AN EFFICIENT ALGORITHM FOR UNWEIGHTED SPECTRAL GRAPH SPARSIFICATION

DAVID G. ANDERSON*, MING GU*, AND CHRISTOPHER MELGAARD*

Abstract. Spectral graph sparsification has emerged as a useful tool in the analysis of large-scale networks by reducing the overall number of edges, while maintaining a comparable graph Laplacian matrix. In this paper, we present an efficient algorithm for the construction of a new type of spectral sparsifier, the unweighted spectral sparsifier. Given a general unweighted graph $G = (V, E)$ and an integer $\ell < |E|$ (the number of edges in $E$), we compute an unweighted graph $H = (V, F)$ with $F \subseteq E$ and $|F| = \ell$ such that for every $x \in \mathbb{R}^V$

$$\frac{x^T L_G x}{\kappa} \leq x^T L_H x \leq x^T L_G x,$$

where $L_G$ and $L_H$ are the Laplacian matrices for $G$ and $H$, respectively, and $\kappa \geq 1$ is a slowly-varying function of $|V|$, $|E|$ and $\ell$. This work addresses an open question of the existence of unweighted graph sparsifiers for unweighted graphs [2]. Additionally, our algorithm can efficiently compute unweighted graph sparsifiers for weighted graphs, leading to sparsified graphs that retain the weights of the original graphs.

Key words. graph sparsification, spectral graph theory, spectral sparsification, unweighted graph sparsification

AMS subject classifications. 68R10, 90C35, 15A18, 15B34, 15B48

1. Introduction. Graph sparsification seeks to approximate a graph $G$ with a graph $H$ on the same vertices, but with fewer edges. Called a sparsifier, $H$ serves as a proxy for $G$ for computations for which $G$ is too large. Sparsifiers also require less storage than the graphs they approximate, evoking their effectiveness in wide-ranging applications of graphs, including social networks, conductance, electrical networks, and similarity [6, 7, 11, 14]. In some applications, graph sparsification improves the quality of the graph, such as in the design of information networks and the hardwiring of processors and memory in parallel computers [3, 9]. Sparsifiers have also been utilized to find approximate solutions of symmetric, diagonally-dominant linear systems in nearly-linear time [8, 13, 14, 15].

Recent work on graph sparsification includes [1, 8, 12, 14, 16]. Batson, Spielman, and Srivastava [2] prove that for every graph there exists a spectral sparsifier where the number of edges is linear in the number of vertices. They further provide a polynomial-time, deterministic algorithm for the sparsification of weighted graphs, which may produce weights that differ greatly from the weights of the original graphs. Our concept of unweighted sparsification, nevertheless, applies to both weighted and unweighted graphs.

Our work introduces a deterministic, greedy algorithm to calculate sparsifiers for weighted and unweighted graphs. Our algorithm selects a subset of edges for the sparse approximation $H$, without assigning or altering weights. Consequently, the algorithm produces an unweighted sparsifier for an unweighted input graph, and can create a weighted sparsifier for a weighted input graph by assigning the original edge weights to the sparsifier. To formalize:

Definition 1.1. Let $G = (V, E, w)$ be a given graph. We define an unweighted
A sparsification of $G$ to be any graph of the form $H = (V, F, w \odot I_F)$, where

$$I_F(e) = \begin{cases} 1, & \text{if } e \in F \\ 0, & \text{otherwise} \end{cases}$$

is the indicator function and $\odot$ is the Hadamard product, i.e.

$$(w \odot I_F)(e) = \begin{cases} w_e, & \text{if } e = (u, v) \in F \\ 0, & \text{otherwise} \end{cases}.$$ 

Several definitions have been proposed for the notion in which a sparsifier approximates a dense graph. Benčúr and Karger [4] introduced cut sparsification, where the sum of the weights of the edges of a cut dividing the set of vertices is approximately the same for the dense graph and the sparsifier. Spielman and Teng [16] proposed spectral sparsification, a generalization of cut sparsification, which seeks sparsifiers with a Laplacian matrix close to that of the input graph. We follow the work of [2, 16] and base our work on spectral sparsification, for which we now present a rigorous definition.

Given a graph $G = (V, E, w)$, define a matrix $B_G \in \mathbb{R}^{E \times V}$

$$(B_G)_{ej} = \begin{cases} -1, & \text{if } e = (u, v) \in E \text{ and } j = u \in V \\ 1, & \text{if } e = (u, v) \in E \text{ and } j = v \in V, \\ 0 & \text{otherwise} \end{cases}$$

where $e = (u, v) \in E$ is an edge from $u$ to $v$, and a diagonal weight matrix $W_G \in \mathbb{R}^{E \times E}$

$$(W_G)_{ij} = \begin{cases} w_e, & \text{if } e = (i, j) \in E, \\ 0, & \text{otherwise} \end{cases}.$$ 

The Laplacian of the graph is

$$L_G = B_G^T W_G B_G.$$ 

Note that

$$x^T L_G x = \sum_{(u, v) \in E} w_{(u, v)} (x_u - x_v)^2$$

for a vector $x \in \mathbb{R}^{|V|}$. Given graphs $X$ and $Y$ defined on the same set of nodes

$$L_X \preceq L_Y \quad \text{if and only if} \quad x^T L_X x \leq x^T L_Y x, \quad \text{for all } x.$$ 

