Constructing Linear-Sized Spectral Sparsification in Almost-Linear Time

Citation for published version:

Digital Object Identifier (DOI):
/10.1137/16M1061850

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
SIAM Journal on Computing

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
CONSTRUCTING LINEAR-SIZED SPECTRAL SPARSIFICATION IN ALMOST-LINEAR TIME∗
YIN TAT LEE† AND HE SUN‡

Abstract. We present an almost-linear time algorithm for constructing a spectral sparsifier
with the number of edges linear in its number of vertices. This improves all previous constructions
of linear-sized spectral sparsifiers, which requires $\Omega(n^2)$ time. A key ingredient in our algorithm is a
novel combination of two techniques used in literature for constructing spectral sparsifiers: random
sampling by effective resistance, and adaptive construction based on barrier functions.

Key words. algorithmic spectral graph theory, spectral sparsification

AMS subject classifications. 05C50, 15B52

1. Introduction. Graph sparsification is a procedure of approximating a graph
G by a sparse graph $G'$ such that certain quantities between $G$ and $G'$ are preserved.
For instance, spanners are defined between two graphs in which the distances between
any pair of vertices in these two graphs are approximately the same [5]; cut sparsifiers
are reweighted sparse graphs of the original graphs such that the weights of every cut
between the sparsifiers and the original graphs are approximately the same [4]. Since
most algorithms run faster on sparse graphs and storing sparse graphs is more space-
efficient, graph sparsification has become one of the most central tools in designing
fast algorithms, including approximately solving Laplacian systems [10, 11, 12, 13, 19,
23], approximately computing the maximum flow in an undirected graph [4, 8, 20],
and solving streaming problems [7, 9]. Moreover, techniques developed for spectral
sparsification are widely used in randomized linear algebra [6, 15, 17], sparsifying
linear programs [14], and many other mathematical problems [2, 18, 22, 25].

In this work we study spectral sparsification introduced by Spielman and Teng [24]:
A spectral sparsifier is a reweighted sparse subgraph of the original graph such that,
for all real vectors, the Laplacian quadratic forms between that subgraph and the orig-
inal graph are approximately the same. Formally, for any undirected and weighted
graph $G$ with $n$ vertices, we call a subgraph $G'$ of $G$, with proper reweighting of the
edges, is a $(1 + \varepsilon)$-spectral sparsifier if it holds for any $x \in \mathbb{R}^n$
that

$$(1 - \varepsilon)x^T L_G x \leq x^T L_{G'} x \leq (1 + \varepsilon)x^T L_G x,$$

where $L_G$ and $L_{G'}$ are the respective Laplacian matrices of $G$ and $G'$.

Spielman and Teng [24] present the first algorithm for constructing spectral spar-
sifiers: for any undirected graph $G$ of $n$ vertices and $m$ edges, their algorithm runs in
$O(m \log^c n/\varepsilon^2)$ time, for some big constant $c$, and produces a spectral sparsifier with
$O(n \log^{c'} n/\varepsilon^2)$ edges for some $c' \geq 2$. Since then, there has been a wealth of work
on better constructions of spectral sparsifiers. For instance, Spielman and Srivas-
tava [21] present a nearly-linear time algorithm for constructing a spectral sparsifier
of $O(n \log n/\varepsilon^2)$ edges. Batson, Spielman and Srivastava [3] present an algorithm
for constructing a spectral sparsifier with $O(n/\varepsilon^2)$ edges, which is optimal up to a
constant.

∗A preliminary version of this paper appeared in the Proceedings of the 56th Annual IEEE
Symposium on Foundations of Computer Science (FOCS 2015).
†Paul G. Allen School of Computer Science & Engineering, University of Washington, Washington, USA (yintat@uw.edu).
‡School of Informatics, University of Edinburgh, Edinburgh, UK. (h.sun@ed.ac.uk).
In this paper we present an almost-linear time algorithm for constructing linear-sized spectral sparsifiers for graphs, which improves all previous constructions that either require $\Omega(n^{2+\varepsilon})$ time in order to produce linear-sized sparsifiers [1, 3, 27], or $O(m \log^{O(1)} n/\varepsilon^2)$ time [21]. Our algorithm is conceptually simple, and is based on a novel combination of two techniques used in literature for constructing spectral sparsifiers: random sampling by effective resistance of edges [21], and adaptive construction based on barrier functions [1, 3]. Our result is summarized as follows:

**Theorem 1.1.** Let $G = (V, E, w)$ be an undirected and weighted graph with $n$ vertices and $m$ edges, and $q \geq 10, 0 < \varepsilon \leq 1/120$ be constants. Then, there is an algorithm that outputs a $(1 + \varepsilon)$-spectral sparsifier of $G$ with $O(qn/\varepsilon^2)$ edges. The algorithm runs in $\tilde{O} \left( \frac{q m n^{5/4}}{\varepsilon^{1+1/4}} \right)$ time, where the $\tilde{O}$ notation suppresses poly-logarithmic factors.

It is known that constructing a spectral sparsifier for graphs is a special case of sparsifying the sum of rank-1 positive semi-definite (PSD) matrices [3, 21]. Our result for constructing a spectral sparsifier in the general setting is summarized as follows:

**Theorem 1.2.** Let $I = \sum_{i=1}^{m} v_i v_i^\top$ be the sum of $m$ rank-1 PSD matrices, and $q \geq 10, 0 < \varepsilon \leq 1/120$ be constants. Then, there is an algorithm that outputs scalers $\{s_i\}_{i=1}^m$ with $|\{s_i : s_i \neq 0\}| = O(qn/\varepsilon^2)$ such that

$$(1 - \varepsilon) \cdot I \preceq \sum_{i=1}^{m} s_i v_i v_i^\top \preceq (1 + \varepsilon) \cdot I.$$ 

The algorithm runs in $\tilde{O} \left( \frac{qm n^{5/4}}{\varepsilon^{1+1/4}} \cdot n^{1-1/3/q} \right)$ time, where $\omega$ is the matrix-multiplication constant.

2. **Preliminaries.** Throughout this paper we assume that $G = (V, E, w)$ is a connected, and undirected graph with $n$ vertices, $m$ edges, and weight function $w : V \times V \to \mathbb{R}_{>0}$. The Laplacian matrix of $G$ is an $n$ by $n$ matrix $L$ defined by

$$L_G(u, v) = \begin{cases} -w(u, v) & \text{if } u \sim v, \\ \deg(u) & \text{if } u = v, \\ 0 & \text{otherwise}, \end{cases}$$

where $\deg(u) = \sum_{u \sim v} w(u, v)$. It is easy to see that

$$x^\top L_G x = \sum_{u \sim v} w(u, v)(x_u - x_v)^2 \geq 0,$$

for any $x \in \mathbb{R}^n$.

For any matrix $A$, let $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ be the maximum and minimum eigenvalues of $A$. The condition number of matrix $A$ is defined by $\lambda_{\max}(A)/\lambda_{\min}(A)$. For any two matrices $A$ and $B$, we write $A \preceq B$ to represent $B - A$ is positive semi-definite (PSD), and $A \sim B$ to represent $B - A$ is positive definite. For any two matrices $A$ and $B$ of equal dimensions, let $A \bullet B \triangleq \text{tr} (A^\top B)$. For any function $f$, we write $\tilde{O}(f) \triangleq O(f \cdot \log^{O(1)} f)$. For matrices $A$ and $B$, we write $A \approx_n B$ if $(1 - \varepsilon) \cdot A \preceq B \preceq (1 + \varepsilon)A$.

The following well-known results from linear algebra are used in our proof.


Lemma 2.1 (Sherman-Morrison Formula). Let $A \in \mathbb{R}^{n \times n}$ be an invertible matrix, and $u, v \in \mathbb{R}^n$. Suppose that $1 + v^TA^{-1}u \neq 0$. Then it holds that

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}.$$

Lemma 2.2 (Lieb Thirring Inequality, [16]). Let $A$ and $B$ be positive definite matrices, and $q \geq 1$. Then it holds that

$$\operatorname{tr}(BAB)^q \leq \operatorname{tr}(B^qA^qB^q).$$

3. Algorithm. In this section we study the algorithm for sparsifying the sum of rank-1 PSD matrices. Our goal is to, for any vectors $v_1, \cdots, v_m$ with $\sum_{i=1}^m v_i v_i^T = I$, find scalars $\{s_i\}_{i=1}^m$ satisfying

$$|\{s_i : s_i \neq 0\}| = O\left(\frac{qn}{\varepsilon^2}\right),$$

such that

$$(1 - \varepsilon) \cdot I \preceq \sum_{i=1}^m s_i v_i v_i^T \preceq (1 + \varepsilon) \cdot I.$$

We use this algorithm to construct graph sparsifiers in section 4.

3.1. Overview of our approach. In this section we give an overview of our approach, while a detailed and formal proof is presented in section 4. At a high level, our algorithm can be viewed as an improved and randomized version of the algorithm presented in Batson et al. [3]. We refer their algorithm BSS for short, and first give a brief overview of the BSS algorithm.

