Thesis for the degree
Master of Science

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

עבודת גמר (תזה) לתואר
מוסמך למדעים

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

By
Yosef Pogrow

מאת
יוסף פגראו

פתרון מערכות ליניאריות סימטריות נשלטות אלכסון בזמן תת-לינארי (וכמה אבחנות על דילול גרפים)

Solving Symmetric Diagonally Dominant Linear Systems in Sublinear Time (and Some Observations on Graph Sparsification)

Advisor:
Prof. Robert Krauthgamer

מנחה:
פרופ' רוברט קראוטגמר

September 2017

תשרי התשע"ח

Abstract

This dissertation is divided into two chapters, each investigating a different topic. The first chapter studies *sublinear* algorithms for solving symmetric diagonally dominant (SDD) linear systems. In the classical version of this problem the input is a matrix $S \in \mathbb{R}^{n \times n}$ which is SDD, and a vector $b \in \mathbb{R}^n$ in the range of S, and the goal is to output $x \in \mathbb{R}^n$ satisfying $Sx = b$. The breakthrough algorithm of Spielman and Teng [STOC 2004] approximately solves this problem in near-linear time (in the input size which is the number of non-zeros in S), and subsequent papers have simplified and improved the runtime even further.

Our goal is to approximately solve such a linear system for a particular coordinate. Formally, given an index $u \in [n]$ together with S and b as above, the goal is to output an approximation $\hat{x}_u$ for x^*_u, where $x^* \in \mathbb{R}^n$ is a fixed solution to $Sx = b$. We present an algorithm that approximates a single coordinate x_u in time that is polylogarithmic in n whenever S has a certain spectral gap (including Laplacians of regular expander graphs). The approximation guarantee is additive $|\hat{x}_u - x^*_u| \leq \epsilon \cdot \|x^*\|_\infty$ for a parameter $\epsilon > 0$. An example application of our algorithm is approximating the effective resistance between a pair of vertices in a constant degree expander within relative error ϵ, in time $\text{poly}(\frac{1}{\epsilon})$. To complement our algorithm, we show that the spectral gap assumption is necessary: we exhibit general SDD matrices S (in fact, Laplacians of regular graphs), for which approximating a single coordinate x_u requires $\Omega(n)$ (randomized) time.

The second chapter studies cut-sparsifiers for graphs/hypergraphs, and spectral-sparsifiers for graphs, which were introduced by Benczúr and Karger [STOC 1996] and by Spielman and Teng [STOC 2004] respectively. In fact, graph sparsification is closely related to solving linear systems in SDD matrices, as the aforementioned solver of Spielman and Teng is based on a (fast) construction of spectral-sparsifiers. A k-cut-sparsifier of a graph G is an (ideally sparse) graph G' such that the weight of every cut S in G', denoted $w_{G'}(S, \bar{S})$, approximates the corresponding $w_G(S, \bar{S})$ within factor $k \geq 1$. This definition naturally extends to hypergraphs. A k-spectral-sparsifier of an n-vertex graph G is an (ideally sparse) graph G' such that for every $x \in \mathbb{R}^n$, the quadratic form $x^T L_{G'} x$ (where $L_{G'}$ is the Laplacian matrix of G') approximates $x^T L_G x$ within factor $k \geq 1$.

We present several new bounds on the size and the approximation factor of sparsifiers in certain families/cases. Specifically, we improve the known upper bounds on the size of $(1+\epsilon)$-cut-sparsifiers of hypergraphs with bounded edge-intersections, i.e., where the intersection of every two distinct hyperedges is small. We show limitations on the approximation factor that can be achieved by sparsifiers with (only) small hyperedges. And we exhibit upper and lower bounds on the approximation factor that can be achieved by tree-sparsifiers, i.e., when the sparsifying graph must be a tree.

BS"D

Acknowledgements

At the end of this two year journey, it is only appropriate I acknowledge those who made it a great one.

First and foremost, I would like to express my deep gratitude to my advisor, Prof. Robert Krauthgamer. For always making himself available to help, for sharing his resourceful ideas, and for not letting me get away with less than perfect writing. Robi, it was great to learn from you, thanks for everything.

I would also like to thank the Weizmann Institute for providing such an amazing research atmosphere, and for all their support.

I wish to thank my friends that accompanied me during this journey, Eran Amar, Ron Shiff, Inbal Rika, Yevgeny Levanzov, Ben Berger, Otniel van Handel, and many others. It was great brainstorming with you.

I thank Prof. Alexandr Andoni for collaborating with me and Robi to work on the research presented in Chapter 1 of this thesis.

Finally, I wish to thank my family. You are a great source of support.

Contents

Chapter 1

Solving SDD Linear Systems in Sublinear Time*

1.1 Introduction

Solving linear systems is a fundamental problem in many areas. A basic version of the problem has as input a matrix $A \in \mathbb{R}^{n \times n}$ and a vector $b \in \mathbb{R}^n$, and the goal is to find $x \in \mathbb{R}^n$ such that $Ax = b$. The fastest known algorithm for general A is by a reduction to matrix multiplication, and takes $O(n^\omega)$ time, where $\omega < 2.372$ [Gal14] is the matrix multiplication exponent. When A is sparse, one can do better (by applying the conjugate gradient method to the equivalent positive semi-definite (PSD) system $A^T Ax = A^T b$, see for example [Spi10]), namely, $O(mn)$ time where $m = \text{nnz}(A)$ is the number of non-zeros in A. Note that this $O(mn)$ bound for exact solvers assumes exact arithmetic, and in practice, one seeks fast approximate solvers.

One interesting subclass of PSD matrices is the class of symmetric diagonally dominant (SDD) matrices — recall that a symmetric matrix $S \in \mathbb{R}^{n \times n}$ is said to be SDD if $S_{ii} \geq \sum_{j \neq i} |S_{ij}|$ for all $i \in [n]$. Many applications require solving linear systems in SDD matrices, and particularly their subclass of graph-Laplacian matrices, see e.g. [Spi10, Vis13, CKM$^+$14]. Solving SDD linear systems received a lot of attention in the past decade after the breakthrough result by Spielman and Teng in 2004 [ST04], showing that a linear system in SDD matrix S can be solved approximately in near-linear time $O(m\log^{O(1)} n \log \frac{1}{\epsilon})$, where $m = \text{nnz}(S)$ and $\epsilon > 0$ is an accuracy parameter. A series of improvements led to the state-of-the-art SDD solver of Cohen et al. [CKM$^+$14] that runs in near-linear time $O(m\sqrt{\log n}(\log\log n)^{O(1)} \log \frac{1}{\epsilon})$.

We continue the line of research on SDD solvers by studying whether better run-times are possible if we only need a particular coordinate of the solution $x \in \mathbb{R}^n$. In particular, given an SDD matrix $S \in \mathbb{R}^{n \times n}$, a vector $b \in \mathbb{R}^n$, and an index $u \in [n]$, output the coordinate x_u where $x \in \mathbb{R}^n$ is a solution to the linear system $Sx = b$. The goal is then to compute x_u in time *sublinear* in n. Indeed, our main result is that, under some conditions on S, we can approximate a single coordinate x_u in polylogarithmic time.

When solving for a particular coordinate, we need to address the issue of consistency between coordinates, which arises when the system is underdetermined. For example, when the all-ones

*The results in this chapter are joint work with Alexandr Andoni, and written in [AKP17].

vector $\vec{1}$ is in the null space of S, then for every $\alpha \in \mathbb{R}$ there is a solution $x \in \mathbb{R}^n$ with $x_u = \alpha$, which renders the aforementioned goal useless. To rectify it, we require that the algorithm approximates a single "global" solution, i.e., if it is invoked with different indices $u \in [n]$, the outputted coordinates will all be consistent with one solution x^*. Formally, given an SDD matrix $S \in \mathbb{R}^{n \times n}$ and a vector $b \in \mathbb{R}^n$ in the range (column space) of S, as well as accuracy parameter $\epsilon > 0$, there exists a solution $x^* \in \mathbb{R}^n$ satisfying $Sx^* = b$, such that for all $u \in [n]$, the algorithm outputs $\hat{x}_u$ satisfy

$$\Pr\left[|\hat{x}_u - x^*_u| \leq \epsilon ||x^*||_\infty\right] \geq \frac{3}{4}.$$

As usual, the success probability can be amplified to $1 - \delta$ by taking the median of $O(\log \frac{1}{\delta})$ independent executions. We note that the above guarantee, when applied to all coordinates, gives a vector $\hat{x}$ with $\|\hat{x} - x^*\|_\infty \leq \epsilon ||x^*||_\infty$, which differs from the guarantee from [ST04], that $||\hat{x} - x^*||_S \leq \epsilon ||x^*||_S$, where $||y||_S \stackrel{\text{def}}{=} \sqrt{y^T S y}$.

The above problem is motivated also by the existence of *quantum* sublinear-time algorithms for solving linear systems, which were introduced in [HHL09] and subsequently improved in [Amb12, CKS15]. In particular, [HHL09] consider the system $Ax = b$, given: (1) oracle access to entries of A (including fast access to the j-th non-zero entry of i-th row), and (2) a fast black-box procedure to prepare a quantum state $|b\rangle = \sum_i \frac{b_i |i\rangle}{\|\sum_i b_i |i\rangle\|}$. Then, if the matrix A has condition number κ, at most d non-zeros per row/column, and $\|A\| = 1$, their quantum algorithm runs in time $\text{poly}(\kappa, d, 1/\epsilon)$, and outputs a quantum state $|\hat{x}\rangle$ within ℓ_2-distance ϵ from $|x\rangle = \frac{\sum_i x_i |i\rangle}{\|\sum_i x_i |i\rangle\|}$. The runtime was later improved by [CKS15] to depend logarithmically on $1/\epsilon$. (Originally, [HHL09] considered a different goal — of outputting a "classical" value, a linear combination of $|x\rangle$ — for which the later improvement in dependence on $1/\epsilon$ is impossible unless $BQP = PP$.) These quantum sublinear-time algorithm naturally raise the question of what are the best classical analogues of these quantum algorithms.

1.1.1 Results

Laplacian systems. Before providing our main result that solves SDD systems (in Theorem 1.1.4), we present a simpler result that conveys our key ideas and solves linear systems in Laplacians of expander graphs (in Theorem 1.1.1). We start with setting up the notation. The Laplacian of an (edge-weighted) undirected graph $G = (V, E)$, is defined as the $|V| \times |V|$ matrix $L_G \stackrel{\text{def}}{=} D - A$, where A is the (weighted) adjacency matrix of G, and D is the diagonal matrix of (weighted) degrees in G. As Laplacians have a non-trivial kernel, namely, the span of the all-ones vector (for connected graphs), a more natural goal is to compute $x_u - x_v$, which has a unique value over all the solutions $\{x : L_G x = b\}$.

Theorem 1.1.1. *There exists a randomized algorithm with the following guarantee. Suppose the input is $\langle G, b, u, v, \epsilon, \mu_2 \rangle$, where*

- *$G = (V, E)$ is a connected d-regular n-vertex graph given as an adjacency list,*
- *$b \in \mathbb{R}^n$ is in the range of L_G (equivalently, orthogonal to the all-ones vector),*
- *$u, v \in [n]$, $\epsilon > 0$, and*

- $\mu_2 > 0$ *is a lower bound on the smallest non-zero eigenvalue of* L_G,

and suppose x *solves* $L_G x = b$. *Then the algorithm outputs* $\hat{\delta}_{u,v} \in \mathbb{R}$ *that satisfies*

$$\Pr\left[\left|\hat{\delta}_{u,v} - (x_u - x_v)\right| \leq \epsilon \cdot \max_{ij \in E} |x_i - x_j|\right] \geq 1 - \frac{1}{s}$$

for $s = O(\frac{d}{\mu_2} \log(\epsilon^{-1} \frac{d}{\mu_2} n))$. *The algorithm runs in time* $O(d\epsilon^{-2} s^3 \log s)$.

Let us point out a few extensions of this theorem that follow easily from our proof in Section 1.2. First, it works even when G has integer edge weights by using weighted degrees. Second, if the algorithm is given also an upper bound B_{up} on $||b||_0 \leq n$, then the expression for s can be refined by replacing n with B_{up}. Third, the algorithm can approximate x_u (rather than $x_u - x_v$) by reporting an estimate $\hat{x}_u$ such that

$$\Pr\left[|\hat{x}_u - x^*_u| \leq \epsilon ||x^*||_\infty\right] \geq 1 - \frac{1}{s},$$

where $x^* = L_G^+ b$ is a fixed solution to $L_G x = b$. Here and throughout, A^+ denotes the Moore-Penrose pseudo-inverse of A.[1] Fourth, the runtime bound can be replaced by $O(f\epsilon^{-2} s^3 \log s)$ whenever the representation of G allows to sample, in $O(f)$ time, a uniformly random neighbor of a vertex.

Whenever $\frac{\mu_2}{d}$ is bounded away from 0, Theorem 1.1.1 achieves polylogarithmic (in n) runtime. However, in graphs with a small spectral gap our runtime bound becomes polynomial, even for fixed d and ϵ. For example, the n-cycle requires plugging $\mu_2 \leq O(1/n^2)$, and then our runtime bound evaluates to $O(s^3 \log s)$ for $s = O(n^2 \log(n^2 \cdot n)) = O(n^2 \log n)$. Our next result shows that polynomial runtime is inevitable; more precisely, the worst-case runtime must be linear in n, even if the graph is sparse and fixed in advance. Its proof appears in Section 1.3.

Theorem 1.1.2. *For infinitely many integers* $n \geq 1$, *there is a (strongly explicit) connected non-bipartite 3-regular graph* $G = (V, E)$ *with* n *vertices, such that the following holds for every randomized algorithm. If on every input* $u, v \in V$ *and* $b \in \{e_z - e_w | z \neq w \in V\}$, *the algorithm's output* $\hat{\delta}_{u,v} \in \mathbb{R}$ *satisfies, for some* x *that solves* $L_G x = b$,

$$\Pr\left[|\hat{\delta}_{u,v} - (x_u - x_v)| \leq 0.49 \cdot \max_{ij \in E} |x_i - x_j|\right] > \frac{3}{4},$$

then the algorithm's worst-case time is $\Omega(n)$. *This holds even if* $u, v \in V$ *are fixed in advance and* G *can be preprocessed arbitrarily.*

SDD systems. Our next result generalizes Theorem 1.1.1 from a Laplacian L_G to an SDD matrix S. When b is in the range of S, or equivalently, orthogonal to the kernel of S, a natural solution to the system $Sx = b$ is $x = S^+ b$. But since our method requires the S to have normalized diagonal entries, we will aim at another solution x^*, construed as follows. Using (only) S to define

$$D \stackrel{\text{def}}{=} \text{diag}(S_{11}, ..., S_{nn}) \quad \text{and} \quad \tilde{S} \stackrel{\text{def}}{=} D^{-1/2} S D^{-1/2} \tag{1.1}$$

[1] For a matrix $A \in \mathbb{R}^{n \times n}$, let its Singular Value Decomposition (SVD) be $A = U \Sigma V^T$ where Σ is a diagonal matrix of the positive singular values, then the Moore-Penrose pseudo-inverse of A is $A^+ = V \Sigma^{-1} U^T$.

our linear system can be written as $\tilde{S}(D^{1/2}x) = D^{-1/2}b$, and leads to a solution x^* written using the pseudo-inverse of $\tilde{S}$ (rather than of S).

Fact 1.1.3. *Let $Sx = b$ be a feasible SDD linear system, and let D and $\tilde{S}$ are as in (1.1). Then*

$$x^* \stackrel{\text{def}}{=} D^{-1/2}\tilde{S}^+D^{-1/2}b$$

always solves the system.

Proof. $D^{-1/2}$ is invertible, thus $b \in \text{range}(S) = \text{range}(SD^{-1/2})$ and $D^{-1/2}b \in \text{range}(D^{-1/2}SD^{-1/2}) = \text{range}(\tilde{S})$. It follows that $\tilde{S}\tilde{S}^+D^{-1/2}b = D^{-1/2}b$ and thus

$$Sx^* = (D^{1/2}D^{-1/2})S(D^{-1/2}\tilde{S}^+D^{-1/2}b) = D^{1/2}\tilde{S}\tilde{S}^+D^{-1/2}b = D^{1/2}D^{-1/2}b = b. \qquad \square$$

Theorem 1.1.4. *There exists a randomized algorithm with the following guarantee. Suppose the input is $\left\langle S, b, u, \epsilon, \tilde{\lambda}_{up} \right\rangle$, where*

- *$S \in \mathbb{R}^{n\times n}$ is an SDD matrix,*
- *$b \in \mathbb{R}^n$ is in the range of S (equivalently, orthogonal to the kernel of S),*
- *$u \in [n]$, $\epsilon > 0$, and*
- *$\tilde{\lambda}_{up} < 1$ is an upper bound on $\max\{\frac{\tilde{\lambda}_i+1}{2} : i \in [n], \tilde{\lambda}_i < 1\}$ where $1 \geq \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n \geq -1$ are the eigenvalues of $D^{-1/2}(D-S)D^{-1/2} = I - \tilde{S}$,*

and let x^* be the solution for $Sx = b$ given in Fact 1.1.3. Then the algorithm outputs $\hat{x}_u \in \mathbb{R}$ that satisfies

$$\Pr\left[|\hat{x}_u - x^*_u| \leq \epsilon||x^*||_\infty\right] \geq 1 - \frac{1}{s}$$

for $s = O(\log_{1/\tilde{\lambda}_{up}}(\epsilon^{-1}(1-\tilde{\lambda}_{up})^{-1}n\frac{\max_{i\in[n]}D_{ii}}{\min_{i\in[n]}D_{ii}}))$. The algorithm runs in time $O(f\epsilon^{-2}s^3\log s)$, where f is the time to make a step in a random walk in the weighted graph formed by the non-zeros of S.

