1 Overview

1. Graph sparsification continued
2. Network Communication: Gossip protocols

2 Graph sparsification (contd.)

Consider any simple undirected graph $G$ with min-cut $k$. In the last class we showed that if we sparsify the graph by independently picking each edge with probability $p_e = \theta\left(\frac{\log n}{k}\right)$ (edge sampling), any min-cut of $G$ is preserved after rescaling with high probability ($\geq 1 - \frac{1}{n^c}$ for any constant $c$). Although this approach, along with the union bound, can be used to argue that any polynomial number of cuts are preserved with high probability (whp), we cannot use this to argue that all the cuts are preserved whp since there are $\Omega(2^n)$ cuts.

We now prove that for graphs with $k = \omega\left(\log n\right)$ the edge sampling algorithm actually preserves all the cuts whp. The intuition for the better analysis is that for any $\alpha > 1$ even though the number of cuts of size $\alpha k$ is large (in the last lecture we showed them to be at most $2^{2\alpha}\binom{n}{2^\alpha}$), Chernoff bound gives that they are far less likely to deviate away from their expected behaviour.

**Theorem 1** For any undirected simple graph $G$ with min-cut $k = \omega\left(\log n\right)$, edge sampling with probability $p_e = \theta\left(\frac{\log n}{k}\right)$ followed by rescaling with a factor $\frac{1}{p_e}$ will preserve all the cuts of $G$ with high probability.

**Proof** Suppose we sparsify the graph $G$ by independently picking each edge with probability $p_e$. Consider any cut $C$ of $G$ of size $\alpha k$. Since in expectation, the sparsification procedure selects only $p_e$ fraction of the edges of $C$, we rescale our solution with a factor $\frac{1}{p_e}$. This procedure preserves the cut at least in expectation. Let us now consider the probability that the cut deviates far away from the mean.

$$\Pr\left[\# \text{ edges sampled from } C \geq (1 + \epsilon) p_e \alpha k\right] \leq \exp\left(-\frac{\epsilon^2 p_e \alpha k}{3}\right) = e^{-c\alpha \log n}$$

for $p_e = \frac{3c \log n}{\epsilon^2}$. Since there are at most $2^{2\alpha}\binom{n}{2^\alpha} < 2^{2\alpha}n^{2\alpha}$ cuts of size $\alpha k$, union bound gives that the probability of one of them having a size greater than $(1 + \epsilon)\alpha k$ is at most $2^{2\alpha}n^{2\alpha}e^{-c\alpha \log n} = 2^{2\alpha}n^{2\alpha}e^{-c\alpha \log n \log e} \leq \frac{1}{n^{\epsilon^2}}$.

Moreover, since $\alpha$ can take at most $n^2$ values (any cut is of size between $k$ and $n^2$), we again take the union bound and claim that whp all cuts will be preserved.

**Remarks**

1. The above proof only works when the min-cut size is $\omega\left(\log n\right)$. For graphs with smaller min-cut, there are generalizations of the above theorem known. The idea is to sample different edges with different probabilities. More precisely, edge $e$ is sampled with probability

$$p_e = \frac{1}{\text{strength}(e)}$$

1 We say $f(x) = \Omega(g(x))$ if $\lim_{x \to \infty} \frac{g(x)}{f(x)} \leq c$ and $f(x) = \omega(g(x))$ if $\lim_{x \to \infty} \frac{g(x)}{f(x)} = 0$
where $\text{strength}(e)$ is the largest min-cut of a subgraph of $G$ containing edge $e$. By carefully using the concentration inequalities, one can argue that any graph can be sparsified to a graph with $O(n \log n)$ edges while preserving all the cuts.

2. Benczur-Karger [1] first showed how the above argument of sparsifying general graphs can be done in polynomial time. In the last decade, new methods of sparsifying graphs have been found. They use spectral techniques and are much more time efficient [2].

3 Network Communication

Consider the scenario where we have a network $G$ and the goal is to quickly disseminate some message $m$ from a root node $r$ to every other node in the network. For simplicity, we will assume that $G$ is the complete graph in today’s lecture. Note that we also assume that every node in the network already knows that it needs to learn $m$, but it does know which node is $r$. Let us look at some intuitive solutions.

The first possible solution is flooding. Here $r$ starts by sending $m$ to each of its neighbours and then every node that receives $m$ forwards it to all its other neighbours. Although this approach will convey $m$ to all the nodes in the network, there are two clear problems with this approach. First, every node needs to interact with all its neighbours in a single time slot. Second, every node receives its degree copies of the message, leading to a lot of redundant communication in the network.