The BSS algorithm proceeds by iterations, and maintains in iteration $j \geq 1$ a matrix $A_j$, which is the sum of $A_{j-1}$ and some rank-1 matrix. To analyze the spectral properties of matrix $A_j$, Batson et al. [3] introduces two barrier values $u_j$ and $\ell_j$, where $u_0 > 0, \ell_0 < 0$ initially. They show that one can always find a vector in $\{v_i\}_{i=1}^m$ and update $u_j, \ell_j$ properly in each iteration, such that the invariant

$$\ell_j I \prec A_j \prec u_j I \tag{1}$$

always holds [3]. To quantitively measure “how close the eigenvalues of $A$ are to the barriers $u$ and $\ell$”, Batson et al. [3] analyzes the following two potential functions defined by

$$\Phi^u(A) \equiv \operatorname{tr}(uI - A)^{-1}$$

and

$$\Phi_\ell(A) \equiv \operatorname{tr}(A - \ell I)^{-1}.$$

Notice that the value of $\Phi^u(A)$ or $\Phi_\ell(A)$ is large if and only if some eigenvalue of $A$ is close to $u$ or $\ell$. With the help of these potential functions, it is known that, when updating $A_j$ and barrier values $u_j, \ell_j$ properly, it holds after $T = \Theta(n/\varepsilon^2)$ iterations that $\ell_T \geq (1 - O(\varepsilon))u_T$. This implies that the resulting matrix $A_T$ is a linear-sized and $A_T \approx O(\varepsilon) I$.

However, the BSS algorithm is deterministic, and the time needed for finding a desired rank-1 matrix in each iteration is high. Hence, the main difficulties for improving the BSS algorithm are to improve the runtime of each iteration, and reduce the number of iterations. With this in mind, let us look at the following randomised
version of the BSS algorithm: in each iteration, the algorithm chooses a vector \( v_i \) with probability proportional to \( v_i^\top (u_j I - A_j)^{-1} v_i + v_i^\top (A_j - \ell_j I)^{-1} v_i \). When \( v_i \) is chosen, the algorithm adds a rank-1 matrix

\[
\Delta_A \triangleq \frac{\varepsilon}{v_i^\top (u_j I - A_j)^{-1} v_i + v_i^\top (A_j - \ell_j I)^{-1} v_i} \cdot v_i v_i^\top
\]

to the current matrix \( A \). See Algorithm 1 for formal description.

\begin{algorithm}
\begin{algorithmic}
\State \( j = 0 \);
\State \( \ell_0 = -2n/\varepsilon, \ u_0 = 2n/\varepsilon \)
\State \( A_0 = 0 \)
\While {\( u_j - \ell_j < 8n/\varepsilon \)} \Do
\State Let \( t = \text{tr}(u_j I - A_j)^{-1} + \text{tr}(A_j - \ell_j I)^{-1} \)
\State Sample a vector \( v_i \) with probability \( p_i \triangleq \left(v_i^\top (u_j I - A_j)^{-1} v_i + v_i^\top (A_j - \ell_j I)^{-1} v_i\right)/t \)
\State \( A_{j+1} = A_j + \frac{\varepsilon}{t} \cdot \frac{1}{p_i} \cdot v_i v_i^\top \)
\State \( u_{j+1} = u_j + \frac{\varepsilon}{t(1-\varepsilon)} \) and \( \ell_{j+1} = \ell_j + \frac{\varepsilon}{t(1+\varepsilon)} \)
\State \( j \leftarrow j + 1 \)
\EndWhile
\State \textbf{return} \( A_j \)
\end{algorithmic}
\end{algorithm}

Let us look at a fixed iteration \( j \), and analyze how the added \( \Delta_A \) impacts the potential functions. For simplicity, we follow [1] and define

\begin{equation}
\Phi_{u,\ell}(A) \triangleq \text{tr}(u I - A)^{-1} + \text{tr}(A - \ell I)^{-1}.
\end{equation}

After adding \( \Delta_A \), the first-order approximation of \( \Phi_{u,\ell}(A) \) gives that

\begin{equation}
\Phi_{u,\ell}(A + \Delta_A) \sim \Phi_{u,\ell}(A) + (u I - A)^{-2} \cdot \Delta_A - (A - \ell I)^{-2} \cdot \Delta_A.
\end{equation}

Since

\[
E[\Delta_A] = \sum_{i=1}^{m} p_i \cdot \left( \frac{\varepsilon}{t} \cdot \frac{1}{p_i} \cdot v_i v_i^\top \right) = \frac{\varepsilon}{t} \cdot \sum_{i=1}^{m} v_i v_i^\top = \frac{\varepsilon}{t} \cdot I,
\]

we have that

\[
E[\Phi_{u,\ell}(A + \Delta_A)] \sim \Phi_{u,\ell}(A) + \frac{\varepsilon}{t} \cdot (u I - A)^{-2} \cdot I - \frac{\varepsilon}{t} \cdot (A - \ell I)^{-2} \cdot I
\]

\[
= \Phi_{u,\ell}(A) + \frac{\varepsilon}{t} \cdot \text{tr}(u I - A)^{-2} - \frac{\varepsilon}{t} \cdot \text{tr}(A - \ell I)^{-2}
\]

\[
= \Phi_{u,\ell}(A) - \frac{\varepsilon}{t} \cdot \frac{d}{du} \Phi_{u,\ell}(A) - \frac{\varepsilon}{t} \cdot \frac{d}{d\ell} \Phi_{u,\ell}(A).
\]

Notice that if we increase \( u \) by \( \varepsilon/t \) and \( \ell \) by \( \varepsilon/t \), \( \Phi_{u,\ell} \) approximately increases by

\[
\frac{\varepsilon}{t} \cdot \frac{d}{du} \Phi_{u,\ell}(A) + \frac{\varepsilon}{t} \cdot \frac{d}{d\ell} \Phi_{u,\ell}(A).
\]
That is, comparing $\Phi_{u,\ell}(A + \Delta A)$ with $\Phi_{u,\ell}(A)$, the increase of the potential function due to the change of barrier values is approximately compensated by the expected decrease of the potential function by the effect of $\Delta A$. Therefore, the key of analyzing the change of the potential function is to study the impact of high-order terms when approximating $\Phi_{u,\ell}(A + \Delta A)$. Batson et al. [3] presents the following result:

**Lemma 3.1** ([3], proof of Lemma 3.3 and 3.4). Let $A \in \mathbb{R}^{n \times n}$, and $u, \ell$ be parameters satisfying $\ell I < A < uI$. Suppose that $w \in \mathbb{R}^n$ satisfies $ww^\top \preceq \delta(uI - A)$ and $ww^\top \preceq \delta(A - \ell I)$ for some $0 < \delta < 1$. Then, it holds that

$$\Phi_{u,\ell}(A + ww^\top) \leq \Phi_{u,\ell}(A) + \frac{w^\top(uI - A)^{-2}w}{1 - \delta} - \frac{w^\top(A - \ell I)^{-2}w}{1 + \delta}.$$

The estimate above shows that the first-order approximation (3) is good as long as $ww^\top \preceq (uI - A)$ and $ww^\top \preceq (A - \ell I)$ for small $\delta$. By setting $\delta = \varepsilon$, it is easy to see that the added matrix $\Delta A$ satisfies the preconditions in Lemma 3.1, since

$$\frac{\varepsilon}{\ell} \cdot \frac{1}{p_i} \cdot v_i v_i^\top = \frac{\varepsilon \cdot v_i v_i^\top}{v_i^\top (uI - A)^{-1} v_i + v_i^\top (A - \ell I)^{-1} v_i} \leq \frac{\varepsilon \cdot v_i v_i^\top}{v_i^\top (uI - A)^{-1} v_i} \leq \varepsilon (uI - A).$$

Here we use the fact that $ww^\top \preceq (v^\top B^{-1} v)B$ for any $v \in \mathbb{R}^n$ and PSD matrix $B \in \mathbb{R}^{n \times n}$. Similarly, we have that

$$\frac{\varepsilon}{\ell} \cdot \frac{1}{p_i} \cdot v_i v_i^\top \preceq \varepsilon (A - \ell I).$$

Hence, if the initial value of the potential function is small, in expectation $\Phi_{u,\ell}(A)$ is small during the execution of the whole algorithm. Up to a constant factor, this gives the same result as [3], and Algorithm 1 constructs an $\Theta(n/\varepsilon^2)$-sized $(1 + O(\varepsilon))$-spectral sparsifier.

However, Algorithm 1 runs for $\Theta(n/\varepsilon^2)$ iterations, and in each iteration the algorithm re-computes the probability distribution for sampling vectors. To improve the runtime of Algorithm 1, we need to overcome two bottlenecks:

1. We need a fast algorithm to approximate the probabilities $\{p_i\}_{i=1}^m$ used to choose vectors;
2. We need to use the computed $\{p_i\}_{i=1}^m$ to choose multiple vectors in each iteration, so that the total number of iterations can be reduced.

To overcome the first bottleneck, we adopt the idea proposed in [1]: instead of defining the potential function by (2), we use the following potential function:

$$\Phi_{u,\ell}(A) \triangleq \text{tr}(uI - A)^{-q} + \text{tr}(A - \ell I)^{-q}.$$

Since $q$ is a large constant, the value of $\Phi_{u,\ell}(A)$ becomes larger when some eigenvalue of $A$ is closer to $u$ or $\ell$. Hence, a bounded value of $\Phi_{u,\ell}(A)$ insures that the eigenvalues of $A$ do not get too close to $u$ or $\ell$, which allows us to compute the sampling probabilities $\{p_i\}_{i=1}^m$ efficiently simply by Taylor expansion. Moreover, with our new potential function (4) one can prove a similar result as Lemma 3.1. This gives an alternative analysis of the algorithm presented in [1], which is the first almost-quadratic time algorithm for constructing linear-sized spectral sparsifiers.