We prove these results in Section 1.4. Similarly to Theorem 1.1.1, if the algorithm is given an upper bound B_{up} on $||b||_0 \leq n$, then the expression for s can be refined by replacing n with B_{up}. We point out that Theorem 1.1.4 makes no assumptions about the multiplicity of the eigenvalue 1 of $D^{-1/2}(D-S)D^{-1/2}$, e.g., when S is a graph Laplacian L_G, the graph need not be connected. The only assumption needed to achieve a polylogarithmic running time (assuming f is not the dominant term) is that all eigenvalues strictly smaller than 1 are actually bounded away from 1 (and that $\frac{\max_{i\in[n]}D_{ii}}{\min_{i\in[n]}D_{ii}} \leq \text{poly}(n)$, which is trivial if S has polynomial entries).

Local Algorithm. Our algorithms in Theorems 1.1.1 and 1.1.4 are *local* in the sense that they query a small portion of their input, usually around the input vertex, when viewed as graph algorithms. Local algorithms for graph problems were studied in several contexts, like graph partitioning [ST04, AP09], Web analysis [CGS04, ABC$^+$08], and distributed computing [Suo13]. Rubinfeld, Tamir, Vardi, and Xie [RTVX11] introduced a formal concept of *Local Computation Algorithms* that requires consistency between the local outputs of multiple executions (namely, these local outputs must all agree with a single global solution). As explained earlier, out algorithms satisfy this consistency requirement.

Applications. An example application of our results is computing quickly the PageRank (defined in [BP98]) of a single node in a d-regular undirected graph. Recall that the PageRank vector of an n-vertex graph with associated transition matrix P is the solution to the linear system $x = \frac{1-\alpha}{n}\vec{1}+\alpha Px$, where $0 < \alpha < 1$ is a given parameter and $\vec{1} \in \mathbb{R}^n$ denotes the all-ones vector. Equivalently, x solves the system $Sx = \frac{1-\alpha}{n}\vec{1}$ where $S = I - \alpha P$ is an SDD matrix with 1's on the diagonal. As all eigenvalues of P are of magnitude at most 1 (recall P is a transition matrix), all eigenvalues of $I - \tilde{S} = I - S = \alpha P$ are of magnitude at most α, and the runtime guaranteed by Theorem 1.1.4 is logarithmic (with base $\frac{2}{\alpha+1}$).

Another example application is computing the effective resistance between a pair of vertices u, v in a graph G (given u,v and G as input). It is well known that the effective resistance, denoted $\mathrm{R}_{\mathrm{eff}}(u, v)$, is given by $x_u - x_v$ where x solves $L_G x = e_u - e_v$. For example, the spectral-sparsification algorithm of Spielman and Srivastava [SS11] relies on a near-linear time algorithm (that they devise) for approximating the effective resistances of all edges in G. We can approximate the effective resistance between a pair of vertices in a constant-degree expander G by applying Theorem 1.1.1. Observe that we can use $B_{up} = 2$, hence the runtime is $O(\frac{1}{\epsilon^2}\operatorname{polylog}\frac{1}{\epsilon})$, independently of n. The additive accuracy is $\epsilon \cdot \max_{ij\in E(G)} |x_i - x_j|$; in fact, each $x_i - x_j$ is the potential difference between i and j when a potential difference of $\mathrm{R}_{\mathrm{eff}}(u, v)$ is imposed between u and v, thus $\max_{ij\in E(G)} |x_i - x_j| \le x_u - x_v = \mathrm{R}_{\mathrm{eff}}(u, v)$, and the output $\hat{\mathrm{R}}_{\mathrm{eff}}(u, v)$ actually achieves a multiplicative guarantee $\Pr[\hat{\mathrm{R}}_{\mathrm{eff}}(u, v) \in (1 \pm \epsilon)\,\mathrm{R}_{\mathrm{eff}}(u, v)] \ge 1 - \frac{1}{s}$.

1.1.2 Techniques and Related Work

Our basic technique (manifested in Theorem 1.1.1) relies on the formula

$$\forall X \in \mathbb{R}^{n\times n}, \|X\| < 1, \qquad (I - X)^{-1} = \sum_{t=0}^{\infty} X^t,$$

where throughout, $||X||$ denotes the spectral norm of a matrix X. It works as follows. Given a Laplacian $L = L_G$ of a d-regular graph G, observe that $\frac{1}{d}L = I - \frac{1}{d}A$ where A is the adjacency matrix of G. Assume for a moment that $||\frac{1}{d}A|| < 1$, then by the above formula $(\frac{1}{d}L)^{-1} = (I - \frac{1}{d}A)^{-1} = \sum_{t=0}^{\infty}(\frac{1}{d}A)^t$, and the solution of the linear system would be $x = L^{-1}b = \frac{1}{d}\sum_{t=0}^{\infty}(\frac{1}{d}A)^t b$. The point is that the summands decay exponentially because $||(\frac{1}{d}A)^t b||_2 \le ||(\frac{1}{d}A)^t|| \cdot ||b||_2 \le ||(\frac{1}{d}A)||^t \cdot ||b||_2$. Therefore, we can estimate x_u using the first t_0 terms, i.e., $\hat{x}_u = e_u^T \frac{1}{d}\sum_{t=0}^{t_0}(\frac{1}{d}A)^t b$ where t_0 is logarithmic (with base $\|\frac{1}{d}A\|^{-1} > 1$). In order to compute each term $e_u^T \frac{1}{d}(\frac{1}{d}A)^t b$, observe that $e_u^T(\frac{1}{d}A)^t e_w$ is exactly the probability that a random walk of length t starting at u will end at vertex w. Thus, if we perform a random walk of length t starting at u, and we denote by z the (random) vertex at which the walk ends, then

$$\mathbb{E}[b_z] = \sum_{w\in V} e_u^T(\frac{1}{d}A)^t e_w b_w = e_u^T(\frac{1}{d}A)^t b.$$

If we perform several random walks (specifically, $\operatorname{poly}(t_0, \frac{1}{\epsilon})$ many suffices), average the resulting b_z's, and then multiply by $\frac{1}{d}$, then with high probability, we will obtain a good approximation to $e_u^T \frac{1}{d}(\frac{1}{d}A)^t b$.

Unfortunately, it is not the case that $||\frac{1}{d}A|| < 1$, as $\frac{1}{d}A\vec{1} = \vec{1}$. Nevertheless, we can still get a meaningful result if all eigenvalues of A except for the largest one are smaller than d (equivalently, the graph G is connected). First, we get rid of any negative eigenvalues by the standard trick of considering $(dI + A)/2$ instead of A, which is equivalent to adding d self-loops at every vertex. Now assuming that A is PSD and b is orthogonal to $\vec{1}$ (otherwise the linear system has no solution), the linear system $Lx = b$ has infinitely many solutions, and we can aim to estimate the specific solution $x^* \stackrel{\text{def}}{=} L^+b$ by $\frac{1}{d}\sum_{t=0}^{t_0}(\frac{1}{d}A)^t b$. Indeed, the idealized analysis above still applies by restricting all our calculations to the subspace that is orthogonal to the all-ones vector.

The idea of approximating the inverse $(I - X)^{-1} = \sum_{t=0}^{\infty} X^t$ (for $||X|| < 1$) by random walks dates back to von Neumann and Ulam [FL50, Was52]. While we approximate each power X^t by separate random walks of length t and truncate the tail (powers above some t_0), their method employs random walks whose length is random, and their expectation gives exactly the infinite sum. This is done by assigning some probability to terminate the walk at each step, and weighting the value of the walk accordingly (to correct the expectation).

The idea of approximating a generalized inverse L^* of $L = dI - A$ by $\frac{1}{d}\sum_{t=0}^{t_0}(\frac{1}{d}A)^t$ on directions that are orthogonal to the all-ones vector was recently used by Doron, Le Gall, and Ta-Shma [DGT17] to show that L^* can be approximated in probabilistic log-space. However, since they wanted to output L^* explicitly, they could not ignore the all-ones direction and they needed to relate L^* to $\frac{1}{d}\sum_{t=0}^{\infty}(\frac{1}{d}A)^t$ by "peeling off" the all-ones direction, inverting using the infinite sum formula, and then adding back the all-ones direction.

The idea of estimating powers of a normalized adjacency matrix $\frac{1}{d}A$ (or more generally, a stochastic matrix) by performing random walks is well known, and was used also in [DSTS17, DGT17] mentioned above. Chung and Simpson [CS15] used it in a context that is related to ours, of solving a Laplacian system $L_G x = b$ but with a boundary condition, namely, a constraint that $x_i = b_i$ for all i in the support of b. Their algorithm solves for a subset of the coordinates $W \subseteq V$, i.e., their algorithm approximates $x|_W$ (the restriction of x to coordinates in W) where x solves $Lx = b$ under the boundary condition. They relate the solution x to the Dirichlet heat-kernel PageRank vector, which in turn is related to an infinite power series of a transition matrix (specifically, to $f^T e^{-t(I-P_W)} = e^{-t} f^T \sum_{k=0}^{\infty} \frac{t^k}{k!} P_W^k$ where P_W is the transition matrix of the graph induced by W, $t \in \mathbb{R}$, and $f \in \mathbb{R}^{|W|}$), and their algorithm uses random walks to approximate the not-too-large powers of the transition matrix, proving that the remainder of the infinite sum is small enough.

Recently, Shyamkumar, Banerjee and Lofgren [SBL16] considered a related matrix-power problem, where the input is a matrix $A \in \mathbb{R}^{n\times n}$, a power $\ell \in \mathbb{N}$, a vector $z \in \mathbb{R}^n$, and an index $t \in [n]$, and the goal is to compute coordinate t of $A^\ell z$. They devised for this problem a sublinear (in $\text{nnz}(A)$) algorithm, under some bounded-norm conditions and assuming $t \in [n]$ is uniformly random. Their algorithm relies, in part, on von Neumann and Ulam's technique of computing matrix powers using random walks, but of prescribed length. It can be shown that approximately solving positive definite systems for a particular coordinate is reducible to the matrix-power problem.[2] However, in contrast

[2] Let $Ax = b$ be a linear system where A is positive definite. Let λ be the largest eigenvalue of A. Let $A' \stackrel{\text{def}}{=} \frac{1}{2\lambda}A$ and $b' \stackrel{\text{def}}{=} \frac{1}{2\lambda}b$. Consider the equivalent system $(I-(I-A'))x = b'$. As the eigenvalues of A' are in $(0, 1/2]$, the eigenvalues of $I - A'$ are in $[1/2, 1)$. Thus, the solution to the linear system is given by $x = (I-(I-A'))^{-1}b' = \sum_{t=0}^{\infty}(I-A')^t b$. Therefore, we can approximate x_u by truncating the infinite sum at some t_0 and approximating each power $t < t_0$ by the algorithm for the matrix-power problem.

to our results, their expected runtime is polynomial in the input size (specifically $\text{nnz}(A)^{2/3}$), and holds only for a random $t \in [n]$.

To generalize from Laplacians of regular graphs to SDD matrices (as in our Theorem 1.1.4), we face three issues. The first one is the irregularity of general SDD matrices, which we resolve by normalizing the SDD matrix S, and solving the equivalent (normalized) system $\tilde{S}(D^{1/2}x) = D^{-1/2}b$ where $D = \text{diag}(S_{11}, ..., S_{nn})$, and $\tilde{S} = D^{-1/2}SD^{-1/2}$. Second, general SDD matrices can have positive off-diagonal elements, and thus we employ a random walk that "remembers" the signs it has seen. Third, diagonal elements may strictly dominate their row, and we address this by terminating the random walk early with some probability.

1.2 A d-regular Laplacian Solver

In this section we prove Theorem 1.1.1. Let $G = (V = [n], E)$ be a connected d-regular graph with adjacency matrix $A \in \mathbb{R}^{n \times n}$. Let the eigenvalues of A be $d = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n$, and let their associated orthonormal eigenvectors be $u_1, \ldots, u_n$. Then $u_1 = \frac{1}{\sqrt{n}} \cdot \vec{1} \in \mathbb{R}^n$, and we can write $A = U\Lambda U^T$ where $U = [u_1\, u_2 \ldots u_n]$ is unitary and $\Lambda = \text{diag}(\lambda_1, ..., \lambda_n)$. For $u, v \in [n]$, let $\chi_{u,v} \stackrel{\text{def}}{=} e_u - e_v$ where e_i is the i-th standard basis vector. Then the Laplacian of G is given by

$$L \stackrel{\text{def}}{=} \sum_{uv \in E} \chi_{u,v}\chi_{u,v}^T = dI - A = U(dI - \Lambda)U^T.$$

Observe that L does not depend on the orientation of each edge uv, and that $\mu_2 \stackrel{\text{def}}{=} d - \lambda_2$ is the smallest non-zero eigenvalue of L. The Moore-Penrose pseudo-inverse of L is

$$L^+ \stackrel{\text{def}}{=} U \cdot \text{diag}(0, (d - \lambda_2)^{-1}, \ldots, (d - \lambda_n)^{-1}) \cdot U^T.$$

We assume henceforth that all eigenvalues of A are non-negative. At the end of the proof, we will remove this assumption (by adding self-loops).

The idea behind the next fact is that $L = d(I - \frac{1}{d}A)$, and $\frac{1}{d}A$ has norm strictly smaller than one when operating on the subspace that is orthogonal to the all-ones vector, and hence, the formula $(I - X)^{-1} = \sum_{t=0}^{\infty} X^t$ for $||X|| < 1$ is applicable for the span of $\{u_2, ..., u_n\}$.

Fact 1.2.1. *For every $x \in \mathbb{R}^n$ that is orthogonal to the all-ones vector, $L^+x = \frac{1}{d}\sum_{t=0}^{\infty}(\frac{1}{d}A)^t x$.*

Proof. It suffices to prove the claim for each of $u_2, \ldots, u_n$ as the fact will then follow by linearity. Fix $i \in \{2, \ldots, n\}$. Then since $|\frac{\lambda_i}{d}| < 1$,

$$\sum_{t=0}^{\infty} \left(\frac{1}{d}A\right)^t u_i = \sum_{t=0}^{\infty} \left(\frac{\lambda_i}{d}\right)^t u_i = \frac{1}{1 - \frac{\lambda_i}{d}} u_i = \frac{d}{d - \lambda_i} u_i = dL^+ u_i. \qquad \square$$

We now describe an algorithm that on input $b \in \mathbb{R}^n$ that is orthogonal to the all-ones vector, and two vertices $u \neq v \in [n]$, returns an approximation $\hat{\delta}_{u,v}$ to $x_u - x_v$, where x solves $Lx = b$. As G is connected, the null space of L is equal to span$\{\vec{1}\}$ and hence $x_u - x_v$ is uniquely defined, and can be written as $x_u - x_v = \chi_{u,v}^T L^+ b$.

Algorithm 1.1 $\hat{\delta}_{u,v} = \text{SolveLinearLaplacian}(G, b, ||b||_0, u, v, \epsilon, d, \mu_2)$

1. Set
$$s = \frac{\log(2\sqrt{2}\epsilon^{-1}\frac{d}{\mu_2}\sqrt{||b||_0})}{\log(\frac{d}{d-\mu_2})},$$
and $\ell = O((\frac{\epsilon}{4s})^{-2}\log s)$.
2. For $t = 0, 1, \ldots, s-1$ do
   (a) Perform ℓ independent random walks of length t starting at u, and let $u_1^{(t)}, \ldots, u_\ell^{(t)}$ be the vertices at which the random walks ended. Independently, perform ℓ independent random walks of length t starting at v, and let $v_1^{(t)}, \ldots, v_\ell^{(t)}$ be the vertices at which the random walks ended.
   (b) Set $\hat{\delta}_{u,v}^{(t)} = \frac{1}{\ell}\sum_{i\in[\ell]}(b_{u_i^{(t)}} - b_{v_i^{(t)}})$.
3. Return $\hat{\delta}_{u,v} = \frac{1}{d}\sum_{t=0}^{s-1}\hat{\delta}_{u,v}^{(t)}$.

Claim 1.2.2. *For b that is orthogonal to the all-ones vector, $|\chi_{u,v}^T L^+ b - \chi_{u,v}^T \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b| \leq \frac{\epsilon}{2d}||b||_\infty$.*

Proof. Using Fact 1.2.1,

$$\chi_{u,v}^T L^+ b - \chi_{u,v}^T \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b = \chi_{u,v}^T \frac{1}{d}\sum_{t=s}^{\infty}(\frac{1}{d}A)^t b,$$

and thus

$$|\chi_{u,v}^T L^+ b - \chi_{u,v}^T \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b| \leq ||\chi_{u,v}^T||_2 \cdot ||\frac{1}{d}\sum_{t=s}^{\infty}(\frac{1}{d}A)^t b||_2.$$

We know that $||\chi_{u,v}^T||_2 = \sqrt{2}$, so it remains to bound $||\frac{1}{d}\sum_{t=s}^{\infty}(\frac{1}{d}A)^t b||_2$. Decomposing $b = \sum_{i=2}^n c_i u_i$ we get that $\sum_{i=2}^n c_i^2 = ||b||_2^2$ and

$$\sum_{t=s}^{\infty}(\frac{1}{d}A)^t b = \sum_{i=2}^n c_i u_i \sum_{t=s}^{\infty}(\frac{\lambda_i}{d})^t = \sum_{i=2}^n \frac{(\frac{\lambda_i}{d})^s}{1-\frac{\lambda_i}{d}} c_i u_i = d\sum_{i=2}^n \frac{(\frac{\lambda_i}{d})^s}{d-\lambda_i} c_i u_i.$$

Hence,

$$||\frac{1}{d}\sum_{t=s}^{\infty}(\frac{1}{d}A)^t b||_2^2 = \sum_{i=2}^n \left(\frac{(\frac{\lambda_i}{d})^s}{d-\lambda_i}\right)^2 c_i^2||u_i||_2^2 \leq \left(\frac{(\frac{\lambda_2}{d})^s}{d-\lambda_2}\right)^2 \sum_{i=2}^n c_i^2 = \left(\frac{(1-\frac{\mu_2}{d})^s}{\mu_2}\right)^2 ||b||_2^2,$$

where the first equality is because the u_i's are orthogonal. Altogether,

$$|\chi_{u,v}^T L^+ b - \chi_{u,v}^T \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b| \leq \sqrt{2}\frac{(1-\frac{\mu_2}{d})^s}{\mu_2}||b||_2 \leq \sqrt{2}\frac{(1-\frac{\mu_2}{d})^s}{\mu_2}\sqrt{||b||_0}\cdot||b||_\infty = \frac{\epsilon}{2d}||b||_\infty,$$

as claimed. □

Claim 1.2.3. $\Pr\left[|\hat{\delta}_{u,v} - \chi_{u,v}^T \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b| > \frac{\epsilon}{2d}||b||_\infty\right] \leq \frac{1}{s}$.

