CS369G: Algorithmic Techniques for Big Data Spring 2015-2016

Lecture 13: Sketching for Connectivity, k-Connectivity, and Min-Cut

Prof. Moses Charikar *Scribe: Xiuyuan Lu*

1 Overview

In this lecture, we continue our discussion of graph streaming algorithms. We consider the setting where edges can both arrive and depart in a stream, and we are asked to answer some queries at the end of the stream. We will look at the connectivity problem, the k-connectivity problem, and the min-cut problem.

2 Connectivity

Given a stream of edge insertions and deletions, we want to answer queries about the connectivity of the resulting graph at the end of the stream. We first consider a simple non-streaming algorithm that constructs the connected components of a graph. We then show how to use linear sketches to support the operations needed by this algorithm.

2.1 Building connected components

Given a graph, we can construct the connected components in $O(\log n)$ stages. In the first stage, we arbitrarily pair up adjacent nodes, and consider these node-pairs as "supernodes." In each of the following stages, we arbitrarily pair up adjacent supernodes and merge them into new supernodes. We stop when there is no edge between any of the supernodes. It is easy to see that this process terminates in $O(\log n)$ stages, and we obtain the connected components of the graph in the end.

2.2 Sketches for connectivity

Since the graph stream involves both edge insertions and edge deletions, we consider using a linear sketch to represent the graph, as edge deletions can be easily handled by the linearity of the sketch. To support the above algorithm, we need to be able to give some cut edge for each supernode. We consider the following graph representation. For each node $i \in V$, let $a^i \in \{-1, 0, 1\}^{\binom{n}{2}}$ be a vector where

$$a^i_{\substack{(j,k)\in E \\ j<k}} = \begin{cases} 1 & \text{if } i = j < k, \\ -1 & \text{if } j < k = i, \\ 0 & \text{otherwise.} \end{cases}$$

We observe that if S is the set of vertices in a supernode, then the nonzero entries of the sum $\sum_{i\in S} a^i$ correspond exactly to the cut edges of the supernode. Thus, we would like to be able to return

some nonzero entry of $\sum_{i \in S} a^i$ in order to build connected components. To do this, we can use the ℓ_0-sampling discussed in the previous lectures. We maintain a ℓ_0-sketch for each a^i, $i \in V$. Then, for any $S \subseteq V$,

$$\ell_0\text{-sketch}\left(\sum_{i \in S} a^i\right) = \sum_{i \in S} \ell_0\text{-sketch}(a^i)$$

by the linearity of the sketches. To obtain a cut edge of a supernode S, we perform an ℓ_0-sampling on the sketch $\sum_{i \in S} \ell_0\text{-sketch}(a^i)$. Note that each ℓ_0-sketch uses polylog space. Therefore, this algorithm needs $O(n \operatorname{polylog} n)$ space in total.

3 k-Connectivity

We define a graph to be k-connected if every cut of the graph has size at least k. We want to know, at the end of the graph stream, whether the resulting graph is k-connected. In the previous section, we learned how to build a spanning forrest in a streaming model. For k-connectivity, we leverage this to construct k spanning forests. For $i = 1, \ldots, k$, let F_i be a spanning forest of $(V, E \setminus \bigcup_{j=1}^{i-1} F_j)$. We call $H = F_1 \cup \cdots \cup F_k$ the k-skeleton of the graph.

We observe that for any cut $S \subseteq V$, $|E_G(S, \overline{S})| \geq k$ implies $|E_H(S, \overline{S})| \geq k$. The reason is that if every F_i contains a cut edge of S, then $|E_H(S, \overline{S})| \geq k$. On the other hand, if some F_i does not contain any cut edge of S, it must be the case that $F_1 \cup \cdots \cup F_{i-1}$ has included all the cut edges already. Thus, $|E_H(S, \overline{S})| = |E_G(S, \overline{S})| \geq k$. Therefore, we see that G is k-connected if and only if H is k-connected.

