Chapter 9

Spectral Sparsification

In this chapter, we introduce the spectral sparsification problem formulated by Spielman and Teng [ST11], which is a generalization of the graph sparsification problem formulated by Karger [Kar99]. Then we will see a random sampling algorithm to solve the problem by Spielman and Srivastava [SS11], matching the result for the graph sparsification problem by Benczur and Karger [BK15].

As we will see in later chapters, the study of spectral sparsification has led to major breakthroughs, and this is a striking example of using linear algebraic techniques to solve combinatorial problems.

9.1 Graph Sparsification

The graph sparsification problem is to find a sparse graph which approximates all cut values of a given graph.

Definition 9.1 (Cut Approximator [Kar99])**.** *Let $G = (V, E)$ be an undirected graph with a weight $w_G(e)$ on each edge $e \in E$, and $H = (V, F)$ be an undirected graph on the same vertex set with a weight $w_H(e)$ on each edge $e \in F$. For $0 \leq \epsilon \leq 1$, we say H is a $(1 \pm \epsilon)$-cut approximator of G if for all $S \subseteq V$,*

$$(1 - \epsilon) \cdot w_G\big(\delta_G(S)\big) \leq w_H\big(\delta_H(S)\big) \leq (1 + \epsilon) \cdot w_G\big(\delta_G(S)\big).$$

This problem was formulated by Karger [Kar99], and the goal is to find a sparse graph H that is a good cut approximator of the input graph G. Note that this definition does not require that H is a subgraph of G (that is, $F \subseteq E$), but all constructions that we will see satisfy this property which is useful in some applications.

Uniform Sampling

A first example to think about is when G is a complete graph. We know from Chapter 7 that a random sparse graph H is an expander graph, which is a good approximation to the complete graph. So it is a natural strategy to construct a sparsifier H by sampling a uniform random subgraph of G. Karger considered the following simple uniform random sampling algorithm, where the idea is that the expected weight of each edge e in H is the same as the weight of e in G.

Algorithm 5 Uniform Sampling Algorithm for Graph Sparsification

Require: An unweighted undirected graph $G = (V, E)$.

1: Set a sampling probability p. For each $e \in E$, with probability p, add e in F with weight $1/p$.
2: **return** $H = (V, F)$.

Karger proved that the uniform sampling algorithm would work to sparsify the input graph G when the minimum cut value of G is $\Omega(\log n)$.

Theorem 9.2 (Uniform Sampling for Graph Sparsification [Kar99]). *Let $G = (V, E)$ be an unweighted undirected graph with $V = [n]$ and minimum cut value c. Set the sampling probability $p = \frac{9 \ln n}{\epsilon^2 c}$. Then H produced by Algorithm 5 is a $(1 \pm \epsilon)$-cut approximator of G with $O(p \cdot |E|)$ edges with probability at least $1 - \frac{4}{n}$.*

The well-known Chernoff bound is used to analyze the success probability.

Theorem 9.3 (Chernoff Bound for Heterogeneous Coin Flips). *Let $X_1, X_2, \ldots, X_n$ be independent random variables with $X_i = 1$ with probability p_i and $X_i = 0$ otherwise. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} p_i$ be the expected value of X. Then, for any $0 < \delta < 1$,*

$$\Pr\left(|X - \mu| \geq \delta\mu\right) \leq 2e^{-\delta^2 \mu/3}.$$

The proof outline of Theorem 9.2 is as follows. With the assumption that the minimum cut value is $\Omega(\log n)$, Chernoff bound can be used to show that the probability that $w_H(\delta_H(S))$ is not a $(1 \pm \epsilon)$-approximation of $w_G(\delta_G(S))$ for a particular subset $S \subseteq V$ is at most $1/\operatorname{poly}(n)$. While this probability is quite small, this is not nearly small enough to apply a union bound on the exponential number of subsets directly. Karger's observation is that there are only a polynomial number of small cuts as stated below, and so a more careful union bound based on the cut value can be used to prove Theorem 9.2.

Proposition 9.4 (Number of Approximate Minimum Cuts). *Let $G = (V, E)$ be an unweighted undirected graph with $V = [n]$ and minimum cut value c. For any $\alpha \geq 1$, the number of subsets S with $|\delta(S)| \leq \alpha c$ is at most $n^{\lceil 2\alpha \rceil}$.*

An interesting way to prove Proposition 9.4 is to use Karger's random contraction algorithm for solving the minimum cut problem. See [Kar99] or L03/L04 of CS761 for proofs of Theorem 9.3 and Proposition 9.4.

