Nearly-Linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving Linear Systems

preliminary draft

Daniel A. Spielman *
Department of Mathematics
Massachusetts Institute of Technology

Shang-Hua Teng †
Department of Computer Science
Boston University and
Akamai Technologies Inc.

September 24, 2008

Abstract

We develop nearly-linear time algorithms for approximately solving sparse symmetric diagonally-dominant linear systems. In particular, we present a linear-system solver that, given an n-by-n symmetric diagonally-dominant matrix A with m non-zero entries and an n-vector $\boldsymbol{b}$, produces a vector $\tilde{\boldsymbol{x}}$ satisfying $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_A \leq \epsilon$, where $\boldsymbol{x}$ is the solution to $A\boldsymbol{x} = \boldsymbol{b}$, in time

$$m \log^{O(1)} m + O\left(m \log(1/\epsilon)\right) + n2^{O(\sqrt{\log n \log\log n})} \log(1/\epsilon).$$

We remark that while our algorithm is designed for sparse matrices, even for dense matrices the dominant term in its complexity is $O(n^{2+o(1)})$.

Our algorithm exploits two novel tools. The first is a nearly-linear time algorithm for approximately computing crude graph partitions. For any graph G having a cut of sparsity ϕ and balance b, this algorithm outputs a cut of sparsity at most $O(\phi^{1/3} \log^{O(1)} n)$ and balance $b(1-\epsilon)$ in time $m((\log m)/\phi)^{O(1)}$.

Using this graph partitioning algorithm, we design fast graph sparsifiers and graph ultra-sparsifiers. On input a weighted graph G with Laplacian matrix L and an $\epsilon > 0$, the graph sparsifier produces a weighted graph $\tilde{G}$ with Laplacian matrix $\tilde{L}$ such that $\tilde{G}$ has $n(\log^{O(1)} n)/\epsilon^2$ edges and such that for all $\boldsymbol{x} \in \mathbb{R}^n$,

$$\boldsymbol{x}^T \tilde{L} \boldsymbol{x} \leq \boldsymbol{x}^T L \boldsymbol{x} \leq (1+\epsilon) \boldsymbol{x}^T \tilde{L} \boldsymbol{x}.$$

The ultra-sparsifier takes as input a parameter t and outputs a graph $\tilde{G}$ with $(n-1) + tn^{o(1)}$ edges such that for all $\boldsymbol{x} \in \mathbb{R}^n$

$$\boldsymbol{x}^T \tilde{L} \boldsymbol{x} \leq \boldsymbol{x}^T L \boldsymbol{x} \leq \left(\frac{n}{t}\right) \boldsymbol{x}^T \tilde{L} \boldsymbol{x}.$$

Both algorithms run in time $m \log^{O(1)} m$.

These ultra-sparsifiers almost asymptotically optimize the potential of the combinatorial preconditioners introduced by Vaidya.

*spielman@math.mit.edu
†steng@cs.bu.edu

1 Introduction

We present a nearly-linear time algorithm for solving symmetric, diagonally-dominant (SDD) linear systems to arbitrary precision. In particular, our algorithm runs in time nearly-linear in the number of non-zero entries in the matrix specifying the system, logarithmic in the precision desired, and logarithmic in the condition number of the matrix. For those not familiar with the condition number, we remark that it is the ratio of the largest to smallest non-zero eigenvalue of the matrix; for SDD matrices it is $O(L\log(n))$, where L is the logarithm of the ratio of the largest to smallest non-zero entry in the matrix; that it is typically much smaller; that it is standard to state the running times of linear system solvers in term of the condition number; and that the log of the condition number is a lower bound on the number of bits of precision required by any linear system solver.

We build upon Vaidya's [Vai90] remarkable construction of provably-good graph theoretic preconditioners that enable the fast solution of linear systems. Vaidya proved that by augmenting spanning trees with a few edges, one could find ϵ-approximate solutions to SDD linear systems of maximum valence d in time $O((dn)^{1.75}\log(\kappa_f(A)/\epsilon))$, and of planar linear systems in time $O((dn)^{1.2}\log(\kappa_f(A)/\epsilon))$. While Vaidya's work was unpublished, proofs of his results as well as extensions may be found in [Jos97, Gre96, GMZ95, BGH$^+$, BCHT, BH]. By recursively applying Vaidya's preconditioners, Reif [Rei98] improved the running time for constant-valence planar linear systems to $O(n^{1+\beta}\log^{O(1)}(\kappa_f(A)/\epsilon))$, for every $\beta > 0$. The running time for general linear systems was improved by Boman and Hendrickson [BH01] to $m^{1.5+o(1)}\log(\kappa_f(A)/\epsilon)$, by Maggs, *et. al.* [MMP$^+$02] to $O(mn^{1/2}\log^2(n\kappa_f(A)/\epsilon))$ after some preprocessing, and by Spielman and Teng [ST03] to $m^{1.31+o(1)}\log(1/\epsilon)\log^{O(1)}(n/\kappa_f(A))$. For more background on how these algorithms work, we refer the reader to this last paper.

In this work, we re-state the problem of building combinatorial preconditioners as that of finding sparsifiers for weighted graphs that approximate the original. We say that a graph is d-sparse if it has at most dn edges. We say that a graph is k-ultra-sparse if it has $n-1+k$ edges, and note that a spanning tree is 0-ultra-sparse. We say that a graph $\tilde{A}$ γ-approximates a graph A if

$$\mathcal{L}(\tilde{A}) \preccurlyeq \mathcal{L}(A) \preccurlyeq \gamma\mathcal{L}(\tilde{A}),$$

where $\mathcal{L}(A)$ is the Laplacian of A (the diagonal matrix of the weighted degrees of A minus the adjacency matrix of A) and $X \preccurlyeq Y$ means that for all $\boldsymbol{x} \in \mathbb{R}^n$,

$$\boldsymbol{x}^T X \boldsymbol{x} \leq \boldsymbol{x}^T Y \boldsymbol{x}.$$

Vaidya's preconditioners and their improvements can be understood as constructions of ultra-sparse graph approximations. For example, Vaidya showed how to construct for any weighted graph A a t^2-ultra-sparse graph $\tilde{A}$ that $(m/t)^2$-approximates A, for any t. Boman and Hendrickson [BH] applied the trees of [AKPW95] to construct a 0-ultra-sparse $m^{1+o(1)}$-approximations of A. Spielman and Teng [ST03] augmented this construction to obtain for any t an $O(t^2\log n)$-ultra-sparse graph that $(m^{1+o(1)}/t)$-approximates A.

In this work, we augment the low-stretch spanning trees of Alon, Karp, Peleg and West [AKPW95] to obtain $tn^{o(1)}$-ultra-sparse graphs that $\left((n/t)\log^{O(1)} n\right)$-approximate A for all $t \geq 1$. Our linear system solver is obtained immediately by plugging this ultra-sparsifier construction into the recursive algorithm of [ST03].

Our ultra-sparsifiers are derived from more ordinary sparsifiers that produce $\log^{O(1)}$-sparse graphs that $(1+\epsilon)$-approximate the original graph, for any $\epsilon > 0$. Our algorithm for constructing these is stated in Section 7.

While the analysis in this paper may be long, the idea behind the construction of our sparsifiers is quite simple: we show that if a graph A has no sparse cuts, then a natural random rounding of A will be a good approximation of A. Thus, to approximate a general graph A, we would like to remove a small fraction of the edges of A so that each remaining component has no sparse cuts. We then sparsify each of these components via a random rounding, and then apply the algorithm recursively to the edges we removed. Thus, to make the algorithm efficient, we need merely find a fast algorithm for removing those edges. This turns out to be tricky. The other part—proving that the random rounding of a graph with no sparse cuts is a good approximation of the original—is cleanly accomplished in Section 5 by adapting techniques of Füredi and Komlós [FK81].

In Section 3, we present an algorithm that quickly finds crude cuts in graphs of approximately optimal balance. Given a graph G containing a set of vertices S such that $\Phi(S) < \phi$ and $\mathrm{Vol}\,(S) \leq \mathrm{Vol}\,(V)\,/2$, our algorithm `Partition` finds a set of vertices T such that $\mathrm{Vol}\,(T) \geq \mathrm{Vol}\,(S)\,/2$ and $\Phi(T) \leq O(\phi^{1/3} \log^{O(1)} n)$ in time $O(m\,((\log n)/\phi)^{O(1)})$, where $\mathrm{Vol}\,(S)$ is the sum of the degrees of vertices in S and $\Phi(S)$, the sparsity of the cut S, is the number of edges leaving S divided by $\mathrm{Vol}\,(S)$ (See Section 2.2). For our purposes, we may apply this algorithm with $\phi = 1/\log^{O(1)} n$. This algorithm approximates the distributions of many random walks on the graph, and its analysis is based on techniques used by Lovasz and Simonovits [LS93] to analyze their volume estimation algorithm.

We would like to show that if we iteratively apply this partitioning algorithm, then each component in the remaining graph has no cut of sparsity less than $O(\phi)$. Instead, we show that each component can be embedded in a component of the original graph that has no cut of sparsity less than $O(\phi)$. The analysis of the iterative application of the cut algorithm is more complicated than one might expect, and appears in Section 4. The key to the analysis is the introduction of a differently scaled isoperimetric number, which we denote Φ.

1.1 Practicality

The algorithm as stated and analyzed is quite far from being practical. However, most of the impracticality stems from the analysis of the graph partitioning algorithm. Fortunately, algorithms that provide good partitions of graphs quickly in practice are readily available [HL94, KK98]. If these were used instead, the algorithm would probably perform much better.

More fundamentally, the key idea of this paper is that sparsifiers can be used to greatly reduce the number of edges needed to augment the spanning trees of Vaidya and [ST03]. We are aware of many other heuristics for sparsifying graphs, and expect that some will produce practically reasonable algorithms.

1.2 Prior Work: Partitioners

We are aware of three theoretically analyzable general-purpose algorithms for graph partitioning: the spectral method, the linear-programming relaxation of Leighton and Rao [LR99], and the random-walk algorithm implicit in the work of Lovasz and Simonovits [LS93]. Of these, the linear-programming based algorithm provides the the best approximation of the sparsest cut, but is by far the slowest. The spectral method partitions by computing an eigenvector of the Laplacian matrix of a graph, and produces a quadratic approximation of the sparsest cut. This algorithm can be sped up by applying the Lanczos algorithm to compute an approximate eigenvector. Given a graph with a cut of sparsity less than ϕ, this sped-up algorithm can compute a cut of sparsity at most $\sqrt{\phi}$ in time $O(n\sqrt{1/\phi})$. However, there seems to be no way to control the balance of the cut it outputs. Finally, Lovasz and Simonovits essentially show that by examining random walks in a graph, one can obtain an algorithm that produces similar cuts in time $O(n/\phi)$. To obtain maximally balanced cuts, our graph partitioning algorithm exploits rounded random walks, and our analysis builds upon the techniques of [LS93].

We remark that the most successful graph partitioning algorithms in practice are the multi-level methods incorporated into Metis [KK98] and Chaco [HL94]. However, there are still no theoretical analyses of the qualities of the cuts produced by these algorithms on general graphs.

1.3 Prior Work: Sparsifiers

The graph sparsifiers most closely related to ours are those developed by Benczur and Karger [BK96]. They develop an $O(n \log^3 n)$ time algorithm that on input a weighted graph G with Laplacian L and a parameter ϵ outputs a weighted graph $\tilde{G}$ with Laplacian $\tilde{L}$ such that $\tilde{G}$ has $O(n \log n/\epsilon)$ edges and such that for all $\boldsymbol{x} \in \{0,1\}^n$

$$\boldsymbol{x}^T \tilde{L} \boldsymbol{x} \leq \boldsymbol{x}^T L \boldsymbol{x} \leq (1+\epsilon) \boldsymbol{x}^T \tilde{L} \boldsymbol{x}. \tag{1}$$

The difference between their sparsifiers and ours is that ours apply for all $\boldsymbol{x} \in \mathbb{R}^n$. To see the difference between these two types of sparsifiers, consider the graph on vertex set $\{0, \ldots, n-1\}$ containing edges between each pair of vertices i and j such that $|(i-j)| \mod n \leq k$, and one additional edge, e, from vertex 0 to vertex $n/2$. If $\tilde{G}$ is the same graph without edge e, then (1) is satisfied with $\epsilon = 1/k$ for all $\boldsymbol{x} \in \{0,1\}^n$. However, for the vector $\boldsymbol{x} = (0, 1, 2, \ldots n/2 - 1, n/2, n/2 - 1, \ldots, 1, 0)$, (1) is not satisfied for any $\epsilon < n/4k$. Moreover, the algorithm of Benczur and Karger does not in general keep the edge e in its sparsifier. That said, some of the inspiration for our algorithm comes from the observation that we must treat sparse cuts as they treat minimum cuts.

Other matrix sparsifiers that randomly sample entries have been devised by Achlioptas and McSherry [AM01] and Frieze, Kannan and Vempala [FKV98]. The algorithm of Achlioptas and McSherry takes as input a matrix A and outputs a sparse matrix $\tilde{A}$ that satisfies inequalities analogous to (1) for all $\boldsymbol{x}$ in the range of the dominant eigenvectors of A. Similarly, if one applies the algorithm of Frieze, Kannan and Vempala to the directed edge-vertex adjacency matrix of a graph G, then one obtains a graph $\tilde{G}$ satisfying (1) for all $\boldsymbol{x}$ in the span of the few singular vectors of largest singular value. In contrast, our sparsifiers must satisfy this equation on the whole space. Again, one can observe that neither of these algorithms is likely to keep the edge e in the example above. That said, we do prove that a rounding similar to that used by Achlioptas and McSherry works for our purposes if the graph A has reasonably large isoperimetric number.

1.4 Outline

We present the graph partitioning algorithm in Section 3. In Section 4, we show how this algorithm can be used to decompose a graph into pieces, each of which is contained in a graph of relatively large isoperimetric number. In Section 5, we prove that a natural random rounding of a graph that has large isoperimetric number will be a good approximation of the original. Finally, in Section 7, we construct our sparsifiers and briefly outline how they can be applied to solving linear systems.

2 Notation and Background

2.1 Linear Systems

We recall that a matrix is diagonally dominant if $A_{i,i} \geq \sum_{j=1}^{n} |A_{i,j}|$ for all i. As explained in [ST03], the reductions introduced in [Gre96, BGH$^+$] allow us to solve SDD systems by merely preconditioning Laplacian systems. We recall that a symmetric matrix is a Laplacian if all its off-diagonals are non-positive and the sum of the entries in each row is 0. For a non-negative matrix A, we let $\mathcal{L}(A)$ denote the corresponding Laplacian.

When A is non-singular, that is when A^{-1} exists, there exists a unique solution $x = A^{-1}\boldsymbol{b}$ to the linear system. When A is singular and symmetric, for every $\boldsymbol{b} \in \mathbf{Span}\,(A)$ there exists a unique $\boldsymbol{x} \in \mathbf{Span}\,(A)$ such that $A\boldsymbol{x} = \boldsymbol{b}$. If A is the Laplacian of a connected graph, then the null space of A is spanned by $\mathbf{1}$.

There are two natural ways to formulate the problem of finding an approximate solution to a system $A\boldsymbol{x} = \boldsymbol{b}$. A vector $\tilde{\boldsymbol{x}}$ has *relative residual error* ϵ if $\|A\tilde{\boldsymbol{x}} - \boldsymbol{b}\| \leq \epsilon \|\boldsymbol{b}\|$. We say that a solution $\tilde{\boldsymbol{x}}$ is an ϵ-approximate solution if it is at relative distance at most ϵ from the actual solution—that is, if $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| \leq \epsilon \|\boldsymbol{x}\|$. One can relate these two notions of approximation by observing that relative distance of $\boldsymbol{x}$ to the solution and the relative residual error differ by a multiplicative factor of at most $\kappa_f(A)$. We will focus our attention on the problem of finding ϵ-approximate solutions.

The ratio $\kappa_f(A)$ is the finite condition number of A. The l_2 norm of a matrix, $\|A\|$, is the maximum of $\|Ax\| / \|x\|$, and equals the largest eigenvalue of A if A is symmetric. For non-symmetric matrices, $\lambda_{max}(A)$ and $\|A\|$ are typically different. We let $|A|$ denote the number of non-zero entries in A.

The condition number plays a prominent role in the analysis of iterative linear system solvers. When A is PSD, it is known that, after $\sqrt{\kappa_f(A)}\log(1/\epsilon)$ iterations, the Chebyshev iterative method and the Conjugate Gradient method produce solutions with relative residual error at most ϵ. To obtain an ϵ-approximate solution, one need merely run $\log(\kappa_f(A))$ times as many iterations. If A has m non-zero entries, each of these iterations takes time $O(m)$. When applying the preconditioned versions of these algorithms to solve systems of the form $B^{-1}A\boldsymbol{x} = B^{-1}\boldsymbol{b}$, the number of iterations required by these algorithms to produce an ϵ-accurate solution is bounded by $\sqrt{\kappa_f(A, B)}\log(\kappa_f(A)/\epsilon)$ where

$$\kappa_f(A, B) = \left(\max_{\boldsymbol{x}: A\boldsymbol{x} \neq \boldsymbol{0}} \frac{\boldsymbol{x}^T A \boldsymbol{x}}{\boldsymbol{x}^T B \boldsymbol{x}} \right) \left(\max_{\boldsymbol{x}: A\boldsymbol{x} \neq \boldsymbol{0}} \frac{\boldsymbol{x}^T B \boldsymbol{x}}{\boldsymbol{x}^T A \boldsymbol{x}} \right),$$

for symmetric A and B with $\mathbf{Span}\,(A) = \mathbf{Span}\,(B)$. However, each iteration of these methods takes time $O(m)$ plus the time required to solve linear systems in B. In our initial algorithm, we will use direct methods to solve these systems, and so will not have to worry about approximate solutions. For the recursive application of our algorithms, we will use our algorithm again to solve these systems, and so will have to determine how well we need to approximate the solution.

For this reason, we will analyze the Chebyshev iteration instead of the Conjugate Gradient, as it is easier to analyze the impact of approximation in the Chebyshev iterations. However, we expect that similar results could be obtained for the preconditioned Conjugate Gradient. For more information on these methods, we refer the reader to [GV89] or [Bru95].

For Laplacian matrices L and $\tilde{L}$ such that the nullspace of $\tilde{L}$ is contained in the nullspace of L, we recall the definition of the *support* of $\tilde{L}$ in L:

$$\sigma_f(L, \tilde{L}) = \max_{\boldsymbol{x}:\tilde{L}\boldsymbol{x}\neq 0} \frac{\boldsymbol{x}^T L \boldsymbol{x}}{\boldsymbol{x}^T \tilde{L} \boldsymbol{x}},$$

and note that for matrices L and $\tilde{L}$ with the same nullspace, we may express

$$\kappa_f(L, \tilde{L}) = \sigma_f(L, \tilde{L}) \sigma_f(\tilde{L}, L),$$

We note that

$$\sigma_f(L, \tilde{L}) \leq \lambda \text{ if and only if } \lambda L \succcurlyeq \tilde{L},$$

and that there exists a scaling factor μ such that $\mu\tilde{L}$ is an $\kappa_f(L, \tilde{L})$-approximation of L. For more information on these quantities, we refer the reader to [BH].

2.2 Cuts and Isoperimetry

Let $G = (V, E)$ be an undirected unweighted graph with n vertices and m edges. For each $S \subseteq V$, we let $G(S)$ be the induced graph on the vertices in S. We also define $\mathrm{Vol}_V(S) = \sum_{v \in S} d(v)$ where $d(v)$ is the degree of vertex v in G. We note that $\mathrm{Vol}_V(V) = 2m$.

Each subset $S \subseteq V$ defines a *cut* and hence a *partition* $(S, \bar{S})$ of G, where $\bar{S} = V - S$. Let $\partial_V(S) = E(S, \bar{S})$ be the set of edges with exactly one endpoint in S and one endpoint in $\bar{S}$.

The *sparsity* of the set is defined to be

$$\Phi_V(S) \stackrel{\text{def}}{=} \frac{|\partial_V(S)|}{\min(\mathrm{Vol}_V(S), \mathrm{Vol}_V(\bar{S}))},$$

and the *isoperimetric number* of the graph is

$$\Phi_V = \min_{S \subset V} \Phi_V(S).$$

The *balance* of a cut S or a partition $(S, \bar{S})$ where $\mathrm{Vol}_V(S) \leq \mathrm{Vol}_V(\bar{S})$ is

$$\mathbf{bal}(S) = \frac{\mathrm{Vol}_V(S)}{\mathrm{Vol}_V(V)}.$$

We also define these terms in the subgraph of G induced by a subset of the vertices $W \subseteq V$: For $S \subseteq W$,

$$\mathrm{Vol}_W(S) \stackrel{\text{def}}{=} \sum_{v \in S} |w \in W : (v, w) \in E|,$$

$$\partial_W(S) \stackrel{\text{def}}{=} \sum_{v \in S} |w \in W - S : (v, w) \in E|,$$

$$\Phi_W(S) \stackrel{\text{def}}{=} \frac{|\partial_W(S)|}{\min(\mathrm{Vol}_W(S), \mathrm{Vol}_W(\bar{S}))}.$$

When W is clear from the context we will just write $\partial(S)$, $\mathrm{Vol}(S)$ and $\Phi(S)$.

Proposition 2.1 (Monotonicity of Volume and Boundary). *If* $W_1 \subset W_2$ *then* $\mathrm{Vol}_{W_1}(S) \leq \mathrm{Vol}_{W_2}(S)$ *and* $\partial_{W_1}(S) \leq \partial_{W_2}(S)$.

Proposition 2.2. *If* $S \subset V$*, then*

$$\mathrm{Vol}_{V-S}(V-S) \geq \mathrm{Vol}_V(V) - 2\mathrm{Vol}_V(S).$$

The following lemma shows when $\mathrm{Vol}_W(W)/\mathrm{Vol}_V(V)$ is large, $\mathrm{Vol}_W(S)$ is a close approximation of $\mathrm{Vol}_V(S)$.

