DISCLAIMER: These notes are not necessarily an accurate representation of what I said during the class. They are mostly what I intend to say, and have not been carefully edited.

A few logistical things:

- due to various travels and PhD visit weekend, Richard’s office hours for the four upcoming weeks are changed to 3pm - 5pm on Wednesday Mar 15, cancelled for the week of Mar 20 - 24, 11am - 1pm on Wednesday Mar 29, 11am - 1pm on Monday Apr 3, and 3pm - 5pm on Wednesday Apr 5.

- Problem Set 3 should be out towards the end of this week.

- Comments from last time: proof of concentration bounds is confusing.

Last time we defined similarities of matrices: we said $A \approx_\epsilon B$ if

$$\exp(-\epsilon)A \preceq B \preceq \exp(\epsilon)A,$$

where $P \preceq Q$ if for all vectors $x$,

$$x^T P x \leq x^T Q x.$$

We then talked about matrix concentration bounds, which are ways of generating similar graphs:

**Theorem 0.1 (Matrix Concentration Bound).** Consider a finite sequence $X_1 \ldots X_m$ of independent, random symmetric matrices with dimension $n$ with

$$0 \preceq X_i \preceq \frac{\epsilon^2}{O(\log n)} I$$

Let their sum be $X = \sum_i X_i$. If

$$\mathbb{E}_{X_1 \ldots X_m} \left[ X = \sum_i X_i \right] = I,$$

then with high probability

$$X \approx_\epsilon I.$$
Plan today is to leverage this to give graph sparsification algorithms. Specifically, for a weighted undirected graph $G$, the goal is to find $H$ with few edges s.t. 

$$L_G \approx \epsilon L_H.$$ 

Note that this implies that $w^{(G)}(S, \bar{S}) \approx w^{(H)}(S, \bar{S})$. This means we will need to sample the graph Laplacians, which we can view as a sum of Laplacians, one per edge. We will get $H$ from sampling, specifically independently keep an edge $e$ with probability $p_e$. The process of getting $H$ close to $G$ can be broken into two steps: maintaining expectation and variance.

1 Graph Sampling Processes

We first start to transform the problem into the matrix concentration space, aka. turn it into a set of random matrices. Note that $L^{(G)}$ can be broken into a sum of Laplacians, one per edge:

$$L^{(G)} = \sum_e L^{(e)}.$$ 

Then for each edge $e$, tossing a coin on it means we get a random variable that is:

$$Y^{(e)} = \begin{cases} 
0 & \text{w.p. } 1 - p_e \\
\ell^{(e)} & \text{w.p. } p_e 
\end{cases},$$

and we can solve for $\alpha$ using the fact that we want $\mathbb{E}_H[L_H] = L_G$, or if we apply linearity of expectation backwards,

$$\mathbb{E}[Y^{(e)}] = L^{(e)}.$$ 

This solves to

$$\alpha p_e \ell^{(e)} = L^{(e)},$$

or $\alpha = p_e$. That is, if we keep the edge $e$, we need to rescale the corresponding edge by a factor of $1/p_e$ in order to maintain expectation. This formally leads to the random variables:

$$Y^{(e)} \overset{\text{def}}{=} \begin{cases} 
0 & \text{w.p. } 1 - p_e \\
\frac{1}{p_e} \ell^{(e)} & \text{w.p. } p_e 
\end{cases},$$

which is almost ‘universal’ for graph sampling.

2 Transforming to Setting of Matrix Concentration

We still have the freedom in choosing the $p_e$ values. However, we also need concentration. By transforming the problem into the setting of matrix concentration, we can derive the ‘right’ value of $p_e$ from the $X^{(e)} \leq \frac{1}{O(\log n)}$ condition.
To do this, we first need to turn the condition of $L^{(H)} \approx L^{(G)}$ into $X \approx I$. This is a consequence of composition of spectral bounds: $A \preceq B$ implies

$$U^T A U \preceq U^T B U.$$  

The proof is by considering Rayleigh quotients involving $B - A$, and considering vectors $x = U y$.

For the case of graph Laplacians, we will compose by compose by $L^{+1/2}$, to get the matrices

$$X^{(e)} \overset{\text{def}}{=} L^{+1/2} Y^{(e)} L^{+1/2}.$$  

As linearity of expectation also holds under matrix multiplications, we have:

$$E \left[ \sum_e X^{(e)} \right] = \sum_e L^{+1/2} E \left[ Y^{(e)} \right] L^{+1/2} = \sum_e L^{+1/2} L^{(e)} L^{+1/2} = I_{\perp 1}$$

Where $I_{\perp 1}$ is on the image space orthogonal to the all 1s vector. The same kind of composition applied over individual sets of coin flips also gives that

$$\Pr \left[ \sum_e X^{(e)} \approx I_{\perp 1} \right] = \Pr \left[ \sum_e Y^{(e)} \approx L \right].$$