Proof. Observe that $e_u^T(\frac{1}{d}A)^t$ is a probability vector over V, and $e_u^T(\frac{1}{d}A)^t e_w$ is exactly the probability that a random walk of length t starting at u will end at w. Thus, for every $t \in \{0,1,\ldots,s-1\}$ and $i \in [\ell]$, we have

$$\mathbb{E}[b_{u_i^{(t)}}] = \sum_{w\in[n]} e_u^T(\frac{1}{d}A)^t e_w b_w = e_u^T(\frac{1}{d}A)^t b,$$

and similarly $\mathbb{E}[b_{v_i^{(t)}}] = e_v^T(\frac{1}{d}A)^t b$. By a union bound over Hoeffding bounds, with probability at least $1-\frac{1}{s}$, for every $t \in \{0,1,\ldots,s-1\}$, we have $|\frac{1}{\ell}\sum_{i\in[\ell]} b_{u_i^{(t)}} - e_u^T(\frac{1}{d}A)^t b| \leq \frac{\epsilon}{4s}||b||_\infty$ and $|\frac{1}{\ell}\sum_{i\in[\ell]} b_{v_i^{(t)}} - e_v^T(\frac{1}{d}A)^t b| \leq \frac{\epsilon}{4s}||b||_\infty$. Recalling that $\hat{\delta}_{u,v} = \frac{1}{d}\sum_{t=0}^{s-1}\frac{1}{\ell}\sum_{i\in[\ell]}(b_{u_i^{(t)}} - b_{v_i^{(t)}})$, with probability at least $1-\frac{1}{s}$ we have $|\hat{\delta}_{u,v} - \chi_{u,v}^T \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b| \leq \frac{\epsilon}{2d}||b||_\infty$, as claimed. □

Combining Claim 1.2.2 and Claim 1.2.3 we get that (with probability $1-\frac{1}{s}$) $|\hat{\delta}_{u,v} - \chi_{u,v}^T L^+ b| \leq \frac{\epsilon}{d}||b||_\infty$. Now, as x solves $Lx = b$, for every $i \in [n]$ we have $\sum_{j\in N(i)}(x_i - x_j) = b_i$ where $N(i)$ is the set of neighbors $\{\, j : ij \in E \,\}$, which implies that for some neighbor j of i, it holds that $|x_i - x_j| \geq \frac{|b_i|}{d}$. Therefore, $\max_{ij\in E}|x_i - x_j| \geq \frac{1}{d}||b||_\infty$. We conclude that $|\hat{\delta}_{u,v} - \chi_{u,v}L^+ b| \leq \epsilon \cdot \max_{ij\in E}|x_i - x_j|$. We now turn to the runtime of Algorithm 1.1, which is dominated by the time it takes to perform the random walks. There are $2s\cdot\ell$ random walks in total. The random walks do not need to be independent for different values of t (as we applied a union bound over the different t), we can extend, at each iteration t, the 2ℓ respective random walks constructed at iteration $t-1$ by an extra step in time $O(d)$ (recall we assume G is given as an adjacency list), obtaining a total runtime $O(s\cdot\ell\cdot d) = O(d\epsilon^{-2}s^3\log s)$. To simplify the expression for s, we need the following bound.

Fact 1.2.4. *For all $\delta \in (0,1)$, $\frac{1}{\ln(1-\delta)^{-1}} \leq \frac{1}{\delta}$.*

Proof. We need to show that $\delta \leq \ln(1-\delta)^{-1}$, or equivalently, $e^{-\delta} \geq 1-\delta$, which is well known. □

Applying Fact 1.2.4 to $\delta = \frac{\mu_2}{d}$, we have $s \leq \frac{d}{\mu_2}\log(2\sqrt{2}\epsilon^{-1}\frac{d}{\mu_2}\sqrt{||b||_0})$, and conclude the following.

Theorem 1.2.5. *Given an adjacency list of a d-regular n-vertex graph G, a vector $b \in \mathbb{R}^n$ that is orthogonal to the all-ones vector, $||b||_0$, $u,v \in [n]$, $\epsilon > 0$, and $\mu_2 = d - \lambda_2$, with probability at least $1-\frac{1}{s}$, Algorithm 1.1 outputs a value $\hat{\delta}_{u,v} \in \mathbb{R}$ such that $|\hat{\delta}_{u,v} - \chi_{u,v}^T L^+ b| \leq \epsilon \cdot \max_{ij\in E}|x_i - x_j|$. Algorithm 1.1 runs in time $O(d\epsilon^{-2}s^3\log s)$ for $s = O(\frac{d}{\mu_2}\log(\epsilon^{-1}\frac{d}{\mu_2}||b||_0))$.*

It remains to show how to remove the assumption that A has no negative eigenvalues. Given an adjacency matrix A which might have negative eigenvalues, consider the PSD matrix $A' = A + dI$, which is the adjacency matrix of the $2d$-regular graph G' obtained from G by adding d self-loops to each vertex. Observe that $A' = U(\Lambda + dI)U^T$ and we can write $L = dI - A = (2dI - A')$, and thus, similarly to Fact 1.2.1, $L^+x = \frac{1}{2d}\sum_{t=0}^{\infty}(\frac{1}{2d}A')^t$, for $x \in \mathbb{R}^n$ that is orthogonal to the all-ones vector. Therefore, if we use A' (which is PSD) to guide Algorithm 1.1's random walks (i.e., at each step of a walk, with probability 1/2 the walk stays put and with probability 1/2 it moves to a uniform neighbor in G) and apply Claims 1.2.2 and 1.2.3 (which apply even when A has self-loops),

an estimate $\hat{\delta}_{u,v}$ satisfying with high probability $|\frac{1}{2}\hat{\delta}_{u,v} - \chi_{u,v}^T L^+ b| \leq \epsilon \max_{ij\in E} |x_i - x_j|$ is obtained. When running Algorithm 1.1 on G', the term s evaluates to $O(\frac{2d}{2d-(\lambda_2+d)} \log(\epsilon^{-1} \frac{2d}{2d-(\lambda_2+d)} ||b||_0)) = O(\frac{d}{\mu_2} \log(\epsilon^{-1} \frac{d}{\mu_2} ||b||_0))$, thus, leaving the guarantee of Theorem 1.2.5 intact (up to constant factors).

Theorem 1.1.1 follows by noting that the analysis in Claim 1.2.2 and Claim 1.2.3 holds also when replacing $||b||_0$ by an upper bound $B_{up} \geq ||b||_0$ and μ_2 by a lower bound on μ_2.

1.3 A Lower Bound for Graphs with Poor Expansion

In this section we prove Theorem 1.1.2. In light of Theorem 1.2.5, a natural question is whether the runtime dependence on $\frac{1}{\mu_2}$ is necessary. For graphs with poor expansion, like the n-cycle, the runtime bound in Theorem 1.2.5 becomes polynomial, even for fixed d and ϵ. We show that for general d-regular graphs, polynomial time is inevitable; more precisely, we present a d-regular graph G (with poor expansion, of course) for which approximating $x_u - x_v$ requires $\Omega(n)$ time. Let $W_1 \stackrel{\text{def}}{=} \{e_z - e_w \in \mathbb{R}^n : z \neq w \in [\frac{n}{2}]\}$, and let $W_2 \stackrel{\text{def}}{=} \{e_z - e_w \in \mathbb{R}^n : z \in [\frac{n}{2}], w \in [n] \setminus [\frac{n}{2}]\}$. Let $\bar{W} \stackrel{\text{def}}{=} W_1 \cup W_2$.

Theorem 1.3.1 (Theorem 1.1.2 restated)**.** *For infinitely many integers $n \geq 1$, there is a (strongly explicit) connected non-bipartite 3-regular graph $G = (V, E)$ with n vertices, such that the following holds for every randomized algorithm. If on every input $u, v \in V$ and $b \in \bar{W}$, the algorithm's output $\hat{\delta}_{u,v} \in \mathbb{R}$ satisfies, for some x that solves $L_G x = b$,*

$$\Pr\left[|\hat{\delta}_{u,v} - (x_u - x_v)| \leq 0.49 \cdot \max_{ij\in E} |x_i - x_j|\right] > \frac{3}{4},$$

then the algorithm's worst-case time is $\Omega(n)$. This holds even if $u, v \in V$ are fixed in advance and G can be preprocessed arbitrarily.

Proof. Consider a 3-regular G which is a disjoint union of two connected graphs G_1 (on $[\frac{n}{2}]$) and G_2 (on $n \setminus [\frac{n}{2}]$) with a single edge uv connecting G_1 and G_2.[3] It is well known that for (unweighted G and) $b = e_z - e_w$, all solutions to $L_G x = b$ are vectors x such that $x_i - x_j$ is the electrical flow from i to j when a unit of current is shipped from z to w (in G). Therefore, for every solution x to $L_G x = b$ where $b = e_z - e_w$, it holds that $\max_{ij\in E} |x_i - x_j| \leq 1$. Consider two possibilities for the vector b. In the first case, $b \in W_1$. Then for every solution x to $L_G x = b$ it holds that $x_u - x_v = 0$ (as $u, z, w \in V(G_1)$, $v \in V(G_2)$, and by summing the flow conservation equation over $V(G_1)$). In the second case, $b \in W_2$. Then for every solution x to $L_G x = b$ it holds that $x_u - x_v = 1$ (again, by summing the flow conservation equation over $V(G_1)$). Therefore, $\epsilon = 0.49$ will suffice to distinguish between the two cases. We now use Yao's minimax principle to show that any randomized algorithm that reads less than $\frac{n}{10}$ coordinates of b, with probability at least $\frac{1}{4}$, fails to distinguish between the two cases (outputting 1 on W_1 and 0 on W_2) , which completes the proof.

Consider the (hard) distribution that assigns probability $\frac{1}{2}\frac{1}{|W_1|}$ to each $b \in W_1$ and $\frac{1}{2}\frac{1}{|W_2|}$ to each $b \in W_2$. Now, let A be any deterministic algorithm that reads less than $\frac{n}{10}$ coordinates of

[3] Such a 3-regular G exists (and is strongly explicit). E.g., (assuming n is even but not divisible by four) take a cycle on $[\frac{n}{2}]$, add a matching on $[\frac{n}{2} - 1]$, and let G_1 be the resulting graph. Notice that all vertices in G_1 have degree three except for $u = \frac{n}{2}$ which has degree two. Do the same for G_2 (replacing u by $v = \frac{n}{2} + 1$) and connect u and v.

b. Let $I \subseteq [n]$ be the coordinates of b that A reads when operating on the all zeros vector $\vec{0}$. Observe that for every $b \in \bar{W}$ such that $b|_I = \vec{0}$, we have $A(b) = A(\vec{0})$. Therefore, as $|I| < \frac{n}{10}$, if $A(\vec{0}) = 0$, then A errs (from the W_1 part) with probability (over the distribution) at least $\frac{1}{2}(1 - \frac{|I|}{n/2})(1 - \frac{|I|}{n/2-1}) \geq \frac{1}{2}(\frac{4}{5})^2 \geq \frac{1}{4}$, and if $A(\vec{0}) = 1$, then A errs (from the W_2 part) with probability (over the distribution) at least $\frac{1}{2}(1 - \frac{|I|}{n/2})^2 \geq \frac{1}{2}(\frac{4}{5})^2 \geq \frac{1}{4}$. □

1.4 An SDD Solver

In this section we prove Theorem 1.1.4, by generalizing the ideas behind Algorithm 1.1 to solving linear systems in SDD matrices. To generalize from Laplacians of regular graphs to SDD matrices, we face several issues, which have been described in section 1.1.2.

We proceed with a formal description of our result for SDD matrices. Let $S \in \mathbb{R}^{n \times n}$ be an SDD matrix (i.e., S is symmetric and $S_{ii} \geq \sum_{j \neq i} |S_{ij}|$ for all $i \in [n]$). We assume that $S_{ii} > 0$ for every i (as otherwise the entire i-th row and column are zero and can be safely ignored). Let $D \stackrel{\text{def}}{=} \text{diag}(S_{11}, \ldots, S_{nn})$, and $A \stackrel{\text{def}}{=} D - S$. Let $\tilde{A} \stackrel{\text{def}}{=} D^{-1/2} A D^{-1/2}$ (for intuition, this is the normalized adjacency matrix when S is a Laplacian) and let $\tilde{S} \stackrel{\text{def}}{=} D^{-1/2} S D^{-1/2} = I - \tilde{A}$ (the normalized Laplacian, respectively). For an eigenvalue μ of $\tilde{A}$, let $E_\mu(\tilde{A}) \stackrel{\text{def}}{=} \{x : \tilde{A}x = \mu x\}$. Observe that $\tilde{A} \preceq I \iff A \preceq D \iff S \succeq 0$; Recalling that S is SDD, we conclude that $\tilde{A} \preceq I$. Moreover,

$$\tilde{A}x = x \iff D^{-1/2}(D - S)D^{-1/2}x = x \iff D^{-1/2} S D^{-1/2} x = 0 \iff S D^{-1/2} x = 0,$$

so $E_1(\tilde{A}) = D^{1/2} \cdot \ker(S)$. Observe that $\tilde{A} \succeq -I \iff A \succeq -D \iff D + A \succeq 0$; Since $D + A$ is SDD, we conclude that $\tilde{A} \succeq -I$. Let $1 \geq \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n \geq -1$ be $\tilde{A}$'s eigenvalues with associated orthonormal eigenvectors $\tilde{u}_1, \ldots, \tilde{u}_n$ (note that $\tilde{A}$ is symmetric). We can write $\tilde{A} = \tilde{U}\tilde{\Lambda}\tilde{U}^T$ where $\tilde{U} = [\tilde{u}_1 \ \tilde{u}_2 \ldots \tilde{u}_n]$ is unitary and $\tilde{\Lambda} = \text{diag}(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n)$. Note that $\tilde{S} = \tilde{U}(I - \tilde{\Lambda})\tilde{U}^T$, and that $\tilde{S}^+ = \tilde{U}(\frac{1}{1-\tilde{\lambda}_1}, \ldots, \frac{1}{1-\tilde{\lambda}_n})\tilde{U}^T$ where by convention $\frac{1}{0}$ stands for 0. Let $d_i \stackrel{\text{def}}{=} D_{ii}$, $d_{max} \stackrel{\text{def}}{=} \max_{i \in [n]} d_i$, and $d_{min} \stackrel{\text{def}}{=} \min_{i \in [n]} d_i$. Let $\tilde{B} \stackrel{\text{def}}{=} \frac{\tilde{A}+I}{2}$. Note that $\tilde{B} = \tilde{U}\frac{\tilde{\Lambda}+I}{2}\tilde{U}^T$, and that the eigenvalues of $\tilde{B}$ are in $[0, 1]$. Let $\tilde{\lambda} \stackrel{\text{def}}{=} \max\{\frac{\tilde{\lambda}_i+1}{2} : i \in [n], \tilde{\lambda}_i < 1\}$ (the largest non-one eigenvalue of $\tilde{B}$). We now describe an algorithm that on input $b \in \mathbb{R}^n$ that is in the range of S (equivalently, is orthogonal to the kernel of S), and $u \in [n]$, returns an approximation $\hat{x}_u$ to x^*_u, where $x^* = D^{-1/2}\tilde{S}^+ D^{-1/2} b$ is the solution for $Sx = b$ given in Fact 1.1.3.

We now prove that Algorithm 1.2 indeed provides a good approximation. Note that b is orthogonal to $\ker(S)$ iff $D^{-1/2}b$ is orthogonal to $D^{1/2} \cdot \ker(S) = E_1(\tilde{A}) = E_1(\tilde{B})$.

