6.897: CSCI8980 Algorithmic Techniques for Big Data October 10, 2013

Lecture 10

Dr. Barna Saha *Scribe: Kaustubh Duraphe*

Overview

In the previous lecture, we studied how, for a given graph $G(V, E)$, the semi-streaming algorithms allow us to acheive the polylogarithmic space bound in terms of the number of edges E. In the context of semi-streaming algorithms, we also studied some properties of graph such as connected components and k-edge connectivity. We studied that in the $\alpha-$ spanner of a graph $G(V, E)$, if we store all edges in $O(n^{1+1/t})$ space then the approximate distance between any pair of vertices in G is no more than $2t - 1$, where $2t$ is some set threshold.

In this lecture, we study how graph sparsifiers can be used to approximate a graph to a sparse graph. This can be done by approximating the cuts in the original graph. We start with the first approach to cut approximation which includes sampling every edge with uniform probability. We see the shortcoming of this approach and consider another approach that samples the edge with higher probability if the edge participates in the cut. Fianlly, we study some basic properties of the graph sparsifiers to conclude this lecture.

1 Graph Sparsifiers

Let $G = (V, E)$ be an unweighted graph and $S \subset V$. Let $\overline{S} = V/S$. Our goal is to find a sparse subgraph of G, say G_p, within the given approximation factor $(1 \pm \epsilon)$. One way to do this is to approximate all cuts of G within the approximation factor.

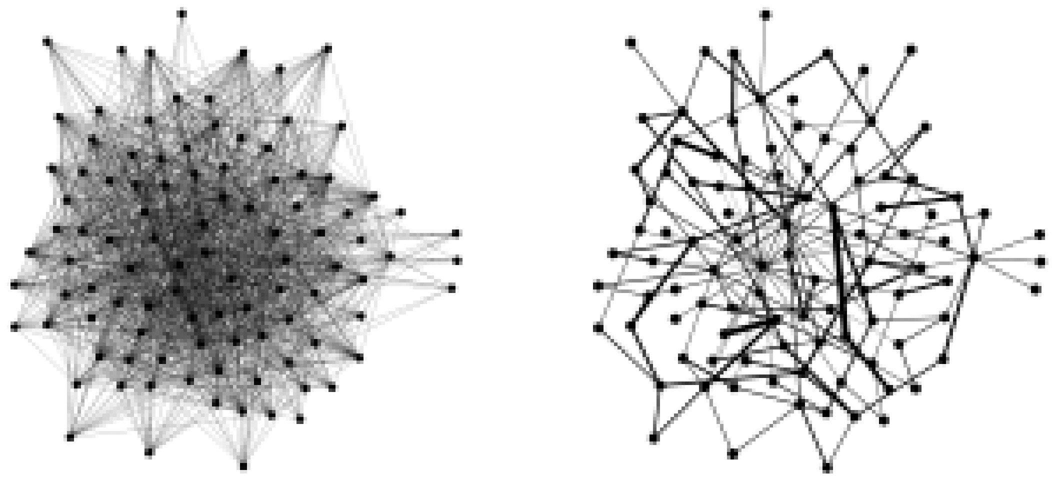

Figure 1: Graph Sparsifiers[5]

1.1 Sampling each edge independently

One way to estimate the cut is to sample each edge $e \in E$ of the graph G independently i.e. with the equal probability, say p. After sampling, the graph G gets transformed to a sparse graph G_p. We then consider a cut $(S, \overline{S})$ and compute the cut value $C_{G_p}(S, \overline{S})$. In expectation, we preserve cuts by scaling them up by the factor of $(\frac{1}{p})$. So, the final value returned is $\frac{C_{G_p}(S,\overline{S})}{p}$.

For every edge $e \in C_G(S, \overline{S})$, we define an indicator random variable X_e such that

$$X_e = \begin{cases} 1, & \text{if } e \text{ is sampled} \\ 0, & \text{otherwise} \end{cases}$$

Therefore, cut size in the sampled graph X is given by,

$$\begin{aligned} X &= C_{G_p}(S, \overline{S}) \\ &= \sum_{e \in C_G(S,\overline{S})} X_e \end{aligned}$$

Now,

$$\begin{aligned} E[X] &= \sum_{e \in C_G(S,\overline{S})} E[X_e] \\ &= \sum_{e \in C_G(S,\overline{S})} p \\ &= p \cdot C_G(S, \overline{S}) \end{aligned}$$

Scaling the result by $\left[\frac{1}{p}\right]$,

$$E\left[\frac{X}{p}\right] = C_G(S, \overline{S})$$

It can be seen that after scaling, the cut value of sampled graph is same as the original graph.

