ID2222 Assignment 3: Mining Data Streams
TRIEST: Counting GLobal Triangles

Akash Singh, Lionel Kusch

We have selected the paper 2: TRIÈST: Counting Local and Global Triangles in Fully-Dynamic Streams with Fixed Memory Size.

We implement the first two algorithms in the paper :

- TRIEST-BASE
- TRIEST-IMPR

Notes:

1) We assume working with directed graphs only as the paper also mentions delaing with undirected graphs in section 2 (Preliminaries). To handle directed graphs we process every edge (u,v) such that:
    i) u < v
    ii) u != v
    iii) Discard edges (u,u)
    iv) If edge (u,v) is already present in the current edge sample S, we discard (u,v) or (v,u).
    v) Evaluation with Advogato dataset, a directed graph gave good results with this handling.

2) In evaluation we compare only the global triangles count as the test datasets provided the true value of only global triangle counts. However the code outputs the values of local Ts and Eta for Triest-Base, and the local triangle estimates for Triest-Impr.

Execution Instructions:

1) Run scripts triest_impr.py for Triest-Impr algorithm and triest_base for Triest-Base algorithm.
2) The desired dataset and values of M can be set in variables datafile and M respectively in the main() method of either script.
3) The code assumes that the first two columns of the datafile are in the form:
    <source_node> <destination_node>
4) The code works with space separated files and works with only the first two fields of each row and ignores the rest.,

1) MAIN TASKS:

- Implementing Reservoir Sampling

This has been implemented as part of the method *sample_edge()*. The coin flip has been simulated by method .

- Implementing Triest-Base

This has been implemented in script *triest_base.py*.

- Implementing Triest-Impr
This has been implemented in script *triest_impr.py.*

- Implementing Edge Sample S
The class *edgestore* has been used as a utility to implement edge sample S as a adjacency list using python dictionaries.

Evaluation:

We evaluate the algorrithms on different datasets. The different datasets are:

1) Advogato: a directed graph dataset
2) Hamsterster Friendships: an undirected graph dataset.

Advogato Dataset
In this dataset there are 6541 vertices, 51,127 edges and and a triangle count of 98,300.
We run the two algorithms for different values of M:
3000, 6000, 9000, 12000, 15000, 18000, 21000, 24000, 27000, 30000, 40000

Both algorithms converge to the correct value at M =40,000 but give good estimates even with much lower values of M.

Also as can be seen from Figure 1 and Figure 2, the Triest-Impr algorithm has lower variance

Triest-Base Results: The detailed results including local_T and Eta can be seen here

