6.889 Sublinear Time Algorithms 11 February 2019

Lecture 2

Lecturer: Ronitt Rubinfeld *Scribe: Kavya Ravichandran*

Contents

Last Time and Context

In the previous lecture, we showed that we can estimate the number of connected components of a graph with degree $\leq d$ to $\pm\epsilon n$ in time $O(\frac{d}{\epsilon^4})$ with failure probability $\leq \frac{1}{4}$. In order for the failure probability to be $\leq \beta$, we require time $O(\frac{d}{\epsilon^4}\log\frac{1}{\beta})$ by using the Chernoff Bound.

There are two kinds of errors that arise in this kind of approximation algorithm. One is the **approximation error** and the other is **failure probability**.

When the algorithm outputs one of two possible answers (e.g. Yes/No or 1/0), the failure probability can be reduced by running the algorithm several times and outputting the majority answer (the one that was seen most often). In Homework 0, we show that running the algorithm $O(\log 1/\beta))$ times reduces the probability of failure to at most β. When the algorithm outputs a number which is a good approximation with failure probability at most 3/4, in Homework 0, we show that running the algorithm $O(\log 1/\beta))$ times and outputting the median value reduces the probability of failure to at most β.

1 Minimum Spanning Tree

In this section, we develop a randomized approximation algorithm for estimating the weight of a minimum spanning tree of a given graph. We use the connected component approximator developed in Lecture 1 as a subroutine in this algorithm.

1.1 Specification

Input We are given a connected graph $G = (V, E)$. This graph is represented as an **adjacency list**. There are $n = |V|$ nodes with maximum degree d. Each edge (i, j) has weight $w_{ij} \in \{1, ..., w\} \cup \infty$. We are also given ϵ, the parameter that controls the accuracy of the approximation.

Output If M is $\min_{T \text{ spans } G}\{w(T)\}$ where $w(T) = \sum_{(i,j)\in T} w_{ij}$, we want to output $\widehat{M}$ such that $(1-\epsilon)M \leq \widehat{M} \leq (1+\epsilon)$. We will get an additive approximation, and then using the bound on the weights, we make it a multiplicative approximation.

1.2 A different characterization of MST

In order to recast the problem of finding the minimum spanning tree weight into finding the number of connected components of a graph, we define auxiliary graphs with only edges with weight less than some value i.

Definition 1 *$G^{(i)} = (V, E^{(i)})$ where $E^{(i)} = \{(u, v)|w_{uv} \in \{1, ..., i\}\}$.*

Definition 2 *$C^{(i)}$ is the number of connected components in $G^{(i)}$.*

By these definitions, we have $C^{(0)} = n$ and $G^{(w)} = w$.

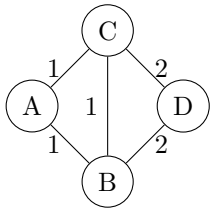
Examples

Example 1: $w = 1$

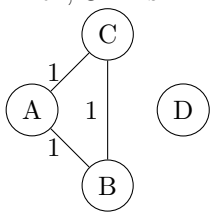
If $w = 1$, then $M = n - 1$, and $G^{(1)}$ is connected.

Example 2: $w = 2$

Consider a simple example of a graph G where each edge has weight $w_{ij} \in \{1, 2\}$.



Then, $G^{(1)}$ is:



The number of connected components is as follows:

$$C^{(1)} = 2, C^{(2)} = 1$$

We can write the total weight of the MST for this graph as:

$$M = (n-1) + 1 \cdot (C^1 - 1) = n - 2 + C^{(1)}$$

This is because all $n-1$ edges in the graph must have at least one unit of weight, and then $C^{(1)} - 1$ edges have an extra unit of weight.

With intuition drawn from the example above, we make following claim:

Claim 3 *In general, the weight of M, the minimum spanning tree of G, is:*

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

Proof Let α_i be the number of edges of weight i in some [1] MST of G. We have that:

$$\sum_{i>l} \alpha_i = C^{(l)} - 1$$

This is because the number of edges required to connect the connected components of a graph without introducing cycles is exactly one less than the number of connected components. Then, because the

[1] "any" is valid here, since all minimum spanning trees of a graph will have the same number of edges of a given weight, even if the specific edges are different.

