CS 59000 CTT Current Topics in Theoretical CS	Aug 30, 2012

Lecture 4

Lecturer: Elena Grigorescu *Scribe: Pinar Yanardag Delul*

1 Estimating the number of connected components and minimum spanning tree

In *Lecture 3*, we showed a test for connectivity of graphs and today we'll see how to estimate the number of connected components of a graph in constant time. In particular, we will focus on the following problems:

- Estimating the number of connected components
- Estimating minimum spanning tree weight

These results are due to Chazelle et al [1]. Recall what we said about testing connectivity in the last lecture. It reduced to showing that if we have a graph that needs to be modified in many places in order to obtain a connected graph then it must have many connected components of small size. And then we showed that we can find these connected components with high probability by sampling random vertices and visiting the components that they belonged to in a BFS manner. We'll use a similar idea here as well and we are going to be estimating the number of connected components by estimating sizes of random connected components.

2 An additive approximation algorithm

Recall that an α-additive approximation algorithm is an algorithm for an optimization problem that outputs a solution within α of the optimum solution. That is, if OPT is the value of the solution to the optimization problem, an α-additive approximation algorithm outputs a value v such that $|v - OPT| \leq \alpha$.

Theorem 1 *There $\exists$ a randomized algorithm that given ϵ and a graph $G(V, E)$, with $|V| = n$ outputs an ϵn-additive approximation to the number of connected components of G, w.p. $2/3$. That is, if C is the number of connected components of G, w.p. $2/3$ the algorithm outputs v s.t. $|v - C| \leq \epsilon n$.*

Notice that this algorithm is useful when G has many connected components i.e. $\Theta(n)$. Recall from last lecture that if a graph has many connected components then it must have many connected components of small size (say, of size $2/\epsilon$. See Figure 2) At a very high level, the algorithm will use the size of the small components to estimate C, the number of connected components.

In what follows we will prove Theorem 1.

For a vertex v let $n_v = \#$ of vertices in $C(v)$ (where $C(v)$ denotes the connected component where v lies).

We are going to use n_v in order to estimate the number of connected components. We can make two simple observations about n_v:

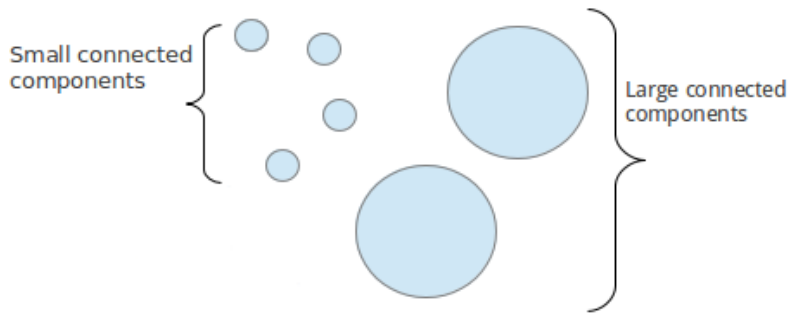

Figure 1: Size of the small connected components is $\frac{2}{\epsilon}$

.

Observation 2 $\sum_{v \in C_i} \frac{1}{n_v} = 1$ *where* C_i *is the connected component* i*. This is true since* $|C_i| = n_v$ *for all* $v \in C_i$*.*

Observation 3 $\sum_{v \in V} \frac{1}{n_v} = C$*. (This is true since each component contributes exactly* 1 *to the sum.)*

Now we have a way to compute the number of connected components in some sort of a local way (i.e. n_v gives us some local information about v). But notice that if we try to calculate exactly the number of connected components by using this quantity, it will take $O(n^2)$ time (for each vertex, we have to look all of its neighbors). Instead, we can try to estimate n_v for a random sample and hope to get something close to the number of connected components.

If we pick a random sample S of vertices and compute the estimate $\sum_{v \in S} \frac{1}{n_v}$, notice that if n_v is large, then $\frac{1}{n_v}$ is small, and so it has a small contribution to the sum. In our algorithm we will see that is enough to round the number of vertices inside a large component to a fixed threshold (in our case $\frac{2}{\epsilon}$).

Definition 4 *Let* $\hat{n}_v = \min\{n_v, 2/\epsilon\}$*. That is,* $\hat{n}_v = \begin{cases} n_v & \text{if } v \in \text{connected component of size} < \frac{2}{\epsilon} \\ \frac{2}{\epsilon} & \text{otherwise} \end{cases}$

In what follows we will denote a component of size $< \epsilon/2$ a *small* component.

Lemma 5 $0 \leq \frac{1}{\hat{n}_v} - \frac{1}{n_v} < \epsilon/2$.

Proof

If v is in small component then $\frac{1}{\hat{n}_v} - \frac{1}{n_v} = 0$, else $0 < \frac{1}{\hat{n}_v} - \frac{1}{n_v} = \frac{\epsilon}{2} - \frac{1}{n_v} < \epsilon/2$. ■

We can now define an estimate for C using the rounding above.

