Homework 3: Mining Data Streams

Group 28: Junjie shan, Yuxin Meng

December 2021

1 Lab

Our lab is written using Jupyter, see details in **TRIEST_BASE.ipynb** and **TRIEST_IMPR.ipynb**.

2 Bonus questions

2.1 What were the challenges you have faced when implementing the algorithm?

The paper *TRIÈST: Counting Local and Global Triangles in Fully Dynamic Streams with Fixed Memory Size* is specific for undirected graphs, so we encountered some problems when we try to estimate the triangles for a directed graph. We fixed this problem by converting directed graph to undirected graph and removing self-referencing edges and same edges in the undirected graph.

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

This algorithm can be parallelized, but might not be easily parallelized. We think the algorithm need to be changed with strict synchronization between threads, which requires the algorithm to update counter while taking the graph or edge participation into account, since two node of one edge can be distributed into two threads. Or there might be a way running several threads with original algorithm and estimate the global or local triangles using the output of all threads, but the estimate algorithm should be carefully designed and formally proved. Anyway, this algorithm is hard to be parallelized.

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

Yes, TRIÈST is designed for estimating global and local triangles of infinite stream. This algorithm making use of reservoir sampling stores a user-specified, fixed amount M of edges. This is different from other approaches, which require hard-to-choose parameters and offer no guarantees on the amount of memory they use. With TRIÈST, there won't be the situation of running out of memory, so this algorithm can work for unbounded graph streams.

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

No. We used TRIÈST_BASE and TRIÈST_IMPR algorithms which don't support edge deletions. We could adopt the TRIÈST_FD to support it. The TRIÈST_FD is using random paring, which works on the idea that edge deletions will be "compensated" by further edge insertions later in the stream, and it keeps two counters to keep track of the numbers of uncompensated edge deletions involving an edge that was (resp. was not) in edge sample at the time the deletion for this edge was on the stream[1].

References

[1] Lorenzo De Stefani et al. "TRIÈST: Counting Local and Global Triangles in Fully Dynamic Streams with Fixed Memory Size". In: *ACM Trans. Knowl. Discov. Data* 11.4 (June 2017). ISSN: 1556-4681. DOI: 10.1145/3059194. URL: https://doi.org/10.1145/3059194.