CMPSCI 690RA: Randomized Algorithms
Lecture 16-18: Sublinear Time Graph Algorithms

Andrew McGregor
Number of Connected Components

- **Goal:** Estimate number of connected components $C$ up to $\pm \epsilon n$ error in time $\text{poly}(1/\epsilon)$.

- **Approach:** For each node $v$, let $n_v$ be the size of the connected component involving $v$. Then,

$$C = \sum_v \frac{1}{n_v}$$

- Estimate $C$ by randomly sampling nodes and estimating the corresponding values of $1/n_v$. 

Estimating $1/n_v$ for a given $v$

- For each node $v$, let $\hat{n}_v = \min\{n_v, 2/\epsilon\}$ and note that
  \[ |1/\hat{n}_v - 1/n_v| \leq \epsilon/2 \]
  because $n_v \geq 2/\epsilon$ implies $|1/\hat{n}_v - 1/n_v| \leq 1/\hat{n}_v = \epsilon/2$.
- Let $\hat{C} = \sum_v 1/\hat{n}_v$ and note that
  \[ |\hat{C} - C| \leq \epsilon n/2 \]
- **Good news:** For a given $v$, can compute $\hat{n}_v$ in $O(1/\epsilon^2)$ time by doing a BFS from $v$ and terminating early if $2/\epsilon$ nodes discovered.
Sampling Nodes: Estimating $\hat{C}$

- Sample $r = 12 \log \left( \frac{2}{\delta} \right) / \epsilon^2$ nodes $v_1, \ldots, v_r$ with replacement.
- Compute $X_i = 1/\hat{n}_{v_i}$ for $i = 1, \ldots, r$ and return

$$T = \frac{n}{r} (X_1 + X_2 + \ldots X_r)$$

- Note $\mathbb{E}[X_i] = \hat{C}/n$ and so $\mathbb{E}[T] = \hat{C}$.
- By the Chernoff bound,

$$\Pr \left[ |X_1 + X_2 + \ldots X_r - r \hat{C}/n| \geq \epsilon r/2 \right] \leq 2 \exp(-\epsilon^2/12 \cdot r) \leq \delta$$

- And so $T = \hat{C} \pm \epsilon n/2 = C \pm \epsilon n$ with probability at least $1 - \delta$. 
Estimating Weight of Minimum Spanning Tree

- **Goal:** Given connected graph $G$ with edge weights in $\{1, 2, \ldots, w\}$, estimate MST weight up to $1 \pm \epsilon$ factor error in time $\text{poly}(w/\epsilon)$.

- **Observation:** If $G_i$ is subgraph consisting of edges of weight $\leq i$.
  - MST includes $n - C(G_1)$ edges of weight 1.
  - MST includes $n - C(G_2)$ edges of weight $\leq 2$ and so $C(G_1) - C(G_2)$ edges of weight 2
  - MST includes $n - C(G_3)$ edges of weight $\leq 3$ and so $C(G_2) - C(G_3)$ edges of weight 3
  - ... MST includes $n - C(G_w)$ edges of weight $\leq w$ and so $C(G_{w-1}) - C(G_w)$ edges of weight $w$

- Hence, weight of MST is:

$$\sum_{i=1}^{w} i(C(G_{i-1}) - C(G_i)) = n - w + \sum_{i=1}^{w} C(G_i)$$

- Estimating each $C(G_i)$ up to $\pm \epsilon n/w$ with failure probability $\delta/w$ gives $\pm \epsilon n$ estimate of MST weight with failure probability $\delta$.

- Since MST weight is at least $n$, we have a $1 + \epsilon$ factor approx.
Vertex Cover

Goal: Return estimate $\alpha$ for size of minimum vertex cover $\text{OPT}$ where

$$\alpha \leq O(\log d)\text{OPT} + \epsilon n$$

on the assumption that the maximum degree at most $d = O(1)$

Consider the following vertex cover algorithm on graph $G = (V, E)$:

1. $i \leftarrow 0$
2. while edges remain in $E$ do
   2.1 Let $U$ be the set of nodes in $V$ with degree $\geq d/2^i$
   2.2 $A \leftarrow A \cup U$
   2.3 Remove $U$ from $V$ and incident edges from $E$
   2.4 Update degrees of remaining nodes in graph
   2.5 $i \leftarrow i + 1$
3. Return $A$
Analysis