weight of the tree is exactly the weighted sum of the number of edges with that weight, this leads to:

$$M = \sum_{i=1}^{w} i\alpha_i \tag{1}$$

$$= \sum_{i=1}^{w} \alpha_i + \sum_{i=2}^{w} \alpha_i + \ldots + \sum_{i=w}^{w} \alpha_i \tag{2}$$

$$= C^{(0)} - 1 + C^{(1)} - 1 + \ldots + C^{(w-1)} - 1 \tag{3}$$

$$= n - 1 + C^{(1)} - 1 + \ldots + C^{(w-1)} - 1 \tag{4}$$

$$= n - w + \sum_{i=1}^{w-1} C^{(i)} \tag{5}$$

■

1.3 Algorithm

From the last lecture, we know how to estimate the number of connected components, using $\widetilde{c}$-Calculator. However, in this setting, we must call it w times, and that means that we must tune the error parameters ϵ and β to limit the error bounds of the MST weight approximation algorithm. In particular, we set $\boxed{\epsilon' = \frac{\epsilon}{2w}}$ and $\boxed{\beta = \frac{1}{10w}}$ (justified in the next section) to get the following algorithm:

Algorithm 1: MST Weight Approximator

Input : Graph $G = (V, E)$, ϵ
Output: $\widehat{M}$.

```
for i = 1 to w − 1 do
    Calculate Ĉ^(i) using c̃-calculator with ε′ and β as defined above
Return M̂ = n − w + Σ_{i=1}^{w−1} Ĉ^(i)
```

Using the values of ϵ' and β defined, to run the algorithm w times, we get a runtime of:

$$O\left(\frac{d}{\epsilon'^4} \cdot \log w \cdot w\right) = O\left(\frac{dw^5 \log w}{\epsilon^4}\right)$$

The best time complexity, given in [Chazelle et al. 2005], is $O(\frac{dw}{\epsilon^2} \log \frac{dw}{\epsilon^2})$ with $\Omega(dw/\epsilon^2)$ space overhead.

1.4 Approximation Guarantee

With the chosen parameters, we are able to show ϵ **multiplicative error** guarantee and **failure probability less than** $\frac{1}{10}$.

Failure Probability Since we call $\widetilde{c}$-calculator with failure probability of $\leq \frac{1}{10w}$, we use the Union Bound to show:

$$\mathbb{P}(\text{any failure}) \leq w \cdot \frac{1}{10w} \leq \frac{1}{10}$$

Approximation Error In the $\frac{9}{10}$ case in which there is no failure, for each call, $\widehat{C}^{(i)} = C^{(i)} \pm \epsilon' n = C^{(i)} \pm \frac{\epsilon}{2w} n$, where $\widehat{C}^{(i)}$ is the approximate number of connected components in $G^{(i)}$ and $C^{(i)}$ is the true number of connected components in $G^{(i)}$. Then, for all calls:

$$\sum \widehat{C}^{(i)} \leq \sum (C^{(i)} \pm \epsilon' n) \tag{6}$$

$$= \sum (C^{(i)}) \pm w \cdot \frac{\epsilon}{2w} \cdot n \tag{7}$$

$$= \sum C^{(i)} \pm \frac{\epsilon n}{2} \tag{8}$$

We have $|M - \widehat{M}| \leq \frac{\epsilon n}{2}$, and since $M \geq n - 1 > \frac{n}{2}$, the first expression can be rewritten as:

$$|M - \widehat{M}| \leq \frac{\epsilon n}{2} < \epsilon M$$

Thus, we have a ϵ-approximation for the weight of the MST of a graph.

Notes Some notes on the presented algorithm.

- If $d \approx n$ (that is, the graph is very dense) or w is large, we don't get much of a benefit from this algorithm relative to Kruskal's.
- In terms of $G^{(i)}$, it might, at first, seem that it would take $O(|E|)$ time to calculate each one, but it turns out that during BFS (called in $\widetilde{c}$-calculator), we can just ignore any edges we reach with weight $> i$ without any overhead.
- An important question is: *which edges are on this MST?* It turns out we cannot know in sublinear time! Consider differentiating between a cycle and a path on n nodes. In order to output a spanning tree, on the path, one must answer "yes" on every edge, whereas on the cycle, one must answer "no" on exactly one edge. We need a linear number of steps to know whether we are walking along a tree or a cycle.

2 Average Degree

In this problem, we seek to estimate the average degree of a graph. Note that if we know the number of edges in the graph, then this problem is trivial. However, we are given an adjacency list representation of the graph along with a vector of degrees. Thus, our algorithm must strategically sample the degree vector to get a good representative sample.