Next we sketch our approach to overcome the second bottleneck: we show that one can get the same guarantee on the potential function, as long as the sampling
probability satisfies

\[ p_i \geq C \cdot \frac{v_i^T (u I - A)^{-1} v_i + v_i^T (A - \ell I)^{-1} v_i}{\sum_{j=1}^{m} \left( v_j^T (u I - A)^{-1} v_j + v_j^T (A - \ell I)^{-1} v_j \right)} \]

for some constant \( C > 0 \). Hence, we only need to re-compute \( \{ p_i \}_{i=1}^{m} \) after every \( \Theta \left( \frac{n}{\varepsilon^2} \right) \) iterations. To informally see the reason, let us assume \( \Delta_A = \sum_{t=1}^{T} \Delta_{A,i} \) is the sum of the sampled matrices within \( T = O \left( \frac{n^{1-1/q}}{q} \right) \) iterations. If a randomly chosen matrix \( \Delta_{A,i} \) satisfies \( \Delta_{A,i} \preceq \frac{1}{q} (u I - A) \), then by the matrix Chernoff bound \( \Delta_A \preceq \frac{1}{2} (u I - A) \) holds with high probability. By scaling every sampled rank-1 matrix \( q \) times smaller, the sampling probability only changes by a constant factor within \( T \) iterations. Since we choose \( \Theta \left( \frac{n}{\varepsilon^2} \right) \) vectors in total, our algorithm only recomputes the sampling probabilities \( \Theta \left( \frac{n^{1/q-1}}{\varepsilon^2} \right) \) times. To compute these probabilities, we solve roughly \( n O(1)/q \) many linear systems each iteration and this explains our running times.

### 3.2. Algorithm description

Our actual algorithm follows the same framework as Algorithm 1, but uses the same probability \( \{ p_i \}_{i=1}^{m} \) to pick multiple vectors, before the algorithm increases the barrier values. To highlight this difference, we always use the term phase (instead of iteration) in the following discussion. At a high-level, our algorithm proceeds by phases, and initially the algorithm sets

\[ u_0 = \frac{2n}{1/q}, \quad \ell_0 = -\frac{2n}{1/q}, \quad A_0 = 0. \]

After phase \( j \) the algorithm updates \( u_j, \ell_j \) by \( \Delta_{u,j}, \Delta_{\ell,j} \) respectively, i.e.,

\[ u_{j+1} = u_j + \Delta_{u,j}, \quad \ell_{j+1} = \ell_j + \Delta_{\ell,j}, \]

and updates \( A_j \) with respect to the chosen matrix in phase \( j \). Specifically, in phase \( j \) the algorithm computes the relative effective resistance of vectors \( \{ v_i \}_{i=1}^{m} \) defined by

\[ R_i(A_j, u_j, \ell_j) = v_i^T (u_j I - A_j)^{-1} v_i + v_i^T (A_j - \ell_j I)^{-1} v_i, \]

and samples \( N_j \) vectors independently with replacement, where vector \( v_i \) is chosen with probability proportional to \( R_i(A_j, u_j, \ell_j) \). The number of samples \( N_j \) in phase \( j \) is defined by

\[ N_j = \frac{1}{n^{2/q}} \left( \sum_{i=1}^{m} R_i(A_j, u_j, \ell_j) \right) \min \{ \lambda_{\text{min}}(u_j I - A_j), \lambda_{\text{min}}(A_j - \ell_j I) \}. \]

For simplicity, let us define

\[ W = \sum_{\text{sampled vector } v_i} \frac{\varepsilon}{q} \cdot \frac{1}{R_i(A_j, u_j, \ell_j)} \cdot v_i v_i^T, \]

and update \( A_j \) by

\[ A_{j+1} = A_j + W. \]

Since

\[ E[W] = \frac{\varepsilon}{q} \cdot \frac{N_j}{\sum_{i=1}^{m} R_i(A, u, \ell)} \cdot I, \]
we increase the barrier values $u_j$ and $\ell_j$ by $\Delta_{u,j}$ and $\Delta_{\ell,j}$ respectively, where

$$\Delta_{u,j} \triangleq (1 + 3\varepsilon) \cdot \frac{\varepsilon}{q} \sum_{i=1}^{m} R_i(A_j, u_j, \ell_j), \quad \Delta_{\ell,j} \triangleq (1 - 3\varepsilon) \cdot \frac{\varepsilon}{q} \sum_{i=1}^{m} R_i(A_j, u_j, \ell_j).$$

Our choice of $N_j$ insures that $0 \preceq W \preceq \frac{1}{2} \cdot (u_j I - A_j)$ holds with high probability (see Lemma 4.2). Moreover, our choice of $\Delta_{u,j}$ and $\Delta_{\ell,j}$ insures that the invariant

$$\ell_j I \prec A_j \prec u_j I$$

always holds for any phase $j$. See Algorithm 2 for formal description.

Algorithm 2 Algorithm for constructing spectral sparsifiers

Require: $\varepsilon \leq 1/120, q \geq 10$

1: $j = 0$
2: $\ell_0 = -(2n)^{1/q}, u_0 = (2n)^{1/q}, A_0 = 0$
3: while $u_j - \ell_j < 4 \cdot (2n)^{1/q}$ do
4: $W_j = 0$
5: Compute $R_i(A_j, u_j, \ell_j)$ for all vectors $v_i$
6: Sample $N_j$ vectors independently with replacement, where every $v_i$ is chosen with probability proportional to $R_i(A_j, u_j, \ell_j)$. For every sampled $v$, add $\varepsilon/q \cdot (R_i(A_j, u_j, \ell_j))^{-1} \cdot vv^\top$ to $W_j$
7: $A_{j+1} = A_j + W_j$
8: $u_{j+1} = u_j + \Delta_{u,j}, \ell_{j+1} = \ell_j + \Delta_{\ell,j}$
9: $j = j + 1$
10: end while
11: Return $A_j$

We remark that, although exact values of $N_j$ and relative effective resistances are difficult to compute, we present almost-linear time algorithms for approximating $\{R_i\}_{i=1}^{m}$ and $N_j$ in section 4. It is easy to see that in each phase an over estimate of the relative effective resistance for every vector $v_i$, and an under estimate of $N_j$ with constant-factor approximation suffice for our purpose.

4. Analysis. In this section we prove that the output matrix returned by Algorithm 2 is a $(1 + \varepsilon)$-spectral sparsifier, and analyze the algorithm’s runtime. To simplify our analysis, we always assume in the rest of the paper that $\varepsilon$ and $q$ are constants such that $0 < \varepsilon \leq 1/120, \text{ and } q \geq 10$.

This section is organized as follows: in subsection 4.1 we show how the potential function (4) evolves after each phase. Combining this with the ending condition of the algorithm, we prove in subsection 4.2 that the algorithm outputs a linear-sized spectral sparsifier. subsection 4.3 shows almost-linear time algorithms for approximately computing all required quantities of Algorithm 2 within each phase when constructing graph sparsifiers. We prove Theorem 1.1 and Theorem 1.2 in subsection 4.4.

4.1. Analysis of a single phase. We analyze the sampling scheme within a single phase, and drop the subscript representing the phase $j$ for simplicity. Recall that in each phase the algorithm samples $N$ vectors independently from $\{v_i\}_{i=1}^{m}$ satisfying $\sum_{i=1}^{m} v_i v_i^\top = I$, where every vector $v_i$ is sampled with probability

$$\frac{R_i(A, u, \ell)}{\sum_{j=1}^{m} R_j(A, u, \ell)}.$$
We use $v_1, \ldots, v_N$ to denote these $N$ sampled vectors, and define the reweighted vectors by

$$w_i \triangleq \frac{\sqrt{\varepsilon}}{q \cdot R_i(A,u,\ell)} \cdot v_i,$$

for any $1 \leq i \leq N$. Let

$$W \triangleq \sum_{i=1}^{N} w_i w_i^T.$$

We show that with high probability matrix $W$ satisfies $0 \preceq W \preceq \frac{1}{2} (uI - A)$. We first recall the following Matrix Chernoff Bound.

