Today we will talk about randomized ways of reducing the density of edges, while preserving the cut and spectral structures of graphs.

1 Uniform Sampling

The purpose of graph sparsification is to find a sparse graph $H$ to represent the dense graph $G$, while preserving all the information about $G$. For example, if the graph $G$ has $n$ vertices and $m$ edges, we want to find a sparse graph $H$ with $O(\log n)$ edges to represent it. This is especially useful when we work with some vertex reduction algorithms, because those algorithms tend to increase the number of edges, and then we can use graph sparsification to reduce the edges.

The approximation of graph $G$ means to find a similar graph $H$ in terms of cuts. That is,

$$\forall \text{ cut } S, E^G(S, \bar{S}) \approx \epsilon E^H(S, \bar{S})$$

where

$$a \approx \epsilon b \iff (1-\epsilon)b \leq a \leq (1+\epsilon)b \text{ for } a, b \text{ positive}$$

Recall from last time we want to reduce the number of edges in a graph with min-cut $\delta$ so that the min-cut is about $O(\log n)$, while preserving all cuts within a constant factor. Such a smaller graph, $H$, enables us to use a tree-packing of $H$ to solve things on $G$.

The sampling scheme we considered was: pick every edge with probability

$$\frac{c_S \log n}{\delta}$$

for some constant $c_S$ that represents the sampling overhead.

This schema works due to the following interpretation of Chernoff bounds:

**Lemma 1.1.** *(meta-statement of Concentration)* If $X_1 \ldots X_t$ are independent positive random variables with ‘magnitude’ at most 1 ($0 < X_i \leq 1$), then the probability that $\sum_{i} X_i$ is close to $\sum_{i} \mathbb{E}[X_i]$ is roughly

$$1 - \exp \left( -c_{\text{Sample}} \cdot \sum_i \mathbb{E}[X_i] \right).$$

for some absolute constant $c_{\text{Sample}}$. 
Now consider a partial cut $E(S, V \setminus S)$: by assumption of the minimum cut having size at least $\delta$, the expected number of edges we sample is at least
\[
\delta \cdot O \left( \frac{\log n}{\delta} \right) = \Theta (\log n).
\]
So by the meta concentration statement, we get that this particular cut as its size preserved with probability at least $1 - 1/poly(n)$.

However, we want to show this is true for any cut. Here we use the cut counting lemma: there are at most $n^{2\alpha}$ different cuts (specified by $S$) of size at most $\alpha \delta$. We can apply union bound on the probability of getting each of these wrong, giving a total failure probability of at most
\[
\sum_{\alpha=1}^{\infty} n^{2\alpha} \exp ( - \tilde{c} \alpha \log n )
\]
which sums to $1/poly(n)$ once we chose the constants in the $O(\cdot)$ so that $c$ is bigger than 2.

This process was used in the nearly-linear time min-cut algorithm by Karger [Kar00]. It can be taken to a much more general setting.

## 2 Importance Sampling: Keeping Expectation

We can generalize uniform sampling to importance sampling. Such a process was first used in constructing general cut sparsifiers by Benczur and Karger [BK96]. It’s done by associating a coin per edge with probability of keeping the edge as $p_e$.

Then we have:
\[
w_e^{(H)} = \begin{cases} 
0 & w.p. 1 - p_e \\
\alpha w_e^{(G)} & w.p. p_e,
\end{cases}
\]
and we can solve for $\alpha$ using the fact that we want $E_H \left[ w_e^{(H)} \right] = w_e^{(G)}$. This can be viewed as applying linearity of expectation backwards, and solves
\[
\alpha p_e w_e^{(G)} = w_e^{(G)},
\]
or $\alpha = p_e$, which is almost ‘universal’ for graph sampling.

A very helpful case for thinking through such a sampling scheme is two size $n$ cliques connected by a length $n$ path (as shown in Fig. 1). This case rules out uniform sampling because sampling any edge on the path might disconnect the graph, but the two cliques need significant size reductions. In this importance sampling language, it means everything on the path should have $p_e = 1$, while everything in the cliques should have $p_e = O(\log n/n)$, which brings the total number of edges down to $O(n \log n)$.

This intuition could actually be made rigorous using expander decompositions. We actually need vertex sparsity there, with the main statement being:
Figure 1: The case with two size $n$ cliques connected by a length $n$ path, in which importance sampling works better than uniform sampling.

**Lemma 2.1.** In an expander with expansion $\phi$, and all vertices have degrees between $[d, 2d]$, the minimum cut has size $\phi \cdot d$.

When combined with a cut sampling statement, this implies that each (degree regular) expander can be down sampled to about $O(n\phi^{-1} \log n)$ edges. With enough fiddlings to expander decompositions (specifically rip out a lot of low degree vertices), this can be turned into an ‘efficient’ algorithm. Such ideas are at the core of the first construction of spectral sparsifiers by Spielman and Teng [ST11]. That paper proved something stronger than preserving cuts: preserving Laplacian operators. As it turns out, this matrix based way of viewing things actually enables a much more direct way of obtaining sparsifiers that also give better guarantees.

