Chapter 10

Barrier Method

In Chapter 9, we have seen that spectral sparsification is a stronger notion than cut sparsification, but this provides a linear algebraic formulation that connects the problem to more general mathematical phenomenon, which leads to an elegant solution that matches the best known result in cut sparsification. In this chapter, we will see that this stronger notion even leads to a surprisingly strong solution that goes beyond what was known (or perhaps thought possible) for cut sparsification. The main theorem that we will study is by Batson, Spielman and Srivastava.

Theorem 10.1 (Linear-Sized Spectral Approximator of Identity Matrix [BSS14]). *For any m vectors $v_1, \ldots, v_m \in \mathbb{R}^n$ satisfying $\sum_{i=1}^m v_i v_i^T = I_n$, there always exist scalars $s_1, \ldots, s_m$ with at most dn nonzeros such that*

$$\left(1 - \frac{1}{\sqrt{d}}\right)^2 \cdot I_n \preccurlyeq \sum_{i=1}^m s_i v_i v_i^T \preccurlyeq \left(1 + \frac{1}{\sqrt{d}}\right)^2 \cdot I_n.$$

It follows from the reduction in Lemma 9.11 that every graph has a linear-sized spectral sparsifier.

Theorem 10.2 (Linear-Sized Spectral Sparsifier [BSS14]). *For any edge-weighted undirected graph $G = ([n], E)$ and any $0 < \epsilon \leq 1$, there is a reweighted subgraph $H = ([n], F)$ on the same vertex set with at most $O(n/\epsilon^2)$ edges such that H is a $(1 \pm \epsilon)$-spectral approximator of G.*

One corollary is that every graph has a $(1 \pm \epsilon)$-cut sparsifier with at most $O(n/\epsilon^2)$ edges, which improves upon Theorem 9.5 by Benczur and Karger. It is quite remarkable that a harder problem leads to a stronger solution in a well-studied special case. Up until now, there is no known alternative way to obtain linear-sized cut sparsifiers without going through the concept of spectral sparsification. It is an interesting challenge especially to those who prefer to see combinatorial algorithms to solve combinatorial problems.

10.1 Deterministic Algorithm and Polynomial Perspective

The approach taken to prove Theorem 10.1 is different from the random sampling approach used in previous algorithms in Theorem 9.2, Theorem 9.5, and Theorem 9.9. It is a deterministic "greedy" approach that uses a potential function to guide the algorithm to add one vector at a time.

Intuition from Characteristic Polynomials

As discussed in [BSS14], the intuition of their approach is from a polynomial perspective to the problem. Let $A \in \mathbb{R}^{n \times n}$ be the current partial solution, where $A = 0$ initially. They considered the characteristic polynomial $p_A(x) = \det(xI - A) = \prod_{j=1}^n (x - \lambda_j)$ whose roots are the eigenvalues of A, and studied how it changes after adding one vector. By the matrix determinant formula in Fact 2.29,

$$p_{A+vv^T}(x) = \det(xI - A - vv^T) = \det(xI - A) \cdot \left(1 - v^T(xI - A)^{-1}v\right) = p_A(x) \cdot \left(1 - \sum_{j=1}^n \frac{\langle v, u_j \rangle^2}{x - \lambda_j}\right).$$

where λ_j are the eigenvalues of A and u_j are the corresponding orthonormal eigenvectors. Suppose we add a uniformly random vector v from $v_1, \ldots, v_m$ to A. Then, by the isotropy assumption,

$$\mathbb{E}\left[\langle v, u_j \rangle^2\right] = \frac{1}{m} \sum_{i=1}^m \langle v_i, u_j \rangle^2 = \frac{1}{m} \cdot u_j^T \left(\sum_{i=1}^m v_i v_i^T\right) u_j = \frac{\|u_j\|^2}{m} = \frac{1}{m}.$$

This implies that the expected characteristic polynomial is

$$\mathbb{E}\left[p_{A+vv^T}(x)\right] = p_A(x) \left(1 - \frac{1}{m} \sum_{j=1}^n \frac{1}{x - \lambda_j}\right) = p_A(x) - \frac{1}{m} \partial_x p_A(x),$$

as $\partial_x p_A(x) / p_A(x) = \sum_{j=1}^n 1/(x - \lambda_j)$. Since we start from $A = 0$, the initial polynomial is $p_A(x) = x^n$. After t iterations, the expected characteristic polynomial becomes

$$p_t(x) = \left(1 - \frac{1}{m} \partial_x\right)^t x^n.$$

This generates a standard family of orthogonal polynomials, called the associated Laguerre polynomials, whose roots are known. After $t = dn$ iterations, the ratio of the largest root to the smallest root of $p_{dn}(x)$ is known to be

$$\frac{d + 1 + 2\sqrt{d}}{d + 1 - 2\sqrt{d}},$$

and this is the ratio of the maximum eigenvalue and the minimum eigenvalue in Theorem 10.1.

