Approximating the average degree

Last Lecture

Last lecture, we introduced sublinear time algorithms to approximate the diameter of a graph, test monotonicity of a sequence numbers, and last, we approximated minimum spanning tree by approximating the number of connected components. This lecture will continue to discuss sublinear algorithms about graphs, particularly approximating the average degree. Worth noticing, the approximation rate is directly related to \( n \), the number of nodes in the graph.

Average Degree Problem Definition

The degree of node, \( v \), in graph \( G(V, E) \) is defined as

\[
d(v) = |\{ u \in V : (v, u) \in E \}|
\]

The average degree \( \bar{d} \) is

\[
\bar{d} = \frac{\sum_{v \in V} d(v)}{|V|}
\]

We want to estimate \( \bar{d} \) given the adjacency list representation of a “simple” graph \( G \) in sublinear time. By simple we mean: 1. \( G \) does not have parallel edges, i.e. between two nodes, there is at most one edge; 2. there are no self loops. An example graph of an adjacency list is as follows.

In this representation, we can query the node’s degree in constant time, and randomly access a node’s neighbors in constant time. Hence, instead of a pointer list, the neighbors of a node \( v \) are stored in an array of length \( d(v) \). This setup does affect the running time and performance of our algorithm.

Naive sampling

Algorithm

- Randomly pick a node, \( v_i \in V \), and query its degree \( d(v_i) \).
- Output the average degree of the samples
Short Analysis

This analysis of Naive sampling does not immediately work, since in a graph the range of the degree can vary substantially. There can be an isolated node with degree 0, and another node connecting to all the rest of nodes with degree $n-1$. The variance of the samples is too high, using the Chernoff bound, we need to sample $\Omega(n)$ to get a reasonable estimation. That is worse than scanning through the list, and calculating $\bar{d}$ directly.

Special Properties of Degrees and Sampling Lower Bound

Examples

Even though degrees can vary from 0 to $n$, they are not a sequence of arbitrary numbers. For example

$$(n-1, 0, ..., 0)$$

is not possible for a degree sequence, because other nodes need to receive the incoming edge.

However

$$(n-1, 1, ..., 1)$$

is valid. It represents a star graph.

$$(D, D, D, D, ..., D)$$

represents a regular graph.

$$(k, ..., k, 0, ..., 0)$$

with $m$ nodes with degree $k$ is valid as well, which could represent a collections of $k$ cliques (for appropriate values of $k$), and some isolated nodes, or more generally, any $k$ regular graph on a subset of the nodes. The average degree of this graph is

$$\bar{d} = \frac{(k^2)m}{n}.$$

And we can have $cn^2$ nodes in $c$ cliques, and $n - cn^2$ nodes in cycle. The average degree of the graph is

$$\bar{d} = \frac{2(n - cn^2) + (cn^2)^2}{n} \approx 2 + c^2.$$

Lower Bound

Several observations we can have:

- If there is only one edge. Then unless we sample that edge, our approximation will always be 0. Hence, it is impossible to achieve a multiplicative approximation in sublinear time.

- In the last example, to get an additive approximation of $c^2n$, we need make at least $\Omega(n^{\frac{1}{2}})$ queries, which is the lower bound of the sample size.
Algorithm 1: only querying $d(v)$

Idea

The reason that we cannot achieve a good estimation from naive sampling is that the variance of degrees is too high. As a result, we cannot use Chernoff bound. The idea of this algorithm is to group the nodes with similar degrees in one bucket, and get an $(1 + \beta)$-approximation for each bucket. If we want to achieve an $(1 + \epsilon)$-approximation overall, let $c$ be some carefully chosen constant

$$\beta = \epsilon / c.$$  

For bucket $B_i$, the degree range of nodes is

$$B_i = \{v \in V : (1 + \beta)^{i-1} \leq d(v) \leq (1 + \beta)^i\}.$$  

So the number of buckets

$$t = O\left(\frac{1}{\epsilon} \log n\right)$$

which is logarithmic.

Total degree of the graph can be bounded

$$\sum_i (1 + \beta)^{i-1}|B_i| \leq \bar{d}n \leq \sum_i (1 + \beta)^i|B_i|$$

by estimating the size of each bucket $|B_i|$ separately. The variance is small within each bucket.

Algorithm

- $S \leftarrow$ sample nodes
- Group sampled nodes with similar degrees into same buckets.
  $$B_i = \{v \in V : (1 + \beta)^{i-1} \leq d(v) \leq (1 + \beta)^i\}.$$  
  $$S_i \leftarrow S \cap B_i.$$  
- Estimate average degree in each bucket from samples nodes that fall in it. If
  $$|S_i| > \left(\frac{\epsilon}{n}\right)\frac{|S|}{ct},$$
  then $\rho_i \leftarrow |S_i|/|S|$, otherwise $\rho_i \leftarrow 0$.
- Output
  $$\sum_i \rho_i(1 + \beta)^{i-1}.$$

Analysis

Let us define “large” $i$ to be the buckets that

$$|B_i|/n > \left(\frac{\epsilon}{n}\right)^{2/3}/(c't)$$

for some constant $c'$; and otherwise, “small” $i$.

