Data Mining, ID2222 HT201

Homework Assignment 3

Frano Rajic, Ivan Stresec

November 23, 2020

1 Objective

Our objective in this homework was to study and implement a streaming algorithm used in one of three research papers:

- 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.
- L. De Stefani, A. Epasto, M. Riondato, and E. Upfal, TRIEST: Counting Local and Global Triangles in Fully-Dynamic Streams with Fixed Memory Size, KDD'16.
- P. Boldi and S. Vigna, In-Core Computation of Geometric Centralities with HyperBall: A Hundred Billion Nodes and Beyond, ICDMW'13.

We had to decide on a paper and then implement and test first the reservoir or the Flajolet-Martin algorithm and then the paper's streaming graph algorithm which uses it.

2 Chosen paper

We decided to implement the TRIEST algorithm from TRIEST: Counting Local and Global Triangles in Fully-Dynamic Streams with Fixed Memory Size by L. De Stefani, A. Epasto, M. Riondato, and E. Upfal. It uses the reservoir algorithm to maintain a sample of edges from a stream. The main idea is to, along with the edges, maintain counters which are used in estimating the local and global numbers of triangles. All estimates are unbiased and have relatively low variance, which we will observe experimentally using our implementation.

3 Implementation

For this assignment, we implement the TRIEST-BASE algorithm and its improved version TRIEST-IMPR. There also exists a fully-dynamic version of the algorithm which we will not implement here.

3.1 Reservoir sampling

All the version of the TRIEST algorithm are based on reservoir sampling, a method which produces a uniformly random samples of the so-far seen data. It uses a fixed-size container, the size is a parameter to be decided and we will denote it as M.

Our implementation was pretty straight forward, we read edges of the graph one by one, increasing our sample (or time) counter t by 1 each step. While the container of fixed size M is not full we simply remember all the incoming edges. Once the container is full ($t > M$), we either remember the new edge and uniformly pick an edge which we will remove, or simply ignore the new edge. The first is done with probability M/t, and the second otherwise.

3.2 TRIEST-BASE

As mentioned, TRIEST algorithms build on top of the reservoir sampling method. The algorithm works on insertion only streams and provides an unbiased and relatively low variance estimate of the number of local and global triangles. Being built on the reservoir technique, the algorithm also returns true values as long as $M > t$.

Several counters are kept and are used for estimating. We have one global counter τ which is used to estimate the number of global triangles, and at any time we also keep counters for all vertices which are observed to be triangles in our current sample. These counters are increased or decreased as we add and remove edges from our fixed-size reservoir. The estimates can be calculated at any moment by multiplying some counter with $\xi = max(1, \frac{t(t-1)(t-2)}{M(M-1)(M-2)})$.

3.3 TRIEST-IMPR

TRIEST-IMPR is an improved version of the base algorithm. In this version, counters aren't used as a sort of average of the stream, but the number of triangles themselves. The following is achieved by increasing counters regardless of the edge being added to the reservoir or not, and by never reducing the counters when removing edges. The counts are also not increased by 1 as in

the base algorithm, but by a weight $\eta = max(1, \frac{(t-1)(t-2)}{(M-1)(M-2)})$. This variant is proven to have lower variance while remaining unbiased, and does so using approximately the same amount of resources.

4 Testing and evaluation

To test our implementation, we took 5 datasets from SNAP, estimated the global number of triangles, and then compared it to the real number of triangles. The summary of the used datasets is shown in table 1

Dataset name	Nodes	Edges	Triangles
Friendster	65 608 366	1 806 067 135	4 173 724 142
Amazon	334 863	925 872	667 129
DBLP	317 080	1 049 866	2 224 385
Youtube	1 134 890	2 987 624	3 056 386
Facebook	4039	88 234	1 612 010

Table 1: These datasets were used. Notice that Friendstar is the by far largest dataset, with a filesize of $31GB$, whereas the Facebook dataset is the smallest.

Table 2 shows the results of running TRIEST-BASE and TRIEST-IMPR on the 5 datasets. The results are very close to the ground truth. TRIEST-IMPR showed better results in that the yielded result was closer to the ground truth than was the case for TRIEST-BASE. It also gave consistantly lower variance.