Non-Uniform Sampling

Without the minimum cut assumption, then it is easy to see that the uniform sampling algorithm could fail. For example, consider the dumbbell graph where there is a bridge connecting two complete graphs.

In 1996, Benczur and Karger [BK15] designed a very clever non-uniform sampling algorithm, where the sampling probability p_e for each edge $e = uv$ is proportional to the "connectivity" of u and v. The idea is that edges with low connectivity are in H with higher probability p_e and smaller weight $1/p_e$ because they are crucial and so we basically just keep them (with the right expectation), while edges with high connectivity are in H with lower probability and larger weight to sparsify the graph. They defined a notion called "strong connectivity" for the non-uniform sampling algorithm and proved that every graph has a cut approximator with only $O(n \log n)$ edges.

Theorem 9.5 (Benczur-Karger Cut Sparsification [BK15]). *For any edge-weighted undirected graph $G = (V, E)$ and any $0 < \epsilon < 1$, there is a reweighted subgraph $H = (V, F)$ on the same vertex set with at most $O(\frac{n \log n}{\epsilon^2})$ edges such that H is a $(1 \pm \epsilon)$-cut approximator of G. Furthermore, H can be computed in nearly linear time $\tilde{O}(|E|)$.*

The definition of "strong connectivity" is a bit unnatural, and Benczur and Karger conjecured that it can be replaced by the more natural edge-connectivity between u and v. This conjecture is proved by Fung, Hariharan, Harvey and Panigrahi in 2011 [FHHP19].

Applications of Graph Sparsifications

An important feature of Theorem 9.2 and Theorem 9.5 is that they provide a near-linear time algorithm to find a cut sparsifer, and they become an important primitive in designing fast graph algorithms. For example, suppose we would like to solve the minimum s-t cut problem in a graph G. Standard algorithms have their time complexity depending on the number of edges in G, so when G is dense with $\Omega(n^2)$ edges the algorithms are quite slow. To design a fast approximation algorithm, we can first use Theorem 9.5 to obtain a $(1 \pm \epsilon)$-cut approximator H of G with only $O(n \log n/\epsilon^2)$ edges. Then, we just run the standard algorithms on H to find an optimal s-t cut in H, and it can be shown that this is a $(1 + 3\epsilon)$-approximate minimum s-t cut in G. More generally, with these sparsification algorithms, for many graph problems about cuts (e.g. graph conductance), one could trade a small loss in the optimality of the solutions for a time complexity that is faster by at least one order of n.

Truly remarkably, Karger [Kar00] used the uniform sampling algorithm in Theorem 9.2 to design a near-linear time algorithm to solve the minimum cut problem *optimally*. It is actually crucial that the sparsifier is unweighted for this application, so for example the stronger Theorem 9.5 cannot be used. It took more than 20 years for researchers to finally find a deterministic near-linear time algorithm for the minimum cut problem [KT19].

9.2 Spectral Sparsification

On their way of designing a near-linear time algorithm for solving Laplacian systems of linear equations, Spielman and Teng [ST11] defined the following stronger notion of spectral sparsification for Laplacian matrices.

Definition 9.6 (Spectral Approximator). *Let $G = (V, E)$ be a weighted undirected graph and $H = (V, F)$ be a weighted undirected graph on the same vertex set. For $0 \leq \epsilon \leq 1$, we say H is a $(1 \pm \epsilon)$-spectral approximator of G if for all $x : V \to \mathbb{R}$,*

$$(1 - \epsilon) \cdot x^T L_G x \leq x^T L_H x \leq (1 + \epsilon) \cdot x^T L_G x,$$

where L_G and L_H are the weighted Laplacian matrices of G and H respectively. Equivalently, H is a $(1 \pm \epsilon)$-spectral approximator of G if

$$(1 - \epsilon) L_G \preccurlyeq L_H \preccurlyeq (1 + \epsilon) L_G.$$

Exercise 9.7 (Spectrum of Spectral Approximator). *Let G and H be weighted undirected graphs with Laplacian spectrums $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and $\gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_n$ respectively. Prove that if H is a $(1 \pm \epsilon)$-spectral approximator of G, then $(1 - \epsilon)\lambda_i \leq \gamma_i \leq (1 + \epsilon)\lambda_i$ for every $1 \leq i \leq n$.*

Again, the goal is to find a sparse graph H that is a good spectral approximator of the input graph G. Their original motivation is to use L_H as a "preconditioner" for solving the equations $L_G \cdot z = b$. Their definition is inspired by the result of Benczur and Karger [BK15] and is indeed a more demanding one.