Proposition 2.3. *If* $S \subseteq W \subseteq V$*, then*

$$\mathrm{Vol}_V(S) \leq \mathrm{Vol}_W(S) + \mathrm{Vol}_V(V-W).$$

2.3 Multiway Partitioning

$\mathcal{C} = \{C_1, \ldots, C_k\}$ is a *multiway partition* of G if $(C_1, \ldots, C_k)$ is a partition of V and for all $i \in [1,k]$, the induced graph of C_i is connected. We call C_i a *component* in the partition $\mathcal{C}$ of G. The *cut-size* of $\mathcal{C}$, $\textbf{cut-size}(\mathcal{C})$, is defined to be the number of edges whose endpoints are in different components in $\mathcal{C}$.

We call $\mathcal{C}$ a (θ, θ_*)-multiway-partition of G if $\textbf{cut-size}(\mathcal{C}) \leq \theta m$ and for all $i \in [1,k]$, $\Phi_{C_i} \geq \theta_*$.

3 Nearly Linear-Time Graph Partitioning

The main result of this section is a nearly linear-time algorithm `Partition` that computes an approximate sparsest cut with approximately optimal balance: for any graph $G = (V, E)$ that has a cut S of sparsity ϕ and balance $b \leq 1/2$, with high probability, `Partition` finds a cut D with $\Phi_V(D) = \phi^{1/3} \log^{O(1)} n$ and $\textbf{bal}(D) \geq b/2$. Actually, `Partition` satisfies an even stronger guarantee: with high probability either the cut is well balanced:

$$(5/12)\mathrm{Vol}_V(V) \leq \mathrm{Vol}_V(D) \leq (5/6)\mathrm{Vol}_V(V),$$

or touches most of the edges touching S:

$$\mathrm{Vol}_{V-D}((V-D) \cap S) \leq \mathrm{Vol}_V(S)/2.$$

The running time of `Partition` is $m \log^{O(1)} n/\phi^5$. Thus, it can be used to quickly find crude cuts.

The algorithm `Partition` works by repeatedly calling a partitioning subroutine called `Nibble` that essentially runs in time proportional to the number of edges in the cut that it finds. In particular, `Nibble` takes as input a seed vertex v and a target cut-volume 2^b. For most $v \in S$, there is a target cut-volume 2^b such that `Nibble` will output a cut of volume at least $(5/14)2^b$, most of whose vertices are in S. Moreover, `Nibble` runs in time at most $O(2^b(\log n/\phi)^{O(1)})$.

`Partition` makes its calls to `Nibble` via a routine called `Random Nibble` that calls `Nibble` with carefully chosen random parameters. `Random Nibble` has a very small expected running time, and is expected to remove a similarly small fraction of any set with small isoperimetic number.

The algorithm `Nibble` works by approximately computing the distribution of a few steps of the random walk starting at the seed vertex v. It is implicit in the analysis of the volume estimation algorithm of Lovasz and Simonovits [LS93] that one can find a small cut from the distributions of the steps of the random walk starting at any vertex from which the walk does not mix rapidly. We first note that a random vertex in S is probably such a vertex. We then extend the analysis of Lovasz and Simonovits to show one can find a small cut from approximations of these distributions, and that these approximations can be computed quickly. In particular, we will truncate all small probabilities that appear in the distributions to 0. In this way, we minimize the work required to compute our approximations.

We will use the definitions of the following two vectors:

$$\chi_S(x) = \begin{cases} 1 & \text{for } x \in S, \\ 0 & \text{otherwise,} \end{cases}$$

$$\psi_S(x) = \begin{cases} d(x)/\mathrm{Vol}_V(S) & \text{for } x \in S, \\ 0 & \text{otherwise.} \end{cases}$$

We note that ψ_V is the steady-state distribution of the random walk, and that ψ_S is the restriction of that walk to the set S.

Given an unweighted graph A, we will consider the walk that at each time step stays put with probability $1/2$, and otherwise moves to a random neighbor of the current vertex. The matrix realizing this walk can be expressed $P = (AD^{-1} + I)/2$, where $d(i)$ is the degree of node i, and D is the diagonal matrix with $(d(1), \ldots, d(n))$ on the diagonal. We will let p_t^v denote the distribution obtained after t steps of the random walk starting at vertex v. In this notation, we have $p_t^v = P^t \chi_v$. We will omit v when it is understood. For convenience, we introduce the notation

$$\rho_t^v(x) = p_t^v(x)/d(x).$$

To describe the rounded random walks, we introduce the truncation operation

$$[p]_\epsilon(v) = \begin{cases} p(v) & \text{if } p(v) \geq 2\epsilon d(i), \\ 0 & \text{otherwise.} \end{cases}$$

We then have the truncated probability vectors

$$\begin{aligned} \tilde{p}_0 &= p_0 \\ \tilde{p}_t &= [P\tilde{p}_{t-1}]_\epsilon . \end{aligned} \tag{2}$$

That is, at each time step, we will evolve the random walk one step from the current density, and then round every $p_t(i)$ that is less than $2d(i)\epsilon$ to 0. We remark that this will result in an odd situation in which the sum of the probabilities that we are carrying around will be less than 1.

$C = \texttt{Nibble}(G, v, \theta_0, b)$
where v is a vertex
$0 < \theta_0 < 1$
b is a positive integer.

(1) Set $\tilde{p}_0(x) = \chi_v$.

(2) Set $t_0 = 49 \ln(me^4)/\theta_0^2$, $\gamma = \frac{5\theta_0}{7 \cdot 7 \cdot 8 \ln(me^4)}$, and $\epsilon_b = \frac{\theta_0}{7 \cdot 8 \ln(me^4) t_0 2^b}$.

(3) For $t = 1$ to t_0

   (a) Set $\tilde{p}_t = [P \tilde{p}_{t-1}]_\epsilon$.

   (b) Compute a permutation $\tilde{\pi}_t$ such that $\tilde{\rho}_t(\tilde{\pi}_t(i)) \geq \tilde{\rho}_t(\tilde{\pi}_t(i+1))$ for all i.

   (c) If there exists a $\tilde{j}$ such that

      i $\Phi(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\})) \leq \theta_0$,

      ii $\tilde{\rho}_t(\tilde{\pi}_t(\tilde{j})) \geq \gamma / \mathrm{Vol}_V(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\}))$, and

      iii $5 \mathrm{Vol}_V(V)/6 \geq \mathrm{Vol}(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\})) \geq (5/7) 2^{b-1}$.

      then output $C = \tilde{\pi}_t(\{1, \ldots, \tilde{j}\})$ and quit.

(4) Return $\emptyset$.

We will use the following notation.

$$\theta_+ \stackrel{\text{def}}{=} \frac{\theta_0^3}{14^4 \ln^2(me^4)}. \tag{3}$$

In Section 3.2, we will prove the following lemma on the performance of `Nibble`.

Lemma 3.1 (`Nibble`). *`Nibble` can be implemented so that on all inputs it runs in time $O\left(2^b \ln^4(m)/\theta_0^5\right)$. If the set C output by `Nibble` is non-empty, it satisfies*

i. $\Phi_V(C) \leq \theta_0$,

ii. $\mathrm{Vol}_V(C) \leq (5/6) \mathrm{Vol}_V(V)$.

Moreover, for each $\theta_0 \leq 1/2$ and for each set S satisfying

$$\mathrm{Vol}_V(S) \leq (2/3) \mathrm{Vol}_V(V) \quad \textit{and} \quad \Phi_V(S) \leq 2\theta_+,$$

there is a subset $S^g \subseteq S$ such that $\mathrm{Vol}_V(S^g) \geq \mathrm{Vol}_V(S)/2$ that can be decomposed into sets S_b^g for $b = 1, \ldots, \lg m$ such that if `Nibble` is started from a vertex $v \in S_b^g$ and run with parameters θ_0 and b, then it will output a set of vertices C such that

iii. $(4/7) 2^{b-1} \leq \mathrm{Vol}_V(C \cap S)$.

To define `Partition`, we first define an intermediate algorithm `Random Nibbles` which calls `Nibble` on carefully chosen random inputs.

$C = \texttt{RandomNibble}(G, \theta_0)$

(1) Choose a vertex v according to ψ_V.

(2) Choose a b in $1, \ldots, \lceil \log m \rceil$ according to

$$\Pr[b = i] = 2^{-i}/(1 - 2^{-\lceil \log m \rceil}).$$

(3) $C = \texttt{Nibble}(G, v, \theta_0, b)$.

Lemma 3.2 (Random Nibble). *The expected running time of **Random Nibble** is $O\left(\ln^4(m)/\theta_0^5\right)$. If the set C output by **Random Nibble** is non-empty, it satisfies*

i. $\Phi_V(C) \leq \theta_0$,

ii. $\mathrm{Vol}_V(C) \leq (5/6)\mathrm{Vol}_V(V)$.

Moreover, for each $\theta_0 \leq 1/2$ and for each set S satisfying

$$\mathrm{Vol}_V(S) \leq (2/3)\mathrm{Vol}_V(V) \quad \textit{and} \quad \Phi_V(S) \leq 2\theta_+,$$

iii. $\mathbf{E}[\mathrm{Vol}(C \cap S)] \geq \mathrm{Vol}(S)/14m$,

where θ_+ is as defined in (3).

Proof. The expected running time of `Random Nibble` may be upper bounded by

$$O\left(\sum_{i=1}^{\lceil \log m \rceil} \left(2^{-i}/(1 - 2^{\lceil \log m \rceil})\right)\left(2^i \ln^4(m)/\theta_0^5\right)\right) = O\left(\ln^4(m)/\theta_0^5\right).$$

Properties i and ii of C follow directly from parts i and ii of Lemma 3.1. To prove part iii, define α_b by

$$\mathrm{Vol}_V\left(S_b^g\right) = \alpha_b \mathrm{Vol}_V(S^g).$$

So, $\sum_b \alpha_b = 1$. For each b, the chance that v lands in S_b^g is $\alpha_b \mathrm{Vol}_V(S^g)/2m$. If v lands in S_b^g, then by part iii of Lemma 3.1, C satisfies

$$\mathrm{Vol}_V(C \cap S) \geq (4/7)2^{b-1}.$$

So,

$$\begin{aligned}
\mathbf{E}[\mathrm{Vol}_V(C \cap S)] &\geq \sum_i 2^{-i}\alpha_i (\mathrm{Vol}_V(S^g)/2m)(4/7)2^{i-1} \\
&= \sum_i (4/14)\alpha_i (\mathrm{Vol}_V(S^g)/2m) \\
&= \mathrm{Vol}_V(S^g)/7m. \qquad\qquad \geq \mathrm{Vol}_V(S^g)/14m.
\end{aligned}$$

$\square$

We now define `Partition` and analyze its performance.

> $D = \texttt{Partition}(G, \theta_0, p)$, where G is a graph, $\theta_0, p \in (0,1)$.
>
> (0) Set $W_1 = V$.
>
> (1) For $j = 1$ to $56m\lceil \lg(1/p) \rceil$.
>
> (a) Set $D_j = \texttt{RandomNibble}(G(W_j), \theta_0)$
>
> (b) Set $W_{j+1} = W_j - D_j$.
>
> (c) If $\mathrm{Vol}_{W_{j+1}}(W_{j+1}) \leq (5/6)\mathrm{Vol}_V(V)$, then go to step (2).
>
> (2) Set $D = V - W_{j+1}$.

$$\theta_0 \stackrel{\text{def}}{=} (5/36)\theta. \tag{4}$$

Theorem 3.3 (`Partition`). *The expected running time of `Partition` is at most* $O\left(m \lg(1/p) \ln^4(m)/\theta_0^5\right)$. *Let D be the output of* $\texttt{Partition}(G, \theta_0, p)$, *where G is a graph and $\theta_0, p \in (0,1)$. Then*

i. $\mathrm{Vol}_D(D) \leq (31/36)\mathrm{Vol}_V(V)$,

ii. $\Phi_V(D) \leq \theta$, *as defined in* (4).

Moreover, for each set S satisfying

$$\mathrm{Vol}_V(S) \leq (2/3)\mathrm{Vol}_V(V) \quad \textit{and} \quad \Phi_V(S) \leq 2\theta_+,$$

with probability at least $1-p$, either

iii.a. $\mathrm{Vol}_{V-D}(V-D) \leq (5/6)\mathrm{Vol}_V(V)$, *or*

iii.b. $\mathrm{Vol}_{V-D}(S \cap (V-D)) \leq (1/2)\mathrm{Vol}_V(S)$.

where θ_+ is as defined in (3).

We say that `Partition` *succeeds* if one of case *iii.a* or *iii.b* occurs.

Our proof of Theorem 3.3 will use the following probabilistic lemma:

Lemma 3.4. *Let $X_1, \ldots, X_{ak}$ be non-negative random variables and let $E_1, \ldots, E_{ak}$ be events such that*

$$\mathbf{E}\left[X_{i+1} \middle| \sum_{k=1}^{i} X_k < 1 \textit{ and } E_i \right] \geq 1/k,$$

Then,

$$\Pr\left[\sum_{i=1}^{ak} X_i < 1 \textit{ and } \bigwedge_i E_i \right] < 2^{-a/2}.$$

Proof. We first prove the lemma in the case $a = 2$. In this case, we define the random variable

$$Y_i = \begin{cases} X_i & \text{if } E_i \text{ and } \sum_{k=1}^{i} X_k < 1 \\ 1/k & \text{otherwise.} \end{cases}$$

We note that $\sum Y_i \geq 1$ implies either $\sum X_i \geq 1$ or E_i is false for some i. So, it suffices to lower bound the probability that $\sum Y_i \geq 1$. To this end, we note that

$$\mathbf{E}\left[\sum Y_i\right] \geq 2, \text{ and}$$
$$\sum Y_i \leq 3.$$

Thus, we may conclude that

$$\Pr\left[\sum Y_i \geq 1\right] \geq 1/2,$$

for otherwise

$$\mathbf{E}\left[\sum Y_i\right] < 1/2 + 3/2 = 2,$$

which would be a contradiction. The proof for general a now follows by applying this argument to each of the $a/2$ consecutive blocks of $2k$ variables. $\square$

Proof of Theorem 3.3. The bound on the expected running time of `Partition` is immediate from the bound on the running time of `RandomNibble`.

Let j be such that $D = V - W_{j+1}$. To prove (i), we let $C = \cup_{i=1}^{j-1}$. So, we have $W_j = V - C$, $W_{j+1} = W_j - D_j$ and $D = D_j \cup C$. We then compute

$$\begin{aligned}
\mathrm{Vol}_{D_j \cup C}\left(D_j \cup C\right) &\leq \mathrm{Vol}_V\left(C\right) + \mathrm{Vol}_{V-C}\left(D_j\right) \\
&\leq \mathrm{Vol}_V\left(V\right) - \mathrm{Vol}_{V-C}\left(V - C\right) + \mathrm{Vol}_{V-C}\left(D_j\right) \\
&\leq \mathrm{Vol}_V\left(V\right) - (1/6)\mathrm{Vol}_{V-C}\left(V - C\right), \text{by Lemma 3.2, } ii \\
&\leq \mathrm{Vol}_V\left(V\right) - (5/36)\mathrm{Vol}_V\left(V\right),
\end{aligned}$$

as $\mathrm{Vol}_{W_j}\left(W_j\right) \geq (5/6)\mathrm{Vol}_V\left(V\right)$.

To prove ii, we note that if $\mathrm{Vol}_V\left(D\right) \leq \mathrm{Vol}_V\left(V\right)/2$, then

$$\begin{aligned}
\Phi_V(D) &= \frac{\partial_V\left(C \cup D_j\right)}{\mathrm{Vol}_V\left(C \cup D_j\right)} \\
&\leq \frac{\partial_V\left(C\right) + \partial_{V-C}\left(D_j\right)}{\mathrm{Vol}_V\left(C\right) + \mathrm{Vol}_V\left(D_j\right)} \\
&\leq \max\left(\frac{\partial_V\left(C\right)}{\mathrm{Vol}_V\left(C\right)}, \frac{\partial_{V-C}\left(D_j\right)}{\mathrm{Vol}_V\left(D_j\right)}\right) \\
&\leq \max\left(\frac{\partial_V\left(C\right)}{\mathrm{Vol}_V\left(C\right)}, \frac{\partial_{V-C}\left(D_j\right)}{\mathrm{Vol}_{V-C}\left(D_j\right)}\right) \\
&\leq \theta_0,
\end{aligned}$$

by Lemma 3.2.

On the other hand, we established above that $\mathrm{Vol}_V(D) \leq (31/36)\mathrm{Vol}_V(V)$, from which it follows that $\mathrm{Vol}_V(V-D) \geq (5/36)\mathrm{Vol}_V(V)$. Thus,

$$\Phi_V(D) \leq \frac{\partial_V(D)}{\mathrm{Vol}_V(V-D)} \leq \frac{36}{5}\frac{\partial_V(D)}{\mathrm{Vol}_V(D)} \leq \frac{36\theta_0}{5}.$$

To prove part *iii*, we will image what happens if we run `Partition` for all $56m\lceil \lg(1/p) \rceil$ iterations. We will apply Lemma 3.4 with

$$X_i = 2\mathrm{Vol}_{W_i}(D_i \cap S \cap W_i) / \mathrm{Vol}_V(S), \tag{5}$$

and

$$E_i = \begin{cases} 1 & \text{if } \mathrm{Vol}_{W_i}(W_i) \leq (5/6)\mathrm{Vol}_V(V) \text{, and} \\ 0 & \text{otherwise.} \end{cases}$$

As

$$\mathrm{Vol}_V(S) - \mathrm{Vol}_{V-D}((V-D) \cap S) = \sum_{i=1}^{j} X_i,$$

it suffices to lower bound the probability that

$$\sum_{i=1}^{j} X_i \geq 1.$$

If $\sum_{k=1}^{i-1} X_k < 1$, then

$$\mathrm{Vol}_{W_i}(S \cap W_i) \geq \mathrm{Vol}_V(S)/2.$$

As $\mathrm{Vol}_V(S) \leq 2\mathrm{Vol}_V(V)/3$, if E_i is true then

$$\mathrm{Vol}_{W_i}(W_i - S) \geq \mathrm{Vol}_V(V-S)/2.$$

As $\partial_{W_{i+1}}(S) \leq \partial_V(S)$, we find that E_i and $\sum_{k=1}^{i} X_k < 1$ implies

$$\Phi_{W_{i+1}}(S) \leq 2\Phi_V(S).$$

By Lemma 3.2, we then have

$$\mathbf{E}\left[X_{i+1} \middle| E_i \text{ and } \sum_{k=1}^{i} X_k < 1\right] \geq \frac{\mathrm{Vol}_{W_i}(S \cap W_i)}{7\mathrm{Vol}_{W_i}(W_i)} \geq 1/28m.$$

Thus, the theorem now follows from Lemma 3.4.

□

3.1 Cuts From Random Walks

Before analyzing the cuts obtained from rounded random walks in `Nibble`, we analyze the cuts obtained from ordinary random walks. Our analysis will reveal that this cut algorithm is resistant to some perturbation of the distributions.

As in the previous section, we let $p_t(v_i)$ denote the probability that the random walk is at vertex v_i at time t and define $\rho_t(v) = p_t(v)/d(v)$. Let π_t denote an ordering on the vertices so that $\rho_t(\pi_t(1)) \geq \rho_t(\pi_t(2)) \geq \cdots$. For all integers $j \in [0, n]$, we define

$$k_j^t = \sum_{i=1}^{j} d(\pi_t(i)),$$

and

$$H_t(k_j^t) = \sum_{i=1}^{j} p_t(\pi_t(i)) - d(\pi_t(i))/2m.$$

For general $x \in [0, 2m]$, we define $H_t(x)$ to be piecewise-linear between these points. That is, for $k_{j-1}^t < x < k_j^t$, if $x = \alpha k_{j-1}^t + (1-\alpha)k_j^t$, we set $H_t(x) = \alpha H_t(k_{j-1}^t) + (1-\alpha)H_t(k_j^t)$. Note that

$$H_t'(x) = \rho_t(\pi_t(j)) - 1/2m. \tag{6}$$

We remark that, up to rescaling, this definition agrees with the definition of $h_t(x)$ used by Lovasz and Simonovits [LS90, LS93].

Our analysis of the random walk will make use of the following definition.