**Definition 1.2.** The graph $H$ is a $\kappa$-approximation of $G$ if

$$\frac{1}{\kappa} L_G \preceq L_H \preceq L_G.$$ 

Because our unweighted sparsification does not change the weights of the edges kept in $H$, it is immediate that $L_H \preceq L_G$:

**Proposition 1.3.** If $H$ is an unweighted sparsification of $G$, then

$$L_H \preceq L_G.$$
Proof. 

\[ x^T L_H x = \sum_{(u,v) \in F} w_{(u,v)} (x_u - x_v)^2 \]

\[ \leq \sum_{(u,v) \in F} w_{(u,v)} (x_u - x_v)^2 + \sum_{(u,v) \in E \setminus F} w_{(u,v)} (x_u - x_v)^2 \]

\[ = x^T L_G x \]

for all \( x \in \mathbb{R}^V \).

Our algorithm does not operate directly on the Laplacian matrix. Rather, we consider the SVD of \( W_G^{1/2} G B_G \).

(1.1) \[ W_G^{1/2} G B_G = U_G^T \Sigma_G V_G, \]

where \( \Sigma_G \) is a diagonal matrix containing all non-zero singular values of \( W_G^{1/2} G B_G \); and where \( U_G \in \mathbb{R}^{n \times m} \) is a row orthonormal matrix, with \( n = |E| - r \), and \( r \) being the number of connected components in \( G \). For the unweighted graph, \( W_G \) is simply the identity matrix. \( U_G \) plays a similar role to that of the matrix \( V_{n \times m} \) in [2]. Our algorithm utilizes the column-orthogonality of \( U_G^T \), highlighting the reason for not working directly with the Laplacian matrix.

We are now in a position to present our main results. In section 2 we present the unweighted column selection algorithm, as well as the spectral bounds for the sparsifiers it calculates. Section 3 provides supporting theory. Some observations and concluding remarks are offered in sections 4 and 5.

2. The Unweighted Column Selection Algorithm. Our algorithm decides which edges to keep solely using the matrix \( U_G \). More specifically, it will make decisions based on the columns \( u_i \) of \( U_G \)

\[ U_G = (u_1 \ u_2 \ \cdots \ u_m) \in \mathbb{R}^{n \times m}, \]

where \( m = |E| \) is the number of edges, and \( n = |V| \) is the number of vertices. Therefore, the edges of \( G \) that are kept are exactly the columns of \( U_G \) that we select. Let us define the number of edges kept to be \( \ell \) such that

\[ \ell = |\Pi_t| = |F| \] for \( t \geq 0 \).

We propose the following greedy algorithm for column selection on \( U_G \). Initially set \( A_0 = 0_{n \times n} \) and \( \Pi_0 = \emptyset \), and choose a constant \( T > 0 \). At step \( t \geq 0 \):

- Solve for the unique \( \lambda < \lambda_{\min}(A_t) \) such that

(2.1) \[ \text{tr}(A_t - \lambda I) = T. \]

- Solve for the unique \( \hat{\lambda} \in (\lambda, \lambda_{\min}(A)) \) such that

(2.2) \[ (\hat{\lambda} - \lambda) \left( m - t + \sum_{j=1}^{n} \frac{1 - \lambda_j}{\lambda_j - \lambda} \right) = \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \lambda)}. \]

where \( \lambda_j \) is the \( j^{th} \) largest eigenvalue of the symmetric matrix \( A_t \).
Find an index $i \notin \Pi_t$ such that
\begin{equation}
\text{tr} \left( A_t - \hat{\lambda}I + u_iu_i^T \right)^{-1} \leq \text{tr} \left( A_t - \lambda I \right)^{-1}.
\end{equation}

• Update $A_t$ and $\Pi_t$.

While equations (2.1) and (2.2) are relatively straightforward to justify and solve, equation (2.3) requires careful consideration, and is the focus of much of section 3. Note that equation (2.1) can be solved in $O(n^3)$ operations, equation (2.2) in $O(n)$ operations, and equation (2.3) in $O(n^2m)$ operations. This last complexity count follows because testing the inequality scales with $O(n^2)$, and potentially all remaining indices must be tested. Thus the total complexity of selecting $\ell$ columns is $O(\ell n^2m)$.

While this procedure will work for any $T > 0$, we will show that an effective choice is
\begin{equation}
T = \hat{T}^* \left( 1 + F \left( \hat{T}^* \right) \right),
\end{equation}
where
\[
F \left( \hat{T} \right) = \left[ \left( 1 - \frac{n}{T} \right) \frac{\ell}{m - \frac{\ell - 1}{2} + \hat{T} - n} \right],
\]
and where $\hat{T}^*$ is the minimizer of $F \left( \hat{T} \right)$, given as
\[
\hat{T}^* = \frac{n \left( m + \frac{\ell + 1}{2} - n \right) + \sqrt{n \ell \left( m - \frac{\ell - 1}{2} \right) \left( m + \frac{\ell + 1}{2} - n \right)}}{\ell - n}.
\]

Our spectral bounds are derived using this choice of $T$. We summarize this procedure in the Unweighted Column Selection algorithm.

