1 Graph Sparsification

Given a graph $G = (V, E)$, a graph $G' = (V, E')$ is a graph sparsification of $G$ if it has the same vertices as $G$ and a reasonable small subset of the edges of $G$. Graph sparsification is used because a graph with less edges is easier to use. If you want to use a certain property of a graph, you off course want that the graph sparsification has the same property. Therefor the weight on the edges are changed in the process of sparsification. We will only focus on edge cuts, which we will denote by $(S, \bar{S})$, such that $S \cap \bar{S}$ is empty and $S \cup \bar{S} = V$. If we consider $G$, $E[S, \bar{S}]$ denotes the number of edges that are cut. $G'$ will have weights on the edges, so we denote the total weight of the edges that are cut by $(S, \bar{S})$ as $w(S, \bar{S})$. Now we can state the theorem we will prove during this lecture.

**Theorem 1** Given a graph $G = (V, E)$, there exists a graph sparsification $G'$ on $O\left(\frac{n \log^3 n}{\epsilon^2}\right)$ edges such that given any cut $(S, \bar{S})$ we have

$$(1 - \epsilon)w(S, \bar{S}) \leq E[S, \bar{S}] \leq (1 + \epsilon)w(S, \bar{S})$$

2 Toy Example

We will first proof the theorem for the complete graph $K_n$. We will try to find a graph $G'$ with $\frac{10 \log n}{n^2}$ edges. We will use the following algorithm to create $G'$. First set $p = \frac{10 \log n}{n^2}$, then pick each edge of $K_n$ independently with probability $p$ and give each picked edge weight $\frac{1}{p}$. Notice that the expected number of edges in $G'$ is $E(\#edges in G') = p\left(\begin{array}{c} n \\ 2 \end{array}\right) \approx \frac{10 \log n}{n^2}$. Also, the expected weight an edge adds to a cut is $p\frac{1}{p} = 1$. Notice that for a cut with $|S| = k$, $E(S, \bar{S}) = k(n - k)$ and thus $E(w(S, \bar{S})) = p\frac{1}{p}k(n - k) = k(n - k)$. So in expectation $G'$ will be as desired, but this doesn’t mean there exist such a graph.

Remember some of the results of Chernoff. Let $X_i$ be independent $(0 - 1)$ random variables, such that $Pr(X_i = 1) = p_i$ and thus $E(X_i) = p_i$. Let $X = \sum_i X_i$, then from Chernoff follows that $Pr(X > (1 + \epsilon)E(X)) \leq e^{-\frac{\epsilon^2}{2}E(X)}$ if $\epsilon \leq 2e - 1$. Also $Pr(X < (1 + \epsilon)E(X)) \leq e^{-\frac{\epsilon^2}{4}E(X)}$, so $Pr(X \notin (1 \pm \epsilon)E(X)) \leq 2e^{-\frac{\epsilon^2}{4}E(X)}$. A corollary is that if $X_i$ were $(0 - w)$ variables, then $Pr(X > (1 + \epsilon)E(X)) \leq e^{-\frac{\epsilon^2}{2w}E(X)}$.

Now the idea is to bound the chance that a cut $(S, \bar{S})$ is messed up by our algorithm. A cut $(S, \bar{S})$ is messed up if $w(S, \bar{S}) > (1 + \epsilon)E[S, \bar{S}]$ or $w(S, \bar{S}) < (1 - \epsilon)E[S, \bar{S}]$. Note that there are $2^n$ possible cuts, so it would be nice if we could bound this chance to something like $\frac{1}{2^n}$. Note that $G'$ can’t have less than $n - 1$ vertices, since $G'$ isn’t connected in that case. And if $G'$ isn’t connected it is messed up, since the cut between connected components is zero in that case. Consider a cut where $S$ is a single vertex $v$. We will calculate the chance that our algorithm will disconnect $v$ from the rest of the graph. $Pr(v \text{ disconnected}) = (1 - p)^{n - 1} = (1 - \frac{10 \log n}{n^2})^{n - 1} \approx e^{-\frac{10 \log n}{n^2}} = e^{-\frac{10 \log n}{n^2}} = \frac{1}{n^{10 \log n}}$. So the chance that a cut which cuts only one vertex will be messed up
is at least $\frac{1}{n^{10\log n}}$. If this were true for all cuts we would have a problem, since the probability that any cut would be messed up would be at least $2^n \frac{1}{n^{10\log n}}$, which is bigger than one for most values of $n$. Luckily this isn’t true for all cuts. The chance that a cut $(S, \bar{S})$ with $|S| = \frac{n}{2}$ is for example $(1 - p)\frac{e^2}{4} \approx e^{-10\log n} \frac{n^2}{e^2} \ll \frac{1}{n^p}$. Note that there are $n$ cuts with $|S| = 1$ and $\left(\frac{n}{2}\right)$ cuts with $|S| = \frac{n}{2}$. So there are way more ”big” cuts.