Definition 3.5 (S^g). *For each set $S \subseteq V$, we define S^g to be the set of nodes x in S such that*

$$\left\langle \chi_{\bar{S}} | P^{t_0} \chi_x \right\rangle \leq 2 \left\langle \chi_{\bar{S}} | P^{t_0} \psi_S \right\rangle.$$

We have

Proposition 3.6 (Mass of S^g).

$$\mathrm{Vol}_V\left(S^g\right) \geq \mathrm{Vol}_V\left(S\right)/2.$$

Proof. By linearity, we have

$$\begin{aligned}
\left\langle \chi_{\bar{S}} | P^{t_0} \psi_S \right\rangle &= \sum_{x \in S} \frac{d(x)}{\mathrm{Vol}_V\left(S\right)} \left\langle \chi_{\bar{S}} | P^{t_0} \chi_x \right\rangle \\
&> \sum_{x \notin S^g} \frac{d(x)}{\mathrm{Vol}_V\left(S\right)} 2 \left\langle \chi_{\bar{S}} | P^{t_0} \psi_S \right\rangle \\
&= \frac{\mathrm{Vol}_V\left(S\right) - \mathrm{Vol}_V\left(S^g\right)}{\mathrm{Vol}_V\left(S\right)} 2 \left\langle \chi_{\bar{S}} | P^{t_0} \psi_S \right\rangle.
\end{aligned}$$

So, we may conclude

$$\frac{\mathrm{Vol}_V\left(S\right) - \mathrm{Vol}_V\left(S^g\right)}{\mathrm{Vol}_V\left(S\right)} < \frac{1}{2},$$

from which the lemma follows. ∎

The purpose of this section is to prove:

Lemma 3.7 (Cut from Random Walk). *For any $\phi > 0$, let $t_0 = \ln(me^4)/\phi^2$, $\alpha = 1/4t_0$. Then, for every set S such that $\mathrm{Vol}(S) \leq \mathrm{Vol}(V)/2$ and*

$$\Phi(S)t_0 < 1/32,$$

for all $v \in S^g$, if we start the random walk at χ_v, then there exists a $t < t_0$ and a $j \leq m$ such that

(a) $\Phi(\pi_t(\{1, \ldots, j\}) \leq \phi$, and

(b) for the j_0 and j_1 defined by $k^t_{j_0-1} < k^t_j - 2\phi\bar{k}^t_j \leq k^t_{j_0}$ and $k^t_{j_1-1} < k^t_j + 2\phi\bar{k}^t_j \leq k^t_{j_1}$,

$$\rho_t(\pi_t(j_0)) - \rho_t(\pi_t(j_1)) > \frac{\phi}{4\ln(me^4)\mathrm{Vol}\left(\pi_t\left(\{1, \ldots, j\}\right)\right)}, \tag{7}$$

where we define $\bar{k}^t_j = \min(k^t_j, 2m - k^t_j)$.

Condition (b) tells us that even if the values of ρ_t are altered slightly, the cut will not change too much.

The proof will make use of the following two lemmas, the first of which can be derived from the proof of Lemma 1.4 in [LS90], and the second of which is a simple extension of an idea used in the proof of Theorem 1.4 of [LS93].

Lemma 3.8 (Lovasz-Simonovits). *For all $j \in [1, n-1]$ such that*

$$\Phi\left(\pi_t(\{1, \ldots, j\})\right) \geq \phi,$$

we have

$$H_t(k^t_j) \leq \frac{1}{2}\Big(H_{t-1}(k^t_j - 2\phi\bar{k}^t_j) + H_{t-1}(k^t_j + 2\phi\bar{k}^t_j)\Big),$$

where we define $\bar{k}^t_j = \min(k^t_j, 2m - k^t_j)$. Moreover, for all $x \in [0, 2m]$,

$$H_t(x) \leq H_{t-1}(x). \tag{8}$$

Lemma 3.9. *If, for all $x \in [0, 2m]$ and $t \leq t_0$*

$$H_0(x) \leq \sqrt{\bar{x}},$$

$$H_t(x) \leq \frac{1}{2}\Big(H_{t-1}(x - 2\phi\bar{x}) + H_{t-1}(x + 2\phi\bar{x})\Big) + \alpha,$$

then for all $x \in [0, 2m]$,

$$H_{t_0}(x) \leq \sqrt{\bar{x}}\left(1 - \frac{\phi^2}{2}\right)^{t_0} + \alpha t_0,$$

where we let $\bar{x} = \min(x, 2m - x)$.

Proof. For the base case, we observe that

$$H_0(x) \leq \min(\sqrt{x}, \sqrt{m-x}).$$

We now assume by way of induction that

$$H_{t-1}(x) \leq \min(\sqrt{x}, \sqrt{m-x}) \left(1 - \frac{\phi^2}{2}\right)^{t-1} + \alpha(t-1).$$

Assume that $x \leq m$. In this case, it suffices to show that

$$\begin{aligned}
&\left(\sqrt{x - 2\phi x} + \sqrt{\min(x + 2\phi x, 2m - x - 2\phi x)}\right)/2 \\
&\leq (\sqrt{x - 2\phi x} + \sqrt{x + 2\phi x})/2 \\
&\leq \sqrt{x}\left(1 - \frac{2\phi}{2} - \frac{(2\phi)^2}{8} - \frac{(2\phi)^3}{16} + 1 + \frac{2\phi}{2} - \frac{(2\phi)^2}{8} + \frac{(2\phi)^3}{16}\right)
\end{aligned}$$

(by examination of the Taylor series)

$$\leq \sqrt{x}\left(1 - \frac{\phi^2}{2}\right).$$

∎

If we merely wanted to establish condition (a) of Lemma 3.7, it would suffice to apply Lemma 3.8 and then Lemma 3.9 with $\alpha = 0$. So that we may also establish condion (b), we intead use Lemma 3.8 and Lemma 3.9 to prove

Lemma 3.10 (Cut or Mix). *For all $\alpha \geq 0$, either there exists a $t < t_0$ and a $j \in [0, n]$ such that,*

(a) $\Phi\left(\pi_t(\{1, \ldots, j\}\right) \leq \phi$, *and*

(b) $H_t(k_j^t) - \frac{1}{2}\left(H_t(k_j^t - 2\phi\bar{k}_j^t) + H_t(k_j^t + 2\phi\bar{k}_j^t)\right) \geq \alpha$,

or

$$H_{t_0}(x) \leq \sqrt{\bar{x}}\left(1 - \phi^2/2\right)^{t_0} + \alpha t_0, \tag{9}$$

for all $x \in [0, 2m]$, where we define $\bar{x} = \min(x, 2m - x)$ and $\bar{k}_j^t = \min(k_j^t, 2m - k_j^t)$.

To convert condition (b) of Lemma 3.10 to condition (b) of Lemma 3.7, we apply the following lemma.

Lemma 3.11. *If*

$$H_t(k) - \frac{1}{2}\left(H_t(k - 2\phi\bar{k}) + H_t(k + 2\phi\bar{k})\right) \geq \alpha,$$

where $\bar{k} = \min(k, 2m - k)$ then

$$H'_t(k - 2\phi\bar{k}) - H'_t(k + 2\phi\bar{k}) > \frac{\alpha}{\phi k}.$$

We now prove these lemmas, and conclude with the proof of Lemma 3.7.

Proof of Lemma 3.10. We consider what happens if one of (a) or (b) fails to hold. If (a) does not hold for j, then Lemma 3.8 implies

$$H_t(k_j^t) \leq \frac{1}{2}\Big(H_{t-1}(k_j^t - 2\phi\bar{k}_j^t) + H_{t-1}(k_j^t + 2\phi\bar{k}_j^t)\Big).$$

If (b) does not hold for j, then

$$\begin{aligned} H_t(k_j^t) &\leq \frac{1}{2}\Big(H_t(k_j^t - 2\phi\bar{k}_j^t) + H_t(k_j^t + 2\phi\bar{k}_j^t)\Big) + \alpha \\ &\leq \frac{1}{2}\Big(H_{t-1}(k_j^t - 2\phi\bar{k}_j^t) + H_{t-1}(k_j^t + 2\phi\bar{k}_j^t)\Big) + \alpha \end{aligned}$$

by (8). Thus, if (a) or (b) fails to hold for j we have

$$H_t(k_j^t) \leq \frac{1}{2}\Big(H_{t-1}(k_j^t - 2\phi\bar{k}_j^t) + H_{t-1}(k_j^t + 2\phi\bar{k}_j^t)\Big) + \alpha.$$

As H_{t-1} is convex and H_t is piece-wise linear between those k_j^t considered in the previous statement, we obtain that for all $x \in [0, 2m]$

$$H_t(x) \leq \frac{1}{2}\Big(H_{t-1}(x - 2\phi\bar{x}) + H_{t-1}(x + 2\phi\bar{x})\Big) + \alpha,$$

Thus, (9) now follows from Lemma 3.9. ∎

Proof of Lemma 3.11. As H_t is convex, $H'(k - 2\phi k)$ is at least the slope of the line from the point $(k - 2\phi k, H(k - 2\phi k))$ to the point $(k, H(k))$, which by assumption is at least

$$\frac{\alpha + \frac{1}{2}H_t(k - 2\phi\bar{k}) + \frac{1}{2}H_t(k + 2\phi\bar{k}) - H(k - 2\phi\bar{k})}{2\phi\bar{k}} = \frac{\alpha + \frac{1}{2}H_t(k + 2\phi\bar{k}) - \frac{1}{2}H_t(k - 2\phi\bar{k})}{2\phi\bar{k}}.$$

Similarly, we find that $H'(k + 2\phi\bar{k})$ is at most

$$\frac{-\alpha + \frac{1}{2}H_t(k + 2\phi\bar{k}) - \frac{1}{2}H_t(k - 2\phi\bar{k})}{2\phi\bar{k}}.$$

So, the difference is at least $\alpha/\phi\bar{k}$. ∎

Proof of Lemma 3.7. We will derive this from Lemma 3.10. We begin by showing that (9) is not satisfied.

As $v \in S^g$, we have

$$\left\langle \chi_{\bar{S}} | P^t p_0(v) \right\rangle \leq 2 \left\langle \chi_{\bar{S}} | P^t \psi_S \right\rangle \leq 1/16,$$

for all $t \leq t_0$. Thus,

$$H_{t_0}(\mathrm{Vol}\,(S)) \leq 15/16 - \mathrm{Vol}\,(S)\,/\mathrm{Vol}\,(V) \leq 15/16 - 1/2 = 7/16.$$

On the other hand,

$$\begin{aligned}\min(\sqrt{x}, \sqrt{m-x})\left(1-\phi^2/2\right)^{t_0} + \alpha t_0 &\le \sqrt{m} e^{-t_0 \phi^2/2} + 1/4 \\ &\le e^{-\frac{1}{2}\left(t_0\phi^2 - \ln m\right)} + 1/4 \\ &\le e^{-\frac{1}{2}\left(t_0\phi^2 - \ln m\right)} + 1/4 \\ &\le e^{-2} + 1/4 \\ &< 7/16.\end{aligned}$$

Thus, by Lemma 3.10, there must exist a $t < t_0$ and j such that

(a) $\phi\left(\pi_t(\{1,\dots,j\}\right) \le \phi$, and

(b) $H_t(k_j^t) - \frac{1}{2}\Big(H_t(k_j^t - 2\phi\bar{k}_j^t) + H_t(k_j^t + 2\phi\bar{k}_j^t)\Big) \ge \alpha$.

Moreover, by applying Lemma 3.11 to (b), we obtain

$$H'(k_j^t - 2\phi\bar{k}_j^t) - H'(k_j^t + 2\phi\bar{k}_j^t) > \frac{\alpha}{\phi\bar{k}_j^t} \ge \frac{\alpha}{\phi k_j^t}.$$

If we now choose j_0 and j_1 as described in part (b) of the theorem, by Equation (6), we obtain

$$\rho_t(\pi_t(j_0)) - \rho_t(\pi_t(j_1)) = H'(k_j^t - 2\phi\bar{k}_j^t) - H'(k_j^t + 2\phi\bar{k}_j^t) > \frac{\alpha}{\phi k_j^t} = \frac{\phi}{4\ln(me^4)\mathrm{Vol}\left(\pi_t\left(\{1,\dots,j\}\right)\right)}.$$

□

3.2 Analysis of `Nibble`

There are two steps to our analysis of `Nibble`: we must show that the cut produced has large overlap with S, and we must show that the truncation of the distributions of the random walks does not cause the quality of the cuts obtained to decrease too much. Both of these steps are accomplished using the following fundamental fact:

Proposition 3.12 (Monotonicity of Mult by P). *For all non-negative vectors p,*

$$\left\|D^{-1}(Pp)\right\|_\infty \le \left\|D^{-1}p\right\|_\infty.$$

Proof. Applying the transformation $r = D^{-1}p$, we see that it is equivalent to show that for all r

$$\left\|D^{-1}PDr\right\|_\infty \le \|r\|_\infty.$$

To prove this, we note that $D^{-1}PD = D^{-1}(AD^{-1}+I)D/2 = P^T$, and all the sum of the entries in each row of this matrix is 1. □

We now show that on input a seed $v \in S^g$, most of the volume of the cut produced by `Nibble` is contained in S. The proof will follow from the next proposition, which points out that if a random walk is started from the distribution ψ_S, then the amount of probability mass leaving S is bounded by the product of $\Phi(S)$ with the number of steps taken.

Proposition 3.13 (Escaping Mass).

$$\left\langle \chi_S | P^{t_0} \psi_S \right\rangle \geq 1 - t_0 \Phi_V(S).$$

Proof. We first note that $\left\| D^{-1}\psi_S \right\|_\infty = 1/\mathrm{Vol}_V(S)$, and so by Proposition 3.12 $\left\| D^{-1}P^t\psi_S \right\|_\infty \leq 1/\mathrm{Vol}_V(S)$ for all t. Thus, the amount of probability mass escaping S in each time step is at most $(1/\mathrm{Vol}_V(S))\, |\partial_V(S)| = \Phi_V(S)$. $\square$

Lemma 3.14 (Overlap with S). *For a set S for which*

$$\Phi(S) \leq \frac{\theta_0^3}{7^4 \cdot 8 \ln^2(me^4)}, \tag{10}$$

if the truncated random walk (2) is started from any vertex $v \in S^g$, then for every $t < t_0 = 7^2 \ln(me^4)/\theta_0^2$ and $\tilde{j}$ satisfying

$$\tilde{\rho}_t(\tilde{j}) \geq \frac{5\theta_0}{7^2 \cdot 8 \ln(me^4) \mathrm{Vol}\left(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\})\right)},$$

we have

$$\mathrm{Vol}\left(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\}) \cap S\right) \geq (4/5)\mathrm{Vol}\left(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\})\right).$$

Proof. Assume by way of contradiction that

$$\mathrm{Vol}\left(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\}) \cap \bar{S}\right) > \mathrm{Vol}\left(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\})\right)/5.$$

Then,

$$\begin{aligned}
\sum_{x \notin S} p_t(x) &\geq \sum_{x \notin S} \tilde{p}_t(x) \\
&= \sum_{x \notin S} d(x) \tilde{\rho}_t(x) \\
&\geq \sum_{x \in \bar{S} \cap \tilde{\pi}_t(\{1, \ldots, \tilde{j}\})} d(x) \tilde{\rho}_t(x) \\
&\geq \frac{5\theta_0}{7^2 \cdot 8 \ln(me^4) \mathrm{Vol}\left(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\})\right)} \sum_{x \in \bar{S} \cap \tilde{\pi}_t(\{1, \ldots, \tilde{j}\})} d(x) \\
&\geq \frac{5\theta_0}{7^2 \cdot 8 \ln(me^4) \mathrm{Vol}\left(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\})\right)} \mathrm{Vol}\left(\tilde{\pi}_t(\{1, \ldots, \tilde{j}\})\right)/5 \\
&= \frac{\theta_0}{7^2 \cdot 8 \ln(me^4)}.
\end{aligned} \tag{11}$$

However, by Proposition 3.13,

$$\sum_{x \notin S} p_t(x) < t_0 \Phi(S) < (11),$$

by (10). $\square$

We now establish that the distributions produced by the truncated random walk do not differ too much from those produced by the standard random walk.

Lemma 3.15 (Low-impact Truncation). *Let the values $\rho_t(v)$ be derived from the ordinary random walk and the values $\tilde{\rho}_t(v)$ be derived from the truncated random walk with truncation factor ϵ_b. Then, for all t and v,*

$$\rho_t(v) \geq \tilde{\rho}_t(v) \geq \rho_t(v) - 2t\epsilon_b.$$

Proof. The left-hand inequality is trivial. To prove the right-hand inequality, we consider $p_t - [p_t]_\epsilon$, and observe that by definition

$$\left\| D^{-1} \left(p_t - [p_t]_\epsilon \right) \right\|_\infty \leq 2\epsilon_b.$$

The inequality now follows from Proposition 3.12. □

We now define the partition of S^g claimed to exist in Lemma 3.1, and show that the cut produced by `Nibble` does not differ too much from the cut that would be been produced from the untruncated random walk.

Definition 3.16 (S_b^g). *For every set $S \subseteq V$, we define S_b^g to be the set of vertices in S^g such that when the random walk is started at that vertex, the first t for which there is a j satisfying conditions (a) and (b) of Lemma 3.7 has the property that for the least such j*

$$2^{b-1} \leq \mathrm{Vol}\left(\pi_t(\{1, \ldots, j\})\right) < 2^b.$$

Lemma 3.17 (Analysis of Rounded Walk). *For each $\theta_0 \leq 1$, if S is a set satisfying $\mathrm{Vol}\,(S) \leq (2/3)\mathrm{Vol}\,(V)$ and*

$$\Phi(S) \leq \frac{\theta_0^3}{7^4 \cdot 8 \ln^2(me^4)}$$

and `Nibble` is started at a vertex $v \in S_b^g$ with parameter b, then there exists a $t < t_0$ and a $\tilde{j}$ such that conditions i, ii and iii of line (3.c) of `Nibble` are satsfied.

Proof. Let t_0, γ, and ϵ_b be as set by `Nibble`. Let $\phi = \theta_0/7$ and hence $t_0 = 49\ln(me^4)/\theta_0^2 = \ln(me^4)/\phi^2$ and our assumption that $\theta_0 \leq 1$ extends to $\phi \leq 1/7$. Simply from the definition of $\Phi(S)$ and t_0, we also have $\Phi(S)t_0 \leq 1/32$.

Let j, j_0 and j_1 be as in Lemma 3.7. Let j' be the element of $\{1, \ldots, j_0\}$ minimizing $\tilde{\rho}_t(\pi_t(j'))$. We then set $\tilde{j}$ so that

$$\tilde{\pi}_t(\tilde{j}) = \pi_t(j').$$

By the definition of j', we have

$$\pi_t\left(\{1, \ldots, j_0\}\right) \subseteq \tilde{\pi}_t\left(\left\{1, \ldots, \tilde{j}\right\}\right).$$

So, we can establish the right-hand-side of condition iii from

$$\mathrm{Vol}\left(\tilde{\pi}_t\left(\left\{1, \ldots, \tilde{j}\right\}\right)\right) \geq \mathrm{Vol}\left(\pi_t\left(\{1, \ldots, j_0\}\right)\right) \geq k_j^t(1 - 2\phi) \geq 2^{b-1}(1 - 2\phi) \geq (5/7)2^{b-1}. \quad (12)$$

To establish the left-hand-side of condition iii, we apply Lemma 3.14, which implies

$$\mathrm{Vol}\left(\tilde{\pi}_t\left(\{1,\ldots,\tilde{j}\}\right)\right) \leq (5/4)\mathrm{Vol}\left(S\right) \leq (10/12)\mathrm{Vol}\left(V\right).$$

By Lemma 3.15, we have that for all v

$$\tilde{\rho}_t(v) \geq \rho_t(v) - t_0\epsilon_b \geq \rho_t(v) - \frac{\phi}{8\ln(me^4)k_j^t}. \tag{13}$$

So,

$$\begin{aligned}
\tilde{\rho}_t(\tilde{\pi}_t(\tilde{j})) - \rho_t(\pi_t(j_1)) &= \tilde{\rho}_t(\pi_t(j')) - \rho_t(\pi_t(j_1)) \\
&\geq \rho_t(\pi_t(j')) - \rho_t(\pi_t(j_1)) - \frac{\phi}{8\ln(me^4)k_j^t} \\
&\geq \rho_t(\pi_t(j_0)) - \rho_t(\pi_t(j_1)) - \frac{\phi}{8\ln(me^4)k_j^t} \\
&\geq \frac{\phi}{4\ln(me^4)k_j^t} - \frac{\phi}{8\ln(me^4)k_j^t} \\
&> 0.
\end{aligned}$$

This last inequality implies

$$\tilde{\pi}_t\left(\{1,\ldots,\tilde{j}\}\right) \cap \pi_t\left(\{j_1,\ldots,n\}\right) = \emptyset,$$

from which we derive

$$\begin{aligned}
\partial\left(\tilde{\pi}_t\left(\{1,\ldots,\tilde{j}\}\right)\right) &\leq \partial\left(\pi_t\left(\{1,\ldots,j_0\}\right)\right) + k_{j_1-1}^t - k_{j_0}^t \\
&\leq \partial\left(\pi_t\left(\{1,\ldots,j_0\}\right)\right) + 4\phi k_j^t.
\end{aligned}$$

Thus,

$$\begin{aligned}
\Phi\left(\tilde{\pi}_t\left(\{1,\ldots,\tilde{j}\}\right)\right) &= \frac{\partial\left(\tilde{\pi}_t\left(\{1,\ldots,\tilde{j}\}\right)\right)}{\mathrm{Vol}\left(\tilde{\pi}_t\left(\{1,\ldots,\tilde{j}\}\right)\right)} \\
&\leq \frac{\partial\left(p_t\left(\{1,\ldots,j\}\right)\right) + 4\phi k_j^t}{k_j^t(1-2\phi)} \\
&\leq \frac{1}{1-2\phi}\left(\frac{\partial\left(p_t\left(\{1,\ldots,j\}\right)\right)}{k_j^t} + 4\phi\right) \\
&\leq \frac{5\phi}{1-2\phi},
\end{aligned}$$

implying that condition i is satisfied. To establish condition ii, we note

$$\rho_t(\pi_t(j')) \geq \rho_t(\pi_t(j_0)) \geq \frac{\phi}{4\ln(me^2)k_j^t}.$$

So,

$$\begin{aligned}
\tilde{\rho}_t(\tilde{\pi}_t(\tilde{j})) &\geq \frac{\phi}{4\ln(me^2)k_j^t} - t_0\epsilon_b \\
&\geq \frac{\phi}{8\ln(me^2)k_j^t} \\
&\geq \frac{\phi(1-2\phi)}{8\ln(me^2)\mathrm{Vol}\left(\tilde{\pi}_t\left(1,\ldots,\tilde{j}\right)\right)} \geq \frac{5\theta_0}{7^2\cdot 8\ln(me^2)\mathrm{Vol}\left(\tilde{\pi}_t\left(1,\ldots,\tilde{j}\right)\right)}
\end{aligned}$$

by (12) and $\phi = \theta_0/7 \leq 1/7$. □

We now prove Lemma 3.1.