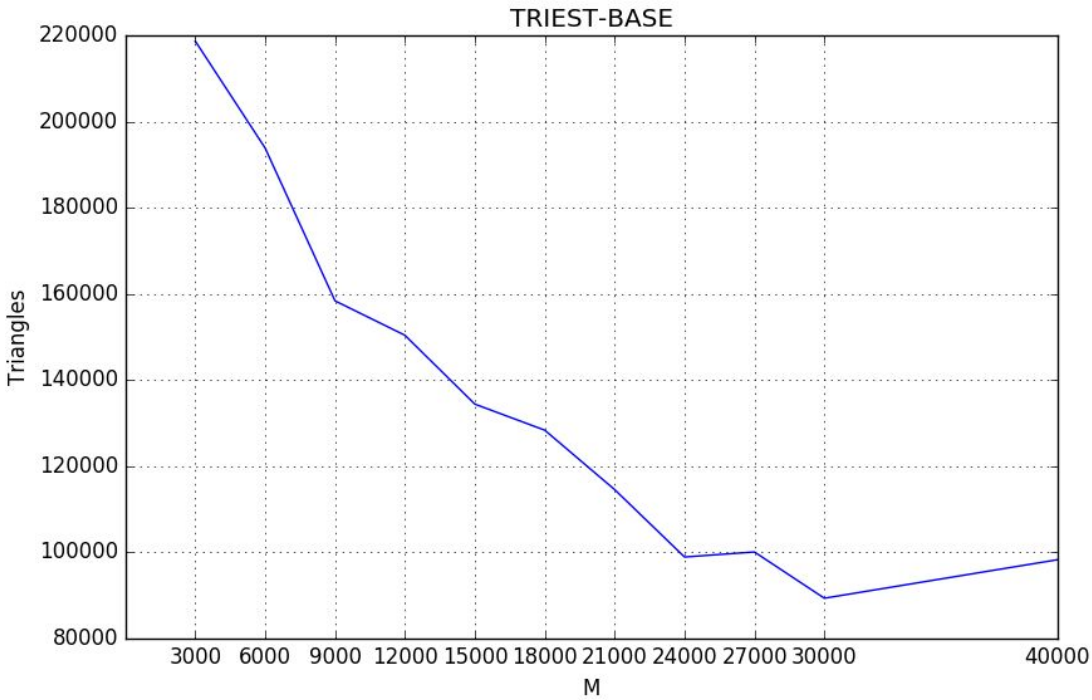

Figure 1: Triangle Count vs M for Advogato dataset with Triest-Base

Triest-Impr Results: The detailed results including local triangle estimates can be seen here

Figure 2: Triangle Count vs M for Advogato dataset with Triest-Impr

Hamsterster Dataset

In this dataset there are 1,858 edges and 12,534 vertices, and a triangle count of 16,750.

The results of the two algorithms are shown in figures 3 & 4. Triangle counts have been calculated for values of M in:

M = 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000.

Both algorithms converge almost to the exact value for M =4000, but start providing good estimates for much lower values.