This is only a heuristic argument, as there may not be any vector v with its characteristic polynomial $p_{A+vv^T}(x)$ equal to the expected characteristic polynomial. The proof of Theorem 10.1 in [BSS14] is also not based on this approach, but this foreshadows the polynomial approach that we will study in the second part of this course.

Algorithm Structure

As discussed in Chapter 9, one advantage of the algebraic formulation for spectral sparsification in Theorem 9.9 is that we "only" need to keep track of the maximum eigenvalue and the minimum eigenvalue of the current partial solution, instead of the exponentially many cut values as was done in the cut sparsification problem. So the general idea is to maintain an upper bound of the maximum eigenvalue and a lower bound on the minimum eigenvalue of the current partial solution, and to control how they evolve over time. This will be done using two potential functions Φ^u and Φ_l that we will define and study in the next section.

Assuming the existence of the two potential functions, we first describe the structure of the deterministic "greedy" algorithm. Initially, we start from the empty solution $A_0 = 0$, some upper bound u_0 of the maximum eigenvalue of A_0, some lower bound l_0 of the minimum eigenvalue, so that the potential values $\Phi^{u_0}(A_0) \leq \phi_u$ and $\Phi_{l_0}(A_0) \leq \phi_l$ for some values ϕ_u and ϕ_l that will be fixed throughout the algorithm. In each iteration t, we find a vector v_i and a scalar s and add $s \cdot v_i v_i^T$ to the current solution so that $A_{t+1} \leftarrow A_t + s v_i v_i^T$, and shift the upper bound $u_{t+1} \leftarrow u_t + \delta_u$ and the lower bound $l_{t+1} \leftarrow l_t + \delta_l$ by some fixed amount δ_u and δ_l to maintain the invariants that $\Phi^{u_{t+1}}(A_{t+1}) \leq \phi_u$ and $\Phi^{l_{t+1}}(A_{t+1}) \leq \phi_l$ and also u_{t+1} and l_{t+1} are upper and lower bounds of the maximum eigenvalue and the minimum eigenvalue of A_{t+1} respectively.

Algorithm 7 Deterministic Greedy Algorithm for Spectral Sparsification

```
Require: Vectors $v_1, \ldots, v_m \in \mathbb{R}^n$ satisfying $\sum_{i=1}^m v_i v_i^T = I_n$.
1: Initialization: $A_0 = 0$ and $\tau = dn$.
2: Choose $u_0, l_0, \phi_u, \phi_l$ so that $\Phi^{u_0}(A_0) \leq \phi_u$ and $\Phi_{l_0}(A_0) \leq \phi_l$.
3: Choose two parameters $\delta_u$ and $\delta_l$ and set $u_t = u_0 + t\delta_u$ and $l_t = l_0 + t\delta_l$ for any $t \geq 1$.
4: for $1 \leq t \leq \tau$ do
5:     Find vector $v \in \{v_1, \ldots, v_m\}$ and scalar $s$ and set $A_t = A_{t-1} + s \cdot vv^T$ to maintain the invariants
       that $\Phi^{u_t}(A_t) \leq \phi_u$ and $\Phi_{l_t}(A_t) \leq \phi_l$ and $\lambda_{\max}(A_t) \leq u_t$ and $\lambda_{\min}(A_t) \geq l_t$.
6: end for
7: return $A_\tau$.
```

There are many parameters $u_0, l_0, \phi_u, \phi_l, \delta_u, \delta_l$ to be chosen, and we will only do so in the end.

10.2 Potential Functions

The magical element in this algorithm is the definition of the potential functions, that will make everything works beautifully. Before we state the potential functions used in [BSS14], let us discuss some natural attempts and see what we need.

Norm of Eigenvalues

A natural first attempt is to simply use the maximum eigenvalue and the minimum eigenvalue as the potential functions (i.e. $\Phi^u(A_t) = \lambda_{\max}(A_t)$ and $\Phi_l(A_t) = \lambda_{\min}(A_t)$), and then inductively prove that $\lambda_{\max}(A_t) \leq \lambda_{\max}(A_{t-1}) + \delta_u$ and $\lambda_{\min}(A_t) \leq \lambda_{\min}(A_{t-1}) + \delta_l$. This way of measuring progress does not work well for this problem, as the matrix A_t is n-dimensional, and just focusing on the maximum direction cannot distinguish between the case where every direction is large or where one direction is large and all other orthogonal directions are small. Ideally, we hope to say something such as after n iterations, every direction is increased by one unit. To prove it inductively, we would need a potential function to let us argue that the maximum direction is increased by $1/n$ unit per edge *on average*, but the maximum eigenvalue is not such a smooth/robust quantity for this.

