparsification. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 678–687, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3055399.3055477.

30 Avi Levy, Harishchandra Ramadas, and Thomas Rothvoss. Deterministic discrepancy minimization via the multiplicative weight update method. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 380–391. Springer, 2017.

31 Huan Li and Zhongzhi 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*, pages 2377–2396. SIAM, 2018. arXiv:1708.05959.

32 Yang P. Liu, Sushant Sachdeva, and Zejun Yu. Short cycles via low-diameter decompositions. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 2602–2615, USA, 2019. Society for Industrial and Applied Mathematics.

33 Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families ii: Mixed characteristic polynomials and the kadison—singer problem. *Annals of Mathematics*, pages 327–350, 2015.

34 Raghu Meka. Discrepancy and beating the union bound. *Windows on theory, a research blog*, 2014.

35 Merav Parter and Eylon Yogev. Optimal Short Cycle Decomposition in Almost Linear Time. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 89:1–89:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2019.89.

36 Richard Peng and Zhuoqing Song. Sparsified block elimination for directed laplacians. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 557–567, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520053.

37 Richard Peng and Daniel A. Spielman. An efficient parallel solver for sdd linear systems. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 333–342, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591832.

38 Victor Reis and Thomas Rothvoss. Linear size sparsifier and the geometry of the operator norm ball. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, pages 2337–2348, USA, 2020. Society for Industrial and Applied Mathematics.

39 Victor Reis and Thomas Rothvoss. Vector balancing in lebesgue spaces. *Random Structures & Algorithms*, 62(3):667–688, 2023.

40 Sushant Sachdeva and Yibin Zhao. A simple and efficient parallel laplacian solver. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '23, pages 315–325, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3558481.3591101.

41 Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 2616–2635, USA, 2019. Society for Industrial and Applied Mathematics.

42 D. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. doi:10.1137/080734029.

43 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 563–568, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1374376.1374456.

44 Daniel A. Spielman and Shang-Hua 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*, STOC '04, pages 81–90, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/1007352.1007372.

45 Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, July 2011. doi:10.1137/08074489X.

46 Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

47 Anastasios Zouzias. A matrix hyperbolic cosine algorithm and applications. In *International Colloquium on Automata, Languages, and Programming*, pages 846–858. Springer, 2012.