**Lemma 4.1 (Matrix Chernoff Bound, [26]).** Let $\{X_i\}$ be a finite sequence of independent, random, and self-adjoint matrices with dimension $n$. Assume that each random matrix satisfies $X_i \succeq 0$, and $\lambda_{\max}(X_i) \leq D$. Let $\mu \geq \lambda_{\max}(\sum_i E[X_i])$. Then, it holds for any $\delta \geq 0$ that

$$\Pr\left[ \lambda_{\max}\left(\sum_i X_i\right) \geq (1 + \delta)\mu \right] \leq n \cdot \left(\frac{e^\delta}{(1 + \delta)^{1+\delta}}\right)^{\mu/D}.$$

**Lemma 4.2.** Assume that the number of sampled vectors satisfies

$$N < \frac{2}{n^2/q} \left(\sum_{i=1}^{m} R_i(A,u,\ell)\right) \cdot \lambda_{\min}(uI - A).$$

Then, it holds that

$$E[w_i w_i^T] = \frac{\varepsilon}{q} \cdot \frac{1}{\sum_{t=1}^{m} R_t(A,u,\ell)} \cdot I,$$

$$E[W] = \frac{\varepsilon}{q} \cdot \frac{N}{\sum_{i=1}^{m} R_i(A,u,\ell)} \cdot I,$$

and

$$\Pr\left[ 0 \preceq W \preceq \frac{1}{2} (uI - A) \right] \geq 1 - \frac{\varepsilon^2}{100qn}.$$

**Proof.** By the description of the sampling procedure, it holds that

$$E[w_i w_i^T] = \sum_{j=1}^{m} \frac{R_j(A,u,\ell)}{\sum_{t=1}^{m} R_t(A,u,\ell)} \cdot \frac{\varepsilon}{q} \cdot \frac{v_j v_j^T}{R_j(A,u,\ell)} = \frac{\varepsilon}{q} \cdot \frac{1}{\sum_{t=1}^{m} R_t(A,u,\ell)} \cdot I,$$

and

$$E[W] = E\left[ \sum_{i=1}^{N} w_i w_i^T \right] = \frac{\varepsilon}{q} \cdot \frac{N}{\sum_{i=1}^{m} R_i(A,u,\ell)} \cdot I,$$

which proves the first two statements.

Now we turn to the third statement. Let

$$z_i = (uI - A)^{-1/2} w_i.$$
It holds that
\[
\text{tr}(z_i^Tz_i) = \text{tr} \left( (uI - A)^{-1/2} w_i w_i^T (uI - A)^{-1/2} \right) \\
= \frac{\varepsilon}{q} \cdot \text{tr} \left( (uI - A)^{-1/2} v_i v_i^T (uI - A)^{-1/2} \right) \\
\leq \frac{\varepsilon}{q} \cdot R_i(A, u, \ell) \\
\leq \frac{\varepsilon}{q} \cdot v_i^T (uI - A)^{-1} v_i
\]
and \( \lambda_{\text{max}}(z_i z_i^T) \leq \frac{\varepsilon}{q} \). Moreover, it holds that
\[
\mathbb{E} \left[ \sum_{i=1}^{N} z_i z_i^T \right] = \frac{\varepsilon}{q} \cdot \frac{\sum_{i=1}^{N} R_i(A, u, \ell)}{\sum_{i=1}^{N} R_i(A, u, \ell)} \cdot (uI - A)^{-1}
\]
(5)
This implies that
\[
\lambda_{\text{max}} \left( \mathbb{E} \left[ \sum_{i=1}^{N} z_i z_i^T \right] \right) \leq \frac{\varepsilon}{q} \cdot \frac{\sum_{i=1}^{N} R_i(A, u, \ell)}{\sum_{i=1}^{N} R_i(A, u, \ell)} \cdot \lambda_{\text{max}} \left( \frac{1}{uI - A} \right).
\]
By setting
\[
\mu = \frac{\varepsilon}{q} \cdot \frac{\sum_{i=1}^{N} R_i(A, u, \ell)}{\sum_{i=1}^{N} R_i(A, u, \ell)} \cdot \lambda_{\text{max}} \left( \frac{1}{uI - A} \right),
\]
it holds by the Matrix Chernoff Bound (Lemma 4.1) that
\[
\Pr \left[ \lambda_{\text{max}} \left( \sum_{i=1}^{N} z_i z_i^T \right) \geq (1 + \delta) \mu \right] \leq n \cdot \left( \frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^{\mu/q/\varepsilon}.
\]
Set the value of \( 1 + \delta \) to be
\[
1 + \delta = \frac{1}{2\mu} = \frac{q}{2eN} \cdot \left( \sum_{j=1}^{m} R_j(A, u, \ell) \right) \cdot \lambda_{\text{max}} \left( \frac{1}{uI - A} \right)
\]
\[
= \frac{q}{2eN} \cdot \left( \sum_{j=1}^{m} R_j(A, u, \ell) \right) \cdot \lambda_{\text{min}}(uI - A)
\]
\[
\geq \frac{q}{4\varepsilon} \cdot n^{2/q},
\]
where the last inequality follows from the condition on \( N \). Hence, with probability at least
\[
1 - n \cdot \left( \frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^{\mu/q/\varepsilon} \geq 1 - n \cdot \left( \frac{e}{1 + \delta} \right)^{(1+\delta)\cdot\mu/q/\varepsilon} \geq 1 - n \cdot \left( \frac{e}{1 + \delta} \right)^{2\mu/4\delta} \geq 1 - \frac{\varepsilon^2}{100qn},
\]
we have that
\[
\lambda_{\text{max}} \left( \sum_{i=1}^{N} z_i z_i^T \right) \leq (1 + \delta) \cdot \mu = \frac{1}{2},
\]
which implies that \( 0 \preceq \sum_{i=1}^{N} z_i z_i^T \preceq \frac{1}{2} \cdot I \) and \( 0 \preceq W \preceq \frac{1}{2} \cdot (uI - A) \). \qed
Lemma 4.3 below shows how the potential function changes after adding a rank-1 matrix, and plays a key role in our analysis. This lemma is first shown in [3] for the case of \( q = 1 \), and a similar lemma is shown in [1].

**Lemma 4.3 ([1]).** Let \( q \geq 10 \) and \( \varepsilon \leq 1/10 \). Suppose that \( w^\top (uI - A)^{-1} w \leq \varepsilon/q \) and \( w^\top (A - \ell I)^{-1} w \leq \varepsilon/q \). Then, it holds that

\[
\text{tr}(A + uw^\top - \ell I)^{-q} \leq \text{tr}(A - \ell I)^{-q} - q(1 - \varepsilon) \ w^\top (A - \ell I)^{-(q+1)} w,
\]

and

\[
\text{tr}(uI - A - tw^\top)^{-q} \leq \text{tr}(uI - A)^{-q} + q(1 + \varepsilon) \ w^\top (uI - A)^{-(q+1)} w.
\]

**Proof.** Let \( Y = A - \ell I \). By the Sherman-Morrison Formula (Lemma 2.1), it holds that

\[
\text{tr}(Y + w^w^\top)^{-q} = \text{tr} \left( Y^{-1} - \frac{Y^{-1}w^w^\top Y^{-1}}{1 + w^w^\top Y^{-1} w} \right)^q.
\]

By the assumption of \( w^\top Y^{-1} w \leq \varepsilon/q \), we have that

\[
(7a) \quad \text{tr}(Y + w^w^\top)^{-q} \leq \text{tr} \left( Y^{-1} - \frac{Y^{-1}w^w^\top Y^{-1}}{1 + \varepsilon/q} \right)^q
\]

\[
= \text{tr} \left( Y^{-1/2} \left( I - \frac{Y^{-1/2}w^w^\top Y^{-1/2}}{1 + \varepsilon/q} \right) Y^{-1/2} \right)^q
\]

\[
\leq \text{tr} \left( Y^{-q/2} \left( I - \frac{Y^{-1/2}w^w^\top Y^{-1/2}}{1 + \varepsilon/q} \right)^q Y^{-q/2} \right)
\]

\[
= \text{tr} \left( Y^{-q} \left( I - \frac{Y^{-1/2}w^w^\top Y^{-1/2}}{1 + \varepsilon/q} \right)^q \right),
\]

where \( 7a \) uses the fact that \( A \preceq B \) implies that \( \text{tr}(A^q) \preceq \text{tr}(B^q) \), \( 7b \) follows from the Lieb-Thirring inequality (Lemma 2.2), and \( 7c \) uses the fact that the trace is invariant under cyclic permutations.

Let

\[
D = \frac{Y^{-1/2}w^w^\top Y^{-1/2}}{1 + \varepsilon/q}.
\]

Note that \( 0 \leq D \leq (\varepsilon/q) \cdot I \), and

\[
(I - D)^q \preceq I - qD + \frac{q(q - 1)}{2} D^2
\]

\[
\preceq I - \left( q - \frac{\varepsilon(q - 1)}{2} \right) D.
\]

Therefore, we have that

\[
(I - \frac{Y^{-1/2}w^w^\top Y^{-1/2}}{1 + \varepsilon/q})^q \preceq I - \left( q - \frac{\varepsilon(q - 1)}{2} \right) \frac{Y^{-1/2}w^w^\top Y^{-1/2}}{1 + \varepsilon/q}
\]

\[
\preceq I - \left( q - \frac{\varepsilon(q - 1)}{2} \right) \left( 1 - \frac{\varepsilon}{q} \right) Y^{-1/2}w^w^\top Y^{-1/2}
\]

\[
\preceq I - q \left( 1 - \frac{\varepsilon(q + 1)}{2q} \right) Y^{-1/2}w^w^\top Y^{-1/2}
\]

\[
\preceq I - q (1 - \varepsilon) Y^{-1/2}w^w^\top Y^{-1/2}.
\]
This implies that
\[
\text{tr}(Y + ww^\top)^{-q} \leq \text{tr} \left( Y^{-q} \left( I - q(1 - \varepsilon)Y^{-1/2}ww^\top Y^{-1/2} \right) \right) \\
\leq \text{tr} \left( Y^{-q} \right) - q(1 - \varepsilon) w^\top Y^{-(q+1)} w,
\]
which proves the first statement.