Proof of Lemma 3.1. We first bound the running time of `Nibble`. The algorithm will take $O(\ln(m)/\theta_0^2)$ iterations. We now show that in each iteration, the algorithm does at most $O(\log(m)/\epsilon_b)$ work, if the multiplication of step (3.a) is implemented correctly. Note that $\tilde{p}_{t-1} = [\tilde{p}_{t-1}]_\epsilon$. Let V_{t-1} be the set of nodes x for which $[\tilde{p}_{t-1}]_\epsilon(x) > 0$. The set V_{t-1} can be determined in $O(|V_{t-1}|)$ time at this stage from $\tilde{\pi}_{t-1}$ (or trivially for $t = 1$). Moreover, given knowledge of V_{t-1}, the vector $P\tilde{p}_{t-1}$ can be produced in time $O(\mathrm{Vol}\left(V_{t-1}\right))$, which satisfies

$$\mathrm{Vol}\left(V_{t-1}\right) = \sum_{x\in V_{t-1}} d(x) \leq \sum_{x\in V_{t-1}} \tilde{p}(x)/2\epsilon_b \leq 1/2\epsilon_b,$$

by the definition of $[\tilde{p}_{t-1}]_\epsilon$. Similarly, step (3.c) can be implemented in $O(\mathrm{Vol}\left(V_t\right))$ time. Finally step (3.b) requires time at most $O(\ln(1/\epsilon_b)/\epsilon_b)$.

In Definition 3.5, we define a set of vertices $S^g \subseteq S$ such that in Lemma 3.17 we prove that for each $v \in S^g$ there exists a suitable b. Our lower bound on the volume of S^g comes from Proposition 3.6. The assertion that $\mathrm{Vol}\left(C\right) \leq (5/6)\mathrm{Vol}\left(V\right)$ follows from Lemma 3.14 by the logic

$$\mathrm{Vol}\left(\tilde{\pi}_t\left(\{1,\ldots,\tilde{j}\}\right)\right) \leq (5/4)\mathrm{Vol}\left(S\right) \leq (10/12)\mathrm{Vol}\left(V\right).$$

□

4 `MultiwayPartition`

Given a partition algorition, one can design a multiway partition algorithm by repeatedly applying the partition algorithm to the subgraphs it generates. On an input parameter θ_*, we would desire to produce a multiway partition $\{C_i\}$ of a given graph G so that $\Phi_{C_i} \geq \theta_*$ for all i. So far it remains open whether one can achieve this guarantee in $m\log^{O(1)} n/\theta_*^{O(1)}$ time.

In this section, we analyze the performence of the following $m\log^{O(1)} n/\theta_*^5$ time algorithm for multiway graph partitioning. We will show that it generates a multiway partition that is good enough for developing nearly linear-time algorithms for graph sparsification and ultra-sparsification, and for solving SDD linear systems.

$\mathcal{C} = \texttt{MultiwayPartition}(G, \theta, p)$

(0) Set $\mathcal{C}_1 = V$ and $S = \emptyset$.

(1) For $t = 1$ to $\lceil \log_{17/16} m \rceil \cdot \lceil \lg m \rceil \cdot \lceil \lg(2/\epsilon) \rceil$

(a) For each component $C \in \mathcal{C}_t$,

$D = \texttt{Partition}(G(C), \theta_0, p/m)$.

Add D and $C - D$ to $\mathcal{C}_{t+1}$.

(2) Return $\mathcal{C} = C_{t+1}$.

We will use the following set of parameters.

Let m be an upper bound on the number of edges of the input graph. We let

$$\epsilon \stackrel{\text{def}}{=} \min\left(\frac{1}{16}, \frac{1}{4\lceil \lg m \rceil}\right). \tag{14}$$

Assuming θ is the sparsity of the cut that our partitioning algorithm is aiming to produce. We then let

$$\theta_* \stackrel{\text{def}}{=} \epsilon\theta_+/32, \tag{15}$$

where θ_+ is defined as in Equation (3).

Theorem 4.1 (MultiwayPartition). *Let $G = (V, E)$ be an undirected graph of n vertices and at most m edges. For any $0 < \theta < 1$, let $\mathcal{C}$ be the set of components returned by MultiwayPartition. Then, with probability at least $1 - p$,*

1 ***cut-size*** $(\mathcal{C}) \leq \left(\theta \log_{17/16} m \cdot \lg m \cdot \lg(2/\epsilon)\right)(m/2)$, *and*

2. *there exists a set $\mathcal{W}$ of subsets of V, an assignment **level** $: \mathcal{W} \to \left\{1, \ldots, \lceil \log_{17/16} m \rceil\right\}$, and a mapping $\pi : \mathcal{C} \to \mathcal{W}$ such that*

   a. *For all $W \in \mathcal{W}$, $\Phi_W \geq \theta_*$,*

   b. *For all $C \in \mathcal{C}$, $C \subseteq \pi(C)$*

   c. *For all $l \in \left\{1, \ldots, \lceil \log_{17/16} m \rceil\right\}$, $\{W \in \mathcal{W} : \textbf{level}(W) = l\}$ are pair-wise disjoint.*

   d. *For each pair $W_i \in \mathcal{W}$ and $W_j \in \mathcal{W}$ such that $\textbf{level}(W_i) > \textbf{level}(W_j)$,*

$$W_i \cap \left(\cup_{C \in \mathcal{C}: \pi(C) = W_j} C\right) = \emptyset.$$

In addition, the expected running time of MultiwayPartition is $m\left(\lg(1/p) \lg^{O(1)}(m)\right)/\theta^5$.

Proof. The bound on the expected running time of `MultiwayPartition` follows immediately from the bound on the expected running time of `Partition` established in Theorem 3.3. As `Partition` always finds cuts of sparsity at most θ, the calls made to `Partition` at each iteration of `MultiwayPartition` remove at most $\theta(m/2)$ edges. Thus,

$$\textbf{cut-size}\,(\mathcal{C}) \leq \theta \log_{17/16} m \cdot \lg m \cdot \lg(2/\epsilon)(m/2).$$

`MultiwayPartition` makes at most m calls are made to `Partition`; so, with probability at least $1-p$ every call made to `Partition` succeeds. To prove (2), we assume that all these calls succeed. We will divide the iterations of `MultiwayPartition` into $\lceil \log_{17/16} m \rceil$ epochs of $\lceil \lg m \rceil \cdot \lceil \lg(2/\epsilon) \rceil$ iterations. Thus the ith epoch starts at time $t_i = (i-1)\lceil \lg m \rceil \lceil \lg(2/\epsilon) \rceil + 1$.

Algorithm `MultiwayPartition` can be viewed as a process to grow a tree T of components. Initially we have a tree T_1 that has one node V. For each component $C \in \mathcal{C}_{t_i}$, the ith epoch generates a set of subcomponents $\mathcal{C}_{t_{i+1}}(C)$. In T_{i+1}, C is then an internal node with children $\mathcal{C}_{t_{i+1}}(C)$. In the end, `MultiwayPartition` generates a tree T with $\mathcal{C}$ its leaves. Note that in step 1.a of `MultiwayPartition`, an application of `Partition` on $C \in C_t$ addes at least one component to $\mathcal{C}_{t+1}$. So all leaf-components $\mathcal{C}$ are at depth $\lceil \log_{17/16} m \rceil$ in T.

We will define the set $\mathcal{W}$ by truncating this tree T: Starting from the root of T, we perform Breadth-First Search (BFS). When BFS visits a node C, we truncate the search of its subtree if one of the following two conditions is satisfied.

1. $\Phi_C \geq \theta_*$, or
2. $\mathrm{Vol}_C(C) > (16/17)\mathrm{Vol}_{P_C}(P_C)$ where P_C be the parent of C in T.

When the first condition holds, we add C to $\mathcal{W}$ and assign $\textbf{level}(C)$ to be the depth of C in T. Also in π we map all leaf-components of the subtree rooted at C to C.

When the second condition is true, it follows from Lemma 4.5, with probability at least $1 - \lg^{O(1)} m/\theta^5 m^3$, there exists a subset $W_C \in P_C$ with $\Phi_{W_C} \geq \theta_*$ and $C \in W_C$. We add W_C to $\mathcal{W}$ and assign $\textbf{level}(W_C)$ to be the depth of P_C in T. In π we map all leaf-components of the subtree rooted at C to W_C.

We first show, with probability at least $1 - \log_{17/16} m \lg^{O(1)} m/\theta^5 m^3 = 1 - \lg^{O(1)} m/\theta^5 m^3$, every leaf-component in $\mathcal{C}$ is mapped to one of the set in $\mathcal{W}$. To apply proof-by-contradication, assume $C \in \mathcal{C}$ is an components that is not mapped to any set in $\mathcal{W}$, which implies that none of the proper ancestors of C satisfies (2). As C is associated with a leaf of T at depth $\lceil \log_{17/16} m \rceil$, $\mathrm{Vol}_C(C)$ is upper bounded by $\mathrm{Vol}_V(V)(16/17)^{\lceil \log_{17/16} m \rceil} \leq 1$ which implies $\Phi_C \geq \theta^*$. Thus if none of the proper ancestor satisfies condition (1), C will be mapped to $\mathcal{W}$. So we have established that the set $\mathcal{W}$ and the mapping π satisfies condition 2.a and 2.b of the lemma.

Condition 2.c of the lemma follows from the fact that for each $l \in \left\{1, \ldots, \lceil \log_{17/16} m \rceil\right\}$ the set of components associated with nodes at depth l in T forms a partition of V.

To establish condition 2.d of the lemma, we only need to consider a pair $W_i \in \mathcal{W}$ and $W_j \in \mathcal{W}$ with $\textbf{level}(W_i) > \textbf{level}(W_j)$ such that $W_i \cap W_j \neq \emptyset$ which is possible only when W_j is defined by a node C in T of condition (2) and W_i is defined by a node C' in the subtree rooted at P_C, the parent of C in T. Because P_C can has at most one child satisfying condition (2), W_i is contained in $P_C - C$. In our definition of π, all components that mapped to W_j is contained in the subtree rooted at C, and hence they have empty intersection with W_i.

Finally, the time complexity of `MultiwayPartition` is bounded by $O\left(\log_{17/16} m \cdot \lg m \cdot \lg(2/\epsilon)\right)$ times the complexity of `Partition`. □

4.1 A New Variant of Sparsity and Isoperimetric Number

We introduce in this subsection a new variant of sparsity and isoperimetric number that will be helpful[1] for the proofs in this section. We will state some of its basic properties and then use in the next subsection to analyze the performance of `MultiwayPartition`.

For a given graph $G = (V, E)$ and for each subset S of V, we define the new sparsity of S to be

$$\Phi_V(S) = \frac{\partial_V(S)}{\min(\mathrm{Vol}(S), \mathrm{Vol}(V-S))^{1+4\epsilon}}.$$

We also define the new isoperimetric number of a subset S to be

$$\Phi_S = \min_{T \subseteq S} \Phi_S(T) = \min_{T \subseteq S} \frac{\partial_S(T)}{\min(\mathrm{Vol}(T), \mathrm{Vol}(S-T))^{1+4\epsilon}}.$$

Note that the induced graph of S is connected if and only if $\Phi_S > 0$.

The purpose of this definition of Φ is to satisfy the following lemma.

Lemma 4.2 (Union of sets with small intersection). *Let $0 < \epsilon < 1/4$. Let S and T be sets of vertices such that $\mathrm{Vol}(S \cap T) \leq \epsilon \min(\mathrm{Vol}(S), \mathrm{Vol}(T))$. Then*

$$\mathrm{Vol}(S \cup T)^{1+4\epsilon} > \mathrm{Vol}(S)^{1+4\epsilon} + \mathrm{Vol}(T)^{1+4\epsilon}.$$

If in addition $\mathrm{Vol}(S \cup T) \leq (1/2)\mathrm{Vol}(V)$, then

$$\Phi_V(S \cup T) \leq \max(\Phi_V(S), \Phi_V(T))$$

Proof. Assume without loss of generality that $\mathrm{Vol}(T) \leq \mathrm{Vol}(S)$. Let $\mathrm{Vol}(T) = \alpha \mathrm{Vol}(S)$, and note $\alpha \leq 1$. Let $\mathrm{Vol}(T \cap S) = \delta \mathrm{Vol}(T)$, and note $\delta \leq \epsilon$. So,

$$\mathrm{Vol}(S \cup T)^{1+4\epsilon} = \mathrm{Vol}(S)^{1+4\epsilon}(1 + \alpha - 2\alpha\delta)^{1+4\epsilon}$$

and

$$\mathrm{Vol}(S)^{1+4\epsilon} + \mathrm{Vol}(T)^{1+4\epsilon} = \mathrm{Vol}(S)^{1+4\epsilon}(1 + \alpha^{1+4\epsilon}).$$

Thus, to prove the first assertion we must show

$$(1 + \alpha - 2\alpha\delta)^{1+4\epsilon} > (1 + \alpha^{1+4\epsilon}).$$

As $\alpha \leq 1$, it suffices to show that

$$(1 + \alpha - 2\alpha\delta)^{1+4\epsilon} > 1 + \alpha.$$

Let

$$f(\alpha) \stackrel{\text{def}}{=} (1 + \alpha - 2\alpha\delta)^{1+4\epsilon}.$$

We note that

$$f'(\alpha) = (1 + 4\epsilon)(1 + \alpha - 2\alpha\delta)^{4\epsilon}(1 - 2\delta), \text{ and}$$
$$f''(\alpha) = (4\epsilon)(1 + 4\epsilon)(1 + \alpha - 2\alpha\delta)^{1-4\epsilon}(1 - 2\delta)^2.$$

[1] This new variant could be useful for other applications.

As $1+\alpha$ is linear with slope 1, and $f''(\alpha) \geq 0$ for $\alpha \in [0,1]$, it suffices to show that $f'(0) \geq 1$, which follows from

$$(1+4\epsilon)(1-2\delta) \geq (1+4\epsilon)(1-2\epsilon) > 1,$$

for $0 < \epsilon < 1/4$. The second part now follows immediately from this inequality. $\square$

The following proposition follows directly from the fact that $m^{1/\lceil \lg m \rceil} \leq 2$.

Proposition 4.3 ($\varPhi$ and Φ). *For any set S, $\Phi_V(S)/2 \leq \varPhi_V(S) \leq \Phi_V(S)$ and hence $\Phi_S/2 \leq \varPhi_S \leq \Phi_S$.*

Proposition 4.4. *For all $S \subseteq V$, if $\partial_V(S) \leq \phi \mathrm{Vol}_V(S)^{1+4\epsilon}$ and $\mathrm{Vol}_V(S) \leq (1-\beta)\mathrm{Vol}_V(V)$ where $\beta < 1/2$ then*

$$\varPhi_V(S) \leq \left(\frac{1-\beta}{\beta}\right)^{1+4\epsilon} \phi.$$

Proof. If $\mathrm{Vol}_V(S) \leq \mathrm{Vol}_V(V)/2$ then $\varPhi_V(S) \leq \phi$. Otherwise,

$$\varPhi_V(S) \leq \frac{\partial_V(S)}{\mathrm{Vol}_V(V-S)^{1+4\epsilon}} \leq \left(\frac{1-\beta}{\beta}\right)^{1+4\epsilon} \phi.$$

$\square$

4.2 Technical Lemmas for the Analysis of `MultiwayPartition`

For each $t \geq 1$, $C \in \mathcal{C}_t$ and $j > t$ let $\mathcal{C}_j(C)$ denote all the components in $\mathcal{C}_j$ that are subsets of C.

Lemma 4.5 (Divided or Covered: Each Epoch). *For each $t \geq 1$ and $C \in \mathcal{C}_t$, let $t' = t + \lceil \lg m \rceil \cdot \lceil \lg(2/\epsilon) \rceil$. If every call made by `MultiwayPartition` to `Partition` succeeds, then either*

- *for all components $C' \in \mathcal{C}_{t'}(C)$, $\mathrm{Vol}_{C'}(C') \leq (16/17)\mathrm{Vol}_C(C)$, or*
- *Let $C_{t'}$ be the unique component in $\mathcal{C}_{t'}(C)$ such that $\mathrm{Vol}_{C_{t'}}(C_{t'}) > (16/17)\mathrm{Vol}_C(C)$. There exists a set $W \in C_t$ with $\Phi_W \geq \theta_*$ and $C_{t'} \subseteq W$.*

Proof. Assume that every call made by `MultiwayPartition` to `Partition` succeeds.

To establish the lemma, we assume

$$\mathrm{Vol}_{C_{t'}}(C_{t'}) > (16/17)\mathrm{Vol}_C(C). \tag{16}$$

Let $C_t = C$ for notational simplicity. For $t \leq j \leq t'$, let C_j be the unique component in $\mathcal{C}_j(C)$ such that $\mathrm{Vol}_{C_j}(C_j) > (16/17)\mathrm{Vol}_C(C)$. Then $C_{t'} \subseteq C_{t'-1} \subseteq \cdots \subseteq C_{t+1} \subseteq C_t$.

Let $V_0 = C_t$. For $i \in [0 : \lceil \lg m \rceil]$ we iteratively define U_{i+1}, V_{i+1}, and W_{i+1} and S_{i+1} by:

- $S_i \subset V_i$ is the largest subset such that $\mathrm{Vol}_{V_i}(S_i) \leq \mathrm{Vol}_{V_i}(V_i)/2$ and $\varPhi_{V_i}(S_i) \leq 2\theta_*$,
- $W_{i+1} = C_{t+i\lg(2/\epsilon)}$ and $U_{i+1} = V_i - W_{i+1}$, and
- $V_{i+1} = V_i - (S_i \cap U_{i+1})$.

As $W_1 \subseteq V_0$, inductivelly it follows from $V_{i+1} = V_i - (S_i \cap U_{i+1}) = V_i - (S_i \cap (V_i - W_{i+1}))$ and $W_{i+1} \subseteq W_i \subseteq V_i$, that $W_{i+1} \subseteq V_{i+1}$. In particularly, $C_{t'} = W_{\lceil \lg m \rceil} \subseteq V_{\lceil \lg m \rceil}$. We will show that $S_{\lceil \lg m \rceil} = \emptyset$. So, $\Phi_{V_{\lceil \lg m \rceil}} \geq \theta_*$ and $W = V_{\lceil \lg m \rceil}$ is the set claimed to exist by the lemma.

The key to our proof will be the observation in Proposition 4.7 that $\mathrm{Vol}_{W_{i+1}}(W_{i+1}) \geq (16/17)\mathrm{Vol}_{W_i}(W_i)$, implies

$$\mathrm{Vol}_{W_{i+1}}(W_{i+1} \cap S_i) \leq (\epsilon/2)\mathrm{Vol}_{V_i}(S_i). \tag{17}$$

We first prove

$$\forall T \subset V_0 = C_t, \mathrm{Vol}_{V_0}(V_0)/2 \geq \mathrm{Vol}_{V_0}(T) \geq \mathrm{Vol}_{V_0}(V_0)/16 \implies \Phi_{V_0}(T) > 8\theta^*. \tag{18}$$

To prove this, assume by way of contradiction that there exists a $T \subseteq V_0$ such that $\mathrm{Vol}_{V_0}(V_0)/2 \geq \mathrm{Vol}_{V_0}(T) \geq \mathrm{Vol}_{V_0}(V_0)/16$ and $\Phi_{V_0}(T) \leq 8\theta^*$. By Proposition 4.7, $\mathrm{Vol}_{C_{t+\lg(2/\epsilon)}}\left(C_{t+\lg(2/\epsilon)} \cap T\right) \leq (\epsilon/2)\mathrm{Vol}_{C_t}(T)$, which implies

$$\mathrm{Vol}_{C_t}\left(C_t - C_{t+\lg(2/\epsilon)}\right) \geq (1-\epsilon/2)\mathrm{Vol}_{C_t}(T) \geq (1/17)\mathrm{Vol}_{C_t}(C_t),$$

However, this would contradict our assumption that $\mathrm{Vol}_{C_{t'}}(C_{t'}) > (16/17)\mathrm{Vol}_{C_t}(C_t)$.

Fact (18) implies that $\mathrm{Vol}_{V_0}(S_0) \leq \mathrm{Vol}_{V_0}(V_0)/16$. In fact, we can use (18) to show that for all $i \in \{1 : \lceil \lg m \rceil - 1\}$,

$$\mathrm{Vol}_{V_{i+1}}(S_{i+1}) \leq \mathrm{Vol}_{V_i}(S_i)/2 \quad \text{and} \quad \mathrm{Vol}_{V_{i+1}}(S_{i+1}) \leq \mathrm{Vol}_{V_{i+1}}(V_{i+1})/16, \tag{19}$$

which implies $S_{\lceil \lg m \rceil} = \emptyset$. Our proof will be by induction on i.

Assume for now that for all i,

$$\mathrm{Vol}_{V_i}(S_i \cup S_{i+1}) < \mathrm{Vol}_{V_i}(V_i)/2. \tag{20}$$

Using this assumption and (17), we may apply Lemma 4.6 to prove (19).