**Algorithm: Unweighted Column Selection (UCS)**

**Inputs:** $G = (V, E, w)$, $T > 0$, $\ell$.

**Outputs:** $H_{uw} = (V, F, w \odot I_F)$

1: Calculate the column-orthogonal matrix $U_G^T$
2: Set $A_0 = 0_{n \times n}$, $\Pi_0 = \emptyset$
3: for $t = 0, \ldots, \ell - 1$ do
4: Solve for $\lambda$ using equation (2.1)
5: Calculate $\hat{\lambda}$ using equation (2.2)
6: Find $i \notin \Pi_t$ such that inequality (2.3) is satisfied
7: Update $A_{t+1} = A_t + u_iu_i^T$
8: Update $\Pi_{t+1} = \Pi_t \cup \{i\}$
9: end for
10: Let $F = \Pi_\ell$ be the selected edges

Theorem 2.1 below confirms the correctness of the Unweighted Column Selection Algorithm. This theorem, along with other properties of the UCS algorithm, will be discussed and proved in Section 3.

**Theorem 2.1.** Let $G = (V, E, w)$ and let $n < \ell < m$. Then the sparsified graph $H$ produced by the UCS algorithm satisfies
\begin{equation}
\frac{1}{\kappa} L_G \preceq L_H \preceq L_G.
\end{equation}
where
\begin{equation}
\frac{1}{\kappa} = \frac{(\ell - n)^2}{\left( \sqrt{n \left( m + \sqrt{\ell} - n \right)} + \sqrt{\ell \left( m - \sqrt{\ell} \right)} \right)^2 + (\ell - n)^2}.
\end{equation}

For the purpose of comparison, we now briefly discuss the weighted sparsification algorithm of [2]. More recently, their algorithm has been generalized in [5] to compute the CX decompositions. Given a weighted graph $G = (V, E, w)$, this algorithm produces a sparsified graph $H_w = (V, F, \hat{w})$, where $F$ is a subset of $E$ and $\hat{w}$ contains new edge weights, such that
\begin{equation}
L_G \preceq L_{H_w} \preceq \left( \frac{\sqrt{d} + 1}{\sqrt{d} - 1} \right)^2 L_G,
\end{equation}
where the parameter $d$ is defined via the equation $\ell = \lceil d(n - 1) \rceil$.

By choosing $d$ to be a moderate and dimension-independent constant, equation 2.7 asserts that every graph $G = (V, E, w)$ has a weighted spectral sparsifier with a number of edges linear in $|V|$. This strong result, nevertheless, is obtained by allowing unrestricted changes in the graph weights. Such changes may be undesirable, especially if $G$ is unweighted, and the UCS algorithm may be preferred.

To compare the effectiveness of these two types of sparsifiers, we simplify equation (2.6):
\begin{equation}
\frac{1}{\kappa} \approx \frac{\left( \sqrt{d} - 1 \right)^2}{m/n + d/2 + \left( \sqrt{d} - 1 \right)^2}.
\end{equation}

It follows that for $\kappa = \Theta(1)$, a dimension-independent constant, we must choose $d = \Theta(m/n)$. This is the price one must pay to retain the original weights. For $d \ll m/n$, the UCS algorithm computes a sparsified graph with a $\kappa$ that grows at most linearly with $m/n$.

3. Correctness Analysis of the UCS Algorithm. The ultimate goal of this section is to prove Theorem 2.1. To this end, we first show that the UCS algorithm can find a matrix $A_\ell$ with desired properties. We accomplish this through a number of lemmas.

3.1. The Existence of a Solution to Equation (2.3). We first show that equation (2.3) always has a solution.

**Lemma 3.1.** At a given iteration $t$ in the UCS algorithm, at step 6 define
\begin{equation}
f(x) \overset{\text{def}}{=} (x - \lambda) \left[ m - t + \sum_{j=1}^{n} \frac{1 - \lambda_j}{\lambda_j - \lambda} \right] - \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - x)}.
\end{equation}

Then there exists $\hat{\lambda}$, with $\lambda < \hat{\lambda} < \lambda_n$, such that $f(\hat{\lambda}) = 0$. Furthermore,
\begin{equation}
0 < (\hat{\lambda} - \lambda) \left[ \frac{\sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})}}{\sum_{j=1}^{n} \frac{1}{(\lambda_j - \lambda)(\lambda_j - \lambda)}} - (\hat{\lambda} - \lambda) \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \lambda)} \right].
\end{equation}
Proof. Clearly \( f(\lambda) < 0 \). Although \( f \) is undefined at \( \lambda_n \), let \( \lambda^\epsilon_n := \lambda_n - \epsilon \), where \( \epsilon > 0 \). Note that

\[
\lim_{\epsilon \to 0^+} \left( \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \lambda^\epsilon_n)} \right) / \left( \sum_{j=1}^{n} \frac{1}{(\lambda_j - \lambda)(\lambda_j - \lambda^\epsilon_n)} \right) = 1 - \lambda_n
\]

because the last term in each sum will dominate the rest of the sum. Furthermore,

\[
\lim_{\epsilon \to 0^+} (\lambda^\epsilon_n - \lambda) \left[ m - t + \sum_{j=1}^{n} \frac{1 - \lambda_j}{\lambda_j - \lambda} \right] = 1 - \lambda_n + \lim_{\epsilon \to 0^+} (\lambda^\epsilon_n - \lambda) \left[ m - t + \sum_{j=1}^{n-1} \frac{1 - \lambda_j}{\lambda_j - \lambda} \right] > 1 - \lambda_n.
\]