Claim 1.4.1. *For b that is orthogonal to the kernel of S,*

$$\left| x^*_u - \frac{1}{2} e_u^T D^{-1/2} \sum_{t=0}^{s-1} \tilde{B}^t D^{-1/2} b \right| \leq \frac{\epsilon}{4} ||D^{-1} b||_\infty. \tag{1.2}$$

Proof. Observe that

$$2\tilde{S}^+ = \left(\frac{I - \tilde{A}}{2}\right)^+ = \left(I - \frac{\tilde{A} + I}{2}\right)^+ = (I - \tilde{B})^+.$$

Algorithm 1.2 $\hat{x}_u = \text{SolveLinearSDD}(S, b, ||b||_0, u, \epsilon, \tilde{\lambda})$

1. Set $s = \log_{1/\tilde{\lambda}}(2\epsilon^{-1}(1-\tilde{\lambda})^{-1}\sqrt{||b||_0} \cdot \sqrt{d_{max}/d_{min}})$, and $\ell = O((\frac{\epsilon}{2s})^{-2}\log s)$.
2. For $t = 0, 1, \ldots, s-1$ do
   (a) Perform ℓ independent (lazy) random walks of length t starting at u, where in one step from vertex v, the walk stays put with probability $\frac{1}{2}$, moves to v' with probability $\frac{|A_{vv'}|}{2d_v}$, and terminates with the remaining probability.
   For each walk $i \in [\ell]$, let $u_i^{(t)}$ be the vertex where the walk ended, and let $\sigma_i^{(t)}$ be the product of the signs along the walk where stay-put steps have sign 1 and others have $\text{sgn}(A_{vv'})$. Formally, if the walk consists of $u = u_0, u_1, .., u_t$ then $\sigma_i^{(t)} = \prod_{j\in[t]} \text{sgn}((D^{-1}A + I)_{u_{j-1},u_j})$ and if it terminated earlier then $\sigma_i^{(t)} = 0$.
   (b) Set $\hat{x}_u^{(t)} = \frac{1}{\ell}\sum_{i\in[\ell]} \sigma_i^{(t)} \frac{b_{u_i^{(t)}}}{d_{u_i^{(t)}}}$.
3. Return $\hat{x}_u = \frac{1}{2}\sum_{t=0}^{s-1} \hat{x}_u^{(t)}$.

Thus, as $D^{-1/2}b$ is in the span of eigenvectors of $\tilde{B}$ with associated eigenvalues in $[0, 1)$, using the same idea as in Fact 1.2.1 we get that

$$\tilde{S}^{+}D^{-1/2}b = \frac{1}{2}\sum_{t=0}^{\infty}\tilde{B}^t D^{-1/2}b,$$

and hence (recall $x_u^* = e_u^T D^{-1/2}\tilde{S}^{+}D^{-1/2}b$)

$$x_u^* - \frac{1}{2}e_u^T D^{-1/2}\sum_{t=0}^{s-1}\tilde{B}^t D^{-1/2}b = \frac{1}{2}e_u^T D^{-1/2}\sum_{t=s}^{\infty}\tilde{B}^t D^{-1/2}b.$$

Similarly to the proof of Claim 1.2.2, we now get that

$$\begin{aligned}
|\frac{1}{2}e_u^T D^{-1/2}\sum_{t=s}^{\infty}\tilde{B}^t D^{-1/2}b| &\leq \frac{1}{2\sqrt{d_u}}||\sum_{t=s}^{\infty}\tilde{B}^t D^{-1/2}b||_2 \\
&\leq \frac{1}{2\sqrt{d_{min}}}\sum_{t=s}^{\infty}\tilde{\lambda}^t||D^{-1/2}b||_2 \\
&\leq \frac{1}{2\sqrt{d_{min}}}\cdot\frac{\tilde{\lambda}^s}{1-\tilde{\lambda}}||D^{-1/2}b||_2 \\
&\leq \frac{1}{2}\sqrt{\frac{d_{max}}{d_{min}}}\cdot\frac{\tilde{\lambda}^s}{1-\tilde{\lambda}}||D^{-1}b||_2 \\
&\leq \frac{1}{2}\sqrt{\frac{d_{max}}{d_{min}}}\cdot\frac{\tilde{\lambda}^s}{1-\tilde{\lambda}}\sqrt{||b||_0}\cdot||D^{-1}b||_\infty = \frac{\epsilon}{4}||D^{-1}b||_\infty.
\end{aligned}$$

□

Claim 1.4.2. *With probability at least* $1 - \frac{1}{s}$,

$$\left|\hat{x}_u - \frac{1}{2} e_u^T D^{-1/2} \sum_{t=0}^{s-1} \tilde{B}^t D^{-1/2} b\right| \leq \frac{\epsilon}{4} ||D^{-1} b||_\infty.$$

Proof. Recalling that $\tilde{A} = D^{-1/2} A D^{-1/2}$, we can write

$$D^{-1/2} \tilde{B}^t D^{-1/2} = D^{-1/2} \Big(\frac{\tilde{A} + I}{2}\Big)^t D^{-1/2} = D^{-1/2} \Big(D^{1/2} \frac{D^{-1} A + I}{2} D^{-1/2}\Big)^t D^{-1/2} = \Big(\frac{D^{-1} A + I}{2}\Big)^t D^{-1}.$$

Hence (by induction), for every $t \in \{0, 1, \ldots, s-1\}$ and $i \in [\ell]$, we have

$$e_u^T D^{-1/2} \tilde{B}^t D^{-1/2} b = e_u^T \Big(\frac{D^{-1} A + I}{2}\Big)^t D^{-1} b = \mathbb{E}[\sigma_i^{(t)} \frac{b_{u_i^{(t)}}}{d_{u_i^{(t)}}}].$$

By a union bound over Hoeffding bounds (as $|\sigma_i^{(t)} \frac{b_{u_i^{(t)}}}{d_{u_i^{(t)}}}| \leq ||D^{-1} b||_\infty$), with probability at least $1 - \frac{1}{s}$, for every $t \in \{0, 1, \ldots, s-1\}$,

$$\Big|\frac{1}{\ell} \sum_{i \in [\ell]} \sigma_i^{(t)} \frac{b_{u_i^{(t)}}}{d_{u_i^{(t)}}} - e_u^T D^{-1/2} \tilde{B}^t D^{-1/2} b\Big| \leq \frac{\epsilon}{2s} ||D^{-1} b||_\infty,$$

which implies that

$$\Big|\frac{1}{2} \sum_{t=0}^{s-1} \hat{x}_u^{(t)} - \frac{1}{2} e_u^T D^{-1/2} \sum_{t=0}^{s-1} \tilde{B}^t D^{-1/2} b\Big| \leq \frac{\epsilon}{4} ||D^{-1} b||_\infty.$$

□

Combining Claim 1.4.1 and Claim 1.4.2 we get that (with probability $1 - \frac{1}{s}$) $|\hat{x}_u - x_u^*| \leq \frac{\epsilon}{2} ||D^{-1} b||_\infty$. Now, letting x denote any solution to the system $Sx = b$, for every $i \in [n]$ we have

$$\frac{|b_i|}{d_i} = \frac{|\sum_{j \in [n]} S_{ij} x_j|}{d_i} \leq \frac{\sum_{j \in [n]} |S_{ij}| \cdot ||x||_\infty}{d_i} \leq \frac{2 d_i ||x||_\infty}{d_i} = 2||x||_\infty$$

where the last inequality is because S is SDD. Therefore, $||D^{-1} b||_\infty \leq 2||x||_\infty$, and we conclude that (with probability $1 - \frac{1}{s}$) $|\hat{x}_u - x_u^*| \leq \epsilon ||x||_\infty$ for every solution x to the system $Sx = b$ (and in particular for x^*). We now turn to the runtime of Algorithm 1.2, which is dominated by the time it takes to perform the random walks. There are $s \cdot \ell$ random walks in total. Let f be the time it takes to make a single step in the random walks of Algorithm 1.2 (it depends on the access method/representation of S and/or its sparsity). The random walks do not need to be independent for different values of t (as we applied a union bound over the different t), we can extend, at each iteration t, the ℓ respective random walks constructed at iteration $t-1$ by an extra step in time f, obtaining a total runtime $O(s \cdot \ell \cdot f) = O(f \epsilon^{-2} s^3 \log s)$. We conclude the following.

Theorem 1.4.3. *Given access to an SDD matrix $S \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ that is orthogonal to the kernel of S, $||b||_0$, $u \in [n]$, $\epsilon > 0$, and $\tilde{\lambda} = \max\{\frac{\tilde{\lambda}_i + 1}{2} : i \in [n], \tilde{\lambda}_i < 1\}$, with probability at least $1 - \frac{1}{s}$, Algorithm 1.2 outputs a value $\hat{x}_u \in \mathbb{R}$ such that $|\hat{x}_u - x^*_u| \leq \epsilon ||x||_\infty$ for every solution x to the system $Sx = b$ (and in particular for x^*). Algorithm 1.2 runs in time $O(f\epsilon^{-2}s^3 \log s)$ where f is the worst-case time to make a step in a random walk in the weighted graph formed by the non-zeros of S, and $s = O(\log_{1/\tilde{\lambda}}(\epsilon^{-1}(1 - \tilde{\lambda})^{-1}||b||_0 \cdot \frac{d_{max}}{d_{min}}))$.*

Theorem 1.1.4 follows by noting that the analysis in Claim 1.4.1 and Claim 1.4.2 holds also when replacing $||b||_0$ by an upper bound $B_{up} \geq ||b||_0$ and $\tilde{\lambda}$ by an upper bound $\tilde{\lambda}_{up} \geq \tilde{\lambda}$.

1.5 Open Questions

We conclude this chapter with a few questions that arise from our work.

- Can the algorithmic result of Section 1.4 be generalized to general PSD matrices? To general (symmetric) matrices? Or, perhaps, can one establish a polynomial/linear-time lower bound for solving linear systems in PSD/symmetric well-conditioned matrices for a particular coordinate? It is worth noting that a recent paper by Kyng and Zhang [KZ17] makes a formal connection between the runtime of approximate solvers (for all coordinates) for certain structured PSD linear systems (that are not SDD) and of approximate solvers for general linear systems.

- Our runtime bounds depend on the eigenvalues of the *normalized* Laplacian $\tilde{L}$ and the *normalized* SDD matrix $\tilde{S}$. Can one get comparable runtime bounds that depend directly on the eigenvalues of the input matrices L and S?

- In Section 1.3 we established an $\Omega(n)$-time lower bound for approximately solving linear systems in ill-conditioned Laplacians for a particular coordinate. Can the lower bound be strengthened to $\Omega(m)$-time? Such a lower bound would show in particular that for ill-conditioned Laplacians, up to logarithmic factors, approximately solving for a particular coordinate takes as much time as approximately solving for all coordinates.

Chapter 2

Sparsification of Graphs and Hypergraphs

2.1 Introduction

Graphs are fundamental objects in computer science. They generally model relationships (edges) between pairs of elements from a ground set (vertices), e.g., the Facebook friendship graph is a pair (V, E), where V is the set of Facebook users, and $uv \in E$ iff u is a Facebook friend of v. Hypergraphs are a generalization of graphs that model higher order relationships, i.e., a relationship (hyperedge) can be any subset of vertices. Going back to the Facebook example, a hyperedge may model a Facebook group. All graphs considered in this chapter are undirected. When referring to *weighted* graphs or hypergraphs in this chapter, we mean that the graph (hypergraph) is augmented by edge-weights (hyperedge-weights) that are non-negative, and by *unweighted* graphs or hypergraphs we mean those with unit edge-weights (hyperedge-weights). Throughout, for a weighted graph (hypergraph) $G = (V, E, w)$, we denote $n = |V|$, and $m = |E|$.

Let us recall some common graph and hypergraph terminology that is used in this chapter. For a weighted graph or hypergraph $G = (V, E, w)$ and two disjoint subsets $A, B \subseteq V$, let

$$w_G(A, B) \stackrel{\text{def}}{=} \sum_{e \in E : e \cap A \neq \emptyset, e \cap B \neq \emptyset} w(e)$$

be the total weight of edges (hyperedges) in G having endpoints in both A and B. A *cut* C in a weighted graph or hypergraph $G = (V, E, w)$ is a proper subset $C \subset V$. Its *weight/capacity/value* is defined as $w_G(C, \bar{C})$, where $\bar{C} \stackrel{\text{def}}{=} V \setminus C$. Given a weighted graph G, its *Laplacian* is defined as the $n \times n$ matrix $L_G \stackrel{\text{def}}{=} D - A$, where D is the diagonal matrix of weighted degrees in G, and A is the weighted adjacency matrix of G. A hypergraph is said to be of *rank* r if all of its hyperedges contain at most r vertices.

Graph cuts found many applications over the years, amongst them in Computer Vision, Computer Graphics, and Machine Learning [Sin04]. Hypergraph cuts found applications in load balancing in parallel computing [CBD+09], and in video object segmentation [HLM09]. A recent paper [YOTI15] showed how to reduce the task of finding the "best" (in some well-defined sense) cyber

attack on an electricity power network to finding the minimum cut in a hypergraph.

Cut sparsification. In 1996, Benczúr and Karger [BK96] introduced the notion of graph *cut-sparsification*: Given a weighted graph $G = (V, E, w)$, a (sparse) weighted graph $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ is said to be a $(1+\epsilon)$-cut-sparsifier of G if

$$\forall C \subset V, \qquad w_G(C, \bar{C}) \leq w_{G_\epsilon}(C, \bar{C}) \leq (1+\epsilon) w_G(C, \bar{C}). \tag{2.1}$$

Note that (2.1) can be equivalently written as

$$\forall x \in \{0,1\}^n, \qquad x^T L_G x \leq x^T L_{G_\epsilon} x \leq (1+\epsilon) x^T L_G x.$$

While the original definition aimed at $(1+\epsilon)$-approximation for $\epsilon \in (0,1)$, the definition can be extended to any approximation factor $k \geq 1$. The definition also extends to hypergraph cut-sparsification. Aside from the combinatorial interest, cut-sparsification turned out to be very useful. As an example application, a $(1+\epsilon)$-approximation of the minimum (st) cut in G can be found by running the best minimum (st) cut algorithm on the sparsifier graph, which improves the running time [BK96].

We note that a different sparsification notion which preserves only the weights of the cuts of weight up to some threshold, was researched as well (see e.g. [CX16]).

Spectral sparsification. In 2004, Spielman and Teng [ST04] introduced the notion of graph *spectral-sparsification*: Given a weighted graph $G = (V, E, w)$, a (sparse) weighted graph $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ is said to be a $(1+\epsilon)$-spectral-sparsifier of G if

$$\forall x \in \mathbb{R}^n, \qquad x^T L_G x \leq x^T L_{G_\epsilon} x \leq (1+\epsilon) x^T L_G x.$$

Note that every spectral-sparsifier is a cut-sparsifier but the converse is not true. Spectral-sparsifiers enabled Spielman and Teng [ST04] to devise nearly linear-time Laplacian systems solvers.

Known sparsification results. Benczúr and Karger [BK96] showed how to construct (with high probability) graph $(1+\epsilon)$-cut-sparsifiers with $O(\epsilon^{-2} n \log n)$ edges. Spielman and Teng [ST04] showed how to construct (with high probability) graph $(1+\epsilon)$-spectral-sparsifiers with $O(\epsilon^{-2} n \log^{O(1)} n)$ edges. A series of improvements by Spielman and Srivastava [SS08], and Batson, Spielman and Srivastava [BSS12] showed how to construct graph $(1+\epsilon)$-spectral-sparsifiers with $O(\epsilon^{-2} n)$ edges. In all these constructions, the sparsifier graph G_ϵ is a reweighted subgraph of G. On the lower bound side, Andoni, Chen, Krauthgamer, Qin, Woodruff and Zhang [ACK+16] showed a lower bound of $\Omega(\epsilon^{-2} n)$ edges for graph $(1+\epsilon)$-cut-sparsification.

Turning to hypergraph cut-sparsification, Newman and Rabinovich [NR13] developed a general function sparsification method which as a corollary yields hypergraph $(1+\epsilon)$-cut-sparsifiers with $O(\epsilon^{-2} n^2)$ edges. Kogan and Krauthgamer [KK15] showed how to construct (with high probability) $(1+\epsilon)$-cut-sparsifiers of rank-r hypergraphs, with $O(\epsilon^{-2} n(r + \log n))$ edges. Their construction is a generalization of that of Benczúr and Karger. We note that the number of hyperedges of the sparsifier in the construction of Kogan and Krauthgamer is not better than that of Newman and

Rabinovich for $r = \Omega(n)$. Recently, Guha, McGregor and Tench [GMT15] gave another Benczúr-Karger flavored algorithm for rank-r hypergraph $(1+\epsilon)$-cut-sparsification, and showed how to implement it in the dynamic hypergraph model. Their construction produces a sparsifier with $\tilde{O}(\epsilon^{-2}nr)$ edges. For hypergraphs, we currently do not know of any lower bound which is stronger than the graph lower bound of $\Omega(\epsilon^{-2}n)$ edges.

In hypergraphs, hyperedges can be potentially very large (contain many vertices). It is natural (definitely when thinking about the representation) to take the sizes of the hyperedges into account when evaluating the sparsity of a hypergraph. To this end, Chekuri and Xu [CX16] (motivated by the aforementioned different notion of cut sparsification they studied) introduced the following notion of *total edge size*: The total edge size of a hypergraph $H = (V, E, w)$ is defined as $\sum_{e \in E} |e|$. As the hyperedges of the sparsifier constructed by Kogan and Krauthgamer [KK15] are potentially of size r, the total edge size of their sparsifier is $O(\epsilon^{-2}nr^2)$ (for $r = \Omega(\log n)$).

Another existing line of research studies graph spectral-sparsification by trees, i.e., the sparsifying graph is required to be a tree. Such sparsifiers are useful in the context of solving linear systems in Laplacian matrices (see e.g. [Spi10, section 5]). Tree sparsifiers are only expected to give large approximation factors. Vaidya (see [BGH$^+$06]) suggested using maximum spanning trees as sparsifiers. It turns out that maximum spanning trees are nm-spectral-sparsifiers [BGH$^+$06]. The best currently known (via low-stretch spanning trees, see [Spi10, section 6] combined with the best low-stretch spanning tree construction of Abraham and Neiman [AN12]) tree spectral-sparsifiers of an arbitrary graph G are of approximation factor $O(m \log n \log\log n)$.