Assume by way of contradiction that (20) fails for some i, and let j be the least such i. We already know that

$$\begin{aligned}
\mathrm{Vol}_{V_i}(S_i \cup S_{i+1}) &= \mathrm{Vol}_{V_i}(S_i) + \mathrm{Vol}_{V_i}(S_{i+1}) \\
&\leq \mathrm{Vol}_{V_i}(S_i) + \mathrm{Vol}_{V_{i+1}}(S_{i+1}) + \mathrm{Vol}_{V_i}(V_i - V_{i+1}), \quad \text{by Proposition 2.3,} \\
&\leq \mathrm{Vol}_{V_{i+1}}(V_{i+1})/2 + 2\mathrm{Vol}_{V_i}(V_i)/16 \\
&\leq (5/8)\mathrm{Vol}_{V_i}(V_i),
\end{aligned} \tag{21}$$

where the second-to-last inequality follows from the definition of S_{i+1}. So, we would have

$$\mathrm{Vol}_{V_i}(V_i)/2 \leq \mathrm{Vol}_{V_i}(S_i \cup S_{i+1}) \leq (5/8)\mathrm{Vol}_{V_i}(V_i)/2.$$

As j is the least index for which (20) fails, we also have

$$\mathrm{Vol}_{V_i}(S_i) \leq \mathrm{Vol}_{V_{i-1}}(S_{i-1})/2 \quad \text{for all} \quad 0 < i \leq j. \tag{22}$$

Hence

$$\mathrm{Vol}_{V_0}(S_0 \cup \cdots \cup S_{j+1}) \geq \mathrm{Vol}_{V_0}(S_j \cup S_{j+1}) \geq \mathrm{Vol}_{V_j}(V_j)/2 \geq (16/34)\mathrm{Vol}_{V_0}(V_0).$$

On the other hand

$$\begin{aligned}\mathrm{Vol}_{V_0}(S_0 \cup \cdots \cup S_{j+1}) &\leq \mathrm{Vol}_{V_0}(S_0 \cup \cdots \cup S_{j-1}) + \mathrm{Vol}_{V_0}(S_j \cup S_{j+1}) \\ &\leq \mathrm{Vol}_{V_0}(S_0 \cup \cdots \cup S_{j-1}) + (5/8)\mathrm{Vol}_{V_j}(V_j) \\ &\leq 2\mathrm{Vol}_{V_0}(S_0) + (5/8)\mathrm{Vol}_{V_0}(V_0) \\ &\leq (3/4)\mathrm{Vol}_{V_0}(V_0),\end{aligned}$$

where the last-to-second inequality used (22). Note also that

$$\partial_{V_0}(S_0 \cup \cdots \cup S_{j+1}) \leq 2\theta_* \left(\sum_{i=0}^{j+1} \mathrm{Vol}_{V_i}(S_i)^{1+4\epsilon}\right) \leq 2\theta_* \mathrm{Vol}_{V_0}(S_0 \cup \cdots \cup S_{j+1})^{1+4\epsilon}.$$

Thus

$$\Phi_{V_0}(S_0 \cup \cdots \cup S_{j+1}) \leq 2 \cdot 3^{1+4\epsilon}\theta_* \leq 8\theta_*.$$

where the last inequality follows from $\epsilon \leq 1/16$ and $3^{1+4\epsilon} \leq 4$. However, this would contradict (18). Therefore, no such j exists and hence for all i, $\mathrm{Vol}_{V_{i+1}}(S_{i+1}) \leq \mathrm{Vol}_{V_i}(S_i)/2$, implying $S_{\lceil \lg m \rceil} = \emptyset$. $\square$

Lemma 4.6 (Reduction). *For V_i, U_i, W_i and S_i as defined in the proof of Lemma 4.5, if*

a. $\mathrm{Vol}_{V_i}(S_i \cap W_{i+1}) < (\epsilon/2)\mathrm{Vol}_{V_i}(S_i)$,

b. $\mathrm{Vol}_{V_i}(S_i \cup S_{i+1}) < \mathrm{Vol}_{V_i}(V_i)/2$, *and*

c. $\mathrm{Vol}_{V_i}(S_i) \leq \mathrm{Vol}_{V_i}(V_i)/16$,

then,

i. $\mathrm{Vol}_{V_{i+1}}(S_{i+1}) \leq \mathrm{Vol}_{V_{i+1}}(V_{i+1})/16$, *and*

ii. $\mathrm{Vol}_{V_{i+1}}(S_{i+1}) \leq \mathrm{Vol}_{V_i}(S_i)/2$.

Proof. If either i or ii is false, we will obtain a contradiction to the maximality of S_i. We begin by showing that if i is false, then ii is false.

Assume by way of contractiction that $\mathrm{Vol}_{V_{i+1}}(S_{i+1}) > \mathrm{Vol}_{V_{i+1}}(V_{i+1})/16$. By Proposition 2.2, we have

$$\mathrm{Vol}_{V_{i+1}}(V_{i+1}) \geq \mathrm{Vol}_{V_i}(V_i) - 2\mathrm{Vol}_{V_i}(S_i) \geq (7/8)\mathrm{Vol}_{V_i}(V_i).$$

So, we would be able to derive

$$\mathrm{Vol}_{V_i}(S_{i+1}) \geq \mathrm{Vol}_{V_{i+1}}(S_{i+1}) \geq \mathrm{Vol}_{V_{i+1}}(V_{i+1})/16 \geq (7/8)\mathrm{Vol}_{V_i}(V_i)/16 \geq \mathrm{Vol}_{V_i}(S_i)/2 \quad (23)$$

We now proceed under the assumption that ii is false. As $\mathrm{Vol}_{V_i}(S_i \cap W_{i+1}) < (\epsilon/2)\mathrm{Vol}_{V_i}(S_i)$ and $S_i \cap S_{i+1} \subseteq W_{i+1}$, we would then obtain

$$\mathrm{Vol}_{V_i}(S_i \cap S_{i+1}) \leq (\epsilon/2)\mathrm{Vol}_{V_i}(S_i) \leq \epsilon\mathrm{Vol}_{V_i}(S_{i+1}),$$

where the second inequality follows from (23). By Lemma 4.2, we would then have

$$\mathrm{Vol}_{V_i}(S_i \cup S_{i+1})^{1+4\epsilon} > \mathrm{Vol}_{V_i}(S_i)^{1+4\epsilon} + \mathrm{Vol}_{V_i}(S_{i+1})^{1+4\epsilon}.$$

Moreover,

$$\partial_{V_i}(S_i \cup S_{i+1}) \leq E(S_i, V_i - S_i) + E(S_{i+1}, V_{i+1} - S_{i+1}) \leq 2\theta_*(\mathrm{Vol}_{V_i}(S_i)^{1+4\epsilon} + \mathrm{Vol}_{V_{i+1}}(S_{i+1})^{1+4\epsilon}),$$

so, $\Phi_{V_i}(S_i \cup S_{i+1}) \leq 2\theta_*$, which contradicts the maximality of S_i. $\square$

Proposition 4.7. *Assume that each call made by* `MultiwayPartition` *to* `Partition` *succeeds. Then, for every $t > 0$, $t' = t + \lg(2/\epsilon)$, $C \in \mathcal{C}_t$, for every set $T \subset C$ with $\mathrm{Vol}_C(T) \leq \mathrm{Vol}_C(C)/2$ such that $\Phi_C(T) \leq 8\theta_*$, and all $C' \in \mathcal{C}_{t'}$, either $\mathrm{Vol}_{C'}(C') \leq (16/17)\mathrm{Vol}_C(C)$ or $\mathrm{Vol}_{C'}(C' \cap T) \leq (\epsilon/2)\mathrm{Vol}_C(T)$.*

Proof. Assume there exists a (unique) component $C_{t'} \in \mathcal{C}_{t'}$ such that $\mathrm{Vol}_{C_{t'}}(C_{t'}) > (16/17)\mathrm{Vol}_C(C)$. We have for all $t < j < t'$, there is a unique $C_j \in \mathcal{C}_j$ such that $C_{t'} \subseteq C_{t'-1} \subseteq \cdots \subseteq C_{t+1} \subseteq C$. Let $T_j = T \cap C_j$. For simplicity let $C_t = C$ and $T_t = T$.

By Proposition 4.3, $\Phi_{C_t}(T_t) \leq 2\Phi_{C_t}(T_t) \leq 16\theta_* \leq (\epsilon/2)\theta_+$.

Note that

$$\partial_{C_j}(T_j) \leq \partial_{C_t}(T_t) \leq (\epsilon/2)\theta_+ \mathrm{Vol}_{C_t}(T_t).$$

So if $\mathrm{Vol}_{C_j}(T_j) \geq (\epsilon/2)\mathrm{Vol}_{C_t}(T_t)$, then $\Phi_{C_j}(T_j) \leq \theta_+$, and therefore, by Theorem 3.3, $\mathrm{Vol}_{C_{j+1}}(T_{j+1}) \leq \mathrm{Vol}_{C_j}(T_j)/2$, and $\mathrm{Vol}_{C_{t'}}(T_{t'}) \leq (\epsilon/2)\mathrm{Vol}_{C_t}(T_t)$. $\square$

5 Random Sampling

If we let $\tilde{L}$ be the result of randomly sampling the edges of L, we can not in general assume that $\kappa_f(L, \tilde{L})$ will be bounded. However, we now prove that we can bound $\kappa_f(L, \tilde{L})$ if the smallest eigenvalue of $D^{-1}L$ is bounded from below.

Lemma 5.1 (Small norm implies good preconditioner). *Let L and $\tilde{L}$ be Laplacian matrices and let D be a diagonal matrix with positive diagonals. If L has co-rank 1 and $\lambda_{max}(D^{-1}(L - \tilde{L})) < (1/2)\lambda_{min}(D^{-1}L)$, then*

$$\begin{aligned}\sigma_f(L, \tilde{L}) &\leq 1 + 2\frac{\lambda_{max}(D^{-1}(L - \tilde{L}))}{\lambda_{min}(D^{-1}L)}, \quad \textit{and} \\ \sigma_f(\tilde{L}, L) &\leq 1 + \frac{\lambda_{max}(D^{-1}(L - \tilde{L}))}{\lambda_{min}(D^{-1}L)}.\end{aligned}$$

Proof. By Proposition A.1, we have

$$\sigma_f(\tilde{L}, L) = \sigma_f(D^{-1}\tilde{L}, D^{-1}L) = \sigma_f(D^{-1/2}\tilde{L}D^{-1/2}, D^{-1/2}LD^{-1/2}),$$

$\lambda_{max}(D^{-1}(L - \tilde{L})) = \lambda_{max}(D^{-1/2}(L - \tilde{L})D^{-1/2})$, and $\lambda_{min}(D^{-1}L) = \lambda_{min}(D^{-1/2}LD^{-1/2})$. We can then apply the following characterization of σ_f for symmetric Laplacian matrices with co-rank 1:

$$\sigma_f(D^{-1/2}\tilde{L}D^{-1/2}, D^{-1/2}LD^{-1/2}) = \max_{D^{-1/2}x \perp 1} \left(\frac{x^T D^{-1/2}\tilde{L}D^{-1/2}x}{x^T D^{-1/2}LD^{-1/2}x} \right)$$

We then observe that

$$\begin{aligned}\frac{x^T D^{-1/2}\tilde{L}D^{-1/2}x}{x^T D^{-1/2}LD^{-1/2}x} &= \frac{x^T D^{-1/2}LD^{-1/2}x + x^T D^{-1/2}(\tilde{L}-L)D^{-1/2}x}{x^T D^{-1/2}LD^{-1/2}x} \\ &= 1 + \frac{x^T D^{-1/2}(\tilde{L}-L)D^{-1/2}x}{x^T D^{-1/2}LD^{-1/2}x}.\end{aligned}$$

Moreover, for $D^{-1/2}x \perp 1$, the absolute value of the right-hand term is

$$\left|\frac{x^T D^{-1/2}(\tilde{L}-L)D^{-1/2}x}{x^T D^{-1/2}LD^{-1/2}x}\right| \leq \frac{\lambda_{max}(D^{-1/2}(L-\tilde{L})D^{-1/2})}{\lambda_{min}(D^{-1/2}LD^{-1/2})}, \tag{24}$$

establishing the bound for $\sigma_f(\tilde{L}, L)$.

To bound $\sigma_f(D^{-1}L, D^{-1}\tilde{L})$ we note that

$$\sigma_f(L, \tilde{L}) = \sigma_f(D^{-1/2}LD^{-1/2}, D^{-1/2}\tilde{L}D^{-1/2}) = \max_{D^{-1/2}x \perp 1}\left(\frac{x^T D^{-1/2}LD^{-1/2}x}{x^T D^{-1/2}\tilde{L}D^{-1/2}x}\right),$$

and

$$\begin{aligned}\frac{x^T D^{-1/2}LD^{-1/2}x}{x^T D^{-1/2}\tilde{L}D^{-1/2}x} &= \frac{x^T D^{-1/2}LD^{-1/2}x + x^T D^{-1/2}(\tilde{L}-L)D^{-1/2}x}{x^T D^{-1/2}LD^{-1/2}x} \\ &= 1 + \frac{x^T D^{-1/2}(L-\tilde{L})D^{-1/2}x}{x^T D^{-1/2}LD^{-1/2}x - x^T D^{-1/2}(L-\tilde{L})D^{-1/2}x} \\ &= 1 + \frac{\frac{x^T D^{-1/2}(L-\tilde{L})D^{-1/2}x}{x^T D^{-1/2}LD^{-1/2}x}}{1 - \frac{x^T D^{-1/2}(L-\tilde{L})D^{-1/2}x}{x^T D^{-1/2}LD^{-1/2}x}} \\ &\leq 1 + 2\frac{\lambda_{max}(D^{-1/2}(L-\tilde{L})D^{-1/2})}{\lambda_{min}(D^{-1/2}LD^{-1/2})},\end{aligned}$$

where the last inequality follows from inequality (24) and the assumption $\lambda_{max}(D^{-1}(L-\tilde{L})) < (1/2)\lambda_{min}(D^{-1}L)$ which implies

$$\frac{x^T D^{-1/2}(L-\tilde{L})D^{-1/2}x}{x^T D^{-1/2}LD^{-1/2}x} \leq 1/2.$$

□

We will use the following algorithm to sparsify graphs with high isoperimetric number. As it requires little more work, we state the algorithm for general non-negative matrices.

`Sample`
Input: A, a symmetric non-negative matrix, and $c \geq 1$.

(1) Set $d(i) = \sum_j a_{i,j}$.

(2) For all i, j for which $a_{i,j} \neq 0$, set $p_{i,j} = \begin{cases} \frac{c a_{i,j}}{\min(d(i),d(j))} & \text{if } c < \min(d(i), d(j))/a_{i,j}, \\ 1 & \text{otherwise.} \end{cases}$

(3) For all i, j for which $a_{i,j} \neq 0$, set $\tilde{a}_{i,j} = \tilde{a}_{j,i} = \begin{cases} \frac{a_{i,j}}{p_{i,j}} & \text{with probability } p_{i,j}, \\ 0 & \text{with probability } 1 - p_{i,j}. \end{cases}$

(4) Return the matrix $\tilde{A}$ of the $\tilde{a}_{i,j}$s.

By adapting techniques used by Füredi and Komlós [FK81] to study random matrices, we prove:

Theorem 5.2 (Sampling). *Let A be a non-negative symmetric matrix and let $c \geq 1$. Let $d(i) = \sum_j a_{i,j}$, and let $D = \mathrm{diag}(d(1), \ldots, d(n))$. Let $\tilde{A}$ be the output of* `Sample` *(A,c). Then, for all $\alpha \geq 1$, and even integers k,*

$$\Pr\left[\lambda_{max}\left(D^{-1}(\tilde{A} - A)\right) \geq \frac{2\alpha k n^{1/k}}{\sqrt{c}}\right] < \alpha^{-k}.$$

Proof. Our goal is to show that it is unlikely that the largest eigenvalue of $\Delta \stackrel{\text{def}}{=} D^{-1}(\tilde{A} - A)$ is large. To this end, we note that for even k, $(\lambda_{max}(\Delta))^k \leq \mathrm{Tr}\left(\Delta^k\right)$, and so it suffices to upper bound $\mathbf{E}\left[\mathrm{Tr}\left(\Delta^k\right)\right]$. We first observe that

$$\Delta_{i,j} = \begin{cases} \frac{a_{i,j}}{d(i)}\left(\frac{1}{p_{i,j}} - 1\right) & \text{with probability } p_{i,j}, \\ -\frac{a_{i,j}}{d(i)} & \text{with probability } 1 - p_{i,j}. \end{cases}$$

We then observe that the i-th diagonal element of Δ^k corresponds to the sum over all length k walks in A that start and end at i of the product of the weights encountered during the walk. Formally,

$$\left(\Delta^k\right)_{v_0,v_0} = \sum_{v_1,\ldots,v_{k-1}} \Delta_{v_{k-1},v_0} \prod_{i=0}^{k-1} \Delta_{v_i,v_{i+1}},$$

and

$$\mathbf{E}\left[\left(\Delta^k\right)_{v_0,v_0}\right] = \sum_{v_1,\ldots,v_{k-1}} \mathbf{E}\left[\Delta_{v_{k-1},v_0} \prod_{i=0}^{k-1} \Delta_{v_i,v_{i+1}}.\right]$$

As the expectation of the product of independent variables is the product of the expectation, and $\mathbf{E}\left[\Delta_{i,j}\right] = 0$ for all i and j, we need only consider walks that traverse no edge just once. So that we can distinguishing which edges in the walk are repeats, we will carefully code the walks. We first let S denote the set of time steps i such that the edge between v_{i-1} and v_i does not appear earlier in the walk. We then let τ denote the map from $[k] - S \to S$, indicating for each time step not in S the time step in which the edge traversed first appeared (regardless of in which direction it is traversed). We let $p = |S|$, and note that we need only consider the cases in which $p \leq k/2$ as otherwise some edge appears only once in the walk.

For each S and τ, we let $\{s_1, \ldots, s_p\} = S$, and consider an assignment of $v_{s_1}, \ldots, v_{s_p}$. We will call an assignment of $v_{s_1}, \ldots, v_{s_p}$ *valid* if it corresponds to a walk on the non-zero variables of Δ. Formally, the assignment is *valid* if

- It corresponds to a walk. That is at each $i \notin S$, $v_{i-1} \in \left\{v_{\tau(i)-1}, v_{\tau(i)}\right\}$. And,
- If we let v_i denote the vertex reached at the ith step, for $0 \leq i \leq k$, then for no i does $a_{v_i, v_{i+1}} = 0$ or $p_{v_i, v_{i+1}} = 1$.

We have

$$\begin{aligned}
\mathbf{E}\left[\left(\Delta^k\right)_{v_0, v_0}\right] &= \sum_{S,\tau} \sum_{\substack{\text{valid} \\ v_{s_1}, \ldots, v_{s_p}}} \mathbf{E}\left[\Delta_{v_{k-1}, v_0} \prod_{i=0}^{k-1} \Delta_{v_i, v_{i+1}}\right] \\
&= \sum_{S,\tau} \sum_{\substack{\text{valid} \\ v_{s_1}, \ldots, v_{s_p}}} \prod_{j=1}^{p} \mathbf{E}\left[\Delta_{v_{s_j-1}, v_{s_j}} \prod_{i:\tau(i)=s_j} \Delta_{v_{i-1}, v_i}\right].
\end{aligned}$$

We now associate a weight with each vertex v_{s_j} and each valid assignment $v_{s_1}, \ldots, v_{s_p}$ by

$$\begin{aligned}
w\left(v_{s_j}\right) &= \frac{a_{v_{s_j-1}, v_{s_j}}}{d(v_{s_j-1})} \\
w\left(v_{s_1}, \ldots, v_{s_p}\right) &= \prod_{i=1}^{p} w\left(v_{s_i}\right).
\end{aligned}$$

As $w(v_{s_j})$ is the probability that vertex v_{s_j} follows v_{s_j-1} in the random walk on A, we have

$$\sum_{\substack{\text{valid} \\ v_{s_1}, \ldots, v_{s_p}}} \prod_{j=1}^{p} w\left(v_{s_j}\right) \leq 1.$$

Below, we will establish

$$\mathbf{E}\left[\Delta_{v_{s_j-1}, v_{s_j}} \prod_{i:\tau(i)=s_j} \Delta_{v_{i-1}, v_i}\right] \leq \frac{1}{c^{|\{i:\tau(i)=s_j\}|}} w(v_{s_j}), \tag{25}$$

from which it follows that

$$\sum_{\substack{\text{valid} \\ v_{s_1}, \ldots, v_{s_p}}} \prod_{j=1}^{p} \mathbf{E}\left[\Delta_{v_{s_j-1}, v_{s_j}} \prod_{i:\tau(i)=s_j} \Delta_{v_{i-1}, v_i}\right] \leq \frac{1}{c^{k-p}}.$$

As there are n choices for v_0, at most 2^k choices for S, and at most k^k choices for τ, we have

$$\mathbf{E}\left[\mathrm{Tr}\left(\Delta^k\right)\right] \leq \frac{n(2k)^k}{c^{k/2}}.$$

Applying Markov's inequality, we obtain

$$\Pr\left[\lambda_{max}(\Delta) > \alpha n^{1/k} 2kc^{-1/2}\right] \leq \alpha^{-k}.$$

It remains to prove inequality (25). To simplify notation, we let $r = v_{s_j - 1}$ and $t = v_{s_j}$. We must then show that, for $k \geq 1$,

$$\mathbf{E}\left[\Delta_{r,t}^{k} \Delta_{t,r}^{l}\right] \leq \frac{1}{c^{k+l-1}} \frac{a_{r,t}}{d(r)}.$$

We first note that $|\Delta_{r,t}| < 1/c$, so it suffices to prove the inequality in the case $k + l = 2$. In this case, we obtain

$$\begin{aligned}
\mathbf{E}\left[\Delta_{r,t}^{k} \Delta_{t,r}^{l}\right] &= \frac{a_{r,t}^2}{d(r)^k d(t)^l} \left(p_{r,t} \left(\frac{1 - p_{r,t}}{p_{r,t}} \right)^2 + (1 - p_{r,t}) \right) \\
&= \frac{a_{r,t}^2}{d(r)^k d(t)^l} \left(\frac{1 - p_{r,t}}{p_{r,t}} \right) \\
&\leq \frac{a_{r,t}^2}{d(r)^k d(t)^l} \left(\frac{1}{p_{r,t}} \right) \\
&= \frac{a_{r,t}^2}{d(r)^k d(t)^l} \frac{\min(d(r), d(t))}{c a_{r,t}} \\
&= \frac{a_{r,t}}{c d(r)} \frac{\min(d(r), d(t))}{d(r)^{k-1} d(t)^l} \\
&\leq \frac{a_{r,t}}{c d(r)},
\end{aligned}$$

as $k - 1 + l = 1$. □

Lemma 5.3 (Close weighted degrees). *Let A be the adjacency matrix of an unweighted graph, and let $\tilde{A}$ be the output of* `Sample`(A, c). *Let $d(1), \ldots, d(n)$ be the degrees of the vertices of A and let $\tilde{d}(1), \ldots, \tilde{d}(n)$ be the corresponding terms for $\tilde{A}$. Then, for $\delta < 1$,*

(*a*) *for all i,* $\Pr\left[\left|1 - d(i)^{-1}\tilde{d}(i)\right| > \delta\right] < 2e^{-c\delta^2/3}$, *and*

(*b*) *the probability that $\tilde{A}$ has more than $2nc$ edges is at most* $(4/e)^{-cn/2}$.