By the above discussion, we would like to have a more global quantity that will take into consideration of all directions. One possible parameter of this kind is $\frac{1}{n}\operatorname{Tr}(A)$, which is the average eigenvalue of the current solution. For this, we can easily argue that the average eigenvalue increases smoothly, but the problem is that we cannot conclude that the maximum eigenvalue is small by using that the average eigenvalue is small.

So, we would like to have a more global quantity that is smooth enough to measure the progress made in each iteration, and also that the maximum eigenvalue is small when this quantity is small. Let $\vec{\lambda} = \big(\lambda_1(A), \lambda_2(A), \ldots, \lambda_n(A)\big)$ be the spectrum of the current solution. Note that $\lambda_{\max}(A) := \|\vec{\lambda}\|_\infty$ is the infinity norm of the spectrum, while $\mathrm{Tr}(A) := \|\vec{\lambda}\|_1$ is the 1-norm of the spectrum. Interpolating between these two extremes, we may consider the quantity $\left(\frac{1}{n}\sum_{i=1}^n \lambda_i^p\right)^{1/p} = n^{-1/p} \cdot \|\lambda\|_p$. We know that setting $p \approx \log n$ would approximate $\|\vec{\lambda}\|_\infty$ well, but the p-norm may not be so convenient for calculations. In convex optimization, there is a softmax function that is defined as $\log \sum_{i=1}^n \exp(\lambda_i)$, which is known to be convex and differentiable and approximates the maximum well. In our setting, the softmax function can be nicely written as $\log \mathrm{Tr}(e^A)$, where $e^A := \sum_{k=0}^\infty \frac{1}{k!} X^k$ is the matrix exponential of A. So this seems to be a good potential function to be used for spectral sparsification, and indeed this function is used in the proof of the matrix Chernoff bound. I think this function can be used in Algorithm 7 to give a deterministic algorithm with the same guarantee as the random sampling algorithm by Spielman and Srivastava in Theorem 9.9 (please see [dCSHS16]), but it is not enough for linear-sized spectral sparsification.

Barrier Functions

Batson, Spielman, and Srivastava mentioned in [BSS14] that the definition of their potential functions is inspired by the calculation of the expected characteristic polynomial in Section 10.1.

Definition 10.3 (Barrier Functions). *Given $u, l \in \mathbb{R}$ and a real symmetric matrix $A \in \mathbb{R}^{n \times n}$ with eigenvalues $\lambda_1, \ldots, \lambda_n$, the upper barrier function and the lower barrier function are defined as*

$$\Phi^A(u) := \Phi^u(A) := \mathrm{Tr}(uI_n - A)^{-1} = \sum_{i=1}^n \frac{1}{u - \lambda_i} \quad \text{and} \quad \Phi_A(l) := \Phi_l(A) := \mathrm{Tr}(A - lI_n)^{-1} = \sum_{i=1}^n \frac{1}{\lambda_i - l}.$$

We will use the notations $\Phi^u(A)$ and $\Phi_l(A)$ when we fix u and l and see the barrier functions as a function of the matrix A, and we will use the notations $\Phi^A(u)$ and $\Phi_A(l)$ when we fix A and see the barrier functions as a function of u or l.

When $u > \lambda_{\max}(A)$ and $l < \lambda_{\min}(A)$, these functions measure how far the eigenvalues of A are from the barriers u and l, and they blow up as any eigenvalue approaches a barrier. Suppose we could maintain the invariant that say $\Phi^{u_t}(A) \leq 1$ for all t. This will ensure that u_t is a "comfortable" upper bound of the maximum eigenvalue, as there could be at most one eigenvalue with value at least $u_t - 1$, at most two eigenvalues with value at least $u_t - 2$, and so on. This is a more global quantity that takes all the eigenvalues into consideration, and has the property that it changes smoothly to measure the progress made in each iteration.

The following properties of the barrier functions are simple but useful. We will see a generalization in the multivariate setting in the second part of the course.