Our contribution. We first examine in Section 2.2 a restricted family of hypergraphs where the intersection of every two distinct hyperedges is of size at most k. For example, the case $k = 1$ already generalizes ordinary graphs. Such restrictions were previously studied in other contexts, see e.g. [RS98, BEGK04, KBEG07], but not in our context of sparsification. We show that for $(1+\epsilon)$-cut-sparsification in such k-bounded-intersections hypergraphs, the upper bound of $O(\epsilon^{-2}n^2)$ edges can be improved to $O(\epsilon^{-2}n^{2-\frac{2}{k+2}})$ edges, and $O(\epsilon^{-2}n^3)$ total edge size improves to $O(\epsilon^{-2}n^{3-\frac{4}{k+2}})$.

We then examine in Section 2.3 the approach of sparsifying large hyperedges by smaller ones. We show that replacing a unit-weight hyperedge of size n by any set of weighted hyperedges of size at most $r < n$, results in a distortion of $\Omega(n/r)$ in the cut values. Previously, this bound was known only for the special case of $r = 2$ (i.e., replacing a hyperedge by ordinary edges of size 2), see e.g. [VH90].

Next, we consider in Sections 2.4 and 2.5 sparsification by trees, i.e., when the sparsifier is required to be a tree. In Section 2.4 we consider sparsifiers that can be arbitrary trees, and show that for every weighted hypergraph H there is a weighted tree T that is an $(n-1)$-cut-sparsifier of H. In Section 2.5 we restrict ourselves to trees that are subgraphs (i.e., spanning trees), and show that every unweighted graph G has a spanning tree that is an $n^{3/2}$-cut-sparsifier of G, and every m-edge weighted graph G has a spanning tree that is an $O(m)$-cut-sparsifier of G. We further show matching lower bounds for cut-sparsification using spanning trees; specifically, there are unweighted graphs G for which every spanning tree can only be a k-cut-sparsifier for $k = \Omega(n^{3/2})$, and there are m-edge weighted graphs G for which every spanning tree can only be a k-cut-sparsifier for $k = \Omega(m)$.

Finally, in Section 2.6 we consider a related question of low *congestion* spanning trees (see definition in Section 2.5). We show that while there are unweighted graphs G for which every

spanning tree of G has $\Omega(n^{3/2})$-congestion, typical graphs in the Erdős-Réni model have spanning trees with $\tilde{O}(n)$-congestion.

2.2 Bounded Intersection Hypergraphs

In this section we consider a restricted family of hypergraphs where the intersection of every two distinct hyperedges is of size at most k. Such restrictions were previously studied in other contexts, see e.g. [RS98, BEGK04, KBEG07], but not in our context of sparsification. We show (in Theorem 2.2.4) that such hypergraphs have $(1+\epsilon)$-cut-sparsifiers with $O(\epsilon^{-2}n^{2-\frac{2}{k+2}})$ hyperedges and $O(\epsilon^{-2}n^{3-\frac{4}{k+2}})$ total edge size— recall the total edge size of a set of hyperedges E, denoted total-size(E), is defined as $\sum_{e\in E}|e|$. This improves upon the known (see Section 2.1) upper bound for general hypergraphs of $O(\epsilon^{-2}n^2)$ hyperedges and $O(\epsilon^{-2}n^3)$ total edge size. For a set of hyperedges E and a continuous range $R\subset\mathbb{R}$, we use E_R to denote the subset $\{\, e\in E : |\, e|\in R\,\}$.

Definition 2.2.1. *Let $H=(V,E,w)$ be a hypergraph. We say that H has k-*bounded-intersections *if $e_1\neq e_2\in E \implies |e_1\cap e_2|\leq k$.*

Note that every graph has 1-bounded-intersections.

Lemma 2.2.2. *Let $H=(V,E,w)$ be a k-bounded-intersections hypergraph. Then for every integer $r\geq k$, we have $|E_{[2r,4r)}|\leq(\frac{n}{r})^{k+1}$, and* total-size$(E_{[2r,4r)})\leq 4\frac{n^{k+1}}{r^k}$.

Proof. As H has k-bounded-intersections, each $S\subseteq V$ of size $k+1$ can be contained in at most one hyperedge in E. On the other hand, each hyperedge in $E_{[2r,4r)}$ contains at least $\binom{2r}{k+1}$ subsets of size $k+1$. Thus,

$$|E_{[2r,4r)}|\leq\frac{\binom{n}{k+1}}{\binom{2r}{k+1}}=\prod_{i=0}^{k}\frac{n-i}{2r-i}\leq\prod_{i=0}^{k}\frac{n}{r}=\left(\frac{n}{r}\right)^{k+1}.$$

The second part of the lemma follows as each hyperedge in $E_{[2r,4r)}$ is of cardinality at most $4r$. □

Corollary 2.2.3. *Let $H=(V,E,w)$ be a k-bounded-intersections hypergraph. Then for every integer $r\geq k$, we have $|E_{[2r,n]}|\leq\ 2\cdot(\frac{n}{r})^{k+1}$, and* total-size$(E_{[2r,n]})\leq 8\cdot\frac{n^{k+1}}{r^k}$.

Proof. Using Lemma 2.2.2 we have

$$|E_{[2r,n]}|=\sum_{t=0}^{\lfloor\log(n/2r)\rfloor}|E_{[2\cdot2^tr,4\cdot2^tr)}|\leq\sum_{t=0}^{\lfloor\log(n/2r)\rfloor}\left(\frac{n}{2^tr}\right)^{k+1}\leq 2\cdot\left(\frac{n}{r}\right)^{k+1}.$$

Similarly,

$$\text{total-size}(E_{[2r,n]})=\sum_{t=0}^{\lfloor\log(n/2r)\rfloor}\text{total-size}(E_{[2\cdot2^tr,4\cdot2^tr)})\leq\sum_{t=0}^{\lfloor\log(n/2r)\rfloor}4\cdot\frac{n^{k+1}}{(2^tr)^k}\leq 8\cdot\frac{n^{k+1}}{r^k}.$$

□

Note that Corollary 2.2.3 implies that every 1-bounded-intersections hypergraph has total edge size at most $O(n^2)$, like ordinary graphs.

Theorem 2.2.4. *Let $H = (V, E, w)$ be a k-bounded-intersections hypergraph. Then there is a $(1+\epsilon)$-cut-sparsifier $H_\epsilon = (V, E_\epsilon, w_\epsilon)$ of H such that $|E_\epsilon| = O(\epsilon^{-2} n^{2-\frac{2}{k+2}})$ and* total-size$(E_\epsilon) = O(\epsilon^{-2} n^{3-\frac{4}{k+2}})$. *Furthermore, H_ϵ can be constructed in polynomial time.*

Proof. We assume that $k < \log n$ as otherwise $n^{\frac{2}{k+2}} = O(1)$ and $n^{\frac{4}{k+2}} = O(1)$, and the theorem follows trivially from known results. Let $r = n^{1-\frac{2}{k+2}} \geq n^{1/3} \geq k$. Let $H_1 = (V, E_{[1,2r)}, w)$ be the hypergraph obtained from H by removing all hyperedges of size at least $2r$, and let $H_2 = (V, E_{[2r,n]}, w)$ be the hypergraph obtained from H by removing all hyperedges of size less than $2r$. Let $H_\epsilon = H_{1_\epsilon} \cup H_2$ where $H_{1_\epsilon} = (V, E_{1_\epsilon}, w_{1_\epsilon})$ is a $(1+\epsilon)$-cut-sparsifier of H_1 constructed by the algorithm of Kogan and Krauthgamer [KK15], which is guaranteed to have (see Section 2.1) $|E_{1_\epsilon}| = O(\epsilon^{-2} nr) = O(\epsilon^{-2} n^{2-\frac{2}{k+2}})$. Clearly H_ϵ is a $(1+\epsilon)$-cut-sparsifier of H. Since all hyperedges in H_1 are of size at most $2r$, we have total-size$(E_{1_\epsilon}) = O(\epsilon^{-2} nr^2) = O(\epsilon^{-2} n^{3-\frac{4}{k+2}})$. By Corollary 2.2.3, $|E_{[2r,n]}| \leq 2 \cdot (\frac{n}{r})^{k+1} = 2n^{2-\frac{2}{k+2}}$ and total-size$(E_{[2r,n]}) \leq 8 \cdot \frac{n^{k+1}}{r^k} = 8 \cdot n^{3-\frac{4}{k+2}}$. The theorem follows. □

2.3 Sparsification by Smaller Hyperedges

Seeking to improve upon the known upper bounds for $(1+\epsilon)$-cut-sparsification of hypergraphs ($O(\epsilon^{-2} n^2)$ edges, and $O(\epsilon^{-2} n^3)$ total edge size, see Section 2.1), we consider larger approximation factors. A simple observation is that if we are willing to settle for approximation factor $2t$ where $t \geq 1$ is an integer, we can construct, for every hypergraph H, a $2t$-cut-sparsifier of H with $O(\frac{n^2}{t})$ edges and $O(\frac{n^3}{t^2})$ total edge size, as follows. Given a hypergraph $H = (V, E, w)$, first replace each hyperedge $e \in E$ by t hyperedges, each of size $\lceil \frac{|e|}{t} \rceil + 1$ and weight identical to the hyperedge they are replacing, which form a connected hypergraph on e (say a sunflower with kernel of size 1), and denote the resulting hypergraph by H'. Observe that all hyperedges in H' are of size at most $O(n/t)$, and H' is a t-cut-sparsifier of H. Now, run Kogan and Krauthgamer's algorithm on H' with say $\epsilon = 0.99$, to obtain a $2t$-cut-sparsifier of H with $O(n \cdot \frac{n}{t})$ edges and $O(n \cdot (\frac{n}{t})^2)$ total edge size (assuming $\frac{n}{t} \geq \Omega(\log n)$, otherwise, the bounds are $O(n \log n)$ and $O(n \log n \cdot \frac{n}{t})$, respectively).

Unfortunately, this tradeoff between the sizes of the hyperedges in the sparsifier and the approximation factor we get is inevitable (in the worst-case). Specifically, as we show next, replacing a unit-weight hyperedge of size n by any set of weighted hyperedges of size at most $r < n$, results in a distortion of $\Omega(n/r)$ in the cut values.

Theorem 2.3.1. *Let $H = (V, \{ V \})$ be a hypergraph with a single unit-weight hyperedge of size n, and suppose that H' is an α-cut-sparsifier of H and H' contains only hyperedges of size $\leq r$. Then $\alpha \geq \Omega(n/r)$.*

The rest of this section is devoted to proving Theorem 2.3.1. Throughout, V denotes a finite ground set of size n, and w.l.o.g. assume $V = [n]$. We use $K_n^{(r)}$ to denote the complete hypergraph $([n], \binom{[n]}{r})$. For a hypergraph H, we use $\beta \cdot H$ to denote the hypergraph obtained from H by multiplying all edge weights in H by $\beta \geq 0$. The plan is as follows. First we will show (in Lemma 2.3.3) that for every $\beta > 0$, sparsifying $H = K_n^{(n)}$ by $\beta \cdot K_n^{(r)}$ results in a distortion of at least $\Omega(n/r)$ in the cut values. Then we will show that among all hypergraphs with hyperedges

of size at most r, the hypergraph $\binom{n-1}{r-1}^{-1} \cdot K_n^{(r)}$ achieves the best approximation factor as a cut-sparsifier of H. We note that the fact that $K_n^{(r)}$ is the best approximator among all hypergraphs with hyperedges of size at most r, has been known for the special case of $r = 2$ (see e.g. [VH90]).

Definition 2.3.2. *Let $H = (V, E, w)$ be a connected hypergraph. Define the best cut-approximation factor that can be achieved by using $K_n^{(r)}$ as a sparsifier of H as*

$$\alpha_r(H) \stackrel{\text{def}}{=} \inf \{ \alpha \geq 1 : \exists \beta > 0 \text{ such that } \beta \cdot K_n^{(r)} \text{ is an } \alpha\text{-cut-sparsifier of } H \}.$$

Lemma 2.3.3. *For every integer $r \geq 2$,*

$$\frac{n}{r} \geq \alpha_r(K_n^{(n)}) = \frac{\binom{n}{r} - \binom{\lfloor \frac{n}{2} \rfloor}{r} - \binom{\lceil \frac{n}{2} \rceil}{r}}{\binom{n-1}{r-1}} \geq \Omega\Big(\frac{n}{r}\Big),$$

where by convention $\binom{a}{b} = 0$ for $b > a$.

Proof. As all cuts in $K_n^{(n)}$ are of equal weight, $\alpha_r(K_n^{(n)})$ is simply the max-cut in $K_n^{(r)}$ divided by the min-cut in $K_n^{(r)}$. Observe that in $K_n^{(r)}$, all cuts with $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$ vertices on one side have the same weight

$$w_{K_n^{(r)}}([k]) = \binom{n}{r} - \binom{k}{r} - \binom{n-k}{r} = \binom{n}{r} - \binom{k-1}{r-1} - \binom{k-1}{r} - \binom{n-k}{r},$$

where the second equality is by Pascal's rule. Similarly,

$$w_{K_n^{(r)}}([k-1]) = \binom{n}{r} - \binom{k-1}{r} - \binom{n-k+1}{r} = \binom{n}{r} - \binom{k-1}{r} - \binom{n-k}{r-1} - \binom{n-k}{r}.$$

Thus, $w_{K_n^{(r)}}([k-1]) \leq w_{K_n^{(r)}}([k])$ as $\binom{n-k}{r-1} \geq \binom{k-1}{r-1}$ for $k \leq \lfloor \frac{n}{2} \rfloor$, and we conclude that

$$w_{K_n^{(r)}}([1]) \leq w_{K_n^{(r)}}([2]) \leq ... \leq w_{K_n^{(r)}}([\lfloor \frac{n}{2} \rfloor]).$$

Therefore,

$$\alpha_r(K_n^{(n)}) = \frac{w_{K_n^{(r)}}([\lfloor \frac{n}{2} \rfloor])}{w_{K_n^{(r)}}([1])} = \frac{\binom{n}{r} - \binom{\lfloor \frac{n}{2} \rfloor}{r} - \binom{\lceil \frac{n}{2} \rceil}{r}}{\binom{n-1}{r-1}}, \tag{2.2}$$

and observe that $\alpha_r(K_n^{(n)}) = \Omega\Big(\frac{\binom{n}{r}}{\binom{n-1}{r-1}}\Big) = \Omega(\frac{n}{r})$ and $\alpha_r(K_n^{(n)}) \leq \frac{\binom{n}{r}}{\binom{n-1}{r-1}} = \frac{n}{r}$. □

We are now ready to prove Theorem 2.3.1.

Proof (of Theorem 2.3.1). As all cuts in H have the same weight, α is at least the max-cut in H' divided by the min-cut in H', and every permutation on the vertices of H' is also an α-cut-sparsifier. Hence, as the average of α-cut-sparsifiers is also an α-cut-sparsifier, by averaging over all permutations, we can assume that there are weights $\beta_2, ..., \beta_r \geq 0$ such that $H' = \cup_{t=2}^{r} \beta_t \cdot K_n^{(t)}$.

Now,

$$\alpha \geq \frac{w_{H'}([\lfloor \frac{n}{2} \rfloor])}{w_{H'}([1])} = \frac{\sum_{t=2}^{r} \beta_t \cdot w_{K_n^{(t)}}([\lfloor \frac{n}{2} \rfloor])}{\sum_{t=2}^{r} \beta_t \cdot w_{K_n^{(t)}}([1])} = \frac{\sum_{t=2}^{r} \beta_t \cdot \alpha_t(K_n^{(n)}) \cdot \binom{n-1}{t-1}}{\sum_{t=2}^{r} \beta_t \cdot \binom{n-1}{t-1}}$$
$$\geq \frac{\sum_{t=2}^{r} \beta_t \cdot \alpha_r(K_n^{(n)}) \cdot \binom{n-1}{t-1}}{\sum_{t=2}^{r} \beta_t \cdot \binom{n-1}{t-1}} = \alpha_r(K_n^{(n)}),$$

where the second equality is by (2.2), and the second inequality is as $\alpha_2(K_n^{(n)}) \geq \alpha_3(K_n^{(n)}) \geq ... \geq \alpha_r(K_n^{(n)})$. Note that one can obtain an approximation factor of $\alpha = \alpha_r(K_n^{(n)})$ by taking $H' = \binom{n-1}{r-1}^{-1} \cdot K_n^{(r)}$ and hence $\beta_2 = \cdots = \beta_{r-1} = 0$ and $\beta_r = \binom{n-1}{r-1}^{-1}$. □

2.4 Sparsification by Trees

In this section we consider sparsification by trees, i.e., sparsifying a hypergraph H by a tree T (that is not restricted to be a subgraph of H). We show (in Theorem 2.4.3) that for every weighted hypergraph H there is a weighted tree T that is an $(n-1)$-cut-sparsifier of H. The sparsifier tree T we use is actually a *cut-equivalent* tree (aka Gomory-Hu tree [GH61], see Definition 2.4.1) of H. Moreover, we show that the $n-1$ term is the best possible, i.e., that there are graphs for which every cut-equivalent tree T can only be a k-cut-sparsifier for $k \geq n-1$. When H is an ordinary graph, we show (in Theorem 2.4.5) that the aforementioned T is also an $(n-1)^3$-spectral-sparsifier. Furthermore, we show that the n^3 term is the best possible up to constant factors, i.e., that there are graphs for which every cut-equivalent tree T can only be a k-spectral-sparsifier for $k = \Omega(n^3)$.