Now we turn to the second inequality. Let \( Z = uI - A \). By the Sherman-Morrison Formula (Lemma 2.1), it holds that
\[
\text{tr}(Z - ww^\top)^{-q} = \text{tr} \left( Z^{-1} + \frac{Z^{-1}ww^\top Z^{-1}}{1 - w^\top Z^{-1}w} \right)^q.
\]
By the assumption of \( w^\top Z^{-1} w \leq \varepsilon/q \), it holds that
\[
(8a) \quad \text{tr}(Z - ww^\top)^{-q} \leq \text{tr} \left( Z^{-1} + \frac{Z^{-1/2}ww^\top Z^{-1/2}}{1 - \varepsilon/q} \right)^q \\
= \text{tr} \left( Z^{-1/2} \left( I + \frac{Z^{-1/2}ww^\top Z^{-1/2}}{1 - \varepsilon/q} \right) Z^{-1/2} \right)^q \\
(8b) \quad \leq \text{tr} \left( Z^{-q/2} \left( I + \frac{Z^{-1/2}ww^\top Z^{-1/2}}{1 - \varepsilon/q} \right)^q Z^{-q/2} \right) \\
(8c) \quad = \text{tr} \left( Z^{-q} \left( I + \frac{Z^{-1/2}ww^\top Z^{-1/2}}{1 - \varepsilon/q} \right)^q \right),
\]
where (8a) uses the fact that \( A \preceq B \) implies \( \text{tr} (A^q) \leq \text{tr} (B^q) \), (8b) follows from the Lieb-Thirring inequality (Lemma 2.2), and (8c) uses the fact that the trace is invariant under cyclic permutations.

Let
\[
E = Z^{-1/2}ww^\top Z^{-1/2}.
\]
Combining \( E \preceq (\varepsilon/q) I \) with the assumption that \( q \geq 10 \) and \( \varepsilon \leq 1/10 \), we have that
\[
\left( I + \frac{E}{1 - \varepsilon/q} \right)^q \leq I + \frac{qE}{1 - \varepsilon/q} + \frac{q(q - 1)}{2} \left( 1 + \frac{\varepsilon/q}{1 - \varepsilon/q} \right)^{q-2} \left( \frac{E}{1 - \varepsilon/q} \right)^2 \\
\leq I + q \left( 1 + 1.1 \frac{\varepsilon}{q} \right) E + 1.4 \frac{q(q - 1)}{2} E^2 \\
\leq I + q (1 + 0.3\varepsilon) E + 0.7\varepsilon q E \\
\leq I + q (1 + \varepsilon) E.
\]
Therefore, we have that
\[
\text{tr}(Z - ww^\top)^{-q} \leq \text{tr} \left( Z^{-q} \right) + q(1 + \varepsilon) w^\top Z^{-(q+1)} w,
\]
which proves the second statement.

The following lemma shows that, with our choice of updated matrices and barrier values, the conditional expected value of the potential function does not increase.

**Lemma 4.4.** It holds that
\[
E \left[ \Phi_{u + \Delta u, \varepsilon + \Delta \varepsilon}(A + W) \mid W \leq \frac{1}{2} (uI - A) \right] \leq \Phi_{u, \varepsilon}(A).
\]
We study the change of the potential function after adding a rank-1 matrix within the current phase. For this reason, we use
\[
\overline{\Delta}_u = \frac{\Delta_u}{N} = (1 + 3\varepsilon) \cdot \frac{\varepsilon}{q \cdot \sum_{i=1}^{m} R_t(A, u, \ell)},
\]
and
\[
\overline{\Delta}_\ell = \frac{\Delta_\ell}{N} = (1 - 3\varepsilon) \cdot \frac{\varepsilon}{q \cdot \sum_{i=1}^{m} R_t(A, u, \ell)}
\]
to express the average change of the barrier values \(\Delta_u\) and \(\Delta_\ell\). We further define for any \(0 \leq i \leq N\) that
\[
\hat{u}_i = u + i \cdot \overline{\Delta}_u, \quad \hat{\ell}_i = \ell + i \cdot \overline{\Delta}_\ell.
\]
Assuming the picked matrices \(w_1 w_1^\top, \ldots, w_N w_N^\top\) satisfy
\[
W = \sum_{i=1}^{N} w_i w_i^\top \leq \frac{1}{2} (u I - A),
\]
we claim that
\[
w_i w_i^\top \preceq \frac{2\varepsilon}{q} \cdot (\hat{u}_i I - B_{i-1}), \quad w_i w_i^\top \preceq \frac{2\varepsilon}{q} \cdot (B_{i-1} - \hat{\ell}_i I),
\]
for any \(1 \leq i \leq N\) and
\[
\varepsilon \left(1 - \frac{\varepsilon}{2}\right) I \leq E \left[ w_i w_i^\top \mid W \leq \frac{1}{2} (u I - A) \right] \leq \varepsilon \left(1 + \frac{\varepsilon}{2}\right) I.
\]
Based on this, we apply Lemma 4.3 and get that
\[
E \left[ \Phi_{\hat{u}_i, \hat{\ell}_i} (B_{i-1} + w_i w_i^\top) \mid W \leq \frac{1}{2} (u I - A) \right] \leq \Phi_{\hat{u}_i, \hat{\ell}_i} (B_{i-1}) + q \Delta_u \text{tr} \left( (\hat{u}_i I - B_{i-1})^{-(q+1)} \right) - q \Delta_\ell \text{tr} \left( (B_{i-1} - \hat{\ell}_i I)^{-(q+1)} \right). \tag{11}
\]
We define a function \(f_i : \mathbb{R} \to \mathbb{R}\) by
\[
f_i(x) = \text{tr} \left( (\hat{u}_{i-1} + x \cdot \Delta_u) I - B_{i-1} \right)^{-q} + \text{tr} \left( B_{i-1} - (\hat{\ell}_{i-1} + x \cdot \Delta_\ell) I \right)^{-q}.
\]
Notice that
\[
\frac{df_i(x)}{dx} = -q \cdot \Delta_u \cdot \text{tr} \left( (\tilde{u}_{i-1} + x \cdot \Delta_u) I - B_{i-1} \right)^{-(q+1)}
+ q \cdot \Delta_t \cdot \text{tr} \left( B_{i-1} - \left( \tilde{\ell}_{i-1} + x \cdot \Delta_t \right) I \right)^{-(q+1)}.
\]
Since \( f \) is convex, we have that
\[
(12) \quad \left. \frac{df_i(x)}{dx} \right|_{x=1} \geq f_i(1) - f_i(0) = \Phi_{u_i,\tilde{\ell}_i}(B_{i-1}) - \Phi_{\tilde{u}_{i-1},\tilde{\ell}_{i-1}}(B_{i-1}).
\]
Putting (11) and (12) together, we have that
\[
E \left[ \Phi_{u_i,\tilde{\ell}_i}(B_i) \mid 0 \preceq W \preceq \frac{1}{2}(uI - A) \right] \leq E \left[ \Phi_{u_i,\tilde{\ell}_i}(B_{i-1}) \right] \leq \Phi_{u_i,\ell}(A),
\]
which proves the statement.
So, it suffices to prove (9) and (10). Since \( vv^\top \preceq (v^\top B^{-1}v)B \) for any vector \( v \) and PSD matrix \( B \), we have that
\[
\frac{v_iv_i^\top}{R_t(A,u,\ell)} \preceq \frac{v_iv_i^\top}{q \cdot R_t(A,u,\ell)} \preceq \frac{v_i^\top}{q} (uI - A) \preceq \frac{2\varepsilon}{q} (\tilde{u}_i I - B_{i-1}).
\]
By the assumption of \( W \preceq \frac{1}{2}(uI - A) \), it holds that
\[
w_iw_i^\top = \frac{\varepsilon}{q \cdot R_t(A,u,\ell)} \cdot v_i^\top \preceq \frac{\varepsilon}{q} (uI - A) \preceq \frac{2\varepsilon}{q} (\tilde{u}_i I - B_{i-1}).
\]
This proves the first statement of the claim.
For the second statement, notice that
\[
\Delta_\ell \leq \frac{\varepsilon N}{q \sum_{t=1}^m R_t(A,u,\ell)} \leq \frac{1}{2} \cdot \lambda_{\min}(A - \ell I),
\]
and hence
\[
w_iw_i^\top \preceq \frac{\varepsilon}{q} (A - \ell I) \preceq \frac{2\varepsilon}{q} \left( A - \tilde{\ell}_i I \right) \leq \frac{2\varepsilon}{q} \left( B_{i-1} - \tilde{\ell}_i I \right).
\]
To prove (10), by Lemma 4.2 it holds that
\[
E \left[ w_iw_i^\top \mid W \preceq \frac{1}{2}(uI - A) \right] \leq \frac{E \left[ w_iw_i^\top \right]}{Pr \left[ W \preceq \frac{1}{2}(uI - A) \right]} \leq \frac{\varepsilon(1 + \frac{\varepsilon}{2})}{q \cdot \sum_{t=1}^m R_t(A,u,\ell)} \cdot I
\]
and
\[
E \left[ w_iw_i^\top \mid W \preceq \frac{1}{2}(uI - A) \right] \geq E \left[ w_iw_i^\top \right] - Pr \left[ W \preceq \frac{1}{2}(uI - A) \right] \cdot \frac{\varepsilon}{q} \cdot (uI - A)
\]
\[
\geq E \left[ w_iw_i^\top \right] - \frac{4\varepsilon^2}{100qn} \cdot \frac{\varepsilon(2n)^{1/2}}{q} \cdot I
\]
\[
\geq E \left[ w_iw_i^\top \right] - \frac{4\varepsilon^2}{50qn} \sum_{t=1}^m \frac{R_t(A,u,\ell)}{q} \cdot I
\]
\[
\geq \frac{\varepsilon(1 - \frac{\varepsilon}{2})}{q \cdot \sum_{t=1}^m R_t(A,u,\ell)} \cdot I.
\]
where we use that
\[
\sum_{t=1}^{m} R_t(A, u, \ell) = \text{tr}(uI - A)^{-1} + \text{tr}(A - \ell I)^{-1} \geq \text{tr}(uI - \ell I)^{-1} \geq 2n \cdot (2n)^{-1/q}.
\]

### 4.2. Analysis of Algorithm 2

Now we prove that the algorithm produces a linear-sized \((1 + O(\varepsilon))\)-spectral sparsifier. We assume that the algorithm finishes after \(k\) phases, and prove that the condition number of \(A_k\) is small, which follows from our setting of parameters.