Lemma 9.8 (Spectral Approximator is Cut Approximator). *If H is a $(1 \pm \epsilon)$-spectral approximator of G, then H is a $(1 \pm \epsilon)$-cut approximator of G.*

Proof. Let S be a subset of vertices and χ_S be the characteristic vector of S. Then, by Lemma 3.17,

$$\chi_S^T L_G \chi_S = \sum_{ij \in E(G)} w(i,j)\big(\chi_S(i) - \chi_S(j)\big)^2 = w_G\big(\delta_G(S)\big),$$

and similarly $\chi_S^T L_H \chi_S = w_H\big(\delta_H(S)\big)$. Since H is a $(1 \pm \epsilon)$-spectral approximator of G, it follows that

$$(1-\epsilon)\cdot\chi_S^T L_G \chi_S \le \chi_S^T L_H \chi_S \le (1+\epsilon)\cdot\chi_S^T L_G \chi_S \implies (1-\epsilon)\cdot w_G\big(\delta_G(S)\big) \le w_H\big(\delta_H(S)\big) \le (1+\epsilon)\cdot w_G\big(\delta_G(S)\big).$$

As this holds for any subset $S \subseteq V$, H is a $(1 \pm \epsilon)$-cut approximator of G. □

Since spectral sparsification is a strictly stronger requirement than cut sparsification, one would expect that it is a strictly harder problem to solve. Initially, Spielman and Teng [ST11] proved that there is always a $(1 \pm \epsilon)$-spectral sparsifier with $O(n \operatorname{polylog}(n)/\epsilon^2)$ edges and gave a fast algorithm for constructing such sparsifiers. This is enough for their grand goal of designing a nearly-linear time algorithm for solving Laplacian equations, which has become the engine for a new generation of fast algorithms for graph problems.

Reduction

In 2008, Spielman and Srivastava [SS11] revisited the spectral sparsification problem and proved that there is always a $(1 \pm \epsilon)$-spectral approximator with $O(n \log n/\epsilon^2)$ edges, thus by Lemma 9.8 generalizing the result of Benczur and Karger in Theorem 9.5 for cut sparsification. They reduced it to the following simpler statement where the objective is to bound only the maximum eigenvalue and the minimum eigenvalue.

Theorem 9.9 (Sparse Spectral Approximator of Identity Matrix [SS11]). *For any m vectors $u_1, \ldots, u_m \in \mathbb{R}^n$ satisfying $\sum_{i=1}^m u_i u_i^T = I_n$, there always exist scalars $s_1, \ldots, s_m$ with at most $O(\frac{n \log n}{\epsilon^2})$ nonzeros such that*

$$(1-\epsilon) I_n \preccurlyeq \sum_{i=1}^m s_i u_i u_i^T \preccurlyeq (1+\epsilon) I_n.$$

For the reduction, we need the concept of pseudoinverse of a matrix in Definition 2.23.

Definition 9.10 (Pseudoinverse of Laplacian Matrix). *Let G be a connected graph. Let $0 = \lambda_1 < \lambda_2 \le \ldots \le \lambda_n$ be the eigenvalues of $L(G)$ and $u_1, \ldots, u_n$ be the corresponding eigenvectors. Then the pseudoinverse of $L(G)$ is $L_G^\dagger = \sum_{i=2}^n \frac{1}{\lambda_i} u_i u_i^T$. And the square root of $L_G^\dagger$ is $L_G^{\dagger/2} = \sum_{i=2}^n \frac{1}{\sqrt{\lambda_i}} u_i u_i^T$*

Lemma 9.11 (Reduction to Identity). *Suppose for any m vectors with $\sum_{i=1}^m v_i v_i^T = I_n$ there are always scalars with at most $O(n \log n/\epsilon^2)$ nonzeros such that $(1-\epsilon)I_n \preccurlyeq \sum_{i=1}^m s_i v_i v_i^T \preccurlyeq (1+\epsilon)I_n$. Then for any graph $G = (V, E)$ with n vertices and m edges, there is always a graph H with $O(n \log n/\epsilon^2)$ edges such that H is a $(1 \pm \epsilon)$-spectral approximator of G.*