Notation. Let us set up some notation that will be used throughout this section. V denotes a finite ground set of size n, and w.l.o.g. assume $V = [n]$. We use $\bar{S}$ to denote $V \setminus S$. We use $\delta_G(S)$ to denote the set of edges (hyperedges) crossing a cut S in a graph or hypergraph G. Abusing notation, we often omit the set brackets for a single vertex. For a tree $T = (V, E_T)$ and an edge $e = uv \in E_T$ oriented arbitrarily, we let

$$V_{T \downarrow e} \stackrel{\text{def}}{=} \{\, w \in V : w \text{ and } u \text{ are in the same connected component of } T \setminus e \,\}.$$

The orientation is not important since we will use $V_{T\downarrow e}$ as an argument to symmetric set functions, e.g., the weight of the cut $(V_{T\downarrow e}, \bar{V}_{T\downarrow e})$. For two graph G_1 and G_2, we write $G_1 \preceq G_2$ for $L_{G_1} \preceq L_{G_2}$, where $\preceq$ is the Löewner order between PSD matrices. For a graph G, we use $\beta \cdot G$ to denote the graph obtained from G by multiplying all edge weights in G by $\beta > 0$.

Definition 2.4.1 (Gomory and Hu [GH61])**.** *Let V be a finite ground set and let $f : 2^V \to \mathbb{R}$ be a symmetric set function (i.e., $f(S) = f(\bar{S})$ for every $S \subseteq V$). For $s \neq t \in V$, let*

$$\alpha_f(s,t) \stackrel{\text{def}}{=} \min \{\, f(W) : W \subseteq V, \mid W \cap \{\, s, t \,\} \mid = 1 \,\}$$

(for example, the minimum st-cut in an undirected graph on vertex set V). We say that a weighted tree $T = (V, E_T, w_T)$ is a cut-equivalent tree *for (V, f) if*

1. *For all $s \neq t \in V$, the value of the minimum st-cut in T is equal to $\alpha_f(s,t)$; and*

2. *For every edge $e = uv \in E_T$, the connected components of $T \setminus e$ realize $\alpha_f(u,v)$ in the sense that $f(V_{T\downarrow e}) = \alpha_f(u,v)$.*

Cut sparsification. Note that the cut-equivalent notion only requires preserving the $\binom{n}{2}$ min *st*-cuts but not all $2^{n-1}-1$ cuts.

Theorem 2.4.2 (Goemans and Ramakrishnan [GR95])**.** *Let V be a finite ground set and let $f : 2^V \to \mathbb{R}$ be a symmetric submodular*[1] *function. Then there exists a cut-equivalent tree for (V, f).*

An immediate corollary from Theorem 2.4.2 is that the cut function in a hypergraph (i.e., given a weighted hypergraph $H = (V, E_H, w_H)$, define f by $f(S) = w_H(S, \bar{S})$) has a cut-equivalent tree, as such cut functions are always symmetric and submodular.

We now show that cut-equivalent trees approximate *all* cuts up to a factor of $n-1$.

Theorem 2.4.3. *Let $H = (V, E_H, w_H)$ be a weighted hypergraph and let $T = (V, E_T, w_T)$ be a cut-equivalent tree of H (guaranteed to exist by Theorem 2.4.2). Then*

$$\forall S \subseteq V, \qquad w_H(S,\bar{S}) \leq w_T(S,\bar{S}) \leq (n-1) \cdot w_H(S,\bar{S}).$$

Moreover, both inequalities are existentially tight.

Proof. We begin with the second inequality. Let $e = uv \in \delta_T(S)$. By the first property of cut-equivalent trees it holds that $w_T(e) = \alpha_f(u,v) \leq w_H(S,\bar{S})$ as the cut $(S,\bar{S})$ in H separates u and v. The inequality follows by summing over all edges $e \in \delta_T(S)$, as T contains $n-1$ edges.

We now turn to the first inequality. By the second property of cut-equivalent trees it holds that $w_T(S,\bar{S}) = \sum_{e\in\delta_T(S)} w_H(V_{T\downarrow e}, \bar{V}_{T\downarrow e})$. Therefore, if we show that $\delta_H(S) \subseteq \cup_{e\in\delta_T(S)}\delta_H(V_{T\downarrow e})$, we would be done. To this end, let $st \in \delta_H(S)$. Then on the path from s to t in T there must be at least one edge $e = uv \in E_T$ such that $uv \in \delta_T(S)$. As $V_{T\downarrow e}$ separates s and t, it holds that $st \in \delta_H(V_{T\downarrow e})$, which completes the proof.

We proceed to show the tightness claim. For the first inequality, consider $S = V_{T\downarrow e}$ for $e \in E_T$ in any hypergraph H. For the second inequality, consider H that is an unweighted complete graph. Its cut-equivalent tree is a star with edges of weight $n-1$. Letting v be the center of the star, we have $w_T(v,\bar{v}) = (n-1)^2 = (n-1)\cdot w_H(v,\bar{v})$. □

Metric space sparsification. Observe that the weight of a cut S in a graph $G = (V, E, w)$ can be expressed as $\sum_{ij\in E} w(e)\cdot|x_i - x_j|$ where $x \in \{0,1\}^n$ is the indicator of S. Thus, Theorem 2.4.3 can be viewed as approximating all cut metrics (V, d_S) given by $d_S(i,j) = |x_i - x_j|$ where $x \in \{0,1\}^n$ is the indicator of S. We now generalize the ideas behind Theorem 2.4.3 to show that when $H = G$ is an ordinary graph, cut-equivalent trees in fact $(n-1)$-approximate all metrics on V and not only the cut metrics.

[1]Recall that $f : 2^V \to \mathbb{R}$ is said to be *submodular* if for every $A, B \subseteq V$ it holds that $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.

As a warm-up, when $H = G$ is an ordinary graph, it follows from Theorem 2.4.3 that

$$\forall x \in \mathbb{R}^n, \qquad \sum_{ij \in E_G} w_G(i,j) \cdot |x_i - x_j| \leq \sum_{ij \in E_T} w_T(i,j) \cdot |x_i - x_j| \leq (n-1) \sum_{ij \in E_G} w_G(i,j) \cdot |x_i - x_j|. \quad (2.3)$$

To see this, let $x \in \mathbb{R}^n$ and w.l.o.g. (by relabeling the vertices if needed) assume $x_1 \leq x_2 \leq ... \leq x_n$. Now, writing the inequalities in Theorem 2.4.3 for each of $S_1 = [1], S_2 = [2], ..., S_{n-1} = [n-1]$, multiplying the two inequalities of S_i by $x_{i+1} - x_i$ and summing the $n-1$ inequalities, we get (2.3). We now show that cut-equivalent trees $(n-1)$-approximate all metrics on V (and not only the absolute difference metric defined by n reals $x_1, x_2, ..., x_n \in \mathbb{R}$).

Theorem 2.4.4. *Let $G = (V, E_G, w_G)$ be a weighted graph and let $T = (V, E_T, w_T)$ be a cut-equivalent tree of G. Then for every n-point metric space $M = (\{x_i : i \in [n]\}, d_M)$,*

$$\sum_{ij \in E_G} w_G(i,j) \cdot d_M(x_i, x_j) \leq \sum_{ij \in E_T} w_T(i,j) \cdot d_M(x_i, x_j) \leq (n-1) \sum_{ij \in E_G} w_G(i,j) \cdot d_M(x_i, x_j).$$

Proof. We begin with the first inequality. For $e = uv \in E_G$, let P_e be the unique path in T from u to v. Observe that for $e \in E_T$ and $e' \in E_G$ we have $e' \in \delta_G(V_{T \downarrow e})$ iff $e \in P_{e'}$. Thus, by the second property of cut-equivalent trees,

$$\begin{aligned} \sum_{ij \in E_T} w_T(i,j) \cdot d_M(x_i, x_j) &= \sum_{e=ij \in E_T} w_G(V_{T \downarrow e}, \bar{V}_{T \downarrow e}) \cdot d_M(x_i, x_j) = \sum_{e=ij \in E_T} \sum_{e' \in E_G} \mathbf{1}_{\{\, e \in P_{e'} \,\}} \cdot d_M(x_i, x_j) \\ &= \sum_{uv \in E_G} w_G(u,v) \sum_{ij \in P_{uv}} d_M(x_i, x_j) \geq \sum_{uv \in E_G} w_G(u,v) \cdot d_M(x_u, x_v) \end{aligned}$$

where the inequality is by the triangle inequality in M.

We now turn to the second inequality. Let $uv \in E_T$. By the first property of cut-equivalent trees it holds that $w_T(u,v) = \alpha_f(u,v)$ which in turn is equal to the maximum flow between u and v in G. Therefore, there is a set of paths $\{\, P_i \,\}_{i \in [\ell]}$ from u to v in G and a set of positive numbers $\{\, \gamma_i \,\}_{i \in [\ell]}$ (the interpretation is that we ship γ_i units of flow through P_i) such that $\sum_{i \in [\ell]} \gamma_i = w_T(u,v)$ and for every $e \in E_G$ it holds that $\sum_{i \in [\ell] : e \in P_i} \gamma_i \leq w_G(e)$. As so,

$$\begin{aligned} w_T(u,v) \cdot d_M(x_u, x_v) &= \sum_{i \in [\ell]} \gamma_i \cdot d_M(x_u, x_v) \leq \sum_{i \in [\ell]} \gamma_i \sum_{ab \in P_i} d_M(x_a, x_b) \\ &= \sum_{ab \in E_G} d_M(x_a, x_b) \sum_{i \in [\ell] : ab \in P_i} \gamma_i \leq \sum_{ab \in E_G} d_M(x_a, x_b) \cdot w_G(a,b) \end{aligned}$$

where the first inequality is by the triangle inequality in M. The proof follows by summing over all edges $e \in E_T$, as T contains $n-1$ edges. □

Spectral sparsification. In light of Theorem 2.4.4, one may ask whether cut-equivalent trees of graphs are also $(n-1)$-spectral-sparsifiers. As we show next, this is not the case, although they are $(n-1)^3$-spectral-sparsifiers as a consequence of Theorem 2.4.4.

Theorem 2.4.5. *Let $G = (V, E_G, w_G)$ be a weighted graph and let $T = (V, E_T, w_T)$ be a cut-equivalent tree of G. Then*

$$G \preceq (n-1) \cdot T \preceq (n-1)^3 \cdot G.$$

Moreover, both inequalities are existentially tight up to constant factors.

Proof. We begin with the first inequality. Fix some $x \in \mathbb{R}^n$ and consider the n-point metric space $K = (\{ x_i : i \in [n] \}, d_K)$ where d_K is the shortest path metric on the n-clique with vertex set $\{ x_i : i \in [n] \}$ and edge weights $w_K(x_i, x_j) = (x_i - x_j)^2$. Let P_{ij} be any shortest path from x_i to x_j in the n-clique with respect to w_K, and let $|P_{ij}|$ be the number of edges in P_{ij}. Fix some $i, j \in [n]$. Then

$$\begin{aligned}(x_i - x_j)^2 &= \left(\sum_{\{ x_u, x_v \} \in P_{ij}} (x_u - x_v) \right)^2 \leq |P_{ij}| \cdot \sum_{\{ x_u, x_v \} \in P_{ij}} (x_u - x_v)^2 \\ &= |P_{ij}| \cdot d_K(x_i, x_j) \leq (n-1) \cdot d_K(x_i, x_j)\end{aligned}$$

where the first inequality is by the Cauchy-Schwartz inequality applied with the all-ones vector, and the last is as all paths in the n-clique are of length at most $n-1$. Hence,

$$\begin{aligned}\sum_{ij \in E_G} w_G(i,j) \cdot (x_i - x_j)^2 &\leq (n-1) \sum_{ij \in E_G} w_G(i,j) \cdot d_K(x_i, x_j) \\ &\leq (n-1) \sum_{ij \in E_T} w_T(i,j) \cdot d_K(x_i, x_j) \\ &\leq (n-1) \sum_{ij \in E_T} w_T(i,j) \cdot (x_i - x_j)^2\end{aligned}$$

where the second inequality is by Theorem 2.4.4, and the last is as $d_K(x_i, x_j) \leq w_K(x_i, x_j)$ (by construction). A similar proof that uses the second inequality of Theorem 2.4.4 proves the second inequality.

We proceed to show the tightness claim. We do so by providing a graph G_1 that has a unique cut-equivalent tree T_1 and a vector $x \in \mathbb{R}^n$ such that $x^T L_{G_1} x \geq \Omega(n) \cdot x^T L_{T_1} x$, and a graph G_2 such that for every cut-equivalent tree T_2 of G_2 there is vector $y \in \mathbb{R}^n$ s.t. $y^T L_{T_2} y \geq \Omega(n^2) \cdot y^T L_{G_2} y$. First, let G_1 be a cycle on the vertex set $[n]$ with distinct edge weights in $(1, 2)$. W.l.o.g. assume that the edge $(n, 1)$ has minimum weight among all edges. Let T_1 be any cut-equivalent tree of G_1. We claim that T_1 must be the path $1 \to 2 \to ... \to n$ with edge weights $w_{T_1}(i, i+1) = w_{G_1}(i, i+1) + w_{G_1}(n, 1)$. Indeed, all min st-cuts in G_1 have exactly two crossing edges, one of which is the edge $(n, 1)$ (by the uniqueness of the weights), and thus the only minimum st-cut sets are the $n-1$ sets $[1], [2], ..., [n-1]$. By the second property of cut-equivalent trees, for each such set S, there must be an edge in T_1 whose removal gives S as a connected component. This directly implies that T_1 must be the path $1 \to 2 \to ... \to n$. Now, for $x \in \mathbb{R}^n$ where $x_i = i$, we have that $x^T L_{T_1} x = (n-1) \cdot \Theta(1) \cdot 1^2 = \Theta(n)$ whereas

$$x^T L_{G_1} x \geq w_{G_1}(n, 1) \cdot (n-1)^2 = \Omega(n^2) = \Omega(n) \cdot x^T L_{T_1} x.$$

Second, consider G_2 that is an unweighted cycle on the vertex set $[n]$ (where four divides n) augmented with the unweighted edges $\{ \{ i, \frac{n}{2} + i \} : i \in [\frac{n}{2}] \}$. It is easy to verify that every singleton

cut in G_2 has weight 3 whereas any cut in G_2 that is not a singleton has weight > 3 and thus the only minimum st-cuts in G_2 are singletons. Hence, any cut-equivalent tree T_2 of G_2 must be a star. Now, for $y \in \mathbb{R}^n$ where

$$y = (1, 2, ..., \frac{n}{4}, \frac{n}{4} - 1, \frac{n}{4} - 2, ..., 1, 0, 1, 2, ..., \frac{n}{4}, \frac{n}{4} - 1, \frac{n}{4} - 2, ..., 1, 0)^T,$$

we have that $y^T L_{G_2} y = n \cdot 1^2 + \frac{n}{2} \cdot 0^2 = n$ whereas $y^T L_{T_2} y \geq \Omega(n^3) = \Omega(n^2) \cdot y^T L_{G_2} y$ as the center of the star T_2 is connected to $\Omega(n)$ vertices whose y values differ by $\Omega(n)$ from the center's y value. □

2.5 Sparsification by Spanning Trees

We will continue using the notation established in Section 2.4. Throughout, a spanning tree T of a weighted graph $G = (V, E_G, w_G)$ inherits its tree edge-weights from G, formally, $T = (V, E_T, w_T)$ where $w_T(e) = w_G(e)$ for each $e \in E_T$.

In this section we consider sparsification by spanning trees, i.e., sparsifying a graph G by a tree T that is restricted to be a spanning tree of G. Spielman [Spi10, problem 4] asks whether every graph G has a spanning tree T that is an $O(n)$-spectral-sparsifier, i.e., $\frac{1}{O(n)} \cdot G \preceq T \preceq G$ where the second inequality always holds as T is a subgraph of G. We use the connection between cut-sparsification and *congestion* to obtain lower bounds on the cut-sparsification approximation factor that answer Spielman's question negatively. Specifically, we show (in Theorem 2.5.1 and preceding discussion) that there are unweighted graphs G for which every spanning tree can only be a k-cut-sparsifier for $k = \Omega(n^{3/2})$, and there are m-edge weighted graphs G for which every spanning tree can only be a k-cut-sparsifier for $k = \Omega(m)$. We further show (in Corollary 2.5.3 and Corollary 2.5.4, respectively) that every unweighted graph G has a spanning tree that is an $n^{3/2}$-cut-sparsifier of G, and every m-edge weighted graph G has a spanning tree that is an $O(m)$-cut-sparsifier of G, matching our lower bounds.

We will need a few definitions. For a weighted graph $G = (V, E_G, w_G)$ and a spanning tree $T = (V, E_T, w_T)$ of G, define the *load* of $e \in E_T$ with respect to G, T as

$$\text{load}(G, T, e) \stackrel{\text{def}}{=} \frac{w_G(V_{T \downarrow e}, \bar{V}_{T \downarrow e})}{w_G(e)}.$$

For intuition, the load of an edge $e \in E_T$ is the amount of flow on e when every pair $uv \in E_G$ ships $w_G(u, v)$ units of flow on the unique path from u to v in T, normalized by $w_G(e) = w_T(e) = w_T(V_{T \downarrow e}, \bar{V}_{T \downarrow e})$. Define the *congestion* of T with respect to G as $\text{cong}(G, T) \stackrel{\text{def}}{=} \max_{e \in E_T} \text{load}(G, T, e)$.

Observe that if $k = \text{cong}(G, T)$ and $e \in E_T$ is an edge attaining this maximum, then $\frac{w_G(V_{T \downarrow e}, \bar{V}_{T \downarrow e})}{w_T(V_{T \downarrow e}, \bar{V}_{T \downarrow e})} = k$, which implies that T can only be an ℓ-cut-sparsifier of G for $\ell \geq k$. Therefore, to establish a lower bound k on the approximation factor one can get using spanning trees, it suffices to exhibit a graph G for which every spanning tree T of G has congestion at least k.