**Lemma 4.5.** The output matrix \(A_k\) has condition number at most \(1 + O(\varepsilon)\).

**Proof.** Since the condition number of \(A_k\) is at most
\[
u_k - \ell_k \geq (1 - \nu_k - \ell_k)^{-1},
\]
it suffices to prove that \((\nu_k - \ell_k)/\nu_k = O(\varepsilon)\).

Since the increase rate of \(\Delta_{u,j} - \Delta_{\ell,j}\) with respect to \(\Delta_{u,j}\) for any phase \(j\) is
\[
\frac{\Delta_{u,j} - \Delta_{\ell,j}}{\Delta_{u,j}} = \frac{(1 + 3\varepsilon) - (1 - 3\varepsilon)}{1 + 3\varepsilon} = \frac{6\varepsilon}{1 + 3\varepsilon} \leq 6\varepsilon,
\]
we have that
\[
\frac{\nu_k - \ell_k}{\nu_k} = \frac{2 \cdot (2n)^{1/q} + \sum_{j=0}^{k-1} (\Delta_{u,j} - \Delta_{\ell,j})}{(2n)^{1/q} + \sum_{j=0}^{k-1} \Delta_{u,j}} \leq \frac{2 \cdot (2n)^{1/q} + \sum_{j=0}^{k-1} (\Delta_{u,j} - \Delta_{\ell,j})}{(2n)^{1/q} + (6\varepsilon)^{-1} \sum_{j=0}^{k-1} (\Delta_{u,j} - \Delta_{\ell,j})}.
\]

By the ending condition of the algorithm, it holds that \(\nu_k - \ell_k \geq 4 \cdot (2n)^{1/q}\), i.e.,
\[
\sum_{j=0}^{k-1} (\Delta_{u,j} - \Delta_{\ell,j}) \geq 2 \cdot (2n)^{1/q}.
\]

Hence, it holds that
\[
\frac{\nu_k - \ell_k}{\nu_k} \leq \frac{2 \cdot (2n)^{1/q} + 2 \cdot (2n)^{1/q}}{(2n)^{1/q} + (6\varepsilon)^{-1} 2 \cdot (2n)^{1/q}} \leq 12\varepsilon,
\]
which finishes the proof. \(\square\)

Now we prove that the algorithm finishes in \(O(qn^{3/q}/\varepsilon^2)\) phases, and picks \(O(qn/\varepsilon^2)\) vectors in total.

**Lemma 4.6.** The following statements hold:

- With probability at least \(4/5\), the algorithm finishes in \(\frac{10qn^{3/q}}{\varepsilon^2}\) phases.
- With probability at least \(4/5\), the algorithm chooses at most \(10qn/\varepsilon^2\) vectors.
Proof. Notice that after phase $j$ the barrier gap $u_j - \ell_j$ is increased by

\[
\Delta u_j - \Delta \ell_j = \frac{6\varepsilon^2}{q} \frac{N_j}{\sum_{i=1}^m R_i(A_j, u_j, \ell_j)} \\
= \frac{6\varepsilon^2}{q} \frac{1}{n^{2/q}} \cdot \min \{\lambda_{\min}(u_j I - A_j), \lambda_{\min}(A_j - \ell_j I)\} \\
\geq \frac{6\varepsilon^2}{q} \frac{1}{n^{2/q}} \cdot (\Phi_{u_j, \ell_j}(A_j))^{-1/q}.
\]

Since the algorithm finishes within $k$ phases if

\[
\sum_{j=0}^{k-1} (\Delta u_j - \Delta \ell_j) \geq 2 \cdot (2n)^{1/q},
\]

it holds that

\[
\Pr\left[ \text{algorithm finishes within } k \text{ phases} \right] \\
\geq \Pr \left[ \sum_{j=0}^{k-1} (\Delta u_j - \Delta \ell_j) \geq 2 \cdot (2n)^{1/q} \right] \\
\geq \Pr \left[ \sum_{j=0}^{k-1} \frac{6\varepsilon^2}{q} \frac{1}{n^{2/q}} \cdot (\Phi_{u_j, \ell_j}(A_j))^{-1/q} \geq 2 \cdot (2n)^{1/q} \right] \\
= \Pr \left[ \sum_{j=0}^{k-1} (\Phi_{u_j, \ell_j}(A_j))^{-1/q} \geq \frac{q}{3\varepsilon^2} \cdot (2n)^{1/q} \right] \\
\geq \Pr \left[ \sum_{j=0}^{k-1} (\Phi_{u_j, \ell_j}(A_j))^{1/q} \leq \frac{3\varepsilon^2 k^2}{q} \cdot \left( \frac{1}{2n^3} \right)^{1/q} \right],
\]

where the last inequality follows from the fact that

\[
\left( \sum_{j=0}^{k-1} (\Phi_{u_j, \ell_j}(A_j))^{-1/q} \right) \cdot \left( \sum_{j=0}^{k-1} (\Phi_{u_j, \ell_j}(A_j))^{1/q} \right) \geq k^2.
\]

By Lemma 4.2, every picked matrix $W_j$ in phase $j$ satisfies

\[
0 \leq W_j \leq \frac{1}{2} \cdot (u_j I - A)
\]

with probability at least $1 - \frac{\varepsilon^2}{100q} n$, and with probability $9/10$ all matrices picked in $k = 10q^{3/q}/\varepsilon^2$ phases satisfy the condition above. Using Lemma 4.4 and that $x^{1/q}$ is concave, it holds that

\[
\mathbb{E} \left[ \sum_{j=0}^{k-1} (\Phi_{u_j, \ell_j}(A_j))^{1/q} \right] \leq k,
\]

(13)
since the initial value of the potential function is at most 1. Therefore, it holds that

\[
\Pr \left[ \text{algorithm finishes in more than } k \text{ phases} \right] \\
\leq \Pr \left[ \sum_{j=0}^{k-1} \left( \Phi_{u_j, \ell_j}(A_j) \right)^{1/q} \geq \frac{3\varepsilon^2 q^2}{q} \left( \frac{1}{2n^3} \right)^{1/q} \right] \\
\leq \Pr \left[ \sum_{j=0}^{k-1} \left( \Phi_{u_j, \ell_j}(A_j) \right)^{1/q} \geq \frac{3\varepsilon^2 q^2}{q} \cdot \left( \frac{1}{2n^3} \right)^{1/q} \mid \forall j : W_j \leq \frac{1}{2}(u_j I - A_j) \right] \\
\cdot \Pr \left[ \forall j : W_j \leq \frac{1}{2}(u_j I - A_j) \right] \\
\leq \frac{q}{3\varepsilon^2 k} \cdot \left( \frac{2n^3}{q} \right)^{1/q} \cdot \frac{9}{10} \leq 1/5,
\]

where the second last inequity follows from Markov’s inequality and (13), and the last inequality follows by our choice of \( k \). This proves the first statement.