- Output is a vertex cover since the removal of $A$ removed all edges.
- Let $\theta$ be any vertex cover: in each round, at most $2|\theta|$ nodes not in $\theta$ are added to $A$.
  - Let $X$ be set of nodes removed during round $i$ that aren’t in $\theta$. Note all neighbors of $X$ must be in $\theta$.
  - Let $E_X$ be edges incident to $X$ and note
    \[ |E_X| \geq |X|d/2^i \]
  - Max degree at start of $i$th round is $\leq d/2^{i-1}$ and therefore,
    \[ |E_X| \leq |\theta|d/2^{i-1} \]
  - Hence $|X| \leq 2|\theta|$ as claimed.
- There are $\log_2 d$ rounds and hence the total number of nodes in $A$ is:
  \[ |\theta| + 2(\log_2 d)|\theta| = (1 + 2\log_2 d)|\theta| \]
Connection to Sublinear Time Algorithm

- Note the algorithm can be run in the following distributed setting:
  - Each processor corresponds to a node and initially only knows the incident edges
  - In each round every processor can send (potentially different) messages to its neighbors.
- Each message from a processor depends on a) its incident edges, b) previous messages received, and c) potentially random bits.
- Hence, after the $k$th round the “state” of the processor at a node only depends on the edges incident to nodes at most $k$ hops away.
- In context of our algorithm, whether a node $v$ is in the vertex cover can be determined if we query the $\leq d\log_2 d$ nodes at most $\log_2 d$ away from $v$.
- We can estimate fraction of nodes in the vertex cover by sampling; introduces additional $\epsilon n$ error.
Matching

- **Goal:** Estimate size of a maximal matching (MM) in degree bounded graph. Note,
  \[ MM \leq \min \text{ vertex cover} \leq 2MM \]

- **Greedy Algorithm:** Pick edges in random order and add edge \((u, v)\) if no adjacent edges already added. Gives a maximal matching \(M\).

- **Sublinear Time Algorithm**
  1. \(S \leftarrow 8/\epsilon^2\) randomly chosen nodes, \(C \leftarrow 0\)
  2. For each \(v \in S\):
     2.1 If there exists \((v, w) \in M\) then \(C \leftarrow C + 1\)
  3. Output: \(\frac{n}{2|S|} \cdot C\)

- **Analysis:** For each \(v \in S\), let \(X_v = 1\) if there exists \((v, w) \in M\).
  
  \[ \mathbb{E}[C] = \mathbb{E} \left[ \sum_{v \in S} X_v \right] = \frac{2|M||S|}{n} \]

  and by an application of the Chernoff bound \(C = |M| \pm \epsilon n\).
How to we determine where $e = (u, v)$ is in $M$?

- Define random order in greedy algorithm by giving a random value $r_e \in [0, 1]$ to edge $e$ and process edges in order of increasing values.

- To check if $e \in M$,
  1. for all $e'$ adjacent to $e$
     1.1 if $r_{e'} < r_e$, recursively check $e'$. If $e' \in M$ return "$e \notin M$" and halt.
  2. return $e \in M$

- **Claim**: Expected number of queries to graph is $2^{O(d)}$ where $d$ is the maximum number of edges sharing an endpoint with an edge.
Proof of Claim

- Suffices to consider all edges $e'$ that end a monotonically decreasing path from $e$.
- Let $E_k$ be the set of edges that can be reached by a length $k$ path.
- Let
  \[
  X_k = |\{ e' \in E_k : \exists \text{ monotonically decreasing length } k \text{ path from } e \text{ to } e' \}| 
  \leq \sum_{e' \in E_k} |\{ P : P \text{ is monotonically decreasing length } k \text{ path from } e \text{ to } e' \}| 
  = |\{ P : P \text{ is monotonically decreasing length } k \text{ path from } e \}| 
  \]
- Then
  \[
  \mathbb{E} [X_k] \leq d^k \cdot \mathbb{P} [\text{length } k \text{ path monotonically decreasing}] \leq \frac{d^k}{(k+1)!} 
  \]
- Therefore total number of edges needing explored
  \[
  \mathbb{E} \left[ \sum_{k} X_k \right] = \sum_{k=1}^{\infty} \frac{d^k}{(k+1)!} = \frac{e^d}{d} = 2^{O(d)} 
  \]