Proof. We assume without loss of generality that G is a connected graph. Let $L_G = \sum_{e \in E} b_e b_e^T$ be the Laplacian matrix of G as written in Definition 3.15. Define $v_e = U^T L_G^{\dagger/2} b_e$ where $L_G^{\dagger/2}$ is from Definition 9.10 and U is the $n \times (n-1)$ matrix where the i-th column is the $(i+1)$-th eigenvector u_{i+1} of $L(G)$ for $1 \le i \le n-1$. Then

$$\sum_{e \in E} v_e v_e^T = \sum_{e \in E} U^T L_G^{\dagger/2} b_e b_e^T L_G^{\dagger/2} U = U^T L_G^{\dagger/2} L_G L_G^{\dagger/2} U = U^T \left(\sum_{i=2}^n u_i u_i^T \right) U = I_{n-1}.$$

By assumption, there are scalars $s_1, \ldots, s_m$ with at most $O(n \log n/\epsilon^2)$ nonzeros such that

$$(1-\epsilon) I_{n-1} \preccurlyeq \sum_{e \in E} s_e v_e v_e^T \preccurlyeq (1+\epsilon) I_{n-1}.$$

Multiplying $L_G^{1/2} U$ on the left and $U^T L_G^{1/2}$ on the right of the above inequalities, then $L_G^{1/2} U U^T L_G^{1/2} = L_G$ and thus

$$(1-\epsilon) L_G \preccurlyeq \sum_{e \in E} s_e b_e b_e^T \preccurlyeq (1+\epsilon) L_G.$$

Let H be the graph with weight s_e on edge e. Then $L_H = \sum_{e \in E} s_e b_e b_e^T$, and thus H is a $(1 \pm \epsilon)$-spectral approximator of G with $O(n \log n/\epsilon^2)$ edges. □

Random Sampling Algorithm

Isotropy Condition: We first get some intuition about the condition $\sum_{i=1}^m v_i v_i^T = I_n$. We have discussed this isotropy condition before in Exercise 5.14 when we studied the higher-order Cheeger's inequality. When $m = n$, then $v_1, \ldots, v_n$ must be an orthonormal basis. When $m > n$, we can think of $v_1, \ldots, v_m$ as an "overcomplete" basis, as we can write any $x \in \mathbb{R}^n$ as $x = I_n x = \left(\sum_{i=1}^m v_i v_i^T\right) x = \sum_{i=1}^m \langle x, v_i \rangle v_i$, which has applications in communication theory. Also, as stated in Exercise 5.14, for any unit vector $y \in \mathbb{R}^n$, it holds that $\sum_{i=1}^m \langle y, v_i \rangle^2 = 1$. Informally, the vectors are "evenly spread out" so that the projections of these vectors to any direction y are the same. Given $\sum_{i=1} v_i v_i^T = I_n$, we would like to find a small subset of vectors $S \subseteq \{1, \ldots, m\}$ and some scaling factors so that $\sum_{i \in S} s_i v_i v_i^T \approx I_n$, and thus $\sum_{i \in S} s_i \langle y, v_i \rangle^2 \approx 1$. So, the subset should still be "evenly spread out", with the contribution in each direction about the same.

Idea: As in the cut sparsification case, uniform sampling may not work. For example, if some vector v_j has $\|v_j\| = 1$, then we must include v_j in the solution, as otherwise that direction will not be covered in the solution and so it won't be a spectral sparsifier. The analogy in the cut sparsification result is that a cut edge must be included in any cut sparsifier. So, as in the cut sparsification case, we need to do non-uniform sampling if we do random sampling.

The idea is similar and very natural. For longer vectors, we should set the sampling probability p_e to be higher because they are crucial and so we basically just keep them. For shorter vectors, we can afford to set the sampling probability p_e to be lower and the weight $1/p_e$ to be larger in order

to reduce the number of vectors. Concretely, we sample each vector v_i with probability $\|v_i\|_2^2$, and if it is chosen, we set the scalar $s_i = \frac{1}{\|v_i\|_2^2}$, so that the expected contribution is

$$\mathbb{E}\left[s_i v_i v_i^T\right] = \frac{v_i v_i^T}{\|v_i\|^2} \cdot \Pr[v_i \text{ is chosen}] = \frac{v_i v_i^T}{\|v_i\|^2} \cdot \|v_i\|^2 = v_i v_i^T.$$

Algorithm: The actual algorithm is basically the same, but we need to repeat the experiment $\Theta(\log n)$ times and take the average, so that we can prove concentration.