Now we turn to the second statement. Notice that for every vector chosen in phase \( j \) the barrier gap \( \Delta_{u,j} - \Delta_{\ell,j} \) is increased on average by

\[
\frac{\Delta_{u,j} - \Delta_{\ell,j}}{N_j} = \frac{6\varepsilon^2}{q \sum_{i=1}^{m} R_i(A_j, u_j, \ell_j)}.
\]

To bound \( R_i(A_j, u_j, \ell_j) \), let the eigenvalues of matrix \( A_j \) be \( \lambda_1, \cdots, \lambda_n \). Then, it holds that

\[
\sum_{i=1}^{m} R_i(A_j, u_j, \ell_j) = \sum_{i=1}^{n} v_i^T (u_j I - A_j)^{-1} v_i + \sum_{i=1}^{n} v_i^T (A_j - \ell_j I)^{-1} v_i \\
\leq \sum_{i=1}^{n} \left( \frac{1}{u_j - \lambda_i} + \frac{1}{\lambda_i - \ell_j} \right) \\
\leq \left( \sum_{i=1}^{n} (u_j - \lambda_i)^{-q} + \sum_{i=1}^{n} (\lambda_i - \ell_j)^{-q} \right)^{1/q} \cdot (2n)^{1-1/q} \\
= \left( \Phi_{u_j, \ell_j}(A_j) \right)^{1/q} \cdot (2n)^{1-1/q}.
\]

Therefore, we have that

\[
\frac{\Delta_{u,j} - \Delta_{\ell,j}}{N_j} \geq \frac{6\varepsilon^2}{q} \cdot \frac{1}{(2n)^{1-1/q} \cdot \left( \Phi_{u_j, \ell_j}(A_j) \right)^{1/q}}.
\]

Let \( v_1, \cdots, v_z \) be the vectors sampled by the algorithm, and \( v_j \) is picked in iteration \( \tau_j \), where \( 1 \leq j \leq z \). We first assume that the algorithm could check the ending condition after adding every single vector. In such case, it holds that

\[
\Pr \left[ \text{algorithm finishes after choosing } z \text{ vectors} \right] \\
\geq \Pr \left[ \sum_{j=1}^{z} \frac{6\varepsilon^2}{q} \cdot \frac{1}{(2n)^{1-1/q} \cdot \left( \Phi_{u_j, \ell_j}(A_{\tau_j}) \right)^{1/q}} \geq 2 \cdot (2n)^{1/q} \right] \\
= \Pr \left[ \sum_{j=1}^{z} \left( \Phi_{u_j, \ell_j}(A_{\tau_j}) \right)^{-1/q} \geq \frac{2qn}{3\varepsilon^2} \right].
\]
Following the same proof as the first part and noticing that in the final phase the algorithm chooses at most $O(n)$ extra vectors, we obtain the second statement.

4.3. Approximating the required quantities quickly for constructing graph sparsifiers. In this subsection we study fast approximation of the required quantities for constructing graph sparsifiers, and show that the number of sampled vectors $N_j$ and the relative effective resistances $\{R_i(A_j, u_j, \ell_j)\}_{i=1}^m$ used in each phase can be approximately computed in almost-linear time. For simplicity we drop the subscript $j$ expressing the iterations in this subsection. We assume that the following Assumption 4.7 holds on $A$, and show later that the matrix under consideration always satisfies this condition.

**Assumption 4.7.** Let $L$ and $\tilde{L}$ be the Laplacian matrices of graph $G$ and its subgraph after reweighting. Let $A = L^{-1/2}LL^{-1/2}$, and assume that

$$\ell + (\ell | \eta) \cdot I \prec A \prec (1 - \eta)u \cdot I$$

holds for some $0 < \eta < 1$.

**Lemma 4.8.** Under Assumption 4.7, the following statements hold:

- We can construct a matrix $S_u$ such that
  $$S_u \approx \frac{\varepsilon}{10} (uI - A)^{-1/2},$$
  and $S_u = p(A)$ for a polynomial $p$ of degree $O \left( \frac{\log(1/\varepsilon \eta)}{\eta} \right)$.

- We can construct a matrix $S_\ell$ such that
  $$S_\ell \approx \frac{\varepsilon}{10} (A - \ell I)^{-1/2}.$$

Moreover, $S_\ell$ is of the form $(A')^{-1/2}q((A')^{-1})$, where $q$ is a polynomial of degree $O \left( \frac{\log(1/\varepsilon \eta)}{\eta} \right)$ and $A' = L^{-1/2}L'LL^{-1/2}$ for some Laplacian matrix $L'$.

**Proof.** By Taylor expansion, it holds that

$$(1 - x)^{-1/2} = 1 + \sum_{k=1}^{\infty} \prod_{j=0}^{k-1} \left( j + \frac{1}{2} \right) \frac{x^k}{k!}.$$

We define for any $T \in \mathbb{N}$ that

$$p_T(x) = 1 + \sum_{k=1}^{T} \prod_{j=0}^{k-1} \left( j + \frac{1}{2} \right) \frac{x^k}{k!}.$$

Then, it holds for any $0 < x < 1 - \eta$ that

$$p_T(x) \leq (1 - x)^{-1/2} = p_T(x) + \sum_{k=T+1}^{\infty} \prod_{j=0}^{k-1} \left( j + \frac{1}{2} \right) \frac{x^k}{k!}$$

$$\leq p_T(x) + \sum_{k=T+1}^{\infty} x^k$$

$$\leq p_T(x) + \frac{(1 - \eta)^{T+1}}{\eta}.$$
Hence, it holds that

\[(uI - A)^{-1/2} = u^{-1/2}(I - u^{-1} A)^{-1/2} \leq u^{-1/2} p_T(u^{-1} A),\]

and

\[(uI - A)^{-1/2} \leq u^{-1/2} \left( p_T(u^{-1} A) + \frac{(1 - \eta)^{T+1}}{\eta} \cdot I \right),\]

since \(u^{-1} A \preceq (1 - \eta) I\). Notice that \(u^{-1/2} I \preceq (uI - A)^{-1/2}\), and therefore

\[(uI - A)^{-1/2} \leq u^{-1/2} p_T(u^{-1} A) + \frac{(1 - \eta)^{T+1}}{\eta} \cdot (uI - A)^{-1/2}.\]

Setting \(T = \frac{c \log((1/\epsilon)\eta)}{\eta}\) for some constant \(c\) and defining \(S_u = u^{-1/2} p_T(u^{-1} A)\) gives us that

\[S_u \approx_{\epsilon/10} (uI - A)^{-1/2}.\]

Now we turn to the second statement. Our construction of \(S_\ell\) is based on the case distinction \((\ell > 0, \ell \leq 0)\).

Case 1 \((\ell > 0)\): Notice that

\[(A' - \ell I)^{-1/2} = A^{-1/2}(I - \ell A^{-1})^{-1/2},\]

and

\[p_T(\ell A^{-1}) \leq (I - \ell A^{-1})^{-1/2} \leq p_T(\ell A^{-1}) + \frac{(1 - \eta)^{T+1}}{\eta} \cdot I.\]

Using the same analysis as before, we have that

\[A^{-1/2}(I - \ell A^{-1})^{-1/2} \approx_{\epsilon/10} A^{-1/2} p_T(\ell A^{-1}).\]

By defining \(S_\ell = A^{-1/2} p_T(\ell A^{-1})\), i.e., \(A' = A\) and \(q((A')^{-1}) = p_T(\ell A^{-1})\), we have that

\[S_\ell \approx_{\epsilon/10} (A - \ell I)^{-1/2}.\]

Case 2 \((\ell \leq 0)\): We look at the matrix

\[A - \ell I = L^{-1/2} L^{-1/2} - \ell I = L^{-1/2} (L - \ell L) L^{-1/2}.\]

Notice that \(L - \ell L\) is a Laplacian matrix, and hence this reduces to the case of \(\ell = 0\), for which we simply set \(S_\ell = (A - \ell I)^{-1/2}\). Therefore, we can write \(S_\ell\) as a desired form, where \(A' = A - \ell I\) and polynomial \(q = 1\).

The following two lemmas present nearly-linear time algorithms for computing the required quantities within each phase. We remark that an almost-linear time algorithm for computing similar quantities is shown in [1].

**Lemma 4.9.** Let \(A = \sum_{i=1}^m v_i v_i^T\), and suppose that \(A\) satisfies Assumption 4.7. Then, we can compute \(\{r_i\}_{i=1}^m\) and \(\{t_i\}_{i=1}^m\) in \(\tilde{O}\left(\frac{m}{\epsilon^2 \omega}\right)\) time such that

\[(1 - \epsilon)r_i \leq v_i^T(uI - A)^{-1} v_i \leq (1 + \epsilon)r_i,\]

and

\[(1 - \epsilon)t_i \leq v_i^T(A - \ell I)^{-1} v_i \leq (1 + \epsilon)t_i.\]
Proof. Define \( u_i = L^{1/2} v_i \) for any \( 1 \leq i \leq m \). By Lemma 4.8, we have that
\[
v_i^T (u I - A)^{-1} v_i \approx_{3\varepsilon/10} \| p(A) v_i \|^2 \\
= \| p \left( L^{-1/2} L^{-1/2} L^{-1/2} u_i \right) \|^2 \\
= \| L^{1/2} p \left( L^{-1} L^{-1} u_i \right) \|^2.
\]
Let \( L = B^T B \) for some \( B \in \mathbb{R}^{m \times n} \). Then, it holds that
\[
v_i^T (u I - A)^{-1} v_i \approx_{3\varepsilon/10} \| B p \left( L^{-1} L^{-1} u_i \right) \|^2.
\]
By the Johnson-Lindenstrauss lemma, there is a random matrix \( Q \in \mathbb{R}^{O(\log n/\varepsilon^2) \times m} \) such that, with high probability, it holds that
\[
v_i^T (u I - A)^{-1} v_i \approx_{4\varepsilon/10} \| Q B p \left( L^{-1} L^{-1} u_i \right) \|^2.
\]
We apply a nearly-linear time Laplacian solver to compute \( \| Q B p \left( L^{-1} L^{-1} u_i \right) \|^2 \) for all \( \{ u_i \}_{i=1}^m \) up to \((1 \pm \varepsilon/10)\)-multiplicative error in time \( \tilde{O} \left( \frac{m}{\eta^2} \right) \). This gives the desired \( \{ r_i \}_{i=1}^m \).