Hence for small \( \epsilon > 0 \), we have \( f(\lambda^\epsilon_n) > 0 \), and, therefore, \( \hat{\lambda} \) exists, with \( \lambda < \hat{\lambda} < \lambda_n \), and \( f(\hat{\lambda}) = 0 \) via Intermediate Value Theorem. Note that if there exists \( 0 \leq \gamma < n \) such that \( \lambda_\gamma = \lambda_{\gamma+1} = \cdots = \lambda_n \), then we repeat the same argument replacing the expression \( 1 - \lambda_n \) with \( \sum_{j=\gamma}^{n} 1 - \lambda_j = (n - \gamma + 1)(1 - \lambda_n) \).

Now we prove inequality 3.1. We use the following version of the Cauchy-Schwartz formula: for \( a_j, b_j \geq 0 \) then \( (\sum a_j b_j)^2 \leq (\sum a_j^2)(\sum b_j) \). Consequently

\[
\left( \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} \right)^2 \leq \left( \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} \right)^2 \left( 0 + \sum_{j=1}^{n} \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} \right) < \left( \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} \right)^2 \left( \sum_{j=1}^{n} \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} \right) \left( \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} \right),
\]

where the last step comes from \( f(\hat{\lambda}) = 0 \). The strict inequality above holds because \( m - t \geq m - \ell + 1 \geq 1 \). After some simple algebra,

\[
(\hat{\lambda} - \lambda) \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} < \left( \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} \right) / \left( \sum_{j=1}^{n} \frac{1}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} \right),
\]

which implies our desired inequality because \( 0 < \hat{\lambda} - \lambda \). \( \square \)

Next, we show that our algorithm is well defined in the sense we can always find a new index \( i \notin \Pi_t \) for each iteration that satisfies 10.

Lemma 3.2. An index \( i \notin \Pi_t \) can always be found to satisfy line 10 of the UCS algorithm for \( 0 \leq t < \ell \).
Proof. Note the two following partial fraction results

\begin{align}
(3.2) \quad \frac{\hat{\lambda} - \lambda}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} &= \frac{1}{\lambda_j - \hat{\lambda}} - \frac{1}{\lambda_j - \lambda} \\
(3.3) \quad \frac{\hat{\lambda} - \lambda}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)^2} + \frac{1}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} &= \frac{1}{(\lambda_j - \hat{\lambda})^2}.
\end{align}

Using the fact that \( f(\hat{\lambda}) = 0 \) followed by the inequality of Lemma 3.1 we have

\[
(\hat{\lambda} - \lambda) \left[ m - t + \sum_{j=1}^{n} \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} \right]
\]

\[
= \left( \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) / \left( \sum_{j=1}^{n} \frac{1}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) + 0
\]

\[
< \left( \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) / \left( \sum_{j=1}^{n} \frac{1}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right)
\]

\[
+ (\hat{\lambda} - \lambda) \left[ \frac{\sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \lambda)^2}}{\sum_{j=1}^{n} \frac{1}{(\lambda_j - \lambda)(\lambda_j - \lambda)}} - (\hat{\lambda} - \lambda) \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \lambda)} \right]
\]

\[
= \left( \hat{\lambda} - \lambda \right) \sum_{j=1}^{n} \left( \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)^2} + \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right)
\]

\[
\left/ \left( \sum_{j=1}^{n} \frac{1}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) - (\hat{\lambda} - \lambda)^2 \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right)
\]

\[
= \frac{\sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \lambda)} - (\hat{\lambda} - \lambda) \left( \sum_{j=1}^{n} \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} - \sum_{j=1}^{n} \frac{1 - \lambda_j}{\lambda_j - \lambda} \right)}{\sum_{j=1}^{n} \frac{1}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})}} ,
\]

where the last line follows from equations (3.2) and (3.3). After some rearranging:

\[
\left( m - t + \sum_{j=1}^{n} \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} \right) \left( \sum_{j=1}^{n} \frac{\hat{\lambda} - \lambda}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) < \sum_{j=1}^{n} \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})^2}.
\]

This inequality can be rewritten using the trace property \( \text{tr} \left( xy^T \right) = y^T x \) and
the identity \( \sum_{i \in \Pi_t} u_i u_i^T = \sum_{i=1}^m u_i u_i^T - \sum_{i \in \Pi_t} u_i u_i^T = I_n - A_t \):

\[
\left( \sum_{i \in \Pi_t} 1 + u_i^T (A_t - \hat{\lambda} I)^{-1} u_i \right) \left( \text{tr} (A_t - \hat{\lambda} I)^{-1} - \text{tr} (A_t - \lambda I)^{-1} \right)
\]

\[
= \left( m - t + \sum_{i \notin \Pi_t} \text{tr} \left[ (A_t - \hat{\lambda} I)^{-1} u_i u_i^T \right] \right) \left( \sum_{j=1}^n \frac{1}{\lambda_j - \hat{\lambda}} - \sum_{j=1}^n \frac{1}{\lambda_j - \lambda} \right)
\]

\[
= \left( m - t + \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} \right) \left( \sum_{j=1}^n \frac{\hat{\lambda} - \lambda}{(\lambda_j - \hat{\lambda}) (\lambda_j - \lambda)} \right)
\]

\[
< \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)^2}
\]

\[
= \text{tr} \left( (A_t - \hat{\lambda} I)^{-2} (I - A_t) \right)
\]

\[
= \sum_{i \notin \Pi_t} u_i^T (A_t - \hat{\lambda} I)^{-2} u_i.
\]