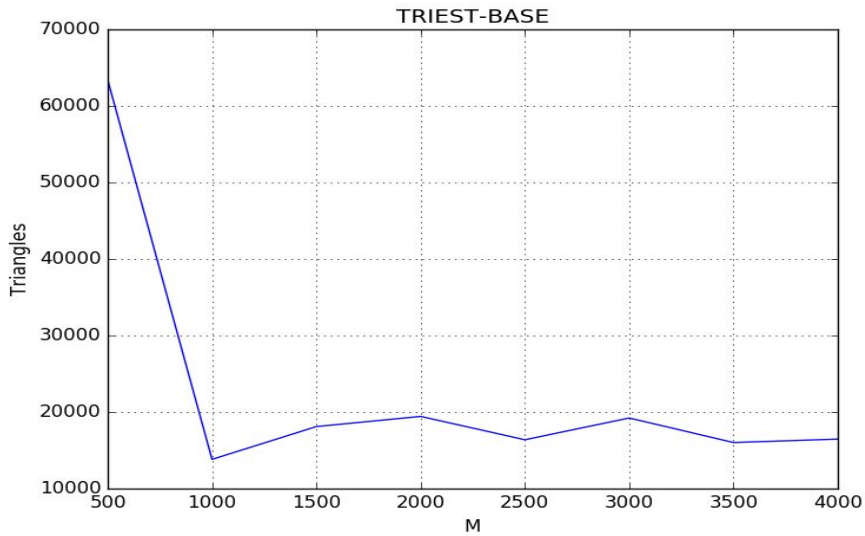

Figure 3: Triangle Count vs M for Hamsterster dataset with Triest-Base

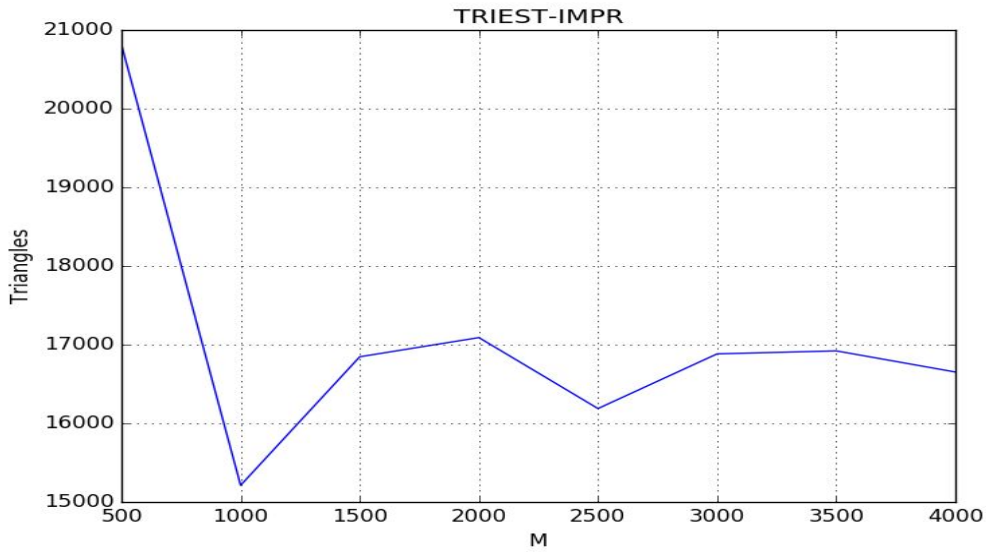

Figure 4: Triangle Count vs M for Hamsterster dataset with Triest-Impr

2) BONUS TASKS

i) Challenges Faced:

There were some challenges faced in the assignment:

- 1) In the current implementation there were some problems in evaluation as the dataset we chose was a directed graph. We did not realize it initially. This caused our implementation to break as we had assumed undirected graphs. However we fixed this by taking the approach described at the start of this document
- 2) We wanted to implement the first two algorithms using a streaming framework but ran into problems with parallelizing the algorithm
- 3)We tried to implement the Triest-FD , the third algorithm presented in the paper. Though had some problems in implementing mainly relating understanding the reasoning behind the algorithm and in approximating Binomials with hypergeometric distribution. So we decided to implement the first two algorithms.

ii) Parallelizable:

It would be difficult to parallelize the algorithm as it depends on a global counter for estimating the global triangles in the graph. Moreover for estimating local triangle counts, it maintains a number of local counters. Both the local and global counters need information about the neighbourhood of the vertices of the edge being processed. Thus a lot of communication would be needed between different workers for updating and synchronizing the various counters. This would also depend on the partitioning of the graph across different nodes on the cluster.

Edge-Cut partitioning in particular would not be suitable as the neighbourhood of the 2 vertices in an edge could potentially get partitioned across many different machines.

Nevertheless it may still be possible to parallelize the algorithm using a suitable vertex-cut partitioning algorithm, but it would not be straightforward

iii) Unbounded Graphs:
Yes the algorithms would work for unbounded graph streams.

Algorithms which require having fixed edge sampling probability p, not only require having some knowledge of the size of the stream, but also implies that the sample size will grow with the size of the stream. This can either result in under-utilization of available memory or running out of memory. Moreover a lower p results in poor approximation of the global triangle count.
On the other hand having a fixed memory size of M and using reservoir sampling(or random pairing) on the other hand ensures that the Triest algorithms can work with unbounded streams and allows them to utilize the available memory as much as possible.
As Theorem 1 (&4) show that Triest algorithms give unbiased estimates and the expected value of the estimates(& Triest-Impr) is the global triangle count.
The paper also provides good concentration bounds for the variance of the estimates given by Triest (Theorems: 2,3,5,6)
Thus the algorithm can work with an unbounded graph streams. Though a larger M gives more accurate estimates, which can also be seen from our evaluation.

iv) Edge Deletions
The Triest-Base and Triest-Impr algorithms are insertion only algorithms. However the Triest-FD algorithm can handle fully dynamic streams including edge deletions. The main modification required is to use *Random Pairing(RP)* instead of reservoir sampling. RP scheme extends reservoir sampling to handle edge deletions compensating edge deletions by future edge insertions. Thus the algorithm works by maintaining counters to keep track of uncompensated edge deletions.