Algorithm 6 Random Sampling 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: $\vec{s} \leftarrow \vec{0}$ and $\tau = \frac{6 \ln n}{\epsilon^2}$.
2: **for** $1 \le i \le m$ **do**
3: **for** $1 \le t \le \tau$ **do**
4: Update $s_i \leftarrow s_i + \frac{1}{\tau p_i}$ with probability $p_i = \|v_i\|_2^2$.
5: **end for**
6: **end for**
7: **return** $\sum_{i=1}^m s_i v_i v_i^T$.

There are two steps in the analysis. One is to show that there are $O(n \log n/\epsilon^2)$ non-zero scalars. Another is to show that the returned solution is a $(1 \pm \epsilon)$-spectral approximator to the identity matrix.

Expectation

We bound the number of non-zero scalars by computing its expected value and using Markov's inequality. The sampling probability is used to bound the expected value.

Lemma 9.12 (Number of Nonzeros)**.** *Let $\vec{s}$ be the output of Algorithm 6 and $S = \text{supp}(\vec{s})$ be the set of vectors with non-zero scalars. Then $|S| = O(n \log n/\epsilon^2)$ with probability at least 0.9.*

Proof. The expected value is

$$\mathbb{E}\left[|S|\right] = \sum_{i=1}^m \Pr[i \in S] = \sum_{i=1}^m \left(1 - (1-p_i)^\tau\right) \le \sum_{i=1}^m \left(1 - (1 - \tau p_i)\right) = \tau \sum_{i=1}^m p_i,$$

where $\tau = \frac{6 \ln n}{\epsilon^2}$ as defined in Algorithm 6. Note that

$$\sum_{i=1}^m p_i = \sum_{i=1}^m \|v_i\|_2^2 = \sum_{i=1}^m v_i^T v_i = \sum_{i=1}^m \text{Tr}(v_i^T v_i) = \sum_{i=1}^m \text{Tr}(v_i v_i^T) = \text{Tr}\left(\sum_{i=1}^m v_i v_i^T\right) = \text{Tr}(I_n) = n,$$

where we used $\text{Tr}(AB) = \text{Tr}(BA)$ in Fact 2.34. Therefore, $\mathbb{E}\left[|S|\right] \le \tau \sum_{i=1}^m p_i = \tau n = 6n \ln n/\epsilon^2$, and the result follows from Markov's inequality that $\Pr\left[|S| \ge 10\mathbb{E}\left[|S|\right]\right] \le 1/10$. □

Matrix Chernoff Bound

There is an elegant generalization of the Chernoff-Hoeffding bound to the matrix setting. The proof uses the Golden-Thompson inequality in Fact 2.36.

Theorem 9.13 (Matrix Chernoff Bound [Tro12]). *Let $X_1, \ldots, X_k$ be independent, $n \times n$ real symmetric matrices with $0 \preccurlyeq X \preccurlyeq rI$. Suppose $\mu_{\min} \cdot I_n \preccurlyeq \sum_{i=1}^{k} \mathbb{E}[X_i] \preccurlyeq \mu_{\max} \cdot I_n$. Then, for any $0 \leq \epsilon \leq 1$,*

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^{k} X_i\right) \geq (1+\epsilon)\mu_{\max}\right] \leq n e^{-\frac{\epsilon^2 \mu_{\max}}{3r}} \text{ and } \Pr\left[\lambda_{\min}\left(\sum_{i=1}^{k} X_i\right) \leq (1-\epsilon)\mu_{\min}\right] \leq n e^{-\frac{\epsilon^2 \mu_{\max}}{2r}}.$$

Note that it is almost an exact analog of the Chernoff-Hoeffding bound in the scalar setting, by using the maximum eigenvalue and the minimum eigenvalue to measure the "size" of a matrix. Informally, it says that if we consider the sum of independent random matrices, when each matrix is not too "big/influential", the sum is concentrated around the expectation in terms of the eigenvalues.

Concentration

The algorithm was designed in a way such that the proof that the solution is a $(1 \pm \epsilon)$-spectral sparsifier is a direct application of the matrix Chernoff bound. The reweighting by the sampling probability is set to ensure that no random variable is too influential.