By Chernoff bound,

$\text{Prob}\left[\left|\frac{X}{p} - C_G(S,\overline{S})\right| > \epsilon \cdot C_G(S,\overline{S})\right]$

$$\begin{aligned} &= \text{Prob}\left[\left|X - p \cdot C_G(S,\overline{S})\right| > \epsilon \cdot p \cdot C_G(S,\overline{S})\right] \\ &= \text{Prob}[|X - E[X]| > \epsilon \cdot E[X]] \\ &\leq e^{\frac{-E[X]\cdot\epsilon^2}{2}} \\ &\leq e^{\frac{-p\cdot c\cdot\epsilon^2}{2}} \end{aligned}$$

where c = $C_G(S, \overline{S})$

In order to reduce the deviation, we would like to have the probability value as small as possible. We have, $e^{\frac{-p\cdot c\cdot \epsilon^2}{2}} = \frac{1}{d}$

We set p = $\frac{2\cdot d\cdot \log n}{c\cdot \epsilon^2}$ i.e. (1)

$$\text{Prob}\left[\left|\frac{X}{p}\cdot \epsilon \cdot (1\pm\epsilon)\cdot c\right|\right] > (1 - \tfrac{1}{n^d})$$

Karger's Theorem: For any graph G, if the minmum cut of G is A, then the number of cuts of value less than $\alpha \cdot A$ for any $\alpha \geq 1$ is almost $n^{2\alpha}$.

Considering minimum cut A of graph G in equation (1),

$$\text{p} = \frac{2\cdot d\cdot \log n}{A\cdot \epsilon^2}.$$

We consider all cuts whose value is α · A. Number of such cuts $\leq n^{2\cdot\alpha}$. For any of such cuts, probability of deviation greater than $\pm\epsilon$ times expected value is

$$\leq e^{\frac{-\alpha\cdot A\cdot p\cdot \epsilon^2}{2}} = e^{-\alpha\cdot d\cdot \log n} = \frac{1}{n^{d\cdot\alpha}}$$

Therefore, probability that there exists at least one cut of value $\alpha \cdot A$ for which deviation is high $\leq \frac{n^{2\cdot\alpha}}{n^{d\cdot\alpha}} = \frac{1}{n^{(d-2)\cdot\alpha}}$.

This method works well when minimum cut value is small.

1.2 Beneczur and Karger's Algorithm for graph sparsification

In the previous algorithm, every edge is sampled independently or with equal probability. The shortcoming of this approach is that the number of edges decreases by the factor of A/log n and in some cases, the graph could still be dense. To give an example, two cliques connected with a single edge (A = 1), could still possibly remain dense after sampling. So, sampling every edge uniformly or independently may not always work well. To overcome this, Beneczur and Karger algorithm samples an edge with high probability if it participates in the cut.

Theorem: For every weighted graph $G(V, E)$ on $|V| = n$ vertices and error parameter $\epsilon > 0$, there exists a weighted subgraph G_p with $O(n \cdot \log n \cdot \epsilon^{-2})$ edges such that $G_p \in (1 \pm \epsilon) \cdot G$.

Main Idea: Sample edges non-uniformly such that each edge e ∈ E will be sampled with the probability p_e which is inversely proportional to its connectivity.

In other words, if an edge participates in the cut, the minimum cut value for the edge is k_e and the probability with which it will be sampled is $(\frac{p}{k_e})$. So, dense regions will be sampled with low probability thereby, reducing the number of edges in such regions.

Goal: Our goal is to find minimum cut value k_e of cuts in which an edge e ∈ E participates such that $0 \leq \frac{k_e}{k} \leq 1$ and $w_e = \frac{k_e}{k}$. So, when w_e becomes small, sampling probability increases and vice versa.

Algorithm 1 Algorithm

```
For each edge e
Compute $w_e = (\frac{k_e}{k})$ where $k_e$ is the minimum value of cuts in which edge e participates.
Sample e with probability $(\frac{p}{w_e})$.
For any cut (S, $\overline{S}$)
**if** we have edges $e_1$, $e_2$, $e_3$ ... $e_l$ crossing the cut **then**
    return the estimated cut value as $(\frac{1}{p} \cdot \sum_{1}^{l} w_e)$
**end if**
```

For every edge $e \in cut(S, \overline{S})$, we define an indicator random variable X_e such that

$$X_e = \begin{cases} 1, & \text{if } e \text{ is sampled with probability } \frac{p}{w_e} \\ 0, & \text{otherwise} \end{cases}$$

Therefore, we have,

$$\begin{aligned} X &= \sum w_e \cdot X_e \\ E[X] &= \sum w_e \cdot E[X_e] \\ &= \sum w_e \cdot (\frac{p}{w_e}) \\ &= p \cdot C_G(S, \overline{S}) \end{aligned}$$

Hence, we can say that the expectation is correct.