One last note is that we need to keep k independent sketches of the graph in order to construct the k-skeleton. Let $\mathcal{A}(G)$ denote the linear sketch for connectivity discussed in the previous section. For k-connectivity, we need to keep k independent sketches, $\mathcal{A}^1(G), \ldots, \mathcal{A}^k(G)$. Then, for $i = 1, \ldots, k$, we construct the spanning forest F_i from

$$\mathcal{A}^i(G - F_1 - \cdots - F_{i-1}) = \mathcal{A}^i(G) - \sum_{j=1}^{i-1} \mathcal{A}^i(F_j).$$

4 Min-Cut

4.1 Sparsification via sampling

We first give a brief review of graph sparsification by random sampling. Consider the following generic graph sampling algorithm.

1. Sample edge e with probability p_e.
2. If e is sampled, set its weight to $\dfrac{1}{p_e}$.

The expected weight of each edge is equal to 1. It turns out that if p_e is sufficiently large, cut sizes are well preserved within $(1 \pm \epsilon)$ with high probability. Such graphs are called cut sparsifiers. Here is a list of useful results.

- Karger in [2] showed that if
$$p_e \geq \min\left(1, \frac{c_1 \log n}{\lambda \epsilon^2}\right),$$
where λ is the size of the min-cut, then the resulting graph is a cut sparsifier with high probability.

- Fung et al. in [1] showed that if
$$p_e \geq \min\left(1, \frac{c_2 \log^2 n}{\lambda_e \epsilon^2}\right),$$
where λ_e is the size of the min-cut that separates the two end points of e, then the resulting graph is a cut sparsifier with high probability.

- Spielman and Srivastava in [4] showed that if
$$p_e \geq \min\left(1, \frac{c_3 r_e \log n}{\epsilon^2}\right),$$
where r_e is the effective resistance of edge e, then we obtain a spectral sparsifier with high probability.

4.2 Min-Cut

We will use Karger's result together with our algorithm for k-connectivity from the previous section to solve the min-cut problem. Karger's result states that if the sampling probability is sufficiently large, then the size of the min cut should be well preserved with high probability. However, ut presupposes knowledge of the min-cut value λ. To get around this, we will create $\log(n)$ sketches, one for each possible value (up to a factor of 2) of λ (or equivalently of the correct sampling probability), and try to recover the correct value by looking at the sketches.

Concretely, we consider a sequence of graphs where the sampling probability decreases geometrically. Let $G_0, G_1, G_2 \ldots$ be a sequence of graphs where G_t is G with edges sampled with probability $\frac{1}{2^t}$. When the sampling probability becomes as low as $\Theta\left(\frac{\log n}{\lambda \epsilon^2}\right)$, the size of the min-cut in the resulting graph is $\Theta\left(\frac{\log n}{\epsilon^2}\right)$ with high probability, according to Karger. Thus, there exists a t such that $\text{mincut}(G_t) = \Theta\left(\frac{\log n}{\epsilon^2}\right)$. We let $k = \Theta\left(\frac{\log n}{\epsilon^2}\right)$, and run k-connectivity on $G_0, G_1, G_2 \ldots$. Then, we find the minimum index j such that G_j is not k-connected,

$$j = \min_t \{\text{mincut}(G_t) < k\}.$$

This means that at G_j, the sampling probability has dropped to $\Theta\left(\frac{\log n}{\lambda \epsilon^2}\right)$ with high probability. Thus, our estimate for the size of the mincut is $2^j \, \text{mincut}(G_j)$.

References

[1] W. S. Fung, R. Hariharan, N. J. A. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *ACM Symposium on Theory of Computing*, pages 71-80, 2011.

[2] D. R. Karger. Random sampling in cut, flow, and network design problems. In *ACM Symposium on Theory of Computing*, pages 648-657, 1994.

[3] A. McGregor. Graph stream algorithms: a survey. In *ACM SIGMOD Record*, 43(1):9-20, 2014.

[4] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(4):981-1025, 2011.