Ostrovskii [Ost04] showed that there is an infinite family of unweighted n-vertex graphs G such that every spanning tree T of G satisfies $\text{cong}(G, T) = \Omega(n^{3/2})$, which implies that $\Omega(n^{3/2})$ is a lower bound on the cut-sparsification approximation factor one can get using spanning trees, even

for unweighted graphs. While for unweighted graphs the lower bound is $\Omega(n^{3/2})$ (and in fact, there is a matching upper bound as we show in Corollary 2.5.3), for weighted graphs the picture is even worse, as established by Theorem 2.5.1.

Theorem 2.5.1. *For every even integer $n \geq 4$ and every integer $m \geq (1 + \Omega(1))n$, there is a weighted n-vertex graph G with m edges such that every spanning tree T of G satisfies* $\mathrm{cong}(G, T) \geq m - n + 2 = \Omega(m)$.

Proof. Consider G that is a union of a tree on $[n/2]$ and a tree on $[n] \setminus [n/2]$, augmented by $m-n+2$ edges crossing from $[n/2]$ to $[n] \setminus [n/2]$, where the tree edges have weight n^2 and the crossing edges have weight 1. Let $T = ([n], E_T)$ be any spanning tree of G. As T is connected, there must be an edge $e \in E_T$ that crosses from $[n/2]$ to $[n] \setminus [n/2]$. If removing e from T disconnects $[n/2]$ or $[n] \setminus [n/2]$, then $\mathrm{load}(G, T, e) \geq n^2 \geq m - n + 2$. Otherwise, removing e from T disconnects $[n]$ to $[n/2]$ and $[n] \setminus [n/2]$ which implies that e is the only edge in E_T that crosses from $[n/2]$ to $[n] \setminus [n/2]$ and hence $\mathrm{load}(G, T, e) = m - n + 2$. □

Turning back to Spielman's question, Theorem 2.5.1 shows that for weighted graphs, $\Omega(m)$ is a lower bound on the cut-sparsification approximation factor one can get using spanning trees. As the upper bound is (see section 2.1) $O(m \log n \log\log n)$ (which gives also spectral sparsification), Theorem 2.5.1 closes the gap up to logarithmic factors.

A natural next question is whether the $\Omega(n^{3/2})$ lower bound for unweighted graphs and the $\Omega(m)$ lower bound for weighted graphs can be matched by upper bounds. The answer is yes as established by Corollary 2.5.3 and Corollary 2.5.4.

Lemma 2.5.2. *Let $G = (V, E_G, w_G)$ be a weighted graph and let $T = (V, E_T, w_T)$ be a spanning tree of G. Construct $T' = (V, E_T, w_{T'})$ from T by multiplying the weight of each edge $e \in E_T$ by* $\mathrm{load}(G, T, e)$. *Then*

$$\forall S \subseteq V, \qquad w_{T'}(S, \bar{S}) \geq w_G(S, \bar{S}).$$

Moreover, for every n-point metric space $M = (\{ x_i : i \in [n] \}, d_M)$,

$$\sum_{ij \in E_T} \mathrm{load}(G, T, ij) \cdot w_T(i, j) \cdot d_M(x_i, x_j) \geq \sum_{ij \in E_G} w_G(i, j) \cdot d_M(x_i, x_j).$$

Proof. We focus on proving the latter statement as it implies the former (by setting x to be the indicator of the cut S, and d_M to be the absolute difference metric). The proof is closely related to the proof of Theorem 2.4.4. For $e = uv \in E_G$, let P_e be the unique path in T from u to v. Then

$$\begin{aligned} \sum_{ij \in E_G} w_G(i, j) \cdot d_M(x_i, x_j) &\leq \sum_{ij \in E_G} \sum_{uv \in P_{ij}} w_G(i, j) \cdot d_M(x_u, x_v) \\ &= \sum_{uv \in E_T} \mathrm{load}(G, T, uv) \cdot w_T(u, v) \cdot d_M(x_u, x_v) \end{aligned}$$

where the inequality is by the triangle inequality, and the equality is as

$$\mathrm{load}(G, T, uv) \cdot w_T(u, v) = \sum_{e \in E_G : uv \in P_e} w_G(e).$$

□

Corollary 2.5.3. *Every unweighted graph $G = (V, E_G)$ has a spanning tree $T = (V, E_T)$ such that for every n-point metric space $M = (\{ x_i : i \in [n] \}, d_M)$,*

$$\sum_{ij \in E_G} d_M(x_i, x_j) \leq n^{3/2} \cdot \sum_{ij \in E_T} d_M(x_i, x_j) \leq n^{3/2} \cdot \sum_{ij \in E_G} d_M(x_i, x_j).$$

Proof. It has been shown in [LRR09] that every unweighted graph G has a spanning tree T with $\text{cong}(G, T) \leq n^{3/2}$. The first inequality then follows from Lemma 2.5.2. The second inequality is trivial as T is a subgraph of G. □

Corollary 2.5.4. *Every weighted graph $G = (V, E_G, w_G)$ has a spanning tree $T = (V, E_T, w_T)$ such that for every n-point metric space $M = (\{ x_i : i \in [n] \}, d_M)$,*

$$\sum_{ij \in E_G} w_G(i,j) \cdot d_M(x_i, x_j) \leq (m-n+2) \cdot \sum_{ij \in E_T} w_T(i,j) \cdot d_M(x_i, x_j) \leq (m-n+2) \cdot \sum_{ij \in E_G} w_G(i,j) \cdot d_M(x_i, x_j),$$

where $m = |E_G|$. Furthermore, T can be constructed in $O(m + n \log n)$ time.

Proof. The second inequality is trivial as T is a subgraph of G. For the first inequality, by Lemma 2.5.2, it suffices to show how to construct T with congestion at most $m - n + 2$. Take T to be a maximum-weight spanning tree of G. The congestion bound follows as every edge $e \in E_T$ induces a cut in G with at most $m-n+2$ crossing edges, each having weight at most $w_G(e)$ (because adding it to T will close a cycle that contains e). □

2.6 Low Congestion Spanning Trees for Random Graphs

We will continue using the notation established in Sections 2.4 and 2.5. For example, recall that given an unweighted graph $G = (V, E_G)$ and a spanning tree $T = (V, E_T)$ of G, the congestion of T with respect to G is $\text{cong}(G, T) = \max_{e \in E_T} \text{load}(G, T, e)$, where $\text{load}(G, T, e)$ is the number of edges crossing the cut $(V_{T \downarrow e}, \bar{V}_{T \downarrow e})$ in G. As mentioned in Section 2.5, Ostrovskii showed [Ost04] that there are unweighted graphs G for which every spanning tree T of G satisfies $\text{cong}(G, T) = \Omega(n^{3/2})$. While such graphs exist, in this section we show that a random graph G drawn from $G(n, p)$ for a large range of values of p, with high probability has a spanning tree T with congestion $\tilde{O}(n)$, and moreover, T can be found in probabilistic linear time (in the number of edges of G). We assume that $C_1 \frac{\log^3 n}{n} \leq p = p(n) \leq C_2 \frac{1}{\log n}$ for some constants $C_1, C_2 > 0$ to be determined later, and at the end of the section we briefly explain what can be done for p's outside this range. This result nearly confirms a conjecture of Ostrovskii [Ost11, Problem 2] who asked whether random graphs have better spanning tree congestion bounds than general graphs, and wrote that it seems plausible that "most" random graphs in $G(n, p)$ have spanning trees with $O(n)$-congestion.

Notation. For a graph $G = (V, E_G)$ and two disjoint subsets $A, B \subseteq V$, let

$$E_G(A, B) \stackrel{\text{def}}{=} \{ e \in E_G : e \cap A \neq \emptyset, e \cap B \neq \emptyset \}$$

be the set of edges in G having one endpoint in A and one in B. Let $\epsilon = \epsilon(n) \stackrel{\text{def}}{=} \frac{1}{\log n}$. Let $d = d(n,p) \stackrel{\text{def}}{=} p(n-1)$ denote the expected degree in $G \sim G(n,p)$. Let $h = h(n,p) \in \mathbb{N}$ be the largest integer such that

$$s = s(n,p) \stackrel{\text{def}}{=} \sum_{i=0}^{h} \Big((1+\epsilon)d\Big)^i \leq n.$$

We assume henceforth that $s \geq C_3 \frac{n \log n}{d}$ for some constant $C_3 > 0$ to be determined later, and at the end of the section we explain how to remove this assumption. Note that $s \geq \Omega(\frac{n}{d})$ necessarily holds as otherwise $\sum_{i=0}^{h+1}((1+\epsilon)d)^i \leq n$, in contradiction to the way h was chosen.

We shall show that with probability $1 - O(n^{-1})$ over the selection of $G \sim G(n,p)$, the following algorithm finds, with probability $1 - O(n^{-1})$ over its internal coins, a spanning tree T of G that satisfies $\text{cong}(G,T) = O(n)$.

Algorithm 2.1 Input: $G = (V, E_G)$

---

1. Fix a vertex $r \in V$ (e.g., the first one lexicographically). Build a BFS tree from r up to (including) level h (where the level of the root is 0), and let $F = (V_F, E_F)$ denote the resulting tree.

2. Connect each remaining vertex $u \in V \setminus V_F$ to a uniformly random vertex among its neighbors in the tree, i.e., $\{\, v \in V_F : (u,v) \in E_G \,\}$ (if the set is empty report failure), and return the resulting tree, denoted $T = (V, E_T)$.

We now state the main result of this section.

Theorem 2.6.1. *With probability at least $1 - O(n^{-1})$ over the selection of $G = (V, E_G) \sim G(n,p)$, with probability at least $1 - O(n^{-1})$ over Algorithm 2.1's coins, the algorithm outputs a spanning tree T of G that satisfies $\text{cong}(G,T) = O(n)$. Moreover, Algorithm 2.1 always runs in linear time (in $|V| + |E_G|$).*

The rest of this section is devoted to proving Theorem 2.6.1. The plan is as follows. In Lemma 2.6.2 and Fact 2.6.3 we provide upper and lower bounds on the size of each level of F, in Lemma 2.6.4 we show why step 2 of the algorithm will "distribute" the remaining vertices roughly equally among the subtrees under different children of r, in Lemma 2.6.5 we present a simple fact that in $G(n,p)$ all cuts have roughly the expected weight, and finally we combine the pieces to prove Theorem 2.6.1.

Let $L^{(j)}(F)$ denote the vertices at level j of the tree F . For $v \in V_F$, let F_v denote the subtree of F rooted at v. Note that for random G, $L^{(j)}(F)$ and V_F are random variables (that are independent of the internal coins of the algorithm). Observe that on input $G = (V, E_G) \sim G(n,p)$, conditioned on the levels $0, 1, ..., \ell$ of F, the edges between $V \setminus \bigcup_{j=0}^{\ell} L^{(j)}(F)$ and $L^{(\ell)}(F)$ are i.i.d. Bernoulli random variables with success probability p. The analysis of the algorithm will use this fact several times. For a level $0 \leq \ell \leq h$, let $\mathcal{F}_\ell$ denote the event that the following conditions hold:

1. Every vertex in $L^{(\ell)}(F)$ has at most $(1+\epsilon)d$ children in F (for $\ell < h$); and

2. If $\ell < h$ then $|L^{(\ell)}(F)| \geq (1-\frac{1}{d})^{2\ell} \cdot (1-\epsilon)^{\ell} d^{\ell}$, and if $\ell = h$ then $|L^{(\ell)}(F)| \geq \frac{1}{2} \cdot (1-\frac{1}{d})^{2\ell-1} \cdot (1-\epsilon)^{\ell} d^{\ell}$.

For $0 \leq \ell \leq h$, let $\mathcal{F}_{\leq \ell}$ denote the event that $\mathcal{F}_j$ holds for every $0 \leq j \leq \ell$, i.e., $\mathcal{F}_{\leq \ell} = \cap_{j=0}^{\ell} \mathcal{F}_j$. Note that the events $\mathcal{F}$ depend solely on G and are independent of the randomness of the algorithm.

Lemma 2.6.2 (BFS growth rate)**.** *Let $G = (V, E_G) \sim G(n,p)$ be the input to Algorithm 2.1. Then for every $0 \leq \ell \leq h$ we have $\Pr[\mathcal{F}_{\leq \ell}] \geq 1 - \frac{\ell+1}{n^2}$. In particular, $\Pr[\mathcal{F}_{\leq h}] \geq 1 - \frac{1}{n}$.*

Proof. We prove the Lemma by induction on ℓ. For $\ell = 0$, (2) clearly holds, and (1) holds with probability at least $1 - \frac{1}{n^2}$ by a Chernoff bound (as $d \geq C_1 \epsilon^{-2} \log n$ for a sufficiently large constant C_1). Assuming correctness for $\ell - 1$, we prove the Lemma for ℓ. It suffices to show that $\Pr[\mathcal{F}_{\ell} | \mathcal{F}_{\leq \ell-1}] \geq 1 - \frac{1}{n^2}$, as then, by the induction hypothesis,

$$\Pr[\mathcal{F}_{\leq \ell}] = \Pr[\mathcal{F}_{\leq \ell-1}] \cdot \Pr[\mathcal{F}_{\ell} | \mathcal{F}_{\leq \ell-1}] \geq (1 - \frac{\ell}{n^2}) \cdot (1 - \frac{1}{n^2}) \geq 1 - \frac{\ell}{n^2} - \frac{1}{n^2}.$$

To this end, assume henceforth that $\mathcal{F}_{\leq \ell-1}$ occurred. For (1), a union bound over Chernoff bounds shows that with probability at least $1 - \frac{1}{2n^2}$, every vertex in $L^{(\ell)}(F)$ has at most $(1+\epsilon)d$ children in F. Toward (2), let $k = |V \setminus \bigcup_{j=0}^{\ell-1} L^{(j)}(F)|$, and let $t = |L^{(\ell-1)}(F)|$. Observe that

$$\sum_{j=0}^{\ell-1} |L^{(j)}(F)| \leq \sum_{j=0}^{\ell-1} (1+\epsilon)^j d^j \leq \sum_{j=0}^{h-1} (1+\epsilon)^j d^j \leq \frac{s}{(1+\epsilon)d} \leq \frac{n}{d},$$

and thus $k \geq n - \frac{n}{d}$. Observe that

$$(1 - \frac{1}{d})^{2\ell-2} \cdot (1-\epsilon)^{\ell-1} d^{\ell-1} \leq t \leq (1+\epsilon)^{\ell-1} d^{\ell-1} < \frac{n}{d^{h-(\ell-1)}}.$$

Now, fixing any $v \in V \setminus \bigcup_{j=0}^{\ell-1} L^{(j)}(F)$,

$$\Pr[v \notin L^{(\ell)}(F)] = (1-p)^t \leq e^{-pt} \leq 1 - pt + \frac{(pt)^2}{2}.$$

Thus,

$$\begin{aligned}\mathbb{E}[|L^{(\ell)}(F)|] \geq k(pt - \frac{(pt)^2}{2}) = kpt(1 - \frac{pt}{2}) \geq (n - \frac{n}{d}) \frac{d}{n} (1 - \frac{1}{d})^{2\ell-2} (1-\epsilon)^{\ell-1} d^{\ell-1} (1 - \frac{1}{2} \cdot \frac{d}{n} \cdot \frac{n}{d^{h-(\ell-1)}}) \\ = (1 - \frac{1}{d})^{2\ell-1} (1-\epsilon)^{\ell-1} d^{\ell} (1 - \frac{1}{2d^{h-\ell}}).\end{aligned}$$

Noting that for $\ell < h$ we have $(1 - \frac{1}{2d^{h-\ell}}) \geq (1 - \frac{1}{d})$ and for $\ell = h$ we have $(1 - \frac{1}{2d^{h-\ell}}) = \frac{1}{2}$, we get that for $\ell < h$ it holds that $\mathbb{E}[|L^{(\ell)}(F)|] \geq (1 - \frac{1}{d})^{2\ell} \cdot (1-\epsilon)^{\ell-1} d^{\ell}$ and for $\ell = h$ it holds that $\mathbb{E}[|L^{(\ell)}(F)|] \geq \frac{1}{2} \cdot (1 - \frac{1}{d})^{2h-1} \cdot (1-\epsilon)^{h-1} d^{h}$. As $d \geq \log n = \epsilon^{-1}$ and $h \leq \log n$, $(1 - \frac{1}{d})^{2\ell}$ and $(1-\epsilon)^{\ell}$ are bounded away from 0. Thus, by a union bound over Chernoff bounds we get that the probability of $|L^{(\ell)}(F)|$ being less than $(1-\epsilon)$ of its expectation (which evaluates exactly to the expression in (2)) is at most $\frac{1}{2n^2}$. Taking a union bound over (1) and (2), the induction step follows. □

Fact 2.6.3 (Cardinality of $L^{(h)}(F)$)**.** *Let $G = (V, E_G)$ be the input to Algorithm 2.1 and suppose that the event $\mathcal{F}_{\leq h}$ holds. Then $|V_F| = \Omega(s)$. Moreover, $|L^{(h)}(F)| = \Omega(s)$.*

Proof. As the event $\mathcal{F}_{\leq h}$ holds,

$$|V_F| \geq \frac{1}{2} \cdot \sum_{j=0}^{h} \Big(1 - \frac{1}{d}\Big)^{2j} (1-\epsilon)^j d^j \geq \frac{1}{2} \cdot \sum_{j=0}^{h} \Big(1 - \frac{1}{d}\Big)^{2h} \Big(\frac{1-\epsilon}{1+\epsilon}\Big)^h \cdot \Big((1+\epsilon)d\Big)^j = \frac{1}{2} \cdot \Big(1 - \frac{1}{d}\Big)^{2h} \Big(\frac{1-\epsilon}{1+\epsilon}\Big)^h \cdot s$$
$$\geq \frac{1}{2} \cdot \Big(1 - \frac{1}{d}\Big)^{2\log n} \Big(\frac{1-\epsilon}{1+\epsilon}\Big)^{\log n} \cdot s = \Omega(1) \cdot s = \Omega(s),$$

as $d \geq \log n = \epsilon^{-1}$. The moreover clause follows as

$$\sum_{j=0}^{h-1} |L^{(j)}(F)| \leq \sum_{j=0}^{h-1} \Big((1+\epsilon)d\Big)^j \leq \frac{s}{(1+\epsilon)d} \leq \frac{s}{d},$$

and hence $|L^{(h)}(F)| \geq \Omega(s) - \frac{s}{d} = \Omega(s)$. □

Lemma 2.6.4. *Let $G = (V, E_G) \sim G(n,p)$ be the input to Algorithm 2.1. Then with probability at least $1 - O(n^{-1})$ over the selection of G, the following holds:*

(a) For every $u \in V \setminus V_F$, $|E_G(u, V_F)| = \Omega(d \cdot \frac{s}{n})$; and

(b) For every child v of r, $\sum_{u \in V \setminus V_F} |E_G(u, V(F_v))| = O(s)$.