The computation for \( \{ t_i \}_{i=1}^m \) is similar. By Lemma 4.8, it holds for any \( 1 \leq i \leq m \) that
\[
v_i^T (A - \ell I)^{-1} v_i \approx_{3\varepsilon/10} \| (A')^{-1/2} q ((A')^{-1}) v_i \|^2 \\
= \| (A')^{-1/2} q \left( L^{1/2} (L')^{-1} L^{1/2} u_i \right) \|^2 \\
= \| (A')^{-1/2} L^{-1/2} q (L(L')^{-1}) u_i \|^2.
\]
Let \( L' = (B')^T (B') \) for some \( B' \in \mathbb{R}^{m \times n} \). Then, it holds that
\[
v_i^T (A - \ell I)^{-1} v_i \approx_{3\varepsilon/10} \| (L')^{-1/2} q \left( L(L')^{-1} u_i \right) \|^2 \\
= \| (L')^{-1/2} (L')^{-1} q (L(L')^{-1}) u_i \|^2 \\
= \| B' (L')^{-1} q (L(L')^{-1}) u_i \|^2.
\]
We invoke the Johnson-Lindenstrauss Lemma and a nearly-linear time Laplacian solver as before to obtain required \( \{ t_i \}_{i=1}^m \). The total runtime is \( \tilde{O} \left( \frac{m}{\eta^2} \right) \). \( \square \)

**Lemma 4.10.** Under Assumption 4.7, we can compute values \( \alpha, \beta \) in \( \tilde{O} \left( \frac{m}{\eta^2} \right) \) time such that
\[
(1 - \varepsilon) \alpha \leq \lambda_{\min}(u I - A) \leq (1 + \varepsilon) \alpha
\]
and
\[
(1 - \varepsilon) \beta \leq \lambda_{\min}(A - \ell I) \leq (1 + \varepsilon) \beta.
\]

**Proof.** By Lemma 4.8, we have \( S_u \approx_{\varepsilon/10} (u I - A)^{-1/2} \). Hence, \( \lambda_{\max}(S_u)^{-2} \approx_{3\varepsilon/10} \lambda_{\min}(u I - A) \), and it suffices to estimate \( \lambda_{\max}(S_u) \). Since
\[
\lambda_{\max}(S_u) \leq \left( \text{tr} \left( S_u^{2k} \right) \right)^{1/2k} \leq n^{1/2k} \lambda_{\max}(S_u),
\]

by picking \( k = \log n/\varepsilon \) we have that \( (\text{tr}(S_u^{2k}))^{1/2k} \approx_{\varepsilon/2} \lambda_{\max}(S_u) \). Notice that
\[
\text{tr} \left( S_u^{2k} \right) = \text{tr} \left( p^{2k} \left( L^{-1/2} \tilde{L} L^{-1/2} \right) \right) = \text{tr} \left( p^{2k} \left( L^{-1} \tilde{L} \right) \right).
\]

Set \( \tilde{L} = \tilde{B}^\top \tilde{B} \) for some matrix \( \tilde{B} \in \mathbb{R}^{m \times n} \), and we have that
\[
\text{tr} \left( S_u^{2k} \right) = \text{tr} \left( p^{2k} \left( \tilde{B} L^{-1} \tilde{B}^\top \right) \right).
\]

Since we can apply \( p^k \left( \tilde{B} L^{-1} \tilde{B}^\top \right) \) to vectors in \( \tilde{O} \left( \frac{m}{\varepsilon^2} \right) \) time, we invoke the Johnson-Lindenstrauss Lemma and approximate \( \text{tr} \left( S_u^{2k} \right) \) in \( \tilde{O} \left( \frac{m}{\varepsilon^2} \right) \) time.

We approximate \( \lambda_{\min}(A - \ell I) \) in a similar way. Notice that
\[
\text{tr} \left( S_A^{2k} \right) = \text{tr} \left( (A')^{-1/2} q((A')^{-1}) \right)^{4k} = \text{tr} \left( q((A')^{-1})(A')^{-1} q((A')^{-1}) \right)^{2k}.
\]

Let \( z \) be a polynomial defined by \( z(x) = xq^2(x) \) and \( L' = (B')^\top (B') \). Then, we have that
\[
\text{tr}(S_A^{2k}) = \text{tr} \left( z^{2k}((A')^{-1}) \right) = \text{tr} \left( z^{2k} \left( L'^{1/2}(L')^{-1} L^{1/2} \right) \right).
\]

Applying the same analysis as before, we can estimate the trace in \( \tilde{O} \left( \frac{m}{\varepsilon^2} \right) \) time. \( \square \)

### 4.4. Proof of the main results.

Now we analyze the runtime of the algorithm, and prove the main results.

**Proof of Theorem 1.1.** Without loss of generality, we assume that graph \( G = (V, E) \) is unweighted. Then, for every edge \( e = \{u, v\} \) in \( G \), we define \( b_e \in \mathbb{R}^n \), where \( b_e(x) = 1 \) if \( x = u \), \( b_e(x) = -1 \) if \( x = v \), and \( b_e(x) = 0 \) otherwise. We can further rewrite the Laplacian matrix of \( G \) as
\[
L = \sum_{e \in E[G]} b_e b_e^\top.
\]

By setting \( v_e = L^{-1/2} b_e \) for \( e \in E[G] \), it is easy to see that constructing a spectral sparsifier of \( G \) is equivalent to sparsifying the matrix \( \sum_{e \in E[G]} v_e v_e^\top \).

We first analyze the correctness of our algorithm. By Lemma 4.6, with probability at least \( 4/5 \) the algorithm chooses at most \( 10qn/\varepsilon^2 \) vectors, and by Lemma 4.5 the condition number of \( A_k \) is at most \( 1+O(\varepsilon) \), implying that the matrix \( A_k \) is a \( (1+O(\varepsilon)) \)-approximation of \( I \). These two results together prove that \( A_k \) is a linear-sized spectral sparsifier.

Next we analyze the runtime of the algorithm. By Lemma 4.2 and the union bound, with probability at least \( 9/10 \) all the matrices picked in \( k = \frac{10qn^{3/4}}{\varepsilon^2} \) phases satisfy
\[
W_j \preceq \frac{1}{2}(u_j I - A_j).
\]

Conditioning on the event,
\[
\mathbb{E} \left[ \Phi_{u_j, \ell_j}(A_j) \mid \forall j : W_j \preceq \frac{1}{2}(u_j I - A_j) \right] \leq 2.
\]
holds for any phase $j$, and by Markov’s inequality with high probability it holds that

$$\Phi_{u,j,\ell_j}(A_j) = O\left(qn^{3/q} / \varepsilon^2\right)$$

for all iterations $j$.

On the other hand, notice that it holds for any $1 \leq j \leq n$ that

$$(u - \lambda_j)^{-q} \leq \sum_{i=1}^{n} (u - \lambda_i)^{-q} < \Phi_{u,\ell}(A),$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $A$. This implies that $\lambda_j < u - (\Phi_{u,\ell}(A))^{-1/q}$. Similarly, it holds that $\lambda_j > \ell + (\Phi_{u,\ell}(A))^{-1/q}$ for any $1 \leq j \leq n$. Therefore, we have that

$$\ell_j + O\left(\left(\frac{\varepsilon^2}{qn^{3/q}}\right)^{1/q}\right) I \prec A_j \prec \left(u_j - O\left(\left(\frac{\varepsilon^2}{qn^{3/q}}\right)^{1/q}\right)\right) I.$$  

Since both of $u_j$ and $\ell_j$ are of the order $O(n^{1/q})$, we set $\eta = O\left((\varepsilon/n)^{2/q}\right)$ and obtain that

$$(\ell_j + |\ell_j|\eta) I \prec A_j \prec (1 - \eta)u_j I.$$  

Hence, we apply Lemma 4.9 and Lemma 4.10 to compute all required quantities in each phase up to constant approximation in time

$$O\left(\frac{m}{\varepsilon^2 \cdot \eta}\right) = O\left(\frac{m \cdot n^{2/q}}{\varepsilon^2 + 2^{1/q}}\right).$$

Since by Lemma 4.6 the algorithm finishes in $\frac{10qn^{3/q}}{\varepsilon^2}$ phases with probability at least 4/5, the total runtime of the algorithm is

$$O\left(\frac{q \cdot m \cdot n^{5/q}}{\varepsilon^{4+1/q}}\right).$$

**Proof of Theorem 1.2.** The correctness of our algorithm follows by the proof of Theorem 1.1. For the runtime, Lemma 4.6 proves that the algorithm finishes in $\frac{10qn^{3/q}}{\varepsilon^2}$ phases, and it is easy to see that all the required quantities in each phase can be approximately computed in $O(q \cdot m \cdot n^{\omega-1})$ time using fast matrix multiplication. Therefore, the total runtime of the algorithm is $O\left(\frac{q \cdot m \cdot n^{\omega-1+3/q}}{\varepsilon^2}\right)$.

**Acknowledgment.** This work was partially supported by NSF awards 0843915 and 1111109. Part of this work was done while both authors were visiting the Simons Institute for the Theory of Computing, UC Berkeley. We would like to thank Michael Cohen for pointing out a gap in a previous version of the paper and his fixes for the gap, as well as Lap-Chi Lau for insightful comments on improving the presentation of the paper.

**REFERENCES**