Lemma 9.14 (Success Probability of Spectral Approximation). *The output of Algorithm 6 satisfies $(1-\epsilon)I_n \preccurlyeq \sum_{i=1}^{m} s_i v_i v_i^T \preccurlyeq (1+\epsilon)I_n$ with probability at least $1 - \frac{2}{n}$.*

Proof. The random variables are

$$X_{i,t} = \begin{cases} \frac{v_i v_i^T}{\tau p_i} & \text{with probability } p_i \\ 0 & \text{otherwise} \end{cases},$$

for vector i in iteration t. The output of the algorithm is $Y := \sum_{i=1}^{m} \sum_{t=1}^{\tau} X_{i,t}$. The expected output is

$$\mathbb{E}[Y] = \sum_{i=1}^{m} \sum_{t=1}^{\tau} \mathbb{E}[X_{i,t}] = \sum_{i=1}^{m} \sum_{t=1}^{\tau} p_i \cdot \frac{v_i v_i^T}{\tau p_i} = \sum_{i=1}^{m} \sum_{t=1}^{\tau} \frac{v_i v_i^T}{\tau} = \sum_{i=1}^{m} v_i v_i^T = I_n.$$

So, the expected output is exactly the identity matrix, with $\mu_{\max} = \mu_{\min} = 1$. To apply the matrix Chernoff bound in Theorem 9.13, it remains to find a bound r so that $X_{i,t} \preccurlyeq rI$. Note that

$$X_{i,t} = \frac{v_i v_i^T}{\tau p_i} = \frac{v_i v_i^T}{\tau \|v_i\|^2} = \frac{1}{\tau}\left(\frac{v_i}{\|v_i\|}\right)\left(\frac{v_i}{\|v_i\|}\right)^T,$$

which is a rank-one matrix of a unit vector, and so the maximum eigenvalue is just $1/\tau$ and thus we can set $r = 1/\tau$. By Theorem 9.13, as $\tau = \frac{6 \ln n}{\epsilon^2}$,

$$\Pr[\lambda_{\max}(Y) \geq 1+\epsilon] \leq n e^{-\frac{\epsilon^2 \tau}{3}} = n e^{-2 \ln n} = \frac{1}{n}.$$

The lower tail follows similarly. So, with probability at least $1 - \frac{2}{n}$, we have $\lambda_{\max}(Y) \leq 1+\epsilon$ and $\lambda_{\min}(Y) \geq 1-\epsilon$, and thus $(1-\epsilon)I_n \preccurlyeq Y \preccurlyeq (1+\epsilon)I$, proving that the solution is a $(1 \pm \epsilon)$-spectral approximator of the identity matrix. □

Combining Lemma 9.12 and Lemma 9.14 by a union bound, we know that a $(1\pm\epsilon)$-spectral approximator of the identity matrix with $O(n\log n/\epsilon^2)$ vectors exists, and indeed the random sampling algorithm will succeed with constant probability. This proves Theorem 9.9, and the reduction in Lemma 9.11 proves the theorem by Spielman and Srivastava that every graph has a $(1\pm\epsilon)$-spectral approximator with $O(n\log n/\epsilon^2)$ edges.

Discussions

By considering spectral sparsification, there is an elegant and arguably simpler proof of Theorem 9.5 for cut sparsification by Benczur and Karger. In the cut sparsification problem, it was not very clear that what is the right sampling probability, and the conjecture that edge-connectivity can be used for sampling was only answered much later [FHHP19]. In the more general spectral sparsification problem, however, there seems to be only one natural choice for the sampling probability and the analysis follows directly from the matrix Chernoff bound. This is a great example that a more general problem can be easier to solve than a special case, where in the special case there seem to be multiple reasonable approaches while the generalization points to the right approach.

Sampling Probability: For spectral sparsification of graphs, the sampling probability of an edge $e=uv$ is proportional to

$$\|v_e\|_2^2 = \|U^T L_G^{\dagger/2} b_e\|_2^2 = b_e^T L_G^{\dagger/2} U U^T L_G^{\dagger/2} b_e = b_e^T L_G^{\dagger} b_e = \mathrm{Reff}_G(u,v),$$

where $\mathrm{Reff}_G(u,v)$ is the effective resistance of the two endpoints u and v in the graph G, when we view the graph as a resistor network with each edge being a resistor of resistance one. An equivalent characterization of effective resistance is

$$\mathrm{Reff}_G(u,v) = \min_{f:E\to R_{\geq 0}} \left\{ \sum_{e\in E} f(e)^2 \;\middle|\; f \text{ is a unit flow from } u \text{ to } v \right\}.$$

This quantity can be thought of as an interpolation between the shortest path distance and the maximum flow value of a graph. Effective resistance is known to be closely related to some quantities in random walks such as the commute time and cover time. Recently, this concept has various applications in designing fast graph algorithms where spectral sparsification is an excellent example.