Proof. By a union bound over the event $\mathcal{F}_{\leq h}$, we may assume that all the vertices of F have at most $d(1+\epsilon)$ children in F and that $|L^{(h)}(F)| = \Omega(s)$. For (a), fix $u \in V \setminus V_F$. Then

$$\mathbb{E}[|E_G(u, V_F)|] = p \cdot |L^{(h)}(F)| = p \cdot \Omega(s) = \Omega(d \cdot \frac{s}{n}).$$

By a union bound over Chernoff bounds (here we use our assumption that $s \geq C_3 \frac{n \log n}{d}$ for a sufficiently large constant C_3) we get that with probability at least $1 - n^{-1}$, for every $u \in V \setminus V_F$ it holds that $|E_G(u, V_F)| = \Omega(d \cdot \frac{s}{n})$. Toward (b), fix a child v of r. Observe that $|V(F_v)| \leq \sum_{j=0}^{h-1} \Big((1+\epsilon)d\Big)^j \leq \frac{s}{d}$. Thus,

$$\mathbb{E}[\sum_{u \in V \setminus V_F} |E_G(u, V(F_v))|] = \mathbb{E}[|E_G(V \setminus V_F, V(F_v))|] = p \cdot |V \setminus V_F| \cdot |V(F_v) \cap L^{(h)}(F)| \leq p(n-1)\frac{s}{d} = s.$$

(b) then follows by a union over Chernoff bounds (as $s \geq \Omega(\frac{n}{d}) \geq \Omega(\frac{\log n}{C_2})$ for a sufficiently small constant C_2). Taking a union bound over (a) and (b) completes the proof. □

Lemma 2.6.5 (Cut weights)**.** *Let $G = (V, E_G) \sim G(n,p)$. Then with probability at least $1 - O(n^{-1})$, for every cut $S \subseteq V$ we have $|E_G(S, \bar{S})| \in (1 \pm \epsilon) \cdot p \cdot |S| \cdot |\bar{S}|$.*

Proof. Fix some $S \subseteq V$ of size $k \leq n/2$. By the Chernoff bound,

$$\Pr[|E_G(S, \bar{S})| \notin (1 \pm \epsilon) \cdot p \cdot |S| \cdot |\bar{S}|] \leq 2\exp(-\epsilon^2 pk(n-k)/3) \leq 2\exp(-\epsilon^2 pkn/6)$$
$$\leq 2\exp(-k \log n \cdot C_1/6) \leq n^{-2k},$$

where we used the assumption that $p \geq C_1 \frac{\log n}{n} \epsilon^{-2}$ for a sufficiently large C_1. Thus, by the union bound, the probability that there is some violating cut is bounded by

$$\sum_{k \in [\lfloor n/2 \rfloor]} \binom{n}{k} n^{-2k} \leq \sum_{k \in [\lfloor n/2 \rfloor]} n^{-k} = O(n^{-1}).$$

□

We are now ready to prove Theorem 2.6.1.

Proof (of Theorem 2.6.1). By a union bound, with probability at least $1 - O(n^{-1})$, we may assume that G admits to the conclusions of Lemma 2.6.2, Lemma 2.6.4 and Lemma 2.6.5. As G admits to the conclusion of Lemma 2.6.4, the algorithm will not fail, as $\Omega(d \cdot \frac{s}{n}) \geq \Omega(\log n) \geq 1$. Toward proving that $\mathrm{cong}(G, T) = O(n)$, it suffices to show that for every child v of r $|V(T_v)| = O(\frac{n}{d})$, because in such a case, any cut $V_{T \downarrow e}$ induced by removing an edge e from T will have one side of size at most $x \stackrel{\text{def}}{=} O(\frac{n}{d}) = O(\frac{1}{p})$, and hence, as G admits to the conclusion of Lemma 2.6.5, we have $|E_G(V_{T \downarrow e}, \bar{V}_{T \downarrow e})| \leq p(1+\epsilon)x(n-x) \leq O((1+\epsilon)n) = O(n)$. As G admits to the conclusion of Lemma 2.6.2, for every child v of r,

$$|V(F_v)| \leq \frac{s}{d} \leq \frac{n}{d}$$

(as seen in the proof of Lemma 2.6.4). It remains to show that step 2 of the algorithm does not add too many vertices to any F_v. For $u \in V \setminus F_V$ and a child v of r, let $p_{u,v}$ be the probability (over the randomness of the algorithm) of connecting u to F_v at step 2. As G admits to the conclusion of Lemma 2.6.4, for every child v of r, the expected number of vertices added to F_v at step 2 is

$$\sum_{u \in V \setminus V_F} p_{u,v} = \sum_{u \in V \setminus F_V} \frac{|E_G(u, V(F_v))|}{|E_G(u, V_F)|} \leq \sum_{u \in V \setminus V_F} \frac{|E_G(u, V(F_v))|}{\Omega(d \cdot \frac{s}{n})} \leq \frac{O(s)}{\Omega(d \cdot \frac{s}{n})} = O(\frac{n}{d}).$$

By a union bound over Chernoff bounds (as $\frac{n}{d} \eqsim \frac{1}{p} \geq \frac{\log n}{C_2}$ for a sufficiently small constant C_2), for every child v of r we have $|V(T_v) \setminus V(F_v)| \leq O(\sum_{u \in V \setminus V_F} p_{u,v}) = O(\frac{n}{d})$, and thus $|V(T_v)| = |V(F_v)| + |V(T_v) \setminus V(F_v)| = O(\frac{n}{d})$. The theorem follows. □

Recall that we assumed $s \geq C_3 \cdot \frac{n \log n}{d}$. We now sketch how to remove this assumption. Removing the assumption will weaken the result of Theorem 2.6.1 to only guarantee $O(n \log n)$-congestion. The assumption was used in two places: (a) for claiming that step 2 of the algorithm will connect all of the remaining vertices, and (b) for bounding $\sum_{u \in V \setminus V_F} p_{u,v}$ in the proof of Theorem 2.6.1. Regarding (b), if $s < C_3 \frac{n \log n}{d}$, we can replace the calculation in the proof of Theorem 2.6.1 by the naive bound

$$\sum_{u \in V \setminus V_F} p_{u,v} = \sum_{u \in V \setminus V_F} \frac{|E_G(u, V(F_v))|}{|E_G(u, V_F)|} \leq \sum_{u \in V \setminus V_F} |E_G(u, V(F_v))| \leq O(s) \leq O(\frac{n \log n}{d}),$$

where the second to last inequality is by part (b) of Lemma 2.6.4 (which holds even if $s < C_3 \frac{n \log n}{d}$, unlike part (a) of the same lemma). Note that this naive bound only incurs an additional $\log n$ factor, and we will get that after step 2, each $V(T_v)$ is of cardinality at most $O(\frac{n \log n}{d})$. Regarding

(a), notice that even when $s < C_3 \cdot \frac{n \log n}{d}$, the tree T will have $\Omega(n)$ vertices as $s \geq \Omega(\frac{n}{d})$ and hence each vertex in $V \setminus V_F$ has at least constant probability to be added to the tree at step 2. We thus modify the algorithm by removing the failure clause from step 2 and adding a step 3 that connects each still-not-connected (after step 2) vertex to a uniformly random neighbor in $V(T)$ (essentially cloning step 2). Using similar analysis to the proof of Lemma 2.6.4 and Theorem 2.6.1, we will get, with probability $1 - O(n^{-1})$, that every vertex in V is in the resulting tree and that we added at most $O(\frac{n \log n}{d})$ vertices to each T_v.

We remark that for the range $p < C_1 \frac{\log^3 n}{n}$, any spanning tree T of G (if G is connected) will trivially satisfy (with probability $1 - O(n^{-1})$) $\text{cong}(G, T) \leq |E_G| = O(n^2 p) = \tilde{O}(n)$. Also, for the range $p > C_2 \frac{1}{\log n}$, replacing the bound of $O(s)$ by $O(s \log n)$ in part (b) of Lemma 2.6.4 will make the analysis go through and only incur an additional $O(\log n)$ factor to the congestion bound. We further remark that step 2 of Algorithm 2.1 can be derandomized using flow techniques.

Bibliography

[ABC+08] R. Andersen, C. Borgs, J. T. Chayes, J. E. Hopcroft, V. S. Mirrokni, and S. Teng. Local computation of pagerank contributions. *Internet Mathematics*, 5(1):23–45, 2008. doi:10.1080/15427951.2008.10129302.

[ACK+16] A. Andoni, J. Chen, R. Krauthgamer, B. Qin, D. P. Woodruff, and Q. Zhang. On sketching quadratic forms. In *Innovations in Theoretical Computer Science*, ITCS' 16, pages 311–319. ACM, 2016. doi:10.1145/2840728.2840753.

[AKP17] A. Andoni, R. Krauthgamer, and Y. Pogrow. Solving SDD linear systems in sublinear time. Manuscript, September 2017.

[Amb12] A. Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 636–647. LIPIcs, 2012. doi:10.4230/LIPIcs.STACS.2012.636.

[AN12] I. Abraham and O. Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 395–406, New York, NY, USA, 2012. ACM. doi:10.1145/2213977.2214015.

[AP09] R. Andersen and Y. Peres. Finding sparse cuts locally using evolving sets. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 235–244. ACM, 2009. doi:10.1145/1536414.1536449.

[BEGK04] E. Boros, K. M. Elbassioni, V. Gurvich, and L. Khachiyan. Generating maximal independent sets for hypergraphs with bounded edge-intersections. In *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium*, pages 488–498, 2004. doi:10.1007/978-3-540-24698-5_52.

[BGH+06] M. W. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo. Support-graph preconditioners. *SIAM J. Matrix Analysis Applications*, 27(4):930–951, 2006. doi:10.1137/S0895479801384019.

[BK96] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *28th Annual ACM Symposium on Theory of Computing*, pages 47–55. ACM, 1996. doi:10.1145/237814.237827.

[BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998. doi:10.1016/S0169-7552(98)00110-X.

[BSS12] J. D. Batson, D. A. Spielman, and N. Srivastava. Twice-ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012. doi:10.1137/090772873.

[CBD+09] U. V. Catalyurek, E. G. Boman, K. D. Devine, D. Bozdağ, R. T. Heaphy, and L. A. Riesen. A repartitioning hypergraph model for dynamic load balancing. *Journal of Parallel and Distributed Computing*, 69(8):711–724, 2009. doi:10.1016/j.jpdc.2009.04.011.

[CGS04] Y.-Y. Chen, Q. Gan, and T. Suel. Local methods for estimating pagerank values. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 381–389. ACM, 2004. doi:10.1145/1031171.1031248.

[CKM+14] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu. Solving SDD linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 343–352, 2014. doi:10.1145/2591796.2591833.

[CKS15] A. M. Childs, R. Kothari, and R. D. Somma. Quantum linear systems algorithm with exponentially improved dependence on precision. *arXiv preprint arXiv:1511.02306*, 2015.

[CS15] F. Chung and O. Simpson. Solving local linear systems with boundary conditions using heat kernel pagerank. *Internet Mathematics*, 11(4-5):449–471, 2015. doi:10.1080/15427951.2015.1009522.

[CX16] C. Chekuri and C. Xu. Computing minimum cuts in hypergraphs. *arXiv preprint arXiv:1607.08682*, 2016.

[DGT17] D. Doron, F. L. Gall, and A. Ta-Shma. Probabilistic logarithmic-space algorithms for laplacian solvers. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017*, pages 41:1–41:20, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.41.

[DSTS17] D. Doron, A. Sarid, and A. Ta-Shma. On approximating the eigenvalues of stochastic matrices in probabilistic logspace. *Comput. Complex.*, 26(2):393–420, June 2017. doi:10.1007/s00037-016-0150-y.

[FL50] G. E. Forsythe and R. A. Leibler. Matrix inversion by a monte carlo method. *Mathematics of Computation*, 4(31):127–129, 1950. doi:10.1090/S0025-5718-1950-0038138-X.

[Gal14] F. L. Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

[GH61] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9:551–570, 1961. doi:10.1137/0109047.

[GMT15] S. Guha, A. McGregor, and D. Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 241–247. ACM, 2015. doi:10.1145/2745754.2745763.

[GR95] M. X. Goemans and V. Ramakrishnan. Minimizing submodular functions over families of sets. *Combinatorica*, 15(4):499–513, 1995. doi:10.1007/BF01192523.

[HHL09] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009. doi:10.1103/PhysRevLett.103.150502.

[HLM09] Y. Huang, Q. Liu, and D. Metaxas. Video object segmentation by hypergraph cut. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1738–1745. IEEE, 2009. doi:10.1109/CVPRW.2009.5206795.

[KBEG07] L. Khachiyan, E. Boros, K. M. Elbassioni, and V. Gurvich. On the dualization of hypergraphs with bounded edge-intersections and other related classes of hypergraphs. *Theor. Comput. Sci.*, 382(2):139–150, 2007. doi:10.1016/j.tcs.2007.03.005.

[KK15] D. Kogan and R. Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 367–376. ACM, 2015. doi:10.1145/2688073.2688093.

[KZ17] R. Kyng and P. Zhang. Hardness results for structured linear systems. *CoRR*, abs/1705.02944, 2017. Available from: `http://arxiv.org/abs/1705.02944`.

[LRR09] C. Löwenstein, D. Rautenbach, and F. Regen. On spanning tree congestion. *Discrete Mathematics*, 309(13):4653–4655, 2009. `doi:10.1016/j.disc.2009.01.012`.

[NR13] I. Newman and Y. Rabinovich. On multiplicative λ-approximations and some geometric applications. *SIAM Journal on Computing*, 42(3):855–883, 2013. `doi:10.1137/100801809`.

[Ost04] M. Ostrovskii. Minimal congestion trees. *Discrete Mathematics*, 285(1):219–226, 2004. `doi:10.1016/j.disc.2004.02.009`.

[Ost11] M. Ostrovskii. Minimum congestion spanning trees in bipartite and random graphs. *Acta Mathematica Scientia*, 31(2):634–640, 2011. `doi:10.1016/S0252-9602(11)60263-4`.

[RS98] J. Radhakrishnan and A. Srinivasan. Improved bounds and algorithms for hypergraph two-coloring. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98*, pages 684–693, 1998. `doi:10.1109/SFCS.1998.743519`.

[RTVX11] R. Rubinfeld, G. Tamir, S. Vardi, and N. Xie. Fast local computation algorithms. In *Innovations in Computer Science - ICS 2010*, pages 223–238, 2011. Available from: `http://conference.itcs.tsinghua.edu.cn/ICS2011/content/papers/36.html`.

[SBL16] N. Shyamkumar, S. Banerjee, and P. Lofgren. Sublinear estimation of a single element in sparse linear systems. In *54th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2016*, pages 856–860, 2016. `doi:10.1109/ALLERTON.2016.7852323`.

[Sin04] S. N. Sinha. Graph cut algorithms in vision, graphics and machine learning–an integrative paper. *UNC Chapel Hill*, 2004.

[Spi10] D. A. Spielman. Algorithms, graph theory, and linear equations in laplacian matrices. In *Proceedings of the International Congress of Mathematicians*, volume 4, pages 2698–2722, 2010. `doi:10.1142/9789814324359_0164`.

[SS08] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *40th Annual ACM Symposium on Theory of Computing*, pages 563–568. ACM, 2008. `doi:10.1145/1374376.1374456`.

[SS11] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, December 2011. `doi:10.1137/080734029`.

[ST04] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *36th Annual ACM Symposium on Theory of Computing*, pages 81–90. ACM, 2004. `doi:10.1145/1007352.1007372`.

[Suo13] J. Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, March 2013. `doi:10.1145/2431211.2431223`.

[VH90] A. Vannelli and S. W. Hadley. A gomory-hu cut tree representation of a netlist partitioning problem. *IEEE Transactions on Circuits and Systems*, 37(9):1133–1139, 1990. `doi:10.1109/31.57601`.

[Vis13] N. K. Vishnoi. Lx = b. *Foundations and Trends in Theoretical Computer Science*, 8(1–2):1–141, 2013. `doi:10.1561/0400000054`.

[Was52] W. R. Wasow. A note on the inversion of matrices by random walks. *Mathematical Tables and Other Aids to Computation*, 6(38):78–81, 1952. `doi:10.2307/2002546`.

[YOTI15] Y. Yamaguchi, A. Ogawa, A. Takeda, and S. Iwata. Cyber security analysis of power networks by hypergraph cut algorithms. *IEEE Transactions on Smart Grid*, 6(5):2189–2199, 2015. doi:10.1109/TSG.2015.2394791.