Proof. For any vertex i, each edge (i, j) of A appears in $\tilde{A}$ with weight $\min(d(i), d(j))/c$ with probability $c/\min(d(i), d(j))$. Thus, $d(i)^{-1}\tilde{d}(i)$ has expectation 1 and is the sum of random variables each of which is always at most $1/c$. So, part (*a*) now follows directly from the Hoeffding inequality in Lemma B.1. We could derive a bound for part (*b*) directly from part (*a*). But, we obtain a stronger bound by letting $X_{i,j}$ be the random variable that is 1 if edge (i, j) appears in $\tilde{A}$. Then,

$$\mathbf{E}\left[\sum X_{i,j}\right] = \sum_{(i,j)} \frac{c}{min(d(i), d(j))} \leq \sum_{(i,j)} \left(\frac{c}{d(i)} + \frac{c}{d(j)} \right) = cn.$$

We can similarly show that $\mathbf{E}\left[\sum X_{i,j}\right] \geq cn/2$. Applying Lemma B.1 we obtain

$$\Pr\left[\sum X_{i,j} > 2cn\right] < (4/e)^{-cn/2}.$$

$\square$

Theorem 5.4 (Preconditioning by Sampling). *Let A be the adjacency matrix of an unweighted graph, L be its Laplacian, D the diagonal matrix of its degrees, and let $\lambda_{min}(D^{-1}A) \geq \lambda$. Let B be the adjacency matrix of a subgraph of A. Let $0 < p < 1$ and*

$$k \stackrel{\text{def}}{=} \max\left(\lceil \lg(2/p) \rceil, \lceil \lg n \rceil\right).$$

For any $\beta < 1$, let $\tilde{B} = \texttt{Sample}(B, c)$, where

$$c = (30k/\beta\lambda)^2.$$

If we then let $\tilde{A} = \tilde{B} + (A - B)$, and let $\tilde{L}$ be its Laplacian, then

$$\Pr\left[\sigma_f(L, \tilde{L}) > 1 + \beta/3 \quad \textit{and} \quad \sigma_f(\tilde{L}, L) > 1 + 2\beta/3\right] < p.$$

Proof. Let D_B be the diagonal matrix of the degrees of B and $D_{\tilde{B}}$ the corresponding matrix for $\tilde{B}$. We then have $L - \tilde{L} = D_B - D_{\tilde{B}} - (B - \tilde{B})$. Applying Theorem 5.2 with $\alpha = 2$, we find

$$\Pr\left[\lambda_{max}\left(D_B^{-1}(B - \tilde{B})\right) \geq \frac{8k}{\sqrt{c}}\right] < p/2.$$

Applying Lemma 5.3 with $\delta = \sqrt{3k/c}$, we find that

$$\Pr\left[\lambda_{max}(D_B^{-1}(D_B - D_{\tilde{B}})) > \frac{\sqrt{3k}}{\sqrt{c}}\right] < p/2.$$

So,

$$\Pr\left[\lambda_{max}\left(D_B^{-1}(L - \tilde{L})\right) \geq \frac{8k}{\sqrt{c}} + \frac{\sqrt{3k}}{\sqrt{c}}\right] < p.$$

By

$$\frac{8k}{\sqrt{c}} + \frac{\sqrt{3k}}{\sqrt{c}} < \frac{10k}{\sqrt{c}} < \frac{\beta\lambda}{3},$$

and applying Proposition A.2, we obtain

$$\Pr\left[\lambda_{max}\left(D^{-1}(L - \tilde{L})\right) \geq \frac{\lambda\beta}{3}\right] < p.$$

So, by applying Lemma 5.1, we find that

$$\Pr\left[\sigma_f(L, \tilde{L}) > 1 + \beta/3 \quad \text{and} \quad \sigma_f(\tilde{L}, L) > 1 + 2\beta/3\right] < p.$$

$\square$

6 Unweighted Sparsifiers

In this section, we show how to sparsify an unweighted graph. We will use the algorithm developed in this section as a subroutine for sparsifying general weighted graphs.

For a multiway partition $\mathcal{C}$, and a set of edges F, we let bridge $(\mathcal{C}, F)$ denote the set of edges of F going between components of the partition.

$\widetilde{E} = \texttt{UnweightedSparsifier}(E, \beta)$,
E is a set of unweighted edges and $\beta < 1$.

(0) Set $\theta = \left(\log_{17/16} m \cdot \lg m \cdot \lg(8 \lg m)\right)^{-1}$ and $\lambda = \theta_*^2/2$, where θ_* is given by (4), (3) and (15).

(1) $\mathcal{C} = \texttt{MultiwayPartition}(E, \theta, 1/n^2)$

(2) For each $C \in \mathcal{C}$ set $\widetilde{C} = \texttt{Sample}(C, c)$, where $c = (30(\lg n + 2)/\beta\lambda)^2$. Set $\tilde{A} = \tilde{A} \cup \widetilde{C}$.

(3) $S = \text{bridge}\,(\mathcal{C}, E)$

(4) $\widetilde{S} = \texttt{UnweightedSparsifier}(S, \beta)$. Set $\tilde{A} = \tilde{A} \cup \widetilde{S}$.

For each symmetric matric A with zero diagonals and non-negative off-diagonals, let $L(A)$ be the Laplacian of A. Let $|A|$ be one half of the number of non-zero entries in A. Suppose A is an n by n matrix. For each subset $U \subseteq \{1, \ldots, n\}$, let A_U denote the n by n matrix defined from A by zero out all the rows and columns not in U. So, if A is the adjacent matrix of $G = (V, E)$ where $V = \{1, \ldots, n\}$, then A_U is the adjacent matrix of the graph induced by U.

Lemma 6.1 (Unweighted Sparsifier). *For a set of unweighted edges E and $\beta < 1$, let $\widetilde{E} =$ $\texttt{UnweightedSparsifier}(E, \beta)$. Then, $\left|\widetilde{E}\right| < n \log^{O(1)} n/\beta^2$ with exponentially high probability. Moreover,*

$$\Pr\left[E \preccurlyeq \left(1 + \frac{\beta}{3}\right)^{O(\lg^2 m)} \widetilde{E},\ \text{and } \widetilde{E} \preccurlyeq \left(1 + \frac{2\beta}{3}\right)^{O(\lg^2 m)} E\right] \geq 1 - \frac{\lg m}{m^2}.$$

Proof. First, by Lemma 5.3 (b), with exponentially small probability, $\tilde{A}$ has more than $2cn \lg m$ non-zeros.

Let $A_{\mathcal{C}} = \sum_{C \in \mathcal{C}} A_C$. We can express A as $A = A_{\mathcal{C}} + A_S$. Algorithm `Unweighted Sparsifier` computes $\tilde{A}_{\mathcal{C}}$ for $A_{\mathcal{C}}$ and recursively computes $\tilde{A}_S$ from A_S and obtain $\tilde{A} = \tilde{A}_{\mathcal{C}} + \tilde{A}_S$. Let k be the total level of recursions in applying `Unweighted Sparsifier`. We will prove the lemma by an induction on k to show that with probability at least $1 - k/n^2$,

$$A \preccurlyeq (1 + \beta/3)^{k \log_{17/16} m} \tilde{A} \quad \text{and} \quad \tilde{A} \preccurlyeq (1 + 2\beta/3)^{k \log_{17/16} m} A.$$

By Theorem 4.1 and our choice of θ, $|A_S| \leq |A|/2$. So $k \leq \lg m$ from which the lemma would follow.

The lemma is true for the case when A is the all zero matrix and $k = 0$, Inductively, we assume it is true for $k_0 < k$, implying with probability at least $1 - (k-1)/n^2$,

$$A_S \preccurlyeq (1+\beta/3)^{(k-1)\log_{17/16} m} \tilde{A}_S \quad \text{and} \quad \tilde{A}_S \preccurlyeq (1+2\beta/3)^{(k-1)\log_{17/16} m} A_S.$$

By Lemma 6.2, it is sufficient to show with probability at least $1 - 1/n^2$

$$A \preccurlyeq (1+\beta/3)^{\log_{17/16} m} \hat{A} \quad \text{and} \quad \hat{A} \preccurlyeq (1+2\beta/3)^{\log_{17/16} m} A, \tag{26}$$

where $\hat{A} = A_S + \tilde{A}_{\mathcal{C}}$.

Let $\mathcal{W}$, π, and $\mathbf{level}\,(\mathcal{W})$ be as defined in Theorem 4.1. Let $A_{\mathcal{W}} = A_{\cup\{W \in \mathcal{W}\}}$ and $A_{\bar{\mathcal{W}}} = A - A_{\mathcal{W}}$. Similarly, let $\hat{A}_{\mathcal{W}} = \hat{A}_{\cup\{W\in\mathcal{W}\}}$. We have $\hat{A} = \hat{A}_{\mathcal{W}} + A_{\bar{\mathcal{W}}}$.

For each $C \in \mathcal{C}$, let $W_C = \pi(C)$. For each $l \in \left\{1 : \log_{17/16} m\right\}$, let

$$A_{C,l} = \sum_{\mathbf{level}(W_C)=l} A_C, \quad \text{and} \quad A_{\mathcal{W},l} = \sum_{\mathbf{level}(W_C)=l} A_{W_C}.$$

Similarly, let

$$\tilde{A}_{C,l} = \sum_{\mathbf{level}(W_C)=l} \tilde{A}_C, \quad \text{and} \quad \hat{A}_{\mathcal{W},l} = \sum_{\mathbf{level}(W_C)=l} \hat{A}_{W_C}.$$

By Theorem 4.1 (2.a and 2.b) we have $\Phi_{W_C} \geq \theta_*$ and $C \subseteq W_C$. By setting $\lambda = \theta_*^2/2$, we can then use Lemma 6.3 to establish a lower bound of λ on the smallest eigenvalue of the Laplacian (scaled by one over the degrees) of the graph induced by W_C.

As $c = 52000 \lg^2 n/(\beta^2\lambda^2)$, by Theorem 5.4 and a union bound, we obtain

$$\begin{aligned}\Pr\left[\forall C \in \mathcal{C} : L(A_{W_C}) \preccurlyeq (1+\beta/3) L(\hat{A}_{W_C}) \quad \text{and} \quad L(\hat{A}_{W_C}) \preccurlyeq (1+2\beta/3) L(A_{W_C})\right] &\geq 1 - 2\,|\mathcal{C}|\, n^{-4} \\ &\geq 1 - 1/n^2.\end{aligned}$$

Thus, by Theorem 4.1 (2.c) and Lemma 6.4, with probability at least $1 - n^{-2}$, for all $l \in \left\{1 : \log_{17/16} m\right\}$

$$L(A_{\mathcal{W},l}) \preccurlyeq (1+\beta/3) L(\hat{A}_{\mathcal{W},l}) \quad \text{and} \quad L(\hat{A}_{W,l}) \preccurlyeq (1+2\beta/3) L(A_{W,l}) \tag{27}$$

Let

$$A_{\mathcal{W}}^{(l)} = A_{\{\cup_{\mathbf{level}(W_C)\geq l} W_C\}} \quad \text{and} \quad \hat{A}_{\mathcal{W}}^{(l)} = \hat{A}_{\{\cup_{\mathbf{level}(W_C)\geq l} W_C\}}.$$

By Theorem 4.1 (2.d), we have for all $l \in \left\{1 : \log_{17/16} m\right\}$,

$$\left(\bigcup_{\mathbf{level}(W_C<l)} C\right) \bigcap \left(\bigcup_{\mathbf{level}(W_C\geq l)} W_C\right) = \emptyset.$$

So we can apply Lemma 6.2 to iteratively show from $l = \log_{17/16} m$ to 1 that if (27) is true, then

$$L(A_{\mathcal{W}}^{(l)}) \preccurlyeq (1+\beta/3)^{\log_{17/16} m - l} L(\hat{A}_{\mathcal{W}}^{(l)}) \quad \text{and} \quad L(\hat{A}_{\mathcal{W}}^{(l)}) \preccurlyeq (1+2\beta/3)^{\log_{17/16} m - l} L(A_{\mathcal{W}}^{(l)}).$$

Then (26) follows from $A_{\mathcal{W}} = A_{\mathcal{W}}^{(0)}$ and $\hat{A}_{\mathcal{W}} = \hat{A}_{\mathcal{W}}^{(0)}$ and Lemma 6.4. $\square$

Lemma 6.2 (Serial Sparsification). *Let A be a matrix that can be written as $A = P + R$. For any $\beta_1 > 0$ and $\beta_2 > 0$, let $\tilde{R}$ be an approximation of R such that $P + \tilde{R} \preccurlyeq (1+\beta_1)A$ and $A \preccurlyeq (1+\beta_1)(P+\tilde{R})$ and let $\tilde{P}$ be an approximation of P such that $\tilde{P} \preccurlyeq (1+\beta_2)P$ and $P \preccurlyeq (1+\beta_2)\tilde{P}$. Let $\tilde{A} = \tilde{P} + \tilde{R}$. Then*

$$\tilde{A} \preccurlyeq (1+\beta_1)(1+\beta_2)A \quad \text{and} \quad A \preccurlyeq (1+\beta_1)(1+\beta_2)\tilde{A}.$$

Proof. On one hand,

$$A \preccurlyeq (1+\beta_1)(P+\tilde{R}) \preccurlyeq (1+\beta_1)((1+\beta_2)\tilde{P}+\tilde{R}) \preccurlyeq (1+\beta_1)(1+\beta_2)(\tilde{P}+\tilde{R}) = (1+\beta_1)(1+\beta_2)\tilde{A}.$$

On the other hand,

$$\tilde{A} \preccurlyeq (1+\beta_2)P + \tilde{R} \preccurlyeq (1+\beta_2)(P+\tilde{R}) \preccurlyeq (1+\beta_1)(1+\beta_2)\tilde{A}.$$

□

Lemma 6.3 (Jerrum and Sinclair [SJ89]). *Let L be the Laplacian of an unweighted graph $G = (V, E)$. Then $\lambda_{\min}(D^{-1}L) \geq (\Phi_V)^2/2$.*

Lemma 6.4 (Splitting Lemma). *Let $A = \sum_{i=1}^{k} A_i$ and $B = \sum_{i=1}^{k} B_i$. For any $\sigma > 0$, if $A_i \preccurlyeq \sigma \cdot B_i$ for all $i \in [1:k]$, then $A \preccurlyeq \sigma \cdot B$.*

7 Preconditioning and Sparsifying Weighted Graphs

In this section, we use `Unweighted Sparsify` and a procedure Rewire to both sparsify and ultra-sparsify weighted graphs. One could use `Unweighted Sparsify` directly to sparsify weighted graphs by dividing the edges of the graphs into classes separated by powers of $(1+\epsilon)^i$, and then applying this sparsifier separately on each class. However, the graphs output by this procedure would have degree depending on the number of weight classes. Instead, we state an algorithm `Sparsify` that outputs a graph with a number of edges that may be bounded without reference to the number of weight classes. We then apply `Sparsify` in algorithm `Ultra-Sparsify`. We remark that if one merely desired an algorithm for dense graphs that takes time $O(n^2 \log^{O(1)} n \log(1/\epsilon))$ to produce solutions $\tilde{\boldsymbol{x}}$ satisfying $|A\tilde{\boldsymbol{x}} - \boldsymbol{b}| < \epsilon$, it would suffice to apply `Sparsify` to the dense graph, and then apply the preconditioned Conjugate Gradient algorithm using the Conjugate Gradient algorithm as an exact algorithm to solve the inner system (see [ST03] for background).

So that we can state `Rewire` in `Ultra-Sparsify`, we need the following variation of a definition from [ST03]:

Definition 7.1. *For a graph $G = (V, E)$, and another set of edges F, we define an F-decomposition of G to be a pair $(\mathcal{W}, \pi)$ where $\mathcal{W}$ is a collection of subsets of V and π is a map from F into sets or pairs of sets in $\mathcal{W}$ satisfying*

1. *for each set $W_i \in \mathcal{W}$, the graph induced by E on W_i is connected,*
2. *$|W_i \cap W_j| \leq 1$ for all $i \neq j$,*
3. *each edge of E lies in exactly one set in $\mathcal{W}$,*

4. *for each edge in $e \in F$, if $|\pi(e)| = 1$, then both endpoints of e lie in $\pi(e)$; otherwise, one endpoint of e lies in one set in $\pi(e)$, and the other endpoint lies in the other.*

For now, it is probably best to first consider the case in which $E = F$ and all the sets in $\mathcal{W}$ are disjoint, in which case π merely maps each edge to the names of subsets in which its endpoints lie. This is how the definition is used in `Sparsify`. We note that, in general, this definition allows there to be sets $W \in \mathcal{W}$ containing just one vertex of V.

`Rewire`$(A, F, (\{W_1, \ldots, W_l\}, \pi), \widetilde{H})$,

A is the weight matrix of a weighted graph $G = (V, E)$,

F is set of unit-weight edges on V,

$(\{W_1, \ldots, W_l\}, \pi)$ is an F-decomposition of G, and

$\widetilde{H}$ is a weighted graph on vertex set $\{1, \ldots, l\}$ with weight matrix $\widetilde{C}$.

(1) Construct a map τ from $\widetilde{H}$ to F as follows:

  (a) For each $(i, j) \in \widetilde{H}$, choose an arbitrary edge $(u, v) \in F$ with $u \in W_i$, $v \in W_j$ and $\pi(u, v) = \{W_i, W_j\}$. Set $\tau(i, j) = (u, v)$.

(2) For each edge (u, v) in the range of τ, set $\tilde{f}_{u,v} = \sum_{(i,j):\tau(i,j)=(u,v)} \tilde{c}_{i,j}$.

(3) Let $\widetilde{F}$ be the set of all the weighted edges $\tilde{f}_{u,v}$. Output $\widetilde{F}$.

Before analyzing `Rewire`, we define the *weighted length* of a path containing edges of weights $\omega_1, \ldots, \omega_l$ to be $(1/\omega_1 + 1/\omega_2 + \cdots + 1/\omega_l)^{-1}$. In particular, the weighted length of a path is *less* then the weight of each of its edges. We will make use of the following inequality, which may be derived from the Rank-One Support Lemma of [BH]

Lemma 7.2. *Let $u_0, u_1, \ldots, u_l$ be a path in a graph in which the edge from u_i to u_{i+1} has weight ω_i. Let ω be the weighted length of the path. Then, for all $\boldsymbol{x} \in \mathbb{R}^n$,*

$$\omega(x_{u_0} - x_{u_l})^2 \leq \sum_{i=0}^{l-1} \omega_i (x_{u_i} - x_{u_{i+1}})^2.$$

Lemma 7.3 (Rewire). *Let $G = (V, E)$ be a weighted graph with weight matrix A, and let F be a set of weight-1 edges on V. Let $(\{W_1, \ldots, W_l\}, \pi)$ be an F-decomposition of G such that for each $f \in F$, $|\pi(f)| = 2$. Let $\widetilde{H}$ be a weighted graph on $\{1, \ldots, l\}$ with weight matrix $\widetilde{C}$. Let $\widetilde{F}$ be the output of* `Rewire` *on these inputs. Let H be the graph on $\{1, \ldots, l\}$ with weight matrix C such that for $i \neq j$,*

$$c_{i,j} = |\{(u, v) \in F : u \in W_i, v \in W_j, \pi(u, v) = \{W_i, W_j\}\}|$$

For each i, let $d_i = \sum_j c_{i,j}$, the weighted degree of node i in H. Let $d_{max} = \max(d_i)$. Assume that for each i, the induced graph $G(W_i, E)$ contains a vertex w_i such that for each edge $(u_i, u_j) \in F$ such that $u_i \in W_i$ and $\pi(u_i, u_j) = \{W_i, W_j\}$, the weighted length of the path from u_i to w_i is at least γd_{max}. Then

$$\mathcal{L}(F) \preccurlyeq \mathcal{L}(E)\left(1 + \sigma_f(H, \widetilde{H})^2 (1 + 2/(\gamma - 2))\right) + \mathcal{L}(\widetilde{F})\left(\sigma_f(H, \widetilde{H})(1 + 2/(\gamma - 2))^2\right), \qquad (28)$$

and

$$\mathcal{L}(\widetilde{F}) \preccurlyeq \mathcal{L}(E)\left(1+\sigma_f(H,\widetilde{H})^2(1+2/(\gamma-2))\right)+\mathcal{L}(F)\left(\sigma_f(H,\widetilde{H})(1+2/(\gamma-2))^2\right). \tag{29}$$

Proof. We begin by creating a multiset of edges K on $\{w_1,\ldots,w_l\}$. For each $i \neq j$, K contains an edge $k_{i,j}$ with endpoints (w_i,w_j) and weight $c_{i,j}$. We note that K is almost isomorphic to H: the only difference is that some vertices may be identified in K as the $w_1,\ldots,w_l$ are not necessarily distinct. However, by treating K as a multigraph, we have a one-to-one correspondence between the edges of H and K. Let $\widetilde{K}$ be the multigraph on $\{w_1,\ldots,w_l\}$ with edges $\tilde{k}_{i,j}$ having endpoints (w_i,w_j) and weight $\tilde{c}_{i,j}$. We also note that $\widetilde{K}$ is almost isomorphic to $\widetilde{H}$, subject to the same identification of vertices. Thus,

$$\sigma_f(K,\widetilde{K}) \leq \sigma_f(H,\widetilde{H}) \quad \text{and} \quad \sigma_f(\widetilde{K},K) \leq \sigma_f(\widetilde{H},H). \tag{30}$$

As each node in H has weighted degree at most d_{max}, each node in $\widetilde{H}$ has weighted degree at most $d_{max}\sigma_f(\widetilde{H},H)$. Thus, $d_{max}\sigma_f(\widetilde{H},H)$ is an upper bound on the weight of each edge in $\widetilde{H}$, and therefore each on edge in $\widetilde{K}$.