This scheme gives us an advantage: if an edge is present in only cuts of large sizes, we can keep it with low probability,which corresponds to setting w_e to be large. On the other hand, if an edge is present in cuts of small size, we will keep it with high probability,which corresponds to setting w_e to be small. In this way,we can approximate cut problems while throwing away more edges which are present in only cuts of high sizes.

Thus, a natural choice for w_e would be the size of the smallest cut containing e. Unfortunately, we do not know w_e; however, it is possible to approximate it quickly. The final result is an multiplicative approximation based on this scheme.

2 Basic Properties of Sparsifiers

Theorem (Batson, Spielman, Srivastava): There exists a (non-streaming) algorithm A that constructs a $(1 \pm \epsilon)$-sparsifier with only $O(\frac{n}{\epsilon^2})$ edges.

Spielman and Teng[1,2] introduced the notion of sparsification and proved that $(1 + \epsilon)$-approximations with O(n/ϵ^2) edges could be constructed in O(m) time.

Spielman and Srivastava [3] proved the existence of spectral sparsifiers with O (n log n/ϵ^2) edges, and showed how to construct them in O(m) time. They conjectured that it should be possible to find such sparsifers with only O (n/ϵ^2) edges.

Lemma 1: Suppose H_1 and H_2 are α sparsifiers of G_1 and G_2. Then, $(H_1 \cup H_2)$ will be a α sparsifier for $(G_1 \cup G_2)$.

Lemma 2: Suppose J is an α sparsifier of H and H is an α sparsifier of G, then J is α^2 sparsifier of G.

Proof: Since J is an α sparsifier of H, we can get J from H in $(1+\epsilon)$ approximation. Similarly, H can be obtained from G in $(1+\epsilon)$ approximation. Hence, J can be obtained from G in $(1+\epsilon)^2$ approximation.

3 Stream Sparsification

The concept of sparsification can be extended to data stream model. Suppose we have a continuous data stream, we can divide it into segments G_1, G_2, ... such that each segment has O(nϵ^{-2}) edges.

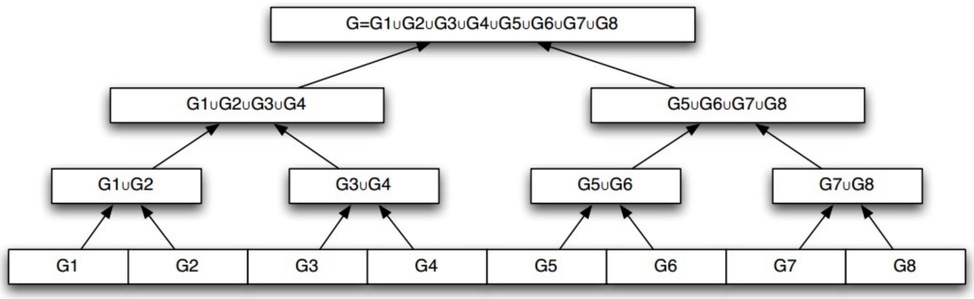

Figure 2: Binary Tree over Segments[4]

From Batson, Spielman, Srivastava theorem[3], we know that there exists an offline algorithm A returning (1+γ) sparcifier with O(nϵ^{-2}) edges. Hence, if we recurcively apply this (1+γ) sparcifier at each of the $\log m$ levels, we will get $(1+\gamma)^{\log m}$ sparcifier for an entire graph G.

At any point of time, we have O($\frac{n}{\gamma^2}$) edges. Hence, the total space required is O(n log m/ ϵ^2) = O(n log n/ ϵ^2).

References

[1] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In STOC '04, pages 81-90, 2004.

[2] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. CoRR, abs/0808.4134, 2008. Available at http://arxiv.org/abs/0808.4134.

[3] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In STOC '08, pages 563-568, 2008.

[4] http://people.cs.umass.edu/ mcgregor/slides/epit-2.pdf

[5] http://www.win.tue.nl/ nikhil/courses/2WO08/Nick-sparsification.pdf