Definition 6 *Let* $\hat{C} = \sum_{v \in V} \frac{1}{\hat{n}_v}$

Lemma 7 $|C - \hat{C}| < \frac{\epsilon n}{2}$

Proof Using Lemma 5 we have $0 \leq \sum_{v \in V} \frac{1}{\hat{n}_v} - \frac{1}{n_v} \leq \epsilon n/2$ and hence $|\hat{C} - C| \leq \epsilon n/2$ ■

We are now ready to see an algorithm that estimates C by estimating $\hat{C}$, which in turn can be achieved by sampling a bunch of vertices and computing $\hat{n}_v$ for each sampled vertex v. In order to compute $\hat{n}_v$ we will search the connected component of v and if we found that it is strictly smaller than $\frac{2}{\epsilon}$ we set $\hat{n}_v$ be the actual size of the component, otherwise we set $\hat{n}_v = \frac{2}{\epsilon}$. Then we will sum up the values $\frac{1}{\hat{n}_v}$, take the average and multiply by n and that will be our output (our estimate).

Algorithm 1 Algorithm that, given G and ϵ estimates the number of connected components in G.

1: Pick $s = O(\frac{1}{\epsilon^2})$ vertices in a sample $S = \{v_1, \ldots, v_s\}$
2: Run BFS around each v_i until the search reaches at most $\frac{2}{\epsilon}$ vertices.
3: For each such $v \in S$ set $\hat{n_v}$ = number of vertices in v's component if it is $< \frac{2}{\epsilon}$ and set $\hat{n_v} = \frac{2}{\epsilon}$ otherwise
4: Output $\left(\frac{1}{s} \sum_{v \in S} \frac{1}{\hat{n_v}}\right) n = C'$

We claim that the output of Algorithm 1 is a very good estimate of the number of connected components:

Theorem 8 *With probability 2/3 (over its randomness), Algorithm 1 outputs C' such that $|C' - C| < \epsilon n$, and runs in time $O(\frac{1}{\epsilon^4})$.*

Runtime analysis Running BFS takes $O((\frac{2}{\epsilon})^2)$ time and this is essentially the complexity of the algorithm (this step can be done even in linear time if we know a bound d on the degrees), but here we will just use a rough estimate).

We will need to analyze the deviation from expectation of the output estimate. We have previously seen a few examples of how to use the Chernoff bound in order to achieve this step. We will use the bound again in our analysis and we recall it next.

Theorem 9 (Chernoff) *Let random variables $X_1, \ldots, X_s \in [0, 1]$ be independent and identically distributed and let $X = \sum_{i=1} X_i$. Then $Pr\left[|X - E[X]| > \frac{\epsilon s}{2}\right] < e^{-\frac{\epsilon^2}{8} s} = e^{-\Omega(\epsilon^2 s)}$*

Lemma 10 $Pr[|C' - \hat{C}| > \frac{\epsilon}{2} n] < \frac{1}{3}$

Proof Let $X_i = \frac{1}{\hat{n_{v_i}}}$ be a random variable for the quantity $\frac{1}{\hat{n_{v_i}}}$ associated to the ith sampled vertex where $1 \leq i \leq s$. Then $E[X_i] = \frac{1}{n} \sum_{v \in V} \frac{1}{\hat{n_v}} = \frac{\hat{C}}{n}$, for every i. Let $X = \sum_{i=1}^{s} X_i$, then $E[X] = E[\sum_{i=1}^{s} X_i] = \sum_{i=1}^{s} E[X_i] = s\frac{\hat{C}}{n}$. So $X = \sum_{i=1}^{s} \frac{1}{\hat{n_{v_i}}}$ is a random variable for the value that we output (the algorithm actually outputs $C' = \frac{n}{s} X$). We want to understand how far $C' = \frac{n}{s} X$ is from its expected value (i.e. $\frac{n}{s} E[X] = \hat{C}$). For this we apply Chernoff's bound and obtain

$$Pr[|X - E[X]| \geq \frac{\epsilon}{2} s] < e^{-\Omega(\epsilon^2 s)} < \frac{1}{3},$$

for $s = O(\frac{1}{\epsilon^2})$. So

$$Pr[|X\frac{n}{s} - \frac{s}{n}\hat{C}\frac{n}{s}| > \frac{\epsilon}{2}s\frac{n}{s}] < \frac{1}{3},$$

and finally

$Pr[|C' - \hat{C}| > \frac{\epsilon n}{2}] < \frac{1}{3}$, concluding the proof.

■

To summarize, the algorithm above outputs an $\epsilon n/2$-additive approximation to $\hat{C}$, with probability 2/3. We will next show that this is also an ϵn-additive approximation to C, the number of connected components.

Proof [of Theorem 1] By Lemma 10 we know that $Pr[|C' - \hat{C}| > \epsilon n/2] < 1/3$ and by Lemma 7 $|\hat{C} - C| < \epsilon n/2$, and recall that we want to show that $Pr[|C' - C| > \epsilon n] < 1/3$.