We will now prove that

$$F \preccurlyeq E + \frac{\gamma}{\gamma-2}K. \tag{31}$$

Consider any edge $(u,v) \in F$, and let $u \in W_i$, $v \in W_j$, and $\rho(u,v) = \{W_i,W_j\}$. Let $u = u_0,u_1,\ldots,u_r = w_i$ be a path in $G(W_i,E)$ of weighted length at least γd_{max}, and let $v = v_0,v_1,\ldots,v_s = w_j$ be an analogous path in W_j. The union of a $1/d_{max}$ fraction of these two paths with an edge from w_i to w_j with weight $\gamma/(\gamma-2)$ has weighted length $(1/\gamma+1/\gamma+(\gamma-2)/\gamma)=1$. So, we obtain the inequality

$$\begin{aligned}(x_u-x_v)^2 \leq & \sum_{\nu=0}^{r-1}(a_{x_{u_\nu},x_{u_{\nu+1}}}/d_{max})(x_{u_\nu}-x_{u_{\nu+1}})^2 \\ & + \sum_{\nu=0}^{s-1}(a_{x_{v_\nu},x_{v_{\nu+1}}}/d_{max})(x_{v_\nu}-x_{v_{\nu+1}})^2 \\ & + (\gamma/(\gamma-2))(x_{w_i}-x_{w_j})^2.\end{aligned}$$

Summing these inequalities over all edges $a_{u,v} \in F$, we obtain $F \preccurlyeq E + (\gamma/(\gamma-2))K$. We similarly obtain the inequalities

$$\begin{aligned}(xw_i-xw_j)^2 \leq & \sum_{\nu=0}^{r-1}(a_{x_{u_\nu},x_{u_{\nu+1}}}/d_{max})(x_{u_\nu}-x_{u_{\nu+1}})^2 \\ & + \sum_{\nu=0}^{s-1}(a_{x_{v_\nu},x_{v_{\nu+1}}}/d_{max})(x_{v_\nu}-x_{v_{\nu+1}})^2 \\ & + (\gamma/(\gamma-2))(x_u-x_v)^2,\end{aligned}$$

which when summed over all $a_{u,v} \in F$, implies

$$K \preccurlyeq E + (\gamma/(\gamma-2))F. \tag{32}$$

We will next prove

$$\widetilde{K} \preccurlyeq \sigma_f(\widetilde{H}, H)E + (\gamma/(\gamma-2))\widetilde{F}. \tag{33}$$

For any edge $\tilde{k}_{i,j} \in \widetilde{K}$, let $(u, v) = \tau(i, j)$. The weight of $\tilde{f}_{u,v}$ will be the sum of the weights of all such edges $\tilde{k}_{i,j}$. For this $\tilde{k}_{i,j}$, let $u = u_0, \ldots, u_r = w_i$ be a path in W_i of weighted length at least γd_{max} and let $v = v_0, \ldots, v_s = w_j$ be an analogous path in W_j. As $\tilde{k}_{i,j} \leq d_{max}\sigma_f(\widetilde{H}, H)$, the weighted length of the union of $\sigma_f(\widetilde{H}, H)$ times these two paths with an edge from u to v with weight $(\gamma/(\gamma-2))\tilde{k}_{i,j}$ is at least $\tilde{k}_{i,j}$. Thus, we obtain the inequality

$$\begin{aligned}\tilde{k}_{i,j}(x_{w_i} - x_{w_j})^2 \leq &\sum_{\nu=0}^{r-1} \sigma_f(\widetilde{H}, H) a_{x_{u_\nu}, x_{u_{\nu+1}}}(x_{u_\nu} - x_{u_{\nu+1}})^2 \\ &+ \sum_{\nu=0}^{s-1} \sigma_f(\widetilde{H}, H) a_{x_{v_\nu}, x_{v_{\nu+1}}}(x_{v_\nu} - x_{v_{\nu+1}})^2 \\ &+ (\gamma/(\gamma-2))\tilde{k}_{i,j}(x_u - x_v)^2.\end{aligned}$$

Recalling that no edge of E lies in two sets in $\mathcal{W}$, we see that the sum of these inequalities over all $(i, j) \in \widetilde{K}$ yields (33). We may similarly obtain the inequality

$$\begin{aligned}\tilde{k}_{i,j}(x_u - x_v)^2 \leq &\sum_{\nu=0}^{r-1} \sigma_f(\widetilde{H}, H) a_{x_{u_\nu}, x_{u_{\nu+1}}}(x_{u_\nu} - x_{u_{\nu+1}})^2 \\ &+ \sum_{\nu=0}^{s-1} \sigma_f(\widetilde{H}, H) a_{x_{v_\nu}, x_{v_{\nu+1}}}(x_{v_\nu} - x_{v_{\nu+1}})^2 \\ &+ (\gamma/(\gamma-2))\tilde{k}_{i,j}(x_{w_i} - x_{w_j})^2,\end{aligned}$$

which when summed over all $(i, j) \in \widetilde{K}$ yields

$$\widetilde{F} \preccurlyeq \sigma_f(\widetilde{H}, H)E + (\gamma/(\gamma-2))\widetilde{K}. \tag{34}$$

Inequality (28) now follows from inequalities (31), (30), and (33). Inequality (29) similarly follows from inequalities (32), (30), and (34).

$\square$

$\texttt{Sparsify}(A, \epsilon)$,
where A is the weight matrix of a weighted graph $G = (V, E)$ normalized to have maximum weight 1.

(0) Set $\gamma = 2 + 4/\epsilon$.

(1) Partition the edges into classes so that class C^t contains all edges with weights in the range $((1+\epsilon)^{-t-1}, (1+\epsilon)^{-t}]$.

(2) For $t = 0, \ldots,$

  (a) Let $\{W_1, \ldots, W_l\}$ be the partition of V obtained by contracting all edges in classes with index less that $t - \log_{1+\epsilon}(\gamma nm \lg(n)/\epsilon^3)$.

  (b) Let H^t be the graph on $\{1, \ldots, l\}$ such that for each $i \neq j$, the weight of $h^t_{i,j}$ is $(1+\epsilon)^{-t} \left|\left\{(u,v) \in C^t : u \in W_i \text{ and } v \in W_j\right\}\right|$.

  (c) Divide the edges in H^t into classes H^t_q of edges of weight $((1+\epsilon)^{t+q-1}, (1+\epsilon)^{t+q})$. Let C^t_q be the set of edges in C^t that are used to make edges in H^t_q.

  (d) For each q, let $\widetilde{H}^t_q = \texttt{Unweighted Sparsify}(H^t_q, \epsilon)$.

  (e) Let $\widetilde{C}^t_q$ be the output of `Rewire` on input C^t_q, $(\{W_1, \ldots, W_l\},)$ and $\widetilde{H}^t_q$.

  (f) Set $\widetilde{C}^t = \cup_q \widetilde{C}^t_q$, and add the edges of $\widetilde{C}^t$ to $\tilde{A}$.

Theorem 7.4 (Sparsify). *Let $\epsilon < 1/2$. **`Sparsify`** can be implemented to have expected running time $O(m \log^{O(1)} m)$. With probability at least $1 - 1/n$ the graph $\tilde{A}$ output by **`Sparsify`** has at most $O(n \log^{O(1)}(n/\epsilon)/\epsilon^2)$ edges and*

$$\sigma_f(A, \tilde{A}) \leq 1 + \epsilon \text{ and } \sigma_f(\tilde{A}, A) \leq 1 + \epsilon. \tag{35}$$

Proof. To establish the bound on the running time, we note that steps (2.c), (2.d) and (2.e) take time quasi-linear in the number of edges in class C_i. All the operations in steps (2.a) and (2.b) over the course of the algorithm can take at most $O(m \log m)$ operations if properly implemented.

Let a and b be constants such that on input ϵ `Unweighted Sparsify` outputs a graph with average degree at most $a \log^b n$ and support ratio $1 + \epsilon$, with probability at least $1 - 1/n^2$. Thus, with probability at least $1 - 1/n$, each of the at most n outputs of `Unweighted Sparsify` satisfy these conditions, and we will perform the remainder of the analysis under the assumption that they do.

To bound the number of edges in the output graph, note that for each $a \log^b n$ edges that we add to $\tilde{A}$, a vertex is contracted out $\log_{1+\epsilon}(\gamma m \lg(n)/\epsilon)$ steps later. Thus, the output graph will have at most

$$n \log_{1+\epsilon}(\gamma m \lg(n)/\epsilon) a \log^b n = n \log^{O(1)}(n/\epsilon)/\epsilon$$

edges.

To prove (35), let A_t be the weighted graph

$$A_t = \sum_{k < t - \log_{1+\epsilon}(\gamma nm \lg(n)/\epsilon^3)} \gamma nm (1+\epsilon)^{-(t-k)} C_k.$$

Note that the weight of each edge in A_t is at least $\gamma nm(1+\epsilon)^{-t}$. Thus, each edge of A_t is at least γnm times the weight of every edge in C^t, and each component of W_r is spanned by such edges. Thus, if we choose any vertex $w_r \in W_r$, each other vertex of W_r is connected to w_r by a path of weighted length at most γm times the largest weight in C^t. So, we may apply Lemma 7.3 to show

$$
\begin{aligned}
C_q^t &\preccurlyeq (1+(1+\epsilon)^2(1+\epsilon))A_t + (1+\epsilon)(1+\epsilon)^2\widetilde{C}_q^t \qquad (36)\\
&\preccurlyeq (2+4\epsilon)A_t + (1+4\epsilon)\widetilde{C}_q^t, \qquad (37)
\end{aligned}
$$

for $\epsilon \le 1/20$. As H^t has at most $\log_{(1+\epsilon)} n \le 2\ln(n)/\epsilon$ weight classes,

$$C^t \preccurlyeq (4/\epsilon)\ln(n)(1+2\epsilon)A_t + (1+4\epsilon)\widetilde{C}^t.$$

Summing these inequalities over all t, we obtain

$$A \preccurlyeq (1+2\epsilon)(4/\epsilon)\ln(n)\left(\sum_t \sum_{k<t-\log_{1+\epsilon}(\gamma nm\ln(n)/\epsilon^3)} (1+\epsilon)^{-(t-k)}\gamma nm\ C_k\right) + (1+5\epsilon)\tilde{A}.$$

As

$$
\begin{aligned}
&\sum_t \sum_{k<t-\log_{1+\epsilon}(\gamma nm\ln(n)/\epsilon^3)} C_k\ \gamma nm(1+\epsilon)^{-(t-k)}\\
&\le \sum_k C_k \sum_{t\ge k+\log_{1+\epsilon}(\gamma nm\ln(n)/\epsilon^3)} \gamma nm(1+\epsilon)^{-(t-k)}\\
&\le \sum_k C_k \left(\epsilon^3/\ln(n)\right)\sum_{i\ge 0}(1+\epsilon)^{-i}\\
&= \sum_k C_k\epsilon^2(1+\epsilon)/\ln(n)\\
&= A\epsilon^2(1+\epsilon)/\ln(n).
\end{aligned}
$$

We thereby obtain the inequality

$$A \preccurlyeq (4\epsilon+8\epsilon^2)A + (1+4\epsilon)\tilde{A},$$

which implies

$$A \preccurlyeq \left(\frac{1+4\epsilon}{1-4\epsilon-8\epsilon^2}\right)\tilde{A}.$$

We may similarly show that

$$\tilde{A} \preccurlyeq \left(\frac{1+4\epsilon}{1-4\epsilon-8\epsilon^2}\right)A.$$

As $\frac{1+4\epsilon}{1-4\epsilon-8\epsilon^2} < 1+11\epsilon$ for $\epsilon < 1/20$, the theorem now follows from setting $\epsilon^* = \epsilon/11$. □

Our ultra-sparsifiers will build upon the low-stretch spanning trees of Alon, Karp, Peleg and West [AKPW95], which we will refer to as AKPW trees. As observed by Boman and Hendrickson [BH], if one runs the AKPW algorithm with the reciprocals of the weights in the graph, then one obtains the following guarantee:

Theorem 7.5 (AKPW). *On input a weighted connected graph G, `AKPW` outputs a spanning tree $T \subseteq G$ such that*

$$\sum_{e \in E} \boldsymbol{wd}_T\,(e) \leq m 2^{O(\sqrt{\log n \log \log n})},$$

where $\boldsymbol{wd}_T\,(e)$ is the reciprocal of the weighted length of the unique path in T connecting the endpoints of e, times the weight of e.

We remark that this algorithm can be implemented to run in time $O(m \log m)$. We also note that if the path in T connecting the endpoints of e has edges with weights $w_1, \ldots, w_l$, then

$$\mathbf{wd}_T\,(e) = \sum_{i=1}^{l} w_e / w_i.$$

$\texttt{Ultra-Sparsify}(A, k)$
where A is the weight matrix of a weighted graph $G = (V, E)$ with maximum weight 1.

(0) $\widehat{A} = \texttt{Sparsify}(A, 1/2)$. let $\widehat{E}$ be the edge set of $\widehat{A}$. Let $\hat{m}$ be the number of edges in $\widehat{E}$.

(1) $T = \texttt{AKPW}(\widehat{A})$.

(2) For every edge $e \in \widehat{E}$, compute $\mathbf{wd}_T(e)$.

Add to $\tilde{A}$ every edge e with $\mathbf{wd}_T(e) > n$. Partition the remaining edges into classes $E_0, \ldots, E_{\log n}$ where E_z contains the edges with $\mathbf{wd}_T(e)$ in the range $[2^z, 2^{z+1})$, and E_0 also contains all edges with $\mathbf{wd}_T(e) < 1$.

(3) For $z = 0, \ldots, \log n$

(a) For $t \geq 1$, let R^t denote the forrest containing the edges in T of weight greater than 2^{-t}. Partition the edges in E_z into classes $C^1, C^2, \ldots$ in which class C^t contains the edges in E_z that go between different trees in R^{t-1} and the same tree in R^t.

(b) For $t = 1, 2, \ldots,$ and $q = -1 - \log_2 z, \ldots, 0, 1, \ldots, 3\log_2 n$,

i. Let C_q^t be the set of edges in C^t with weights in the range $(2^{-t-q}, 2^{-t-q+1}]$.

ii. Apply the algorithm `tree-decomposition` from [ST03] to produce a C_q^t-decomposition of R^t, $(\{W_1, \ldots, W_s\}, \pi)$, such that for each non-singleton set W_i, $\left|\{(u, v) \in C_q^t : W_i \in \pi(u, v)\}\right| \leq 4\hat{m}/k2^z$ and $s \leq \left|C_q^t\right| k2^z/\hat{m}$.

iii. Form the graph H on vertex set $\{1, \ldots, s\}$ by setting the weight of the edge (i, j) to

$$h_{i,j} = \left|\{(u, v) \in C_q^t : u \in W_i, v \in W_j, \pi(u, v) = \{W_i, W_j\}\}\right|$$

iv. if $s > 1$,
Let $\widetilde{H}$ be the output of $\texttt{Unweighted Sparsify}(H, 1/2)$.
Let $\widehat{C}_q^t$ be the output of `Rewire` on inputs R^t, C_q^t, $(\{W_1, \ldots, W_s\}, \pi)$ and $\widetilde{H}$.
Let $\widetilde{C}_q^t$ be the subgraph of C_q^t containing the edges that have non-zero weight in $\widehat{C}_q^t$.
Add the edges of $\widetilde{C}_q^t$ to $\tilde{A}$.

(4) Output $T \cup \tilde{A}$.

Theorem 7.6 (Ultra-Sparsify). *Algorithm `Ultra-Sparsify` can be implemented so that runs in time $O(m \log^{O(1)} m)$. With probability at least $1 - 2^{O(\sqrt{\log n \log\log n})}/n$, the graph $T \cup \tilde{A}$ output by `Ultra-Sparsify` has at most $n - 1 + k2^{O(\sqrt{\log n \log\log n})}$ edges and*

$$\kappa_f(A, \tilde{A}) \leq (n/k) \log^{O(1)} n. \tag{38}$$

Proof. In our analysis, we assume that the call to `Sparsify` and each of the calls to `Unweighted Sparsify` is successful. We will see below that `Unweighted Sparsify` is called at most $2^{O(\sqrt{\log n \log\log n})}k$ times. So, the probability that this assumption is wrong is at most $1 - 2^{O(\sqrt{\log n \log\log n})}/n$. In particular, we will assume that $\hat{m} = n2^{O(\sqrt{\log n \log\log n})}$.

We now justify our claimed bound on the running time. The first complicated task is the computation of $\mathbf{wd}_T(e)$ for each edge $e \in \widehat{E}$. To perform this computation in time $O((n + \left|\widehat{E}\right|) \log n)$, note that every tree contains a *center* vertex whose removal breaks the tree into components having at most $2n/3$ vertices, and that this vertex can be identified in linear time. In time $O(n + \left|\widehat{E}\right|)$, one can compute $\mathbf{wd}_T(e)$ for each edge e whose endpoints lie in different components after the center is removed. One can then remove the center vertex, and recursively apply this computation in the resulting components.

The second complicated task is the partitioning of the edges in E_z into sets $C^1, C^2, \ldots$. This can be accomplished in time $O((n + |E_z|) \log n)$ by a simple merging procedure. The components of R^t are obtained by merging components of R^{t-1}. An edge is eliminated and assigned a class when its two endpoints become part of the same cluster. By only re-labeling the vertices and edges of the smaller cluster in a merge, and measuring size as the sum of vertices and edges, we bound the total work by $O((n + |E_z|) \log n)$.

The implementations of the other steps of the algorithm are straightforward.

To bound the number of edges in $\tilde{A}$, we note that at most $2^{O(\sqrt{\log n \log \log n})}$ edges $e \in \widehat{A}$ have $\mathbf{wd}_T(e) > n$ and are added to $\tilde{A}$ in step (2). By Theorem 7.5,

$$\sum_{e \in E_z} 2^z |E_z| \leq \hat{m} 2^{O(\sqrt{\log n \log \log n})}.$$

If each call to `Unweighted Sparsify` is successful, then the number of edges in $\widetilde{C}_q^t$ is at most $\left(\log^{O(1)} n\right) \left|C_q^t\right| k 2^z / \hat{m}$. So, for each z, the number of edges added is at most

$$\left(\log^{O(1)} n\right) (k 2^z / \hat{m}) \sum_{t,q} \left|C_q^t\right| \leq \left(\log^{O(1)} n\right) (k 2^z / \hat{m}) |E_z|$$

Summing over z, we find that the total number of edges added to $\tilde{A}$ is at most

$$\left(\log^{O(1)} n\right) (k / \hat{m}) \sum_z 2^z |E_z| \leq \left(\log^{O(1)} n\right) 2^{O(\sqrt{\log n \log \log n})} k = 2^{O(\sqrt{\log n \log \log n})} k.$$

We may similarly show that the total number of calls to `Unweighted Sparsify` is at most $2^{O(\sqrt{\log n \log \log n})} k$.

To prove (38), we first note that step (*b.iv*) ensures that $T \cup \tilde{A}$ is a subgraph of $\widehat{A}$, and so $T \cup \tilde{A} \preccurlyeq \widehat{A} \preccurlyeq (3/2) A$, implying

$$\sigma_f(T \cup \tilde{A}, A) \leq 3/2.$$

To complete the proof, we will show that

$$A \preccurlyeq ((n/k) \lg^{O(1)} n)(T \cup \tilde{A}), \tag{39}$$

which implies

$$\sigma_f(A, T \cup \tilde{A}) \leq (n/k) \lg^{O(1)} n.$$

In particular, we will prove for each z that

$$E_z \preccurlyeq ((n/k) \lg^{O(1)} n)(T \cup \tilde{A}),$$

which implies (39) by summing over z. For the rest of the argument, we restric our attention to an arbitrary z.

For each z, we note that each edge in C^t has weight at most 2^{-t+z+2}. To see why this is true, note that the endpoints of each such edge e are connected by a path in T that contains an edge of weight at most 2^{-t+1}. Thus, $2^{z+1} > \mathbf{wd}_T(e) > w_e/2^{-t+1}$. So, each edge in C^t will lie in a class C_q^t for $q \geq -1 - \log_2 z$.

We now define S^t, a weighted sub-graph of R^t, by

$$S^t = (R^t - R^{t-\lg(n)}) + \sum_{i \geq 1} 2^{-i}(R^{t-\lg(n)-i} - R^{t-\lg(n)-i-1}).$$

That is, S^t has the same set of edges as R^t, but every edge in R^t with weight greater than $2^{-t}n$ has weight between $2^{-t}n$ and $2^{-t+1}n$ in S^t. The following critical inequality is immediate from the definition of S^t:

$$\sum_t S^t \preccurlyeq (1 + \lg n)T.$$

We now establish

$$\forall e \in C^t, \quad \mathbf{wd}_{S^t}(e) \leq \mathbf{wd}_{R^t}(e) + 1 \leq 2^{z+1} + 1.$$

To see this, note that the path in S^t connecting the endpoints of e contains a bunch of edges of the same weight as in R^t, plus at most n edges of weight at least n times the weight of e, which can contribute at most 1 to $\mathbf{wd}_{S^t}(e)$.