Moving terms to the right and dividing by \( \left( \text{tr} (A_t - \hat{\lambda} I)^{-1} - \text{tr} (A_t - \lambda I)^{-1} \right) > 0 \)

(because \( \hat{\lambda} > \lambda \)) gives

\[
\sum_{i \notin \Pi_t} \left( \frac{u_i^T (A_t - \hat{\lambda} I)^{-2} u_i}{\text{tr} (A_t - \lambda I)^{-1} - \text{tr} (A_t - \lambda I)^{-1}} - \left( 1 + u_i^T (A_t - \hat{\lambda} I)^{-1} u_i \right) \right) > 0.
\]

For this to be true, there must exist an \( i \notin \Pi_t \) such that

\[
\left( \frac{u_i^T (A_t - \hat{\lambda} I)^{-2} u_i}{\text{tr} (A_t - \lambda I)^{-1} - \text{tr} (A_t - \lambda I)^{-1}} - \left( u_i^T (A_t - \hat{\lambda} I)^{-1} u_i \right) \right) > 1.
\]

This last relation gives

\[
\text{tr} (A_t - \lambda I)^{-1} > \text{tr} (A_t - \hat{\lambda} I)^{-1} - \frac{u_i^T (A_t - \hat{\lambda} I)^{-2} u_i}{1 + u_i^T (A_t - \hat{\lambda} I)^{-1} u_i}
\]

\[
= \text{tr} (A_t - \hat{\lambda} I)^{-1} - \text{tr} \left( \frac{u_i u_i^T (A_t - \hat{\lambda} I)^{-1}}{1 + u_i^T (A_t - \hat{\lambda} I)^{-1} u_i} \right)
\]

\[
= \text{tr} (A_t - \hat{\lambda} I + u_i u_i^T)^{-1}.
\]
Applying equation (3.5) to both sides:

We now state a technical and tedious recurrence relation on $\ell$. Let $\lambda(t)$, $\hat{\lambda}(t)$ and $\lambda_j(t)$ represent the values of $\lambda$, $\hat{\lambda}$ and $\lambda_j$, respectively, determined in iteration $t$. Then note that by the definitions of $\lambda$ and $\hat{\lambda}$ we have

$$\lambda(0) < \hat{\lambda}(0) \leq \lambda(1) < \hat{\lambda}(1) \leq \cdots \leq \lambda(t-1) < \hat{\lambda}(t-1).$$

Define the following quantity and functions respectively

$$\hat{T} \stackrel{\text{def}}{=} T \left(1 - \hat{\lambda}(t-1)\right)$$

$$g(t) \stackrel{\text{def}}{=} \frac{\ell \left(1 - \frac{m}{T}\right)}{m-t+T-n} \quad \text{and} \quad F(\hat{T}) \stackrel{\text{def}}{=} \frac{\ell \left(1 - \frac{m}{T}\right)}{m - \frac{t-1}{2} + T - n} - \frac{n}{T}$$

We now state a technical and tedious recurrence relation on $\hat{\lambda}(t-1)$ in the form of a lemma. This is needed to prove our eigenvalue lower bound.

**Lemma 3.3.** After the last iteration of the UCS algorithm, we have

$$\hat{\lambda}(t-1) \geq \left(1 - \hat{\lambda}(t-1)\right) \left[\frac{\ell}{T} \sum_{t=0}^{\ell-1} g(t) - \frac{n}{T}\right].$$

**Proof.** Remember that $T = \text{tr} \left( A - \lambda(t) I \right)^{-1} = \sum_{j=1}^{n} \frac{1}{\lambda_j(t) - \lambda(t)}$, and note that

$$\frac{1 - \lambda_j(t)}{\lambda_j(t) - \lambda(t)} = \frac{1 - \lambda(t)}{\lambda_j(t) - \lambda(t)} + \frac{\lambda(t) - \lambda_j(t)}{\lambda_j(t) - \lambda(t)} = \frac{1 - \lambda(t)}{\lambda_j(t) - \lambda(t)} - 1.

The equation $f \left(\hat{\lambda}(t)\right) = 0$ gives

$$\left(\hat{\lambda}(t) - \lambda(t)\right) \left(m - t + \sum_{j=1}^{n} \frac{1 - \lambda_j(t)}{\lambda_j(t) - \lambda(t)}\right) = \sum_{j=1}^{n} \frac{1 - \lambda_j(t)}{\lambda_j(t) - \lambda(t)} \left(\lambda_j(t) - \lambda(t)\right).$$

Applying equation (3.5) to both sides:

$$\left(\hat{\lambda}(t) - \lambda(t)\right) \left(m - t + \left(1 - \lambda(t)\right) T - n\right) = 1 - \lambda(t) - \frac{\sum_{j=1}^{n} \frac{1}{\lambda_j(t) - \lambda(t)}}{\sum_{j=1}^{n} \frac{1}{\lambda_j(t) - \lambda(t)}}