## 3 Matrix Based View of Graph Approximations

It turns out that the notion of cut approximation can be generalized through Laplacian matrices. We first need to state a few notations:

**Indicator vector of a graph cut** For a graph cut $S$, the $n$-by-$1$ indicator vector $\chi(S)$ is defined as

$$\chi(s)_i = \begin{cases} 0 & \text{if } v_i \in S, \\ 1 & \text{if } v_i \notin S. \end{cases}$$

**Positive Semidefinite Ordering** A matrix $M$ is positive semidefinite iff $M$ is symmetric and $v^T M v \geq 0$ for all $v \in V$. When we write $L(G) \preceq L(H)$, it indicates

$$U^T (L(G) - L(H)) U \succeq 0$$
$$U^T L(G) U \succeq U^T L(H) U$$
Laplacian of an edge  The graph Laplacian of an edge \( e \) connecting nodes \( u, v \) with weight \( w_{uv} \) is simply

\[
L(e)_{u,v} = L(e)_{v,u} = -w_{uv} \\
L(e)_{u,u} = L(e)_{v,v} = w_{uv}
\]

and 0 elsewhere. It can be equivalently written as

The Laplacian matrix for a graph \( G \) is a \( n \)-by-\( n \) matrix with:

\[
L(G)_{u,v} = \begin{cases} 
  d_u = \sum_\hat{v} w_{u\hat{v}} & \text{if } u = v, \\
  -w_{uv} & \text{otherwise.}
\end{cases}
\]

or

\[
L(G) = \sum_e w_e L(e).
\]

The presentation below follows the Spielman-Srivastava paper, which drew this connection in a slightly different manner [SS11].

This matrix has the property that if \( x(S) \) is an indicator vector of \( S \), then

\[
x(S)^\top L x(S) = |\partial(S)|.
\]

For matrices, we can define the notion of \( L(G) \preceq L(H) \) as \( L(H) - L(G) \) being positive-semidefinite \( [3] \). This means for a weighted undirected graph \( G \), the goal is to find \( H \) with few edges s.t.

\[
(1 - \epsilon) L(H) \preceq L(G) \preceq (1 + \epsilon) L(H),
\]

which we can in-turn short hand it to

\[
L(G) \approx \epsilon L(H).
\]

Note that this implies that \( |\partial(G)(S)| \approx |\partial(H)(S)| \). This means we will need to sample the graph Laplacians, which we can view as a sum of Laplacians, one per edge.

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 \( [3] \):

\[
L(G) = \sum_e L(e).
\]

We will get \( H \) from sampling, specifically independently keep an edge \( e \) with probability \( p_e \). By the importance sampling scheme described above, we need to rescale the corresponding edge by a factor of \( 1/p_e \) in order to maintain expectation. This formally leads to the \( L(H) \) being the sum of random matrices \( \sum_e Y^{(e)} \):

\[
Y^{(e)} = \begin{cases} 
  0 & \text{w.p.} 1 - p_e \\
  \frac{w_G^{(e)}}{p_e} L^{(e)} & \text{w.p. } p_e,
\end{cases}
\]

The whole reason for doing this is that matrix concentration, as stated in Lemma 1.1 still holds for matrices [Tro12]. The formal matrix version, as stated in [Tro12] is:
Lemma 3.1. Let $X_1 \ldots X_n$ be random matrices such that $0 \preceq X_i \preceq \frac{1}{\mathcal{O}(\log n)} I$ and $\sum_i \mathbb{E} [X_i] = I$, then w.h.p.
$$\sum_i X_i \approx I.$$  

4 Picking Probabilities to Increase Concentration

We still have the freedom in choosing the $p_e$ values, and we will pick them to obtain concentration: observe that $p_e = 1$ definitely works, but leads to too many edges. By transforming the problem into the setting of matrix concentration, we can derive the ‘right’ value of $p_e$ from the $X(e) \preceq \frac{1}{\mathcal{O}(\log n)} I$ 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 $L + 1/2$, to get the matrices (Note: we use $L$ for $L(G)$ in the rest of the document)
$$X(e) \overset{\text{def}}{=} \frac{1}{\mathcal{O} \log n} L(e) L + 1/2.$$  

As linearity of expectation also holds under matrix multiplications, we have:
$$\mathbb{E} \left[ \sum_e X(e) \right] = \sum_e \frac{1}{\mathcal{O} \log n} \mathbb{E} [L(e)] L + 1/2 = \sum_e \frac{1}{\mathcal{O} \log n} L(e) L + 1/2 = I_{\perp 1}$$

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

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:
$$\lambda_{\max} \left( \frac{1}{p_e} L + 1/2 L(e) L + 1/2 \right).$$

This is a rank 1 matrix due to $L(e)$ is a rank 1 matrix, so
$$\lambda_{\max} \left( \frac{1}{p_e} L(e) L + 1/2 \right) = \text{tr} \left[ L + 1/2 L(e) L + 1/2 \right]$$
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 \( n \)-by-1 indicator vector with 1 at \( u \) and \( -1 \) at \( v \). Then factoring out the \( w_e \) gives:
\[ \text{tr} \left[ L^{+1/2} L^e L^{+1/2} \right] = w_e \text{tr} \left[ L^{+1/2} \chi_{u,v} \chi_{u,v}^T L^{+1/2} \right] = w_e \text{tr} \left[ \chi_{u,v}^T L^\dagger \chi_{u,v} \right]. \]

Another consequence of this is that we can also bound the expected number of edges sampled:
\[
O(\log n) \sum_e \text{tr} \left[ L^{+} L^e \right] = O(\log n) \text{tr} \left[ L^+ \sum_e w_e L^e \right] = O(n \log n).
\]

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 \( n - 1 \).

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

References