3 Variance and Effective Resistances

Then it’s a matter of figuring out the ‘radius’, or the maximum deviation that a single entry can cause. Specifically, we need to bound:

$$\lambda_{\text{max}} \left( \frac{1}{p_e} L^{+1/2} L^{(e)} L^{+1/2} \right),$$  

which is rank 1 because $L^{(e)}$ is a rank 1 matrix. This means

$$\lambda_{\text{max}} (L^{(e)}) = \text{tr} \left[ L^{+1/2} L^{(e)} L^{+1/2} \right],$$  

and it suffices to set

$$p_e \geq O(\log n) \text{tr} \left[ L^{+1/2} L^{(e)} L^{+1/2} \right].$$

Recall that the trace of a matrix is the sum of its diagonal, and equals to the sum of all its eigenvalues. As the trace is a linear function on the matrix, it is well behaved under summation and expectation. Furthermore, trace is preserved under cyclic rotations, specifically

$$\text{tr} [AB] = \text{tr} [BA].$$

The first consequence of this is that these values can be related back to effective resistances, by writing $L^{(e)}$ as:

$$L^{(e)} = w_e \chi_{u,v} \chi_{u,v}^T,$$
where $\chi_{u,v}$ is the indicator vector with 1 at $u$ and $-1$ at $v$. Then factoring out the $w_e$ gives:

$$\text{tr} \left[ L^{-1/2} L_e L^{-1/2} \right] = w_e \text{tr} \left[ L^{-1/2} \chi_{u,v} \chi_{u,v}^T L^{-1/2} \right] = w_e \text{tr} \left[ \chi_{u,v}^T L \chi_{u,v} \right].$$

Another consequence of this is that we can also bound the expected number of edges sampled:

$$O(\log n) \sum_e \text{tr} \left[ L^+ L_e \right] = O(\log n) \text{tr} \left[ L^+ \sum_e L_e \right] = O(\log n) \text{tr} \left[ L^+ L \right] = O(\log n) \text{tr} \left[ I \right] = O(n \log n).$$

This is known as Foster’s theorem, and its combinatorial instantiation is that the sum over all edges of weight times effective resistances is exactly $n - 1$. Richard is not aware of a simple combinatorial proof of this fact :-(.

The takeaway message can be summarized as: sampling any graph with probability more than weight times effective resistances produces a sparsifier.

4 Sparsifying Graph Polynomials

We now go on to sparsify graph that we don’t have access to: For simplicity we assume that the graph is a random walk, aka

$$L_G = I - A$$

where $A$ is undirected, non-negative, and has row/column sum 1. The goal is to directly produce a graph $H$ s.t.

$$L_H \approx I - A^2.$$

The first observation is that $A^2$ is still a random walk transition matrix, as its row sums satisfy:

$$\sum_j (A)_{ij}^2 = \sum_j \sum_k A_{ik}A_{kj}, = \sum_k A_{ik} \left(\sum_j A_{kj}\right) \leq \sum_k A_{ik} = 1.$$

However, this also means that $A^2$ is dense (e.g. consider $A$ being the degree $n - 1$ star), so we can’t even afford to generate it before producing $H$. Instead, we need to view it as a sum of “product demand” graphs, and sparsify each one individually. The weight between vertices $i$ and $j$ in $A^2$ is

$$A_{ij}^2 = \sum_k A_{ki}A_{kj}.$$
Regrouping this summation over $k$ allows us to view $A^2$ as a sum of ‘product demand graphs’. So the plan is to first sparsify each of these graphs separately to ones with size about $deg(k)$, the combinatorial degree of $k$, and then sparsify the sum again.

We now focus on a single vertex $k$. Assume that it is connected to vertices $1 \ldots deg(k)$. The goal will be the same as before: finding good upper bounds to $w(e)r(e)$ for all edges in the product demand graph. For a pair of vertices $i$ and $j$, consider voltages of $\frac{1}{A_{ik}}$ at $i$, $-\frac{1}{A_{jk}}$ at $j$ and 0 everywhere else. It can be verified that this routes $\sum_i A_{ik}$ units of flow from $i$ to $j$. The resistance produced by this voltage setting is:

$$\frac{1}{\sum_i A_{ik}} \left( \frac{1}{A_{ik}} + \frac{1}{A_{jk}} \right).$$

This means that the probability of sampling the edge $ij$ in the ‘clique graph’ generated by $k$ is at most

$$O(\log n\epsilon^{-2}) \frac{1}{\sum_i A_{ik}} (A_{ik} + A_{jk}),$$

which gives a total of:

$$\frac{O(\log n)}{\sum_i A_{ik}} \sum_i \sum_j A_{ik} = O\left(deg(k) \log n\epsilon^{-2}\right).$$

Therefore we can sparsify each of these cliques into one with $O(deg(k) \log n\epsilon^{-2})$ edges, producing an overall sparsifier of $O(m \log n\epsilon^{-2})$ edges.