Exercise 10.4 (Monotonicity and Convexity). *Let $A \in \mathbb{R}^{n \times n}$ be a real symmetric matrix. For any $u > \lambda_{\max}(A)$ and any $\delta > 0$, the upper barrier function satisfies*

$$\Phi^A(u) \geq \Phi^A(u + \delta) \quad \text{and} \quad \Phi^A(u) + \delta \cdot \big(\Phi^A(u + \delta)\big)' \geq \Phi^A(u + \delta).$$

For any $l + \delta < \lambda_{\min}(A)$ and any $\delta > 0$, the lower barrier function satisfies

$$\Phi_A(l) \leq \Phi_A(l + \delta) \quad \text{and} \quad \Phi_A(l) + \delta \cdot \big(\Phi_A(l + \delta)\big)' \geq \Phi_A(l + \delta).$$

The strategy in Algorithm 7 is to ensure that u_t is increased slowly while maintaining the invariant that the potential value $\Phi^{u_t}(A_t)$ is small. More explicitly, we can define a family of “soft” bounds on the max/min eigenvalue, parameterized by the value of the potential functions.

Definition 10.5 (ϕ-Soft-Max and ϕ-Soft-Min). *Given a real symmetric matrix $A \in \mathbb{R}^{n\times n}$ and a parameter $\phi > 0$, the ϕ-max of A and the ϕ-min of A are defined as*

$$\phi\text{-}\max(A) := \max\{u \mid \Phi^A(u) = \phi\} \quad \textit{and} \quad \phi\text{-}\min(A) := \min\{l \mid \Phi_A(l) = \phi\}.$$

They can be understood as the inverse of the upper and lower barrier functions.

The parameter ϕ can be thought of as a sensitivity parameter, which controls the tradeoff between how accurate the bound is and how smoothly it varies. The strategy in Algorithm 7 is to fix an ϕ and then bound $\phi\text{-}\max(A_t)$ and prove that $\phi\text{-}\max(A_t) \le \phi\text{-}\max(A_{t-1}) + \delta_u$ for all $t \ge 1$.

The barrier functions are similar to the log-barrier functions used in the interior point method for convex optimization.

Remark 10.6 (Log-Barrier Functions). *Let $p_A(x) = \det(xI - A)$ be the characteristic polynomial of A. Note that*

$$\Phi^x(A) = \frac{\partial_x p_A(x)}{p_A(x)} = \partial_x \log\big(p_A(x)\big) \quad \textit{and} \quad \Phi_x(A) = -\partial_x \log\big(p_A(x)\big).$$

These functions blow up when x is getting close to a root.

10.3 Changes of Potential Values

There are nice formulas to analyze the change of the barrier functions when we add a vector and do a rank-one update.

Upper Barrier Function

For the upper barrier function $\Phi^u(A)$, adding a vector $s \cdot vv^T$ would increase the potential value, but increasing the upper bound u would compensate for it to maintain the invariant $\Phi^{u+\delta_u}(A + s\cdot vv^T) \le \Phi^u(A)$.

Lemma 10.7 (Upper Barrier Change). *Suppose $u > \lambda_{\max}(A)$. For any vector v, if*

$$\frac{1}{s} \ge \frac{v^T\big((u+\delta_u)I - A\big)^{-2} v}{\Phi^u(A) - \Phi^{u+\delta_u}(A)} + v^T\big((u+\delta_u)I - A\big)^{-1} v =: U_A(v),$$

then

$$\Phi^{u+\delta_u}\big(A + s\cdot vv^T\big) \le \Phi^u(A) \quad \textit{and} \quad \lambda_{\max}\big(A + s\cdot vv^T\big) < u + \delta_u.$$

Proof. Let $u' := u + \delta_u$. By the Sherman-Morrison rank-one update formula in Fact 2.20,

$$\begin{aligned}
\Phi^{u+\delta_u}(A + s \cdot vv^T) &= \operatorname{Tr}\left(\left(u'I - A - s \cdot vv^T\right)^{-1}\right) \\
&= \operatorname{Tr}\left(\left(u'I - A\right)^{-1} + \frac{s(u'I-A)^{-1}vv^T(u'I-A)^{-1}}{1 - s \cdot v^T(u'I-A)^{-1}v}\right) \\
&= \Phi^{u+\delta_u}(A) + \frac{s \cdot v^T(u'I-A)^{-2}v}{1 - s \cdot v^T(u'I-A)^{-1}v} \\
&= \Phi^u(A) - \underbrace{\left(\Phi^u(A) - \Phi^{u+\delta_u}(A)\right)}_{\text{gain}} + \underbrace{\frac{v^T(u'I-A)^{-2}v}{1/s - v^T(u'I-A)^{-1}v}}_{\text{loss}}
\end{aligned}$$