\geq 1 - \lambda(t) - \frac{\sum_{j=1}^{n} \frac{1}{\lambda_j(t) - \lambda(t)}}{\sum_{j=1}^{n} \frac{1}{\lambda_j(t) - \lambda(t)}}

= 1 - \lambda(t) - \frac{n}{T}.$$

where the last line was accomplished with the trace property previously indicated and the Sherman-Morrison formula. □

**3.2. Lower Bound on $\lambda_{\min}(A_t)$.** Lemma 3.3 ensures that the UCS algorithm can indeed find all $\ell$ indices. We now estimate an eigenvalue lower bound on $A_t$. Let $\lambda(t)$, $\hat{\lambda}(t)$ and $\lambda_j(t)$ represent the values of $\lambda$, $\hat{\lambda}$ and $\lambda_j$, respectively, determined in iteration $t$. Then note that by the definitions of $\lambda$ and $\hat{\lambda}$ we have

$$\lambda(0) < \hat{\lambda}(0) \leq \lambda(1) < \hat{\lambda}(1) \leq \cdots \leq \lambda(t-1) < \hat{\lambda}(t-1).$$

Define the following quantity and functions respectively

$$\hat{T} \stackrel{\text{def}}{=} T \left(1 - \hat{\lambda}(t-1)\right)$$

$$g(t) \stackrel{\text{def}}{=} \frac{\ell \left(1 - \frac{m}{T}\right)}{m-t+T-n} \quad \text{and} \quad F(\hat{T}) \stackrel{\text{def}}{=} \frac{\ell \left(1 - \frac{m}{T}\right)}{m - \frac{t-1}{2} + T - n} - \frac{n}{T}$$

We now state a technical and tedious recurrence relation on $\hat{\lambda}(t-1)$ in the form of a lemma. This is needed to prove our eigenvalue lower bound.

**Lemma 3.3.** After the last iteration of the UCS algorithm, we have

$$\hat{\lambda}(t-1) \geq \left(1 - \hat{\lambda}(t-1)\right) \left[\frac{\ell}{T} \sum_{t=0}^{\ell-1} g(t) - \frac{n}{T}\right].$$

**Proof.** Remember that $T = \text{tr} \left( A - \lambda(t) I \right)^{-1} = \sum_{j=1}^{n} \frac{1}{\lambda_j(t) - \lambda(t)}$, and note that

$$\frac{1 - \lambda_j(t)}{\lambda_j(t) - \lambda(t)} = \frac{1 - \lambda(t)}{\lambda_j(t) - \lambda(t)} + \frac{\lambda(t) - \lambda_j(t)}{\lambda_j(t) - \lambda(t)} = \frac{1 - \lambda(t)}{\lambda_j(t) - \lambda(t)} - 1.$$
AN ALGORITHM FOR UNWEIGHTED GRAPH SPARSIFICATION

Since
\[
\left( \tilde{\lambda}(t) - \lambda(t) \right) \leq 0, \quad \text{and} \quad \left( \tilde{\lambda}(t) - \lambda(t) \right) \geq \frac{1 - \lambda(t) - \frac{n}{T}}{m - t + (1 - \lambda(t)) T - n},
\]
we have
\[
\hat{\lambda}(\ell - 1) \geq \hat{\lambda}(\ell - 1) + \sum_{t=1}^{\ell-1} \left( \tilde{\lambda}(t) - \lambda(t) \right) - \lambda(0) + \lambda(0)
\]
\[
= \sum_{t=0}^{\ell-1} \left( \tilde{\lambda}(t) - \lambda(t) \right) + \lambda(0)
\]
\[
\geq \sum_{t=0}^{\ell-1} \frac{1 - \lambda(t) - \frac{n}{T}}{m - t + (1 - \lambda(t)) T - n} - \frac{n}{T}
\]
\[
\geq \sum_{t=0}^{\ell-1} \frac{1 - \hat{\lambda}(t) - \frac{n}{T}}{m - t + (1 - \hat{\lambda}(t)) T - n} - \frac{n}{T}.
\]

Inequality (3.7) follows by noting that the terms in the sum are decreasing in \( \lambda(t) \).

The final substitution is necessary because solving the preceding recurrence relation is impractical. To further simplify calculations, we define
\[
\hat{T} := T \left( 1 - \hat{\lambda}(\ell - 1) \right).
\]

Therefore,
\[
\hat{\lambda}(\ell - 1) \geq \left( 1 - \hat{\lambda}(\ell - 1) \right) \sum_{t=0}^{\ell-1} \frac{1 - \frac{n}{T}}{m - t + \hat{T} - n} - \frac{n}{T}
\]
\[
= \left( 1 - \hat{\lambda}(\ell - 1) \right) \left[ \frac{1}{\ell} \sum_{t=0}^{\ell-1} g(t) - \frac{n}{T} \right]. \quad \square
\]

Next, to demonstrate the effectiveness of the algorithm, we derive a lower bound for \( \lambda_n \) after \( \ell \) iterations. This analysis will involve selecting an appropriate \( T \) to maximize the lower bound.