A second possible solution is to structurally send the message $m$. For example, one can design a spanning tree of depth $O(\log n)$ rooted at $r$ in the network and then transfer $m$ only along the edges of the tree. This takes care of the above two problems as every node only interacts with a small number of nodes and the number of messages transferred in the network is $n – 1$. However, this solution is centralized because the tree needs to be constructed by looking at the entire network. Moreover, the solution is not robust as if any of the links of the tree fails then the nodes in that subtree do not receive $m$. This motivates us to find a solution that satisfies the following properties:

1. Simple
2. Each node receives $m$ in $O(\log n)$ time slots
3. Every node sends a ‘small’ number of packets in each time slot
4. Total number of packets transmitted in the network is $O(n)$
5. Distributed
6. Robust

Surprisingly, all of the above properties can be simultaneously achieved by using the gossip protocol.

Gossip protocol

The intuition behind gossip protocol is that even if every node communicates with exactly one other node in a time slot, the message $m$ can spread in the entire network in $O(\log n)$ time slots. The protocol has two operations:

- **Push**: Any node that knows message $m$ randomly contacts one of its neighbours and sends them $m$.
- **Pull**: Any node that does not know message $m$ randomly contacts one of its neighbours and asks if they know $m$. If that neighbour knows $m$ then this node also learns $m$.

The gossip protocol has every node performing the push operation for $\theta(\log n)$ time slots and then every node performing the pull operation for $O(\log \log n)$ time slots. The idea is that in the beginning the push operation ensures that whp in each time slot the number of nodes that know message $m$ ‘doubles’. Once more than $\frac{n}{2}$ nodes have received $m$, which takes $\theta(\log n)$ time, pull operation in the second half of the protocol ensures that the remaining nodes also receive $m$ quickly. Next we formally prove these claims.
Lemma 2  After $O(\log n)$ rounds of push operation, with high probability more than $\frac{n}{4}$ nodes in the graph have received message $m$. Moreover, the total number of messages communicated in the network are $\Theta(n)$.

Proof  Let $X_i$ denote the set of nodes that know message $m$ at time slot $i$. In the beginning when $|X_i| < \log n$, we see that the whp the number of nodes that know $m$ doubles in each time slot. This is because

$$\Pr[|X_{i+1}| < 2|X_i|] \leq \Pr[\exists u \in X_i : u pushes to node that knows m] + \Pr[\exists u, v \in X_i : both u and v push to the same node] \leq \log n \frac{\log n}{n} + \left(\frac{\log n}{2}\right) \frac{1}{n} \leq O\left(\frac{\log^2 n}{n}\right)$$

Hence whp in $O(\log \log n)$ time slots the number of nodes that know $m$ will be at least $\log n$.

Now consider any time $i$ when $\log n \leq |X_i| < \frac{n}{4}$. Note that even after time slot $i + 1$ there are at least $\frac{n}{2}$ nodes that do not know $m$. Hence for every node $u$ in $X_i$, with at least half probability it pushes to a node $v$ that does not know $m$ and no other node in $X_i$ tries to push to $v$ in the same time slot. Thus $E[|X_{i+1}|] \geq \frac{3}{2}|X_i|$. Now using the Chernoff bound,

$$\Pr[|X_{i+1}| < \left(\frac{3}{2} - \epsilon\right)|X_i|] \leq \exp\left(-\frac{\epsilon^2}{3} \frac{|X_i|}{2}\right) = \exp\left(-\frac{\epsilon^2}{2}|X_i|\right)$$

Since $|X_i| \geq \log n$ whp Chernoff bound gives that in each time slot cardinality of set $X_i$ increases by at least a factor of $\frac{3}{2} - \epsilon$. Hence the total number of time slots taken are $O(\log n)$.

Moreover, since in each iteration less than half fraction of the messages are wasted, i.e. they reach a node that already knows $m$, the total number of messages sent are $\Theta(n)$.

After being a little careful in the above analysis one can even argue that after $O(\log n)$ time slots at least $\frac{n}{2}$ nodes will learn $m$. Assuming this, we now claim that the pull protocol will quickly spread $m$ to the remaining nodes.

Lemma 3  If more than $\frac{n}{2}$ nodes know message $m$, pull protocol ensures that with high probability every node in the graph knows $m$ in another $O(\log \log n)$ time slots. Moreover, the total number of messages communicated in the network are $\Theta(n)$.

Proof  Here we outline the proof. The idea is that in the first time slot, any node that does not know $m$ will pull it with probability at least half. Hence whp at the end of the first time slot $\approx \frac{3n}{4}$ nodes will know $m$. Now for the second time slot, each node that does not know $m$ pulls it with probability $\approx \frac{3}{4}$. Hence whp after two time slots $\approx \frac{15n}{16}$ nodes will know $m$. This intuition can be formalized into a proof and we leave the details to the interested reader.

Remark: Note that only the push protocol or only the pull protocol will also ensure that whp every node in the graph receives message $m$ in $\Theta(\log n)$ time slots. However, then the total number of messages sent in the network will be $\omega(n)$.

References