Rearranging shows that $\Phi^{u+\delta_u}\left(A + s \cdot vv^T\right) \leq \Phi^u(A)$ when $1/s \geq U_A(v)$. This also implies that $\lambda_{\max}(A + s \cdot vv^T) \leq u + \delta_u$, as otherwise $\lambda_{\max}(A + s' \cdot vv^T) = u + \delta_u$ for some $s' \leq s$ and thus $\Phi^{u+\delta_u}\left(A + s' \cdot vv^T\right) = \infty$, but this contradicts that $\Phi^{u+\delta_u}\left(A + s' \cdot vv^T\right) \leq \Phi^u(A)$ is bounded. □

Lower Barrier Function

For the lower barrier function $\Phi_l(A)$, adding a vector $s \cdot vv^T$ would decrease the potential value, but increasing the lower bound l would increase the potential value. Note that there is an additional condition about the barrier value to ensure that we still have a lower bound on the minimum eigenvalue.

Lemma 10.8 (Lower Barrier Change)**.** *Suppose $\lambda_{\min}(A) > l$ and $\Phi_l(A) \leq 1/\delta_l$. For any vector v, if*

$$0 < \frac{1}{s} \leq \frac{v^T\left(A - (l+\delta_l)I\right)^{-2}v}{\Phi_{l+\delta_l}(A) - \Phi_l(A)} - v^T\left(A - (l+\delta_l)I\right)^{-1}v =: L_A(v),$$

then

$$\Phi_{l+\delta_l}\left(A + s \cdot vv^T\right) \leq \Phi_l(A) \quad \textit{and} \quad \lambda_{\min}\left(A + s \cdot vv^T\right) > l + \delta_l.$$

Proof. Note that $\lambda_{\min}(A) > l$ and $\Phi_l(A) = \sum_{i=1}^n 1/(\lambda_i - l) \leq 1/\delta_l$ imply that $1/(\lambda_{\min} - l) < 1/\delta_l$ and thus $\lambda_{\min} > l + \delta_l$. So, $\lambda_{\min}(A + s \cdot vv^T) \geq \lambda_{\min}(A) > l + \delta_l$. Then, by a similar calculation using the Sherman-Morrison formula as in Lemma 10.7,

$$\Phi_{l+\delta_l}(A + s \cdot vv^T) = \Phi_l(A) + \underbrace{\left(\Phi_{l+\delta_l}(A) - \Phi_l(A)\right)}_{\text{loss}} - \underbrace{\frac{v^T(A - l'I)^{-2}v}{1/s + v^T(A - l'I)^{-1}v}}_{\text{gain}}.$$

Rearranging shows that $\Phi_{l+\delta_l}\left(A + s \cdot vv^T\right) \leq \Phi_l(A)$ when $1/s \leq L_A(v)$. □

10.4 Averaging Argument

We need to prove that there exists a vector v and a scalar s such that both the assumptions in Lemma 10.7 and Lemma 10.8 hold, so that we can conclude that the invariants $\Phi^{u+\delta_u}\left(A+s\cdot vv^T\right) \leq \Phi^u(A)$ and $\lambda_{\max}\left(A+s\cdot vv^T\right) < u+\delta_u$ and $\Phi_{l+\delta_l}\left(A+s\cdot vv^T\right) \leq \Phi_l(A)$ and $\lambda_{\min}\left(A+s\cdot vv^T\right) > l+\delta_l$ in Algorithm 7 hold simultaneously.

The idea in [BSS14] is to prove that $\sum_{i=1}^m L_A(v_i) \geq \sum_{i=1}^m U_A(v_i)$, and so there exists a vector v_i such that $L_A(v_i) \geq U_A(v_i)$. Therefore, by setting s to be a scalar such that $L_A(v_i) \geq 1/s \geq U_A(v_i)$, then both the assumptions in Lemma 10.7 and Lemma 10.8 are satisfied and thus all the invariants hold simultaneously for $A + s \cdot v_i v_i^T$.

The calculations work out quite nicely using the isotropy condition $\sum_{i=1}^m v_i v_i^T = I_n$.