We did not complete the discussion of the algorithm in lecture today but provided lower bounds and considered the strategy of **bucketing**, as presented in [Goldreich and Ron 2005].

2.1 Setup

Definition 4 *The average degree of a graph G is defined as follows:*

$$\bar{d} = \frac{\sum_{u \in V} d(u)}{n}$$

Assumption 5 *G is simple (no parallel edges or self-loops) and has $\Omega(n)$ edges. Further, G is represented as an adjacency list such that each node u has associated with it a array of size $d(u)$, (where $d(\cdot)$ represents the degree of the node). We have two different kinds of queries we can make:*

- ***degree query*** *we can query the degree of a given node.*

- ***neighbor query*** *we can access any element of the adjacency list of a given node in* $O(1)$.

Using naive sampling, if we pick some s sample nodes $\{v_1...v_s\}$ and output $\frac{1}{s}\sum_{i=1}^{s} d(v_i)$, we can imagine a pathological case, such as a star graph, where most of the nodes have very low degree but one node is connected to all of the rest. Here, the average degree is almost two, but if we only sample the degree-one nodes (which is likely, given all but one node have degree one), then we will report one as the degree. Then, the answer produced by a naive sampling approach is far from the true answer [2].

2.2 Lower Bounds

2.2.1 "Ultra-sparse"[3] Graphs

The **empty graph** (i.e., n nodes and no edges) has $\bar{d} = 0$, while adding exactly one edge changes $\bar{d}_1 = \frac{2}{n}$. To distinguish between these two graphs, we need to check roughly all of the nodes, i.e., $\Omega(n)$.

2.2.2 n-cycle and cycle with clique

In the n-cycle graph, $\bar{d} = 2$. In the graph that contains a clique of size $c\sqrt{n}$ and a path among the rest of the nodes:

$$\bar{d} = \frac{2 \cdot (n - c\sqrt{n}) + (c\sqrt{n} - 1)(c\sqrt{n})}{n}$$

where the former is the contribution of the nodes in the path and the latter is from the clique. This is approximately $2 + c^2$, so for large c, there is a significant difference between the two. However, it will take $\sqrt{n}/c$ steps to find the clique, if we are in that case, thus requiring checking $\Omega(\sqrt{n})$.

2.3 Bucketing

Through truly random sampling, we end up with a large enough variance that Chernoff cannot save us. Thus, we want to narrow the variance, and we accomplish this by *putting similar things together*, otherwise known as *bucketing*.

Here, we split the nodes into a logarithmic number of buckets, $t = O(\frac{\log n}{\epsilon})$, and we define $\beta = \epsilon/c$. Then, bucket B_i is defined as:

$$B_i = \{v | (1+\beta)^{i-1} < d(v) \leq (1+\beta)^i\} \text{ for } i \in \{0...t-1\}$$

For completeness, we define $B_0 = \{v | d(v) = 1\}$ and $B_{-1} = \{v | d(v) = 0\}$. Note that $B_1 = \{v | 1 < d(v) \leq (1+\beta)\}$ is empty by definition.

The total degree of nodes in $B_i \in \{(1+\beta)^{i-1} \cdot |B_i|, ..., (1+\beta)^i \cdot |B_i|\}$. Then, the total degree of the graph is:

$$\frac{\sum_i (1+\beta)^{i-1}|B_i|}{n} < \frac{m}{n} \leq \frac{\sum_i (1+\beta)^i |B_i|}{n}$$

We know β and n, so we only need to approximate $|B_i|$ in order to approximate $\bar{d} = \frac{m}{n}$. We will show how to do this via sampling in the next lecture.

[2] This depends on how we define "far" (i.e., what kind of approximation are we looking for?). The naive sampling algorithm would produce one with high likelihood, whereas the true answer would be $2 - \frac{2}{n}$. This is not too "far" if we consider additive error (less than one).

[3] "Ultra-sparse" is not formal verbiage but simply an intuitive description of the types of graphs for which we provide these lower bounds.

References

1. Chazelle, B., Rubinfeld, R. & Trevisan, L. Approximating the Minimum Spanning Tree Weight in Sublinear Time. SIAM Journal on Computing 34, 1370-1379 (2005).

2. Oded Goldreich & Ron, D. Approximating Average Parameters of Graphs. (2005). Available at: `http://www.wisdom.weizmann.ac.il/~oded/p_aver-graph.html`. (Accessed: 13th February 2019)