Data Mining Assignment 3

Name : Rigved S. Patki
E-mail : patki@kth.se
Group: Assignment group 8

Topic : Mining Data Streams

Solution :
For this assignment I am going to implement the algorithm in the following paper :

- M. Jha, C. Seshadhri, and A. Pinar, A Space-Efficient Streaming Algorithm for Estimating Transitivity and Triangle Counts Using the Birthday Paradox ., ACM TKDD, 9-3, 2015

I have implemented this algorithm based on the Section 2 in the above mentioned paper. This algorithm was implemented using Typescript which is transpiled to Javascript to be executed in the Node js environment. The following are the steps I used to implement the algorithm.

- The datasets used in this implementation is from SNAP (http://snap.stanford.edu) . These datasets were mentioned in the paper in Section 4 (Experimental results). The list of the datasets used is as follows:
    - Amazon0312
    - Amazon0505
    - Amazon0601
    - email-Eu-core , etc.
- In order to implement this algorithm in a short period of time we used email-Eu-core as main sample set because it had the least number of edges.
- As most of these datasets were very large in size (eg: 42MB) I stored these datasets on Google Cloud Storage. The details of which are as follows :
    - Project name : data-mining-assignments
    - Region: europe-west1
    - Bucket name : data-mining-assignment-3
- The code can be executed on both the Google Compute Engine by the name of of data-mining-assignment-3.
- Now let's discuss the main objective of this paper. A mentioned in the paper the main objective of this paper is to "**design a space efficient algorithm that approximates the transitivity (global clustering coefficient) and total**

triangle count with only a single pass through a graph given as a stream of edges”

- This paper was based on the concept of classic probabilistic result called the birthday paradox. Where we assume that if the transitivity coefficient is constant and the number of edges is more than the number of wedges in the graph then this algorithm can prove that the required space is O(√ n) (n is the number of vertices) to provide accurate estimates of transitivity/number of triangles of a graph.

Optional bonus task :

1. What were the challenges you have faced when implementing the algorithm?

Answer : One of the biggest challenges was to interpret the mathematical aspect of the algorithm into the code. The authors of this paper did not explain clearly how they implemented the algorithm. From a programmer’s perspective it is difficult to interpret how pseudo code will actually be implemented due to its vagueness. Another challenge that I was face was whether or not to implement the incoming data synchronously or asynchronously. In order to be true to the algorithm I would have needed to use streams but that would then create an overhead of getting reservoir sampling of edges and wedges beforehand. I wish they would have concentrated more on how to implement reservoir sampling for wedges.

2. Can the algorithm be easily parallelized? If yes, how? If not, why? Explain.

Answer : Yes, it can be parallelized as only minority of the new edges will be inserted to the array of reservoir edges, so we can use parallelized algorithm to filter the edges which can not be inserted to the array of reservoir edges but only one thread can update the array of reservoir at a certain time.

3. Does the algorithm work for unbounded graph streams? Explain.

Answer: Yes, it can support unbounded graph streams as we do not need to care about the total length of the graph streams, we just need to keep the sample with the fixed size and the sample contains each element seen so far with probability s/n

4. Does the algorithm support edge deletions? If not, what modification would it need? Explain.

Answer: No this algorithm does not support edge deletion as it uses a single pass over the dataset. In order to support deletion of edges there needs to either multiple passes to detect the deleted edge and update the reservoir sample of edges as well as wedges. This would consume more cpu and memory and would basically go against the main goal of this algorithm.