Upper Barrier Function

Lemma 10.9 (Total Upper Barrier Shift)**.** *Given $v_1, \ldots, v_m \in \mathbb{R}^n$ such that $\sum_{i=1}^m v_i v_i^T = I_n$,*

$$\sum_{i=1}^m U_A(v_i) \leq \frac{1}{\delta_u} + \Phi^u(A).$$

Proof. Using the isotropy assumption $\sum_{i=1}^m v_i v_i^T = I_n$, it follows that

$$\sum_{i=1}^m v_i^T \big((u+\delta_u)I - A\big)^{-2} v_i = \sum_{i=1}^m \operatorname{Tr}\Big(\big((u+\delta_u)I - A\big)^{-2} v_i v_i^T\Big) = \operatorname{Tr}\Big(\big((u+\delta_u)I - A\big)^{-2}\Big),$$

and similarly

$$\sum_{i=1}^m v_i^T \big((u+\delta_u)I - A\big)^{-1} v_i = \operatorname{Tr}\Big(\big((u+\delta_u)I - A\big)^{-1}\Big) = \Phi^{u+\delta_u}(A).$$

By the convexity of the barrier function $\Phi^u(A) = \Phi^A(u)$ in terms of u in Exercise 10.4, the "gain" is

$$\Phi^u(A) - \Phi^{u+\delta_u}(A) = \Phi^A(u) - \Phi^A(u+\delta_u) \geq -\delta_u \cdot \Big(\Phi^A(u+\delta_u)\Big)' = \delta_u \cdot \operatorname{Tr}\Big(\big((u+\delta_u)I - A\big)^{-2}\Big).$$

Therefore,

$$\begin{aligned}
\sum_{i=1}^m U_A(v_i) \quad &:= \quad \sum_{i=1}^m \left(\frac{v_i^T \big((u+\delta_u)I - A\big)^{-2} v_i}{\Phi^u(A) - \Phi^{u+\delta_u}(A)} + v_i^T \big((u+\delta_u)I - A\big)^{-1} v_i \right) \\
&= \quad \frac{\operatorname{Tr}\big((u+\delta_u)I - A\big)^{-2}}{\Phi^u(A) - \Phi^{u+\delta_u}(A)} + \Phi^{u+\delta_u}(A) \\
&\leq \quad \frac{1}{\delta_u} + \Phi^u(A).
\end{aligned}$$

□

Lower Barrier Function

The calculations for the total lower barrier shift is similar, but is a bit trickier. Note that the following lemma also requires the assumption that $\Phi_l(A) \leq 1/\delta_l$ as in Lemma 10.8.

Lemma 10.10 (Total Lower Barrier Shift)**.** *Given $v_1, \ldots, v_m \in \mathbb{R}^n$ such that $\sum_{i=1}^m v_i v_i^T = I_n$, if $\Phi_l(A) \leq 1/\delta_l$, then*

$$\sum_{i=1}^m L_A(v_i) \geq \frac{1}{\delta_l} - \Phi_l(A).$$

Proof. As in the proof of Lemma 10.9, using the isotropy assumption $\sum_{i=1}^m v_i v_i^T = I_n$,

$$\sum_{i=1}^m v_i^T \big(A-(l+\delta_l)I\big)^{-2} v_i = \mathrm{Tr}\left(\big(A-(l+\delta_l)I\big)^{-2}\right) \quad \text{and} \quad \sum_{i=1}^m v_i^T \big(A-(l+\delta_l)I\big)^{-1} v_i = \Phi_{l+\delta_l}(A).$$

Therefore,

$$\begin{aligned}\sum_{i=1}^m L_A(v_i) &:= \sum_{i=1}^m \left(\frac{v_i^T \big(A-(l+\delta_l)I\big)^{-2} v_i}{\Phi_{l+\delta_l}(A)-\Phi_l(A)} - v_i^T \big(A-(l+\delta_l)I\big)^{-1} v_i \right) \\ &= \frac{\mathrm{Tr}\big(A-(l+\delta_l)I\big)^{-2}}{\Phi_{l+\delta_l}(A)-\Phi_l(A)} - \Phi_{l+\delta_l}(A).\end{aligned}$$

Using convexity in Exercise 10.4 as in the proof of Lemma 10.9,

$$\Phi_{l+\delta_l}(A)-\Phi_l(A) = \Phi_A(l+\delta_l)-\Phi_A(l) \le \delta_l \cdot \big(\Phi_A(l+\delta_l)\big)' = \delta_l \cdot \mathrm{Tr}\big(A-(l+\delta_l)I\big)^{-2}.$$

This gives $\sum_{i=1}^m L_A(v_i) \ge \frac{1}{\delta_l} - \Phi_{l+\delta_l}(A)$, which is slightly weaker than the statement and is not enough for the invariants to hold throughout the algorithm. To prove the statement, we need to work harder and show that

$$\frac{\mathrm{Tr}\big(A-(l+\delta_l)I\big)^{-2}}{\Phi_{l+\delta_l}(A)-\Phi_l(A)} - \Phi_{l+\delta_l}(A) \ge \frac{1}{\delta_l} - \Phi_l(A),$$

which is equivalent to the following claim by rearranging.