The output is upper bounded. Suppose for all $i$, $\rho_i = |B_i|/n$ then

$$\sum_i \rho_i(1 + \beta)^{i-1} = \sum_i |B_i|(1 + \beta)^{i-1}/n \leq \sum_v d(v)/n.$$
Similarly if $\rho_i \leq |B_i|(1 + \gamma)/n$, then

$$\sum_i \rho_i(1 + \beta)^{i-1} \leq \sum_v (1 + \gamma)d(v)/n.$$ 

Sampling guarantees that for all $i$, $\rho_i \leq |B_i|(1 + \gamma)/n$. Hence the output is bounded with high probability. The algorithm above might undercount in the following two ways:

- We are using the lower bound for the degree of each bucket.
- For small buckets $B_i$, we set $\rho_i$ to be 0.

First check “large” buckets. Suppose $\rho_i = |B_i|/n$, then

$$\sum_i \rho_i(1 + \beta)^{i-1} = \sum_i |B_i|(1 + \beta)^{i-1}/n \geq \sum_v (1 - \beta)d(v)/n.$$ 

Similarly, if $\rho_i \geq |B_i|(1 - \gamma)/n$, then

$$\sum_i \rho_i(1 + \beta)^{i-1} \geq \sum_v (1 - \gamma)(1 - \beta)d(v)/n.$$ 

And sampling guarantees that for all “large” $i$, $\rho_i \geq (|B_i|/n)(1 - \gamma)$, hence the approximation for the “large” buckets will not be too small.

But for small buckets, we might undercount. There are three types of edges:

- “large - large”: both endpoints are in large buckets. Then we count the edge twice.
- “large - small”: one endpoint in large bucket, one in small. We only count once.
- “small - small”: both endpoints are in small buckets. Then we never count them.

For a “large” bucket such that $|B_i|/n > (\epsilon n)^{1/2}/(ct)$, by Chernoff bound with high probability

$$|S_i| > \left(\frac{\epsilon}{n}\right)^{1/2}|S|/(ct).$$ 

Chernoff also implies with high probability all “small-small” edges are between “small” buckets with less than

$$\left(\frac{\epsilon}{n}\right)^{1/2}/(ct) = O((\epsilon n)^{1/2})$$ 

nodes. There are at most

$$O(\epsilon n)$$

“small - small” edges. This affects the average degree approximation by at most $\epsilon$ additive and multiplicative factor.

For “large-small” edges, we undercount by $1/2$-multiplicative factor. In total, Algorithm 1 yields $(2 + \epsilon)$-approximation. And so far we are only using degree queries.

**Algorithm 2: allowing neighbor queries**

**Idea**

The main improvement that needs to be made is on “large-small” edges. The goal of Algorithm 2 is to estimate the fraction of “large-small” edges, and correct the undercount. If we could pick a random edge, we could just use standard sampling and Chernoff bound to bound the result. However, given the representation model, we can only do neighbor queries such that for any $(v, j)$, return $j$-th neighbor of $v$. And we can do random neighbor queries, such that for any $v$, return $j$-th neighbor of $v$ where $j$ is chosen randomly from $[1...d(v)]$. Instead of picking a random edge, we can pick a random edge in a given bucket; and that can also help us estimate the fraction of “large-small” edges.
Algorithm

Same as Algorithm 1.

1. **For all “large” buckets** $S_i$:
   2. **repeat** $O(1/\delta)$ times:
      1. For all $v \in S_i$, pick a random neighbor $u$ of $v$. Let $\chi(v)$ be 1 if $u$ is “small”; 0 otherwise.
      2. **Average** $\alpha_i \leftarrow |\{v \in S_i | \chi(v) = 1\}|/|S_i|$.
   3. **Output** $\sum_i \rho_i (1 + \alpha_i)(1 + \beta)^i - 1$.

Analysis

If all nodes in a bucket have same degree $d$. Let $T = \text{number of “large - small” edges in } B_i$. For “large - small” edge $e$,

$$\Pr[e \text{ chosen in a loop}] = 1/(d|B_i|).$$

The expectation is

$$E[\alpha_i] = T/(d|B_i|).$$

Since all nodes in a bucket have degree $d$ within a $(1 + \beta)$ factor. Rewrite the above analysis, we have For “large - small” edge $e$,

$$1/(1 + \beta)(d|B_i|) \leq \Pr[e \text{ chosen in a loop}] \leq (1 + \beta)/(d|B_i|).$$

The expectation is

$$T/((1 + \beta)d|B_i|) \leq E[\alpha_i] \leq (1 + \beta)T/(d|B_i|).$$

If we pick $\delta$ corresponding to $\epsilon$, we can get an estimation $(1 + \epsilon)$-approximation for the average degree.