Note that if $|C' - C| > \epsilon n$ then $\epsilon n < |C' - C| \leq |C' - \hat{C}| + |\hat{C} - C|$ and so $|C' - \hat{C}| > \epsilon n/2$, which, by Lemma happens w.p. 1/3. It follows that $Pr[|C' - C| > \epsilon n] < 1/3$ which concludes the proof.

■

What if we want a smaller probability of error, that is we want that that to increase our confidence to $1 - \delta$, where δ is can be set arbitrarily small?

Observation 11 *If we pick $s = O(\frac{1}{\epsilon^2} log \frac{1}{\delta})$ samples in the above algorithm we have that*

$$Pr[|C' - \hat{C}| > \frac{\epsilon}{2}n] < e^{-\Omega(\epsilon^2 s)} < \delta,$$

3 Estimating the Minimum spanning tree weight

Now, we will see how to estimate the MST of a weighted connected graph. We will use the algorithm for estimating the number of connected components from the previous section as a subroutine(in fact the one where we pick $s = O(\frac{1}{\epsilon^2} log \frac{1}{\delta})$ samples.

For a given undirected graph G and n vertices and edge-weights $W = \{w_e | e \in E(G)\}$ are a given set of e where $w_e \in \{1, 2, \ldots, w\}$. The goal is to estimate the MST (a spanning tree of minimum weight), $w(MST(G))$ i.e to output an estimate $\hat{w}(MST(G))$ s.t $|\hat{w}(MST(G)) - w(MST(G))| < \epsilon n$.

In order to do it in sublinear time, we will try to express the weight of the MST as a function of some local quantities, and those local quantities will be the sizes of connected components of some carefully chosen subgraphs. We will look at subgraphs of G and count the number of connected components in these subgraphs and we are going to use estimates of these to estimate $w(MST)$.

Let G_1 = subgraph of G with edges of weight ≤ 1. Let C_1 be the number of connected components in C_1.

Let T be some $MST(G)$ and we want to estimate how many edges in T have weight strictly greater than 1.

We can now make a few observations.

1. T cannot have an edge of weigth > 1 between the vertices of a connected component of G_1 since that would result in a cycle. So in order to get T we must add edges of wt > 1 between disconnected components of C_1.

2. Each time an edge is added to C_1 (greedily, maintaining the tree spanning structure) the number of disconnected components in the new graph decreases by exactly 1.

Therefore, we have proved that the # of edges of weight > 1 in T is exactly $C_1 - 1$.

More generally, define another graph G_i which is the subgraph of G with only weight $\leq i$, where $i \leq w$.

By the same argument as above, the number of edges of weight $> i$ in T is exactly $C_i - 1$,

Observe that this is true even for $i = 0$ as the number of components of G_0 is n and the number of edges in T is $n - 1$.

Lemma 12 *w(MST(G))* $= \sum_{0 \leq i \leq w}(C_i - 1)$

Proof Let N_i be the number of edges of weight i in T. Then

$$\begin{aligned} w(MST) &= \sum_{i=1}^{w} i \cdot N_i \\ &= \sum (1 + 1 + \ldots + 1) N_i \\ &= \sum_{i=1}^{w} N_i + \sum_{i=2}^{w} N_i + \ldots + \sum_{i=w-1}^{w} N_i + \sum_{i=w}^{w} N_i \\ &= \sum_{i=0}^{w} C_i - 1, \end{aligned}$$

where the last identity uses the claim proved above that $\sum_{i=j}^{w} N_j = C_j - 1$. ■

This is gives us a more local way to express the weight of the MST. From here, we will use the previous lemmas for estimating the number of connected components for each of the $G_i's$.

Algorithm 2 Algorithm that, given G and ϵ and set of weights W estimates MST(G).

1: For $i = 1, ..w$ compute $\hat{C}_i$, an approximate to the number of connected components of G_i by running Algorithm 1 on the graph G_i, with $\epsilon' = \epsilon/w$ and confidence parameter $\delta = \frac{1}{3w}$.
2: Output $\hat{w}(MST) = -w + \sum_{i=1}^{w-1} \hat{C}_i$

Analysis If all estimates $\hat{C}_i$ are within an additive error $\frac{\epsilon}{w}n$, i.e. $|\hat{C}_i - C_i| \leq \frac{\epsilon}{w}n$ then

$$|\sum_i \hat{C}_i - C_i| = |\hat{w}(MST) - w(MST)| \leq \sum_{i=0}^{w-1} |\hat{C}_i - C_i| \leq w \cdot \frac{\epsilon}{w} n = \epsilon n.$$

So, with $O(\frac{1}{\epsilon^2} \log \frac{w}{\epsilon})$ samples, for each i $Pr[|\hat{C}_i - C_i| \geq \frac{\epsilon}{w}n] < \frac{1}{3w}$. By a union bound

$$Pr[\text{ one of the estimates is bad }] \leq w \cdot \frac{1}{3w} = 1/3.$$

References

[1] Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005.