Fast Algorithm: Spielman and Srivastava also gave a nearly linear time algorithm to estimate the effective resistances of all edges. The main tools are a nearly linear time algorithm to solve a Laplacian system of equations (which is a breakthrough result by Spielman and Teng), and also the dimension reduction result by Johnson and Lindenstrauss in Theorem 8.8. As a consequence, there is a nearly linear time randomized algorithm for constructing a spectral sparsifier of a graph, which is important for designing fast algorithms for other graph problems.

Tight Example: The analysis of the random sampling algorithm is tight. In a complete graph, the effective resistance of every edge is the same. So, the random sampling algorithm on a complete graph is just the uniform sampling algorithm. A "coupon collector" argument can be used to prove that random sampling won't work to find a cut sparsifier with $o(n\log n)$ edges. It is a good exercise to work out the details.

Randomized Linear Algebra: Random sampling and dimension reduction are very useful in designing fast algorithms for numerical linear algebra problems. We illustrate these ideas in a basic problem, the least square problem. In the least square problem, we are given an $n \times d$ matrix A and $b \in \mathbb{R}^n$, and the objective is to find an $x \in \mathbb{R}^d$ that minimizes $\|Ax - b\|_2$. We are usually interested in the case when $n \gg d$, so the problem is over-constrained. Exact algorithms require $\Omega(n \operatorname{poly}(d))$ time, which is too slow when n is large.

We would like to find an approximation algorithm with $\|Ax' - b\| \leq (1+\epsilon) \min_x \|Ax - b\|_2$ in $\tilde{O}(nd + \operatorname{poly}(\frac{d}{\epsilon}))$ time, which is near linear when $n \gg d$. The idea is to use a nearly linear time algorithm to compress the matrix A into a $k \times d$ matrix $B = SA$ with $k = \operatorname{poly}(\frac{d}{\epsilon})$, and then solve the least square problem on $\min_x \|S(Ax-b)\|_2$ exactly as our approximate solution. The techniques in spectral sparsification can be used for the compression.

Given $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, we first reduce the problem to the case when the columns of A are orthonormal. This is reminiscent to the reduction to the identity matrix in Lemma 9.11, so that $A^T A = I_d$ or equivalently $\sum_{i=1}^n a_i a_i^T = I_d$ where a_i is the i-th row of A. Then, we construct a matrix B by sampling and rescaling each row proportional to its squared length, so that $B = \sum_{i=1}^n s_i a_i a_i^T \approx I_d$ with only $O(d \log d/\epsilon^2)$ nonzero scalars. Therefore, B has $O(d \log d/\epsilon^2)$ rows, where each row of B is $\sqrt{s_i} a_i$ so that $(1-\epsilon)A^T A \preccurlyeq B^T B \preccurlyeq (1+\epsilon)A^T A$. This is a good "subspace embedding" as $\|Ax\|_2^2 \approx \|Bx\|_2^2$ because $x^T A^T A x \approx x^T B^T B x$. All the technique details are similar to those in spectral sparsification, e.g. using matrix Chernoff bound. The sampling probability is called the leverage score of a row, a generalization of effective resistance.

9.3 References

[BK15] András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.*, 44(2):290–319, 2015. 87, 88, 89, 90

[FHHP19] Wai Shing Fung, Ramesh Hariharan, Nicholas J. A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. *SIAM J. Comput.*, 48(4):1196–1223, 2019. 89, 94

[Kar99] David R. Karger. Random sampling in cut, flow, and network design problems. *Math. Oper. Res.*, 24(2):383–413, 1999. 87, 88

[Kar00] David R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, 2000. 89

[KT19] Ken-ichi Kawarabayashi and Mikkel Thorup. Deterministic edge connectivity in near-linear time. *J. ACM*, 66(1):4:1–4:50, 2019. 89

[SS11] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011. 87, 90

[ST11] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011. 87, 89, 90

[Tro12] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012. 93