Now we will fix a cut $(S, \bar{S})$ with $|S| = k$ and without loss of generality $k \leq \frac{n}{2}$. Remember that $E[S, \bar{S}] = k(n - k) = E(w(S, \bar{S}))$. We will bound the probability that this cut is messed up using the corollary of Chernoff.

$$Pr(w(S, \bar{S}) - E(w(S, \bar{S})) \geq \epsilon k(n-k)) \leq e^{-\frac{\epsilon^2}{2}k(n-k)p} \leq e^{-\frac{\epsilon^2}{8}kn\log n} = e^{-1.25\log nk} = n^{-1.25k}.$$ 

So the chance that any cut with $|S| = k$ is messed up is $\leq \binom{n}{k}n^{-1.25k} \leq n^k n^{-1.25k} = n^{-0.25k}$. And the chance that any cut is messed up is

$$\sum_{k=1}^{\frac{n}{2}} n^{-0.25} = \frac{1 - n^{-0.125n}}{1 - n^{0.25}} \ll \frac{1}{n^{0.2}}.$$ 

Which goes to zero of $n$ grows, so there is a big chance that no cut is messed up and thus $G'$ is as desired.

3 General Graph

Now we will look at a general graph. Like in the case of a complete graph our first try will be to pick edges with probability $p \approx \frac{\log^2 n}{n}$. If there are some edges with degree one, this approach won’t work because the chance, that the one edge, which connects the vertex to the rest, is picked, is to small. To counter this problem we include connectivity $k_e$ of an edge $e = (u, v)$ in our strategy. $k_e$ is defined such that any cut separating $u$ and $v$ has value at least $k_e$. For example in a cycle $k_e = 2$ and in a complete graph $k_e = n - 1$. Notice that $k_e$ is equal to the value of the minimal $(u, v)$-cut and thus can be computed in polynomial time, with a mincut algorithm.

The algorithm we will use for general graphs is: Pick each edge $e$ with probability $p(e) = \min(1, \frac{10\log^2 n}{e^2k_e})$ and give $e$ weight $w(e) = \frac{1}{p(e)}$. We will use the following lemma to prove that this algorithm gives indeed a graph with $O\left(\frac{n\log^2 n}{e^2}\right)$ edges.

**Lemma 2** For any graph $G = (V, E)$ with $n$ vertices

$$\sum_{e \in E} \frac{1}{k_e} \leq n - 1.$$ 

**Proof** We will proof this by repeating one step, namely: Find a mincut in a connected component and cut that component there. Notice that this creates an extra connected component in every step. So all edges will be cut in maximally $n - 1$ steps, because there are only $n$ vertices, so there can’t be more than $n$ connected components. We will call the number of edges that are cut by the mincut in step $i$, $k^i$. So in step 1 there are $k^1$ edges removed. For those edges $k_e = k^1$, so the contribution of the $k^1$ edges we removed in step 1 to $\sum_{e \in E} \frac{1}{k_e}$ is $k^1 \frac{1}{k_e} = 1$. In step $i$ we remove $k^i$
edges, which is equal to the \(k_e\) these edges have in this connected component and that number is smaller than or equal to the \(k_e\) these edges had in the original graph. So the contribution of the \(k^i\) edges we removed in step \(i\) to \(\sum_{e \in E} \frac{1}{k_e}\) is \(k^i \frac{1}{k_e} \leq 1\). So in \(n - 1\) steps we cut all edges and in every step the contribution to the sum is 1 or less. We conclude \(\sum_{e \in E} \frac{1}{k_e} \leq n - 1\). \(\square\)

Now we can see that the expected value of the number of edges of the graph our algorithm gives is \(E(\#\text{edges}) = \sum_{e \in E} \frac{10\log^3 n}{e^2} \frac{1}{k_e} \leq \frac{10\log^3 n}{e^2} (n - 1)\). So \(E(\#\text{edges}) = O\left(\frac{n \log^3 n}{e^2}\right)\).