**Lemma 3.4.** If \( \hat{T} > n \), then
\[
\lambda_{\min}(A_{\ell}) \geq \frac{F \left( \hat{T} \right)}{1 + F \left( \hat{T} \right)}.
\]

**Proof.** A key observation is that \( g(t) \) is strictly convex in \( t \), which is easily verified by showing that the second derivative \( \frac{d^2 g}{dt^2}(t) \) is positive by our assumptions that \( \hat{T} > n \) and \( m \geq \ell > t \). Next, we apply Jensen’s Inequality for discrete sums [17].
to the recurrence relation in Lemma 3.3:
\[
\hat{\lambda}(\ell-1) \geq \left(1 - \hat{\lambda}(\ell-1)\right) \left[\frac{1}{\ell} \sum_{t=0}^{\ell-1} \frac{\ell (1 - \frac{m}{T})}{m - t + T - n} - \frac{n}{T}\right]
\]
\[
> \left(1 - \hat{\lambda}(\ell-1)\right) \left[\frac{\ell (1 - \frac{m}{T})}{\ell \sum_{t=0}^{\ell-1} m - t + T - n} - \frac{n}{T}\right] \quad \text{ (strict Jensen's Inequality)}
\]
\[
= \left(1 - \hat{\lambda}(\ell-1)\right) \left[\frac{\ell (1 - \frac{m}{T})}{m - \frac{\ell-1}{2} + T - n} - \frac{n}{T}\right] \overset{\text{def}}{=} \left(1 - \hat{\lambda}(\ell-1)\right) F\left(\hat{T}\right).
\]
Along with \(\lambda_n > \hat{\lambda}\) from Lemma 3.1, this finally leads to
\[
\lambda_{\min} = \lambda_n > \hat{\lambda}(\ell-1) > \frac{F\left(\hat{T}\right)}{1 + F\left(\hat{T}\right)}.
\]
(3.9)

The expression on the right-hand side of (3.9) is monotonically increasing in \(F\). So, maximizing \(F\left(\hat{T}\right)\) will also maximize the lower bound on \(\hat{\lambda}(\ell-1)\).

**Lemma 3.5.** The function \(F(\hat{T})\) is maximized at
\[
\hat{T}^* = \frac{n \left(m + \frac{\ell+1}{2} - n\right) + \sqrt{n\ell \left(m - \frac{\ell-1}{2}\right) \left(m + \frac{\ell+1}{2} - n\right)}}{\ell - n}.
\]

**Proof.** Setting the derivative of \(F\left(\hat{T}\right)\) to zero:
\[
\frac{dF}{d\hat{T}} = \frac{(n - \ell)T^2 + 2n \left(m + \frac{\ell+1}{2} - n\right) T + n \left(m + \frac{\ell+1}{2} - n\right) \left(m - \frac{\ell-1}{2} - n\right)}{\hat{T}^2 \left(m - \frac{\ell-1}{2} - n + \hat{T}\right)^2} = 0.
\]
Solving and picking the desired root of the numerator:
\[
\hat{T}^* = \frac{n \left(m + \frac{\ell+1}{2} - n\right) + \sqrt{n\ell \left(m - \frac{\ell-1}{2}\right) \left(m + \frac{\ell+1}{2} - n\right)}}{\ell - n}.
\]
We see that \(\hat{T}^*\) is the global maximum on the region \(\hat{T} \in (n, \infty)\) via the first derivative test since \(\frac{dF}{d\hat{T}}>0\) for \(n < \hat{T} < \hat{T}^*\) and \(\frac{dF}{d\hat{T}}<0\) for \(\hat{T}^* < \hat{T}\).

We remark that combining (3.4) and (3.9) implies that the UCS algorithm should choose \(T = \hat{T}^* \left(1 + F\left(\hat{T}^*\right)\right)\) for effective column selection. We are now ready to estimate \(\lambda_{\min}(A_\ell)\).

**Theorem 3.6.** If \(T\) is chosen according to Lemma 3.5 in the UCS algorithm, then
\[
\lambda_{\min}(A_\ell) > \frac{1}{\kappa},
\]
where \(\kappa\) is defined in (2.6).
Proof. We wish to apply our choice of \( \hat{T} \) to Lemma 3.4. We satisfy the assumption

\[
\hat{T}^* = \frac{n (m + \ell \frac{1}{2} - n) + \sqrt{n \ell (m - \ell \frac{1}{2}) (m + \ell \frac{1}{2} - n)}}{\ell - n} \geq \frac{n (m - n)}{\ell - n} \geq n.
\]

Therefore, plugging \( \hat{T}^* \) into (3.9) of Lemma 3.4, we arrive at:

\[
\lambda_{\min} (A_\ell) > \frac{F (\hat{T})}{1 + F (\hat{T})} = \frac{(\ell - n)\hat{T} - n (m + \ell \frac{1}{2} - n)}{\hat{T} (m - \ell \frac{1}{2} - n + \hat{T}) + (\ell - n)\hat{T} - n (m + \ell \frac{1}{2} - n)} \geq \frac{(\ell - n)^2}{\left( \sqrt{n (m + \ell \frac{1}{2} - n)} + \sqrt{\ell (m - \ell \frac{1}{2})} \right)^2 + (\ell - n)^2}.
\]

(3.10)

3.3. Correctness of the Unweighted Column Selection Algorithm. We are now in a position to prove Theorem 2.1. Our arguments are similar to those of the weighted sparsifier algorithm in [2].