Claim 10.11 (Lemma 4.3 of [MSS21]). *If $\Phi_l(A) \le 1/\delta_l$, then*

$$\big(\Phi_{l+\delta_l}(A)-\Phi_l(A)\big)^2 \le \mathrm{Tr}\big(A-(l+\delta_l)I\big)^{-2} - \frac{1}{\delta_l}\big(\Phi_{l+\delta_l}(A)-\Phi_l(A)\big).$$

Proof. By definition of the lower barrier function in Definition 10.3

$$\big(\Phi_{l+\delta_l}(A)-\Phi_l(A)\big)^2 = \left(\sum_{i=1}^n \frac{1}{\lambda_i-(l+\delta_l)} - \frac{1}{\lambda_i - l}\right)^2 = \left(\sum_{i=1}^n \frac{\delta_l}{\big(\lambda_i-(l+\delta_l)\big)\cdot(\lambda_i-l)}\right)^2$$

Using Cauchy-Schwarz inequality and then the assumption $\delta_l \cdot \Phi_l(A) \le 1$, the RHS is

$$\le \left(\sum_{i=1}^n \frac{\delta_l}{(\lambda_i-l)}\right)\left(\sum_{i=1}^n \frac{\delta_l}{\big(\lambda_i-(l+\delta_l)\big)^2\cdot(\lambda_i-l)}\right) \le \left(\sum_{i=1}^n \frac{\delta_l}{\big(\lambda_i-(l+\delta_l)\big)^2\cdot(\lambda_i-l)}\right).$$

Check that this is equal to the RHS of the statement of this claim. □

The claim completes the proof of this lemma. □

Both Barrier Functions

Combining Lemma 10.9 and Lemma 10.10 with the averaging argument in the beginning of this section, we arrive at the following conditions for the invariants in Algorithm 7 to hold throughout.

Lemma 10.12 (Invariants). *Let $A_0 = 0$. If we choose $u_0 > 0, l_0 < 0, \phi_u, \phi_l, \delta_u, \delta_l$ so that*

$$\Phi^{u_0}(A_0) \leq \phi_u \quad \textit{and} \quad \Phi_{l_0}(A_0) \leq \phi_l \quad \textit{and} \quad \phi_l \leq \frac{1}{\delta_l} \quad \textit{and} \quad \frac{1}{\delta_l} - \phi_l \geq \frac{1}{\delta_u} + \phi_u,$$

then Algorithm 7 can always find a vector v and a scalar s in each iteration t to maintain the invariants that $\Phi^{u_t}(A_t) \leq \phi_u$ and $\Phi_{l_t}(A_t) \leq \phi_l$ and $\lambda_{\max}(A_t) \leq u_t$ and $\lambda_{\min}(A_t) \geq l_t$, where $u_t = u_0 + t\delta_u$ and $l_t = l_0 + t\delta_l$ as defined in Algorithm 7.

Proof. The proof is by a simple induction. The induction hypothesis is that $\Phi^{u_t}(A_t) \leq \phi_u$ and $\Phi_{l_t}(A_t) \leq \phi_l$ and $\lambda_{\max}(A_t) \leq u_t$ and $\lambda_{\min}(A_t) \geq l_t$. This holds at $t = 0$ by our assumptions. For the induction step, by Lemma 10.9 and Lemma 10.10 and our assumption,

$$\sum_{i=1}^{m} U_{A_t}(v_i) \leq \frac{1}{\delta_u} + \Phi^{u_t}(A_t) \leq \frac{1}{\delta_u} + \phi_u \leq \frac{1}{\delta_l} - \phi_l \leq \frac{1}{\delta_l} - \Phi_{l_t}(A_t) \leq \sum_{i=1}^{m} L_{A_t}(v_i).$$

So there exists some $v \in \{v_1, \ldots, v_m\}$ such that $U_{A_t}(v) \leq L_{A_t}(v)$. Let s be a scalar such that $U_{A_t}(v) \leq 1/s \leq L_{A_t}(v)$. Then, it follows from Lemma 10.7 and Lemma 10.8 that the invariants hold for $t+1$ with $A_{t+1} = A_t + s \cdot vv^T$ and $u_{t+1} = u_t + \delta_u$ and $l_{t+1} = l_t + \delta_l$. □