Increasing the value of the parameter M, which determines the number of buckets used in reservoir sampling, gives results closer to the ground truth. When m is larger than the total number of edges, then there is no variance and the result is precisely equal to the ground truth. This example can be seen for both TRIEST-BASE and TRIEST-IMPR for the *Facebook* dataset and value of M equal to 100 000, as the dataset has only 88 234 edges.

Dataset name		M	Global triangles	Best time
Friendster *ground truth:* 4 173 724 142	Impr.	10 000	814 269 856.18	-
		100 000	4 444 248 139.44	-
		1 000 000	4 092 690 331.18	-
Amazon *ground truth:* 667 129	Base	10 000	714 536.23 ± 95 050.05	4.7966
		50 000	705 474.87 ± 8178.73	6.9082
		100 000	673 228.89 ± 4011.94	8.2966
	Impr.	10 000	687 216.10 ± 3463.12	5.6954
		50 000	666 914.72 ± 1015.81	6.4025
		100 000	668 094.54 ± 643.85	8.7697
DBLP *ground truth:* 2 224 385	Base	10 000	2 430 804.14 ± 1 151 077.12	5.0659
		50 000	2 185 809.98 ± 160 320.32	7.2415
		100 000	2 253 441.38 ± 67 275.69	8.441
	Impr.	10 000	2 141 017.67 ± 117 862.58	6.3098
		50 000	2 238 330.06 ± 58 558.50	7.2527
		100 000	2 234 617.35 ± 38 895.86	10.4615
Youtube *ground truth:* 3 056 386	Base	10 000	8 002 559.95 ± 18 004 387.66	14.1223
		50 000	2 837 560.10 ± 675 047.87	10.4693
		100 000	2 928 146.16 ± 261 835.74	15.3155
	Impr.	10 000	2 975 988.73 ± 320 489.40	16.1659
		50 000	3 064 527.73 ± 102 093.39	20.2397
		100 000	3 026 584.26 ± 51 423.85	14.6142
Facebook *ground truth:* 1 612 010	Base	10 000	1 617 789.94 ± 59 342.88	0.905
		50 000	1 610 232.22 ± 11 807.07	4.2985
		100 000	1 612 010.00 ± 0.00	3.8228
	Impr.	10 000	1 615 253.49 ± 16 739.94	1.5413
		50 000	1 609 853.77 ± 3672.33	4.492
		100 000	1 612 010.00 ± 0.00	4.537

Table 2: Results of running TRIEST-BASE (*Base* in table) and TRIEST-IMPR (*Impr.* in table) on different datasets and for different values of the parameter M (used for reservoir sampling). Number of resulting global triangles are shown, with average and standard deviations on 10 runs. Also, the best (shortest) time the algorithm needed is shown in the last column (except for Friendster, which we did not measure at the moment of recording, but which took about 1 hour). For convenience, ground truth number of global triangles are shown below dataset names.

5 Bonus questions

5.1 Challenges

The implementation of the described algorithms wasn't especially challenging. The pseudo-code of the algorithm is more-or-less self-apparent and was easily implemented in Python. There were minor bugs we had to work out, as well as some slight complications when using a dictionary (hash table) to remember neighbourhoods of vertices, but nothing we couldn't solve with relative ease. By far the hardest part was creating the big results table in latex.

5.2 Parallelization

The two TRIEST algorithms used in this assignment were not designed to be run in parallel. Both run sequentially for each new edge and processing of new edges includes updating the reservoir, a shared data structure also designed to be used in a sequential manner.

For the purpose of parallelization, a complete redesign of the algorithm would be needed, which, if possible, is probably not simple at all and would need major changes at its core.

5.3 Unbounded graph streams

The TRIEST algorithm has no problems with unbounded graph streams, as the algorithm can produce unbiased estimates of the number of triangles at any moment. However, one challenge that would be faced with really big streams is choosing an appropriate value of parameter M, as a smaller value would give much higher variance and thus less accurate results.

5.4 Edge deletion

Both TRIEST-BASE and TRIEST-IMPL have the assumption of an insertion-only stream and will not work for edge deletion.

The original paper, however, proposes an algorithm called TRIEST-FD (fully dynamic TRIEST) that computes unbiased estimates of the global and local triangle counts in a fully-dynamic stream, where edges are inserted/deleted in any arbitrary, adversarial order. It is based on random pairing, which is a sampling scheme that extends reservoir sampling and can handle deletions. The idea behind the random pairing scheme is that edge deletions seen on the stream will be âcompensatedâ by future edge insertions.