To understand the proof of the main theorem we need one more concept. As a toy example we will consider the complete graph \(K_n\). If we look at a cut \((S, \bar{S})\) with \(|S| = k \leq \frac{n}{2}\) the number edges that are cut is \(k(n-k) \geq k \frac{n^2}{4}\). Now we could ask ourselves how many cuts are there that cut \(\frac{nk}{2}\) edges. Any cut that cuts \(\frac{nk}{2}\) edges has \(|S| \leq k\), so there are \(\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{k} \approx nk\) cuts that cut \(\frac{nk}{2}\) edges. We define an \(\alpha\)-mincut as a cut which cuts maximally \(\alpha\) times the size of the mincut. So in our toy example there are about \(nk^{\frac{\alpha}{2}}\) mincuts, since the mincut of \(k_n\) has size \(n - 1\). We now proof the following theorem using Karger’s mincut algorithm.

**Theorem 3** Given a connected graph \(G = (V, E)\). For any integer \(\alpha\), the number of \(\alpha\)-mincuts is \(\leq n^{2\alpha}\).

**Proof:** It is sufficient to prove that a particular \(\alpha\)-mincut \(C\) survives Karger’s mincut algorithm with probability \(\geq \frac{1}{n^{2\alpha}}\). If the mincut has size \(k\), \(C\) has size \(\leq \alpha k\). Since the mincut has size \(k\) there are in total at least \(\frac{nk}{2}\) edges. So the chance that one of the edges of \(C\) is subtracted in the first subtraction is \(\leq \frac{\alpha k}{2} = \frac{2\alpha}{n}\) and in the \((i+1)\)th contraction this is \(\leq \frac{\alpha k}{(n-1)k} = \frac{2\alpha}{n-1}\). Hence the chance that \(C\) will survive until there are only \(2\alpha + 1\) vertices left is at least \((1 - \frac{2\alpha}{n}) \cdot (1 - \frac{2\alpha}{n-1}) \cdots (1 - \frac{2\alpha}{2\alpha+1}) = \frac{(n-2\alpha)}{n} \cdot \frac{(n-2\alpha)}{n-1} \cdots \frac{(n-2\alpha)}{2\alpha+1} = \frac{2^{2\alpha}}{n(n-1) \cdots (n-2\alpha+1)} \geq \frac{2^{2\alpha}}{n^{2\alpha} 2\alpha} \geq \frac{1}{n^{2\alpha}}\). \(\square\)

Notice that like in our toy example we defined the chances and weights such that \(E(w(S, \bar{S})) = \sum_{e \in (S, \bar{S})} p(e)w(e) = E[S, \bar{S}]\). Now we would like to use Chernoff to show that with high probability \(E(S, \bar{S})\) is indeed close to \(E[S, \bar{S}]\). But this doesn’t work because of how \(w(e)\) is defined. \(w(e) = \frac{1}{p(e)}\) can be anything from one to something very big. The Chernoff bound cannot directly give useful tail bounds in this case because it is designed to show that the probability of being \(\beta\) times larger than the expectation is exponentially decreasing in \(\beta\), and that is not true if there are edges with a big weight and edges with a small weight in one cut.

To solve this problem we put all edges in \(\log n\) buckets, depending on the weight of the edges. In the first bucket we put edges of weight 1 to 2, in the second bucket we put edges of weight 2 to 4, in the third bucket we put edges of weight 4 to 8 and so on. Now we use Chernoff per bucket, which works well and gives the desired result in the end. We skipped quite some details here, but the idea of the proof should be clear. The details can be found in the notes of Nick Harvey.