Wrapping Up

With Lemma 10.12, it remains to choose $u_0, l_0, \phi_u, \phi_l, \delta_u, \delta_l$ to prove Theorem 10.1. Batson, Spielman and Srivastava set

$$l_0 := -\sqrt{d}n, \quad u_0 := \Big(\frac{d+\sqrt{d}}{\sqrt{d}-1}\Big)n, \quad \phi_l := \Phi_{l_0}(A_0) = -\frac{n}{l_0} = \frac{1}{\sqrt{d}}, \quad \phi_u := \Phi^{u_0}(A_0) = \frac{n}{u_0} = \frac{\sqrt{d}-1}{d+\sqrt{d}},$$

so that the first three conditions in Lemma 10.12 are satisfied. Then, they set

$$\delta_l := 1 \quad \text{and} \quad \delta_u := \frac{\sqrt{d}+1}{\sqrt{d}-1} \quad \implies \quad \frac{1}{\delta_l} - \phi_l = \frac{1}{\delta_u} + \phi_u,$$

and so the last condition in Lemma 10.12 is also satisfied. Therefore, after dn iterations of Algorithm 7,

$$\frac{\lambda_{\max}(A_{dn})}{\lambda_{\min}(A_{dn})} \leq \frac{u_{dn}}{l_{dn}} = \frac{u_0 + dn \cdot \delta_u}{l_0 + dn \cdot \delta_l} = \frac{\frac{d+\sqrt{d}}{\sqrt{d}-1} + d \cdot \frac{\sqrt{d}+1}{\sqrt{d}-1}}{-\sqrt{d}+d} = \left(\frac{\sqrt{d}+1}{\sqrt{d}-1}\right)^2,$$

completing the proof of Theorem 10.1.

10.5 Discussions

There are many subsequent work on spectral sparsification and we discuss some of them here.

- Allen-Zhu, Liao, and Orecchia [ALO15] constructed linear-sized spectral sparsifier using the regret minimization framework in convex optimization. This provides a more systematic way to derive the result and a different interpretation of Batson, Spielman, Srivastava's result as using a different regularizer in the regret minimization framework. Their tools developed are also more convenient in some applications as we will discuss more in the next chapter.

- Lee and Sun gave an almost linear time algorithm [LS18] and then a nearly linear time algorithm [LS17] to construct linear-sized spectral sparsifiers. Their first algorithm in [LS18] is an adaptive sampling algorithm that is interesting and easy to describe. In the first iteration, the algorithm samples say $n^{0.99}$ vectors using effective resistance as in Theorem 9.9. Then, it will update the sampling probability using barrier functions, and repeat this process for $n^{0.01}$ iterations. Their intuition is from balls-and-bins that the maximum load is only a constant after throwing $n^{0.99}$ balls to n bins.

- One may wonder whether there are other sampling algorithms for constructing spectral sparsifiers. An interesting result by Kyng and Song [KS18] is that the union of $\log n/\epsilon^2$ random spanning trees is an $(1 \pm \epsilon)$-spectral approximator, and this is tight.

- de Carli Silva, Harvey and Sato [dCSHS16] generalized the result in Theorem 10.1 to sparsifying sums of positive semidefinite matrices that have arbitrary rank, with applications in hypergraph sparsification.

A main open question is if every vector is of the same length (or more generally when every vector is short), then is there an efficient algorithm to construct an *unweighted* sparsifier? This is closely related to the Kadison-Singer problem that we will study in the second part of the course.

10.6 References

[ALO15] Zeyuan Allen-Zhu, Zhenyu Liao, and Lorenzo Orecchia. Spectral sparsification and regret minimization beyond matrix multiplicative updates. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 237–245. ACM, 2015. 105, 108, 113, 114

[BSS14] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Rev.*, 56(2):315–334, 2014. 97, 98, 99, 100, 103, 108, 110

[dCSHS16] Marcel Kenji de Carli Silva, Nicholas J. A. Harvey, and Cristiane M. Sato. Sparse sums of positive semidefinite matrices. *ACM Trans. Algorithms*, 12(1):9:1–9:17, 2016. 100, 106

[KS18] Rasmus Kyng and Zhao Song. A matrix chernoff bound for strongly rayleigh distributions and spectral sparsifiers from a few random spanning trees. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 373–384. IEEE Computer Society, 2018. 106

[LS17] Yin Tat Lee and He Sun. An sdp-based algorithm for linear-sized spectral sparsification. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 678–687. ACM, 2017. 106

[LS18] Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. *SIAM J. Comput.*, 47(6):2315–2336, 2018. 106

[MSS21] Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families iii: Sharper restricted invertibility estimates. *Israel Journal of Mathematics*, pages 1–28, 2021. 104, 111