Proof of Theorem 2.1. By Proposition 1.3, we only need to show \( \frac{1}{\kappa} L_G \preceq L_H \).

Consider the SVD of \( W_G^{1/2} B_G \) in equation (1.1), and let \( x \) be any vector such that \( y = \Sigma_G V_G x \neq 0 \). Then

\[
L_G = B_G^T W_G B_G = V_G^T \Sigma_G^2 V_G,
\]

\[
L_H = B_G^T W_H B_G = B_G^T W_G^{1/2} \Pi \Pi^T W_G^{1/2} B_G
\]

\[
= V_G^T \Sigma_G (U_G \Pi^T \Pi U_G^T) \Sigma_G V_G.
\]

It follows that

\[
\frac{x^T L_H x}{x^T L_G x} = \frac{x^T (V_G^T \Sigma_G (U_G \Pi^T \Pi U_G^T) \Sigma_G V_G) x}{x^T (V_G^T \Sigma_G^2 V_G) x} = \frac{y^T U_G \Pi^T \Pi U_G^T y}{y^T y}.
\]

(3.11)

On the other hand, by construction we have

\[
A_\ell = \sum_{j \in \Pi_\ell} u_j u_j^T = U_G \Pi^T \Pi U_G^T.
\]

With equation (3.11), the Courant-Fisher min-max property gives

\[
\frac{x^T L_H x}{x^T L_G x} = \frac{y^T U_G \Pi^T \Pi U_G^T y}{y^T y} \geq \lambda_{\min} (A_\ell) > \frac{1}{\kappa},
\]

where the last line is due to Theorem 3.6. \( \square \)
4. Relation to the Kadison-Singer Conjecture. Let $p \geq 2$ be an integer, and let $U = (u_1, \cdots, u_m) \in \mathbb{R}^{n \times m}$ be a matrix that satisfies

$$\sum_{k=1}^{n} u_k u_k^T = I, \quad \text{and} \quad \|u_k\|_2 \leq \delta, \quad \text{for} \quad k = 1, \cdots, m,$$

where $0 < \delta < 1$. Equation (4.1) implies that $U$ is a row-orthonormal matrix and that each column of $U$ is uniformly bounded away from 1 in 2-norm. Marcus et al. ([10]) show that there exists a partition

$$P = P_1 \cup \cdots \cup P_p$$

of $\{1, \cdots, n\}$ such that

$$\|U(:, P_k)\|_2 \leq \frac{1}{\sqrt{p}} + \delta, \quad \text{for} \quad k = 1, \cdots, p.$$

When the graph $G$ is sufficiently dense, equation (4.2) implies the existence of an unweighted graph sparsifier (see Batson, et al. [2].)

5. Conclusion. We have presented an efficient algorithm for the construction of unweighted spectral sparsifiers for general weighted and unweighted graphs, addressing the open question of the existence of such graph sparsifiers [2] for general graphs. Our algorithm is supported with theoretical spectral bounds. An important feature of our sparsification algorithm is the deterministic unweighted column selection algorithm on which it is based. This algorithm accepts a sparsity input parameter $\ell$, which allows for an exactly specified number of edges in the sparsifier. Open questions include the existence of a larger lower spectral bound, either with the same $T$ or a new one.

REFERENCES

[1] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In PODS, pages 5–14. ACM, 2012.
[2] J. D. Batson, D. A. Spielman, and N. Srivastava. Twice-ramanujan sparsifiers. SIAM J. Comput., 41(6):1704–1721, 2012.
[3] J. D. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng. Spectral sparsification of graphs: theory and algorithms. Commun. ACM, 56(8):87–94, 2013.
[4] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $O(n^2)$ time. In Gary L. Miller, editor, STOC, pages 47–55. ACM, 1996.
[5] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal column-based matrix reconstruction. CoRR, abs/1103.0995, 2011.
[6] F. Chierichetti, S. Lattanzi, and A. Panconesi. Rumour spreading and graph conductance. In SODA, pages 1657–1663. SIAM, 2010.
[7] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In Lance Fortnow and Salil P. Vadhan, editors, STOC, pages 273–282. ACM, 2011.
[8] M. Kapralov and R. Panigrahy. Spectral sparsification via random spanners. In Shafi Goldwasser, editor, ITCS, pages 393–398. ACM, 2012.
[9] C. Leiserson. Fat-trees: Universal Networks for Hardware-efficient Supercomputing. IEEE Trans. Comput., 34(10):992–991, 1985.
[10] A. W. Marcus, D. A. Spielman, and N. Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison-Singer problem. CoRR, abs/1306.3969, 2014.
[11] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In Chid Apt, Joydeep Ghoel, and Padhraic Smyth, editors, KDD, pages 529–537. ACM, 2011.
[12] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

[13] D. A. Spielman and S.-H. Teng. Solving sparse, symmetric, diagonally-dominant linear systems in time $O(n^{1.31})$. *CoRR*, cs.DS/0310036, 2003.

[14] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC’04*, pages 81–90, 2004.

[15] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *CoRR*, abs/cs/0607105, 2006.

[16] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *CoRR*, abs/0808.4134, 2008.

[17] V. A. Zorich. *Mathematical Analysis I*. Springer, Berlin, 2004.