For each q, let D_q^t be the set of edges e in C_q^t for which $|\pi(e)| = 2$. As the maximum degree of a vertex in H is at most $4\hat{m}/k2^z$, we may apply Lemma 7.3 with $\gamma = 4$ to show that

$$D_q^t \preccurlyeq (128\hat{m}/k)S^t(1 + (3/2)^2 2) + \widehat{C}_q^t((3/2)4) \preccurlyeq (768\hat{m}/k)S^t + 6\widehat{C}_q^t.$$

Let $B_q^t = C_q^t - D_q^t$. To show that

$$B_q^t \preccurlyeq (16\hat{m}/k) \cdot S^t,$$

we apply Lemma 7.2 to obtain an inequality for each edge in B_q^t routed over $(2^{z+1}+1)S^t$. The inequality follows by recalling that each component in S^t will be involved in at most $(4\hat{m}/k2^z)$ of these inequalities. We thus obtain

$$C_q^t \preccurlyeq 784(\hat{m}/k)S^t + 6\widehat{C}_q^t.$$

As

$$\widehat{C}_q^t \leq (4\hat{m}/k2^z)\widetilde{C}_q^t \leq (4\hat{m}/k)\widetilde{C}_q^t,$$

we obtain

$$C_q^t \preccurlyeq 784(\hat{m}/k)S^t + 24(\hat{m}/k)\widetilde{C}_q^t.$$

Summing these inequalities over the $(4\lg n + 1)$ values for $q \leq 3\lg n$, we obtain

$$\sum_{q \leq 3\lg n} C_q^t \preccurlyeq (4\lg n + 1)784 S^t + 24 \sum_{q \leq 3\lg n} \widetilde{C}_q^t.$$

For those edges in C_q^t for $q > 3\lg n$, we note that each of these edges has $\mathbf{wd}_{S^t}(e) \leq 1/n^2$, and there are at most n^2 of them, so they are all supported by S^t. Thus,

$$C^t \preccurlyeq 3921(\hat{m}/k)\lg n S^t + 24(\hat{m}/k) \sum_{q \leq 3\lg n} \widetilde{C}_q^t.$$

Summing over t, we find

$$E_z \preccurlyeq 3921(\hat{m}/k)(\lg n)(\lg n + 1)T + 6(\hat{m}/k)\tilde{A}.$$

By recalling that $\hat{m} = n \log^{O(1)} n$, we complete the proof. $\square$

8 Solving Linear Systems

In this section, we will show how the output of `UltraSparsify` can be used to solve linear systems in A with the preconditioned Chebyshev method and the preconditioned conjugate gradient.

We begin by recalling the basic outline of the use of sparsifiers established by Vaidya [Vai90]. Given a matrix A and an ultra-sparsifier $B = T \cup \tilde{A}$ of A, after appropriatedly reordering the vertices of A and B, we can perform partial Cholesky factorization of B to obtain $B = L[I, 0; 0, A_1]L^T$. Here, L is a lower-triangular matrix with at most $O(n)$ non-zero entries and A_1 is a square matrix of size at most $4\left|\tilde{A}\right|$ with at most $10\left|\tilde{A}\right|$ non-zero entries, where $\left|\tilde{A}\right|$ denotes the number of edges in $\tilde{A}$ (see [ST03, Proposition 1.1]). Moreover, if A is SDD then A_1 is as well.

We can then solve linear systems in B by solving a corresponding linear system in A_1 and performing $O(n)$ additional work: given b, one can solve $By = b$ by solving for s in $[I, 0; 0, A_1]s = L^{-1}b$, and then computing $y = L^{-T}s$ by back-substitution.

If we use the output of `UltraSparsify` with $k = \sqrt{m}$ as a preconditioner and solve systems in A_1 using the conjugate gradient method as an exact solver, we obtain the followin "one-shot" result

Theorem 8.1 (One-Shot Algorithm). *Let A be an n-by-n SDD matrix with m non-zero entries. If one solves A using the preconditioned conjugate gradient using $B = \texttt{UltraSparsify}(A, \sqrt{m})$ as a preconditioner, factors B into $[I, 0; 0, A_1]$, and solves systems in A_1 using conjugate gradient as an exact solver, then one can produce an approximate solution $\tilde{\boldsymbol{x}}$ to the system $A\boldsymbol{x} = \boldsymbol{b}$ with $\left\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\right\|_A \leq \epsilon$ in time $m \log^{O(1)} m + m^{3/4} n^{1/2+o(1)} \log(1/\epsilon)$, with probability $1 - o(1)$.*

Proof. The time taken by `UltraSparsify` is $m \log^{O(1)} m$, and the time taken by the Cholesky factorization is $O(n)$. Having produced B and A_1, the algorithm solves $A\boldsymbol{x} = \boldsymbol{b}$ to accuracy ϵ in A-norm by applying at most $\sqrt{\kappa_f(A, B)} \log(1/\epsilon)$ iterations of the preconditioned conjugate gradient. Using the conjugate gradient as an exact algorithm, we can solve the system in A_1 in time $O(|A_1|^2)$. Thus, each iteration of the PCG takes time $O(m + n + \left|\tilde{A}\right|^2)$. Setting $k = \sqrt{m}$, and assuming that `UltraSparsify` succedes, we obtain $\kappa_f(A, B) \leq n/m^{1/2}$ and $\left|\tilde{A}\right| \leq m^{1/2+o(1)}$. So, the time taken by the PCG algorithm will be $m^{3/4} n^{1/2+o(1)} \log(1/\epsilon)$. $\square$

Alternatively, we may solve the system A_1 by a recursive application of our algorithm. In this case, we let $A_0 = \texttt{Sparsify}(A, 1/2)$, and let B_1 denote the output of `UltraSparsify` on input A_0. Generally, we will let B_{i+1} denote the output of $\texttt{UltraSparsify}(A_i, k_i)$, and let $L_i[D_i, 0; 0, A_i]L_i^T$ be the partial Cholesky factorization of B_i. We will let the recursion depth be r and will specify k_i's later. We solve systems in A_r using an exact method and solve systems in A_i using B_{i+1} preconditioner. At the top level, we will then use A_0 as a preconditioner for A.

In our recursion, all the inner applications of the preconditioned Chebyshev method will run for the same, predetermined, number of iterations. To bound the number of iterations we require, we use the following extension of Joshi ([Jos97]: Corollary 5.5, page 73) of a theorem of Golub and Overton ([GO88], Theorem 2, page 579).

Theorem 8.2 (Preconditioned Inexact Chebyshev Method). *Assume B and A are SDD matrices such that $\sigma(B, A) \geq 1$. Let $\boldsymbol{x}$ be the solution to $A\boldsymbol{x} = \boldsymbol{b}$. Let $\kappa > \kappa_f(A, B)$. Let $\delta = (\kappa - 1)/(200\kappa^2)$. If, in kth iteration of the preconditioned Chebyshev Method when the solution of $B\boldsymbol{z} = \boldsymbol{r}_k$ is needed, a vector $\boldsymbol{z}_k$ is returned satisfying*

$$\left\|\boldsymbol{z}_k - \boldsymbol{z}\right\|_B \leq \delta \left\|\boldsymbol{z}\right\|_B,$$

then $\boldsymbol{x}_k$ generated by the Preconditioned Inexact Chebyshev Method after k iterations satisfies

$$\left\|\boldsymbol{x}_k - \boldsymbol{x}\right\|_A \leq 6\sqrt{\kappa}\left(1 - \frac{1}{\sqrt{\kappa_f(A, B)}}\right)^k \left\|\boldsymbol{x}\right\|_A.$$

Corollary 8.3. *Under the assumptions of Theorem 8.2, for any $\kappa > \max(8, \kappa_f(A, B))$, after $5\sqrt{\kappa_f(A, B)} \ln \kappa$ iterations the Preconditioned Inexact Chebyshev Method outputs an approximate solution $\tilde{\boldsymbol{x}}$ to $A\boldsymbol{x} = \boldsymbol{b}$ with $\left\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\right\|_A \leq \delta \left\|\boldsymbol{x}\right\|_A$.*

Proof. By Theorem 8.2, to find an approximate solution $\tilde{\boldsymbol{x}}$ to $A\boldsymbol{x} = \boldsymbol{b}$ with $\left\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\right\|_A \leq \delta \left\|\boldsymbol{x}\right\|_A$ one can stop the Preconditioned Inexact Chebyshev Method after k iterations provided

$$6\sqrt{\kappa}\left(1 - \frac{1}{\sqrt{\kappa_f(A, B)}}\right)^k \leq \frac{\kappa - 1}{200\kappa^2}.$$

So, it is sufficient to set

$$k = \sqrt{\kappa_f(A, B)} \ln\left(\frac{1200\kappa^2\sqrt{\kappa}}{\kappa - 1}\right) \leq 5\sqrt{\kappa_f(A, B)} \ln \kappa,$$

where last inequality follows from the assumption $\kappa \geq 8$. □

By carefully choosing r and k_i, we obtain the following bound on the time of a recursive algorithm.

Theorem 8.4 (Recursive). *Let A be an n-by-n SDD matrix with m non-zero entries (assuming $n > 2$). Using the recursive algorithm, one can produce an approximate solution $\tilde{\boldsymbol{x}}$ to $A\boldsymbol{x} = \boldsymbol{b}$ with $\left\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\right\|_A \leq \epsilon$ in time*

$$m \log^{O(1)} m + O\left(m \log(1/\epsilon)\right) + n 2^{O(\sqrt{\log n \log\log n})} \log(1/\epsilon)$$

with probability $1 - o(1)$.

Proof. Let c be the constant greater than 1 such that the $\tilde{A}$ output by `UltraSparsify` has at most $k2^{c\sqrt{\log n \log\log n}}$ edges (as in Theorem 7.6), and let a be the constant hidden in the $O(1)$ in (38).

As $A_0 = \texttt{Sparsify}(A, 1/2)$, A_0 will have $n \log^{O(1)} n$ edges. Let c_1 be the constant hidden in the $O(1)$ above, so A_0 has $n \log^{c_1} n$ edges.

Let $\kappa_0 = 2^{(2c)\sqrt{\log n \log\log n}} \log^{2a} n$ and $\kappa = \max(8, \kappa_0)$. We will set r so that

$$n^{1/2r} = \kappa_0^{1/4}.$$

As $c \geq 1$, we have that $r \leq \sqrt{\log n / \log\log n}$. We then let $B_{i+1} = \texttt{UltraSparsify}(A_i, k_i)$, where $k_i = n^{1-(i+1)/r}/2^{c\sqrt{\log n \log\log n}}$. Thus, for $1 \leq i \leq r$, A_i will have at most $n^{1-i/r}$ edges. Let $n_i = n^{1-i/r}$ for $i = 1 : r$ and $n_0 = n$

When solving system $A\boldsymbol{x} = \boldsymbol{b}$ the recursive algorithm defines a sequence of systems in A_i for $0 \leq i \leq r$. As A_r has a constant size, we solve systems in A_r (and hence their corresponding systems in B_r) using a direct method. Systems in A_i (and their corresponding systems in B_i) with $i < r$ are solved approximately using B_{i+1} as the preconditioner.

By Theorem 7.6, we have

$$\kappa_f(A_i, B_{i+1}) \leq (n_i/k_i) \log^a n = n^{1/r} 2^{c\sqrt{\log n \log\log n}} \log^a n = \kappa_0.$$

By Corollary 8.3, in

$$(5 \ln \kappa)\sqrt{\kappa_f(A_i, B_{i+1})} \leq (5 \ln \kappa) n^{1/r}$$

iterations, an approximate solution with error in A_i-norm that is less than $\delta = (\kappa - 1)/(200\kappa^2)$ can be obtained for the system in A_i. From Theorem 8.2 and its Corollary 8.3, we know that this error is small enough that solutions with this error in A_i can be used to by the preconditioned inexact Chebyshev method to solve systems in A_{i-1}.

Let T_i be the time needed to find an approximate solution, whose error in A_i norm is at most δ, to a system in A_i. Let T be the time needed to find an approximate solution $\tilde{\boldsymbol{x}}$ to $A\boldsymbol{x} = \boldsymbol{b}$ such that $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_A \leq \epsilon$, using A_0 as the preconditioner.

By Corollary 8.3, there exists a constant $g > 1$ such that

$$\begin{aligned}
T_r &\leq g \\
T_i &\leq g(n_i + T_{i+1}) n^{1/r} \ln \kappa && r > i \geq 1. \\
T_0 &\leq g(n \log^{c_1} n + T_1) n^{1/r} \ln \kappa \\
T &= O\left(\ln(1/\epsilon)(m + T_0)\right)
\end{aligned}$$

Notice that κ is a funtion in n and is independent of n_i. We now prove by induction that $r \geq i \geq 1$, $T_i \leq (r - i + 1) g^{r-i+1} (\ln \kappa)^{r-i} n^{(r-i+1)/r}$. Our base case is $i = r$ which is clearly true. Assuming that the assertionhas been proved for $i + 1$, we now prove it for T_i:

$$\begin{aligned}
T_i &\leq (g \ln \kappa) n^{1/r} (n_i + T_{i+1}) \\
&\leq (g \ln \kappa) n^{1/r} \left(n^{(r-i)/r} + (r-i) g^{r-i} (\ln \kappa)^{r-i-1} n^{(r-i)/r} \right) \\
&\leq (g \ln \kappa) n^{1/r} \left((r-i+1) g^{r-i} (\ln \kappa)^{r-i-1} n^{(r-i)/r} \right) \\
&\leq (r-i+1) g^{r-i+1} (\ln \kappa)^{r-i} n^{(r-i+1)/r},
\end{aligned}$$

where the second to the last inequality follows from $1 < g^{r-i}(\ln \kappa)^{r-i-1}$.

Thus

$$T_0 = n2^{O\left(\sqrt{\log\log\log n}\right)} \quad \text{and} \quad T = O\left(m \log(1/\epsilon)\right) + n2^{O\left(\sqrt{\log\log\log n}\right)} \log(1/\epsilon).$$

Since $\texttt{Sparsify}(A, 1/2)$ takes $m \log^{O(1)} m$, the total time required to find $\tilde{\boldsymbol{x}}$ with $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_A \leq \epsilon$ is

$$m \log^{O(1)} m + O\left(m \log(1/\epsilon)\right) + n2^{O\left(\sqrt{\log\log\log n}\right)}.$$

□

References

[AKPW95] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, February 1995.

[AM01] Dimitris Achlioptas and Frank McSherry. Fast computation of low rank matrix approximations. In ACM, editor, *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing: Hersonissos, Crete, Greece, July 6–8, 2001*, pages 611–618, New York, NY, USA, 2001. ACM Press.

[BCHT] Erik Boman, Doron Chen, Bruce Hendrickson, and Sivan Toledo. Maximum-weight-basis preconditioners. *to appear in Numerical Linear Algebra and Applications.*

[BGH$^+$] M. Bern, J. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo. Support-graph preconditioners. *submitted to SIAM J. Matrix Anal. & Appl.*

[BH] Erik Boman and B. Hendrickson. Support theory for preconditioning. *submitted to SIAM J. Matrix Anal. & Appl (Revised 10/02).*

[BH01] Erik Boman and B. Hendrickson. On spanning tree preconditioners. Manuscript, Sandia National Lab., 2001.

[BK96] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $O(n^2)$ time. In *Proceedings of The Twenty-Eighth Annual ACM Symposium On The Theory Of Computing (STOC '96)*, pages 47–55, New York, USA, May 1996. ACM Press.

[Bru95] Are Magnus Bruaset. *A Survey of Preconditioned Iteratitve Methods*. Longman Scientific and Technical, 1995.

[FK81] Z. Füredi and J. Komlós. The eigenvalues of random symmetric matrices. *Combinatorica*, 1(3):233–241, 1981.

[FKV98] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. In IEEE, editor, *39th Annual Symposium on Foundations of Computer Science: proceedings: November 8–11, 1998, Palo Alto, California*, pages 370–378, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1998. IEEE Computer Society Press.

[GMZ95] Keith Gremban, Gary Miller, and Marco Zagha. Performance evaluation of a new parallel preconditioner. In *9th IPPS*, pages 65–69, 1995.

[GO88] G. H. Golub and M. Overton. The convergence of inexact Chebychev and Richardson iterative methods for solving linear systems. *Numerische Mathematik*, 53:571–594, 1988.

[Gre96] Keith Gremban. *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*. PhD thesis, Carnegie Mellon University, CMU-CS-96-123, 1996.

[GV89] G. H. Golub and C. F. Van Loan. *Matrix Computations, 2nd. Edition*. The Johns Hopkins University Press, Baltimore, MD, 1989.

[HL94] B. Hendrickson and R. Leland. The chaco user's guide, version 2.0. Tech. Rep. SAND94-2692, Sandia National Labs, Albuquerque, NM, October 1994.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[Jos97] Anil Joshi. *Topics in Optimization and Sparse Linear Systems*. PhD thesis, UIUC, 1997.

[KK98] George Karypis and Vipin Kumar. *MeTis: Unstrctured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0*, September 1998.

[LR99] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, November 1999.

[LS90] L. Lovasz and M. Simonovits. The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume. In IEEE, editor, *Proceedings: 31st Annual Symposium on Foundations of Computer Science: October 22–24, 1990, St. Louis, Missouri*, volume 1, pages 346–354, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1990. IEEE Computer Society Press.

[LS93] Lovasz and Simonovits. Random walks in a convex body and an improved volume algorithm. *RSA: Random Structures & Algorithms*, 4:359–412, 1993.

[MMP+02] Bruce M. Maggs, Gary L. Miller, Ojas Parekh, R. Ravi, and Shan Leung Maverick Woo. Solving symmetric diagonally-dominant systems by preconditioning. 2002.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[Rei98] John Reif. Efficient approximate solution of sparse linear systems. *Computers and Mathematics with Applications*, 36(9):37–58, 1998.

[SJ89] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82(1):93–133, July 1989.

[ST03] Daniel Spielman and Shang-Hua Teng. Solving sparse, symmetric, diagonally-dominant linear systems in time $o(m^{1.31})$. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 416–427, 2003. Most recent version available at `http://arxiv.org/cs.DS/0310036`.

[Vai90] Pravin M. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. Unpublished manuscript UIUC 1990. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991, Minneapolis., 1990.

A Algebraic Facts

Proposition A.1. *If D is a non-negative diagonal matrix and A is a symmetric matrix, then the sets of eigenvalues of $D^{-1}A$ and $D^{-1/2}AD^{-1/2}$ are identical. If L and $\tilde{L}$ are symmetric positive-semi definite matrices with identical null-spaces, then*

$$\sigma_f(D^{-1}L, D^{-1}\tilde{L}) = \sigma_f(D^{-1/1}LD^{-1/2}, D^{-1/2}\tilde{L}D^{-1/2}) = \sigma_f(L, \tilde{L})$$

Proof. The first fact is standard. The second follows from [BH, Proposition 3.12]. □

Proposition A.2. *Let M be a matrix and let D_A and D_B be non-negative diagonal matrices such that $D_A \geq D_B$. Then,*

$$\lambda_{max}(D_A^{-1}M) \leq \lambda_{max}(D_B^{-1}M).$$

B A Hoeffding Bound

The following lemma is sometimes attributed to Hoeffding [Hoe63]. However, its proof does not appear in his work. We prove it by following the exposition of Motwani and Raghavan [MR95]

Lemma B.1 (A Hoeffding Bound). *Let $\alpha_1, \ldots, \alpha_n$ all lie in $[0,1]$ and let let $X_1, \ldots, X_n$ be independent random variables such that X_i equals α_i with probability p_i and 0 with probability $1 - p_i$. Let $X = \sum_i X_i$ and $\mu = \mathbf{E}[X] = \sum \alpha_i p_i$. Then,*

$$\Pr[X > (1+\delta)\mu] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}.$$

For $\delta < 1$, we remark that this probability is at most $e^{-\mu\delta^2/3}$. Also, for $\delta < 1$,

$$\Pr[X < (1-\delta)\mu] < e^{-\mu\delta^2/2}.$$

Proof. Applying Markov's inequality and using the fact that the X_is are independent, we have that for all $t > 0$

$$\begin{aligned}
\Pr\left[X > (1+\delta)\mu\right] &< \frac{\prod \mathbf{E}\left[\exp(tX_i)\right]}{\exp(t(1+\delta)\mu)} \\
&= \frac{\prod \left(p_i e^{\alpha_i t} + 1 - p_i\right)}{\exp(t(1+\delta)\mu)} \\
&\leq \frac{\prod \left(\exp(p_i \left(e^{\alpha_i t} - 1\right))\right)}{\exp(t(1+\delta)\mu)}, && \text{applying } 1 + x \leq e^x \text{ with } x = p_i\left(e^{\alpha_i t} - 1\right), \\
&\leq \frac{\prod \left(\exp(p_i \alpha_i \left(e^{t} - 1\right))\right)}{\exp(t(1+\delta)\mu)}, && \text{as } \alpha_i \leq 1, \\
&= \frac{\exp(\mu \left(e^t - 1\right))}{\exp(t(1+\delta)\mu)}, \\
&\leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu,
\end{aligned}$$

by the choice of $t = \ln(1+\delta)$. To bound this last term for $\delta < 1$, we take the Taylor series $(1+\delta)\ln(1+\delta)$, and observe that in this case it is alternating and decreasing after the first term, and so we may bound

$$(1+\delta)\ln(1+\delta) \geq \delta + \delta^2/2 - \delta^3/6 = \delta + \delta^2/3.$$

The other inequality follows from a similar argument using

$$\Pr\left[X > (1-\delta)\mu\right] < \frac{\prod \mathbf{E}\left[\exp(-tX_i)\right]}{\exp(-t(1+\delta)\mu)}$$

by applying the identity $e^{-at} - 1 \leq a(e^{-t} - 1)$ and setting $t = \ln(1/(1-\delta))$. $\square$