Homework 3: Mining Data Streams

Konstantin Sozinov, sozinov@kth.se
Kim Hammar, kimham@kth.se

November 17, 2017

1 Solution

We implemented the TRIÈST-IMPR algorithm[1] for estimating triangle counts on the Euroroad graph dataset [2]. The graph is undirected, nodes represent cities and an edge between two nodes denotes that they are connected by an E-road. The algorithm estimates both global and local triangle counts. The dataset is bounded but we process it in a streaming fashion by reading edge by edge and applying the reservoir sampling.

2 Questions

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

   One problem that we encountered was about choosing the right streaming graph processing platform. To the best of our knowledge Apache Flink does not support streaming graph processing. We tried to use plain Apache Flink and stream every event as an edge in our graph but it was not clear to us how our TRIÈST-IMPR counters were updated since Flink uses updated counters in parallel way. Flink streaming typically considers the data as unbounded and if using this approach we would generate estimates per window rather than a global estimate of the triangle count. Since our dataset in this were bounded it would over-complicate things to use Flink so we simply streamed the edges ourself in a non-parallel fashion.

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

   Yes it can be parallelized, different streaming nodes can run the TRIÈST-IMPR algorithm in parallel and maintain local estimates. When querying the stream or materializing the results, the local estimates have to be merged to create the final estimate. This can for example be done in Flink-Streaming, where the stream of edges can be split uniformly among a set of nodes and each node updates its local sample and estimates.

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

   Yes, since the algorithm uses reservoir sampling it is meant to be used for unbounded graph streams. The difference if the stream of edges is unbounded is that the mindset have to be shifted. With a unbounded stream we cannot wait until all edges have been received to materialize the estimates but rather some form of windowing strategy should be applied to construct rolling estimates for given time periods. What notion of time to use depends on the characteristics of the stream, for instance if the edges are time-stamped we could use event-time, otherwise we could use processing-time. Data-driven windows are also a possibility.

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

   TRIÈST-IMPR does not support edge deletions. Our stream for this lab did not contain any edge deletions so we did not implement the edge deletion part. To extend the algorithm for edge deletion, two counters, d_i and d_o should be maintained to keep track of how many times edges have been removed versus in the stream. This is necessary since the stream

[1] http://www.kdd.org/kdd2016/papers/files/rfp0465-de-stefaniA.pdf
[2] http://konect.uni-koblenz.de/networks/subelj_euroroad

might arrive unordered. The reservoir should only keep those edges that have been inserted more times than deleted. Effectively the counters works like a sort of tombstone.

3 How to run

Clone this repository and navigate to *mining_ data_ streams* project. Then use:

```
sbt compile //compile
sbt test //test
sbt run //run
sbt assembly //generate fat jar
```

4 Evaluation and results

Number of Edges in a Sample, M	Estimated Number of Triangles	Actual Number of Triangles
350	20	32
750	28	32
1000	30	32
1100	32	32

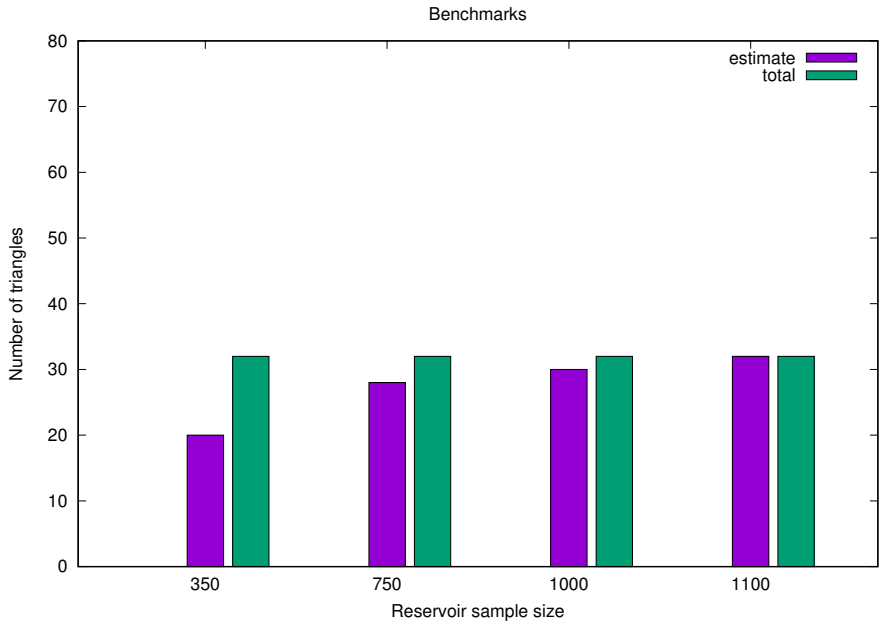

Figure 1: Estimate accuracy as sample size increases, when sample size is 1100 the estimate is correct (32).

The number of total edges in the graph we used is 1417, total number of vertices is 1,174 and average degree is ≈ 2.4. As we increase number of edges in the sample the precision of our implementation gets better. This is based on the second implementation of the TRIÈST-IMPR algorithm. The intuition behind this is that the algorithm was meant to be use at the very large graphs (number of edges order of 10^9) and precision gets better if number of edges in the reservoir increases. As we saw in the actual paper, in order to estimate number of triangles for the Twitter network graph (contains billions of edges), the authors choose very high M, $M = 10^6$.