Nora Al-Naami
Mohammadmehdi Satei

Introduction and problem definition

In this lab we use Triest to implement streaming graph algorithms in order to find the count of triangles in an undirected graph represented as streamed edges. Triest algoirthms use reservoir sampling which means ” Definition 1. The first step of any reservoir algorithm is to put the first n records of the file into a “reservoir.” The rest of the records are processed sequentially; records can be selected for the reservoir only as they are processed. An algorithm is a reservoir algorithm if it maintains the invariant that after each record is processed a true random sample of size n can be extracted from the current state of the reservoir.” [1]

Method

The data chosen for this lab is Contiguous USA with 49 nodes and 107 edges. The data is undirected and unweighted. We downloaded the data as TSV file and then uploaded it as a list where each element is a tuple of two vertices u and v. Each element of the list is seen a stream whose elements are edges.

The input of the Triest algorithms implemented in this is M which is the fixed amount of memory in order to maintain a sample of edges for computing the count of traingles in of the graph stream. We also use an edge sample S with the size M. Therefore, the algorithms uses the space O(M) but in the following case we might use a bigger space than O(M) ” If instead we are interested in estimating the local triangle counts, at any time t trièst maintains (non-zero) local counters only for the nodes u such that at least one triangle with a corner u has been detected by the algorithm up until time t. The number of such nodes may be greater than O(M)” [2].

The triest algorithms used in this lab are triest base and triest, both of them work for insertion-only streams while edge deletions are not allowed. For the implementation of the algorithms we use a global counter to save the estimate of the number of triangles of the graph.

The first algorithm which we implemented is triest base which uses standard reservoir sampling [2]. The pseudocode for the algorithm is shown from fig 1. Accroding to the algorithm we add an edge = (u, v) at time t if t<=M in sample S of size M. Otherwise if t>M we flip a biased coin with heads having the probability M/T. If we get heads we remove a randomly chosen edge= (w, z) from S and insert the new edge (u, v) otherwise we change nothing. After every insertion or removal of an edge we update the global and local counters for each trangle found in the sample so far shown by the algorthm as $\mathcal{N}^{\mathcal{S}}_{u,v}$.

The second algorithm shown in fig 2 that we use which improves the the estimations of triangle count, lowering the variance is triest impr. The biggest difference between triest base and triest impr is that triest impr does not decrement the the counters when an edge is removed from S and it performs a weighted increase of the counters $\eta^{(t)} = \max\{1, (t-1)(t-2)/(M(M-1))\}$

Nora Al-Naami
Mohammadmehdi Satei

Algorithm 1 TRIÈST-BASE

Input: Insertion-only edge stream Σ, integer $M \geq 6$

1: $\mathcal{S} \leftarrow \emptyset$, $t \leftarrow 0$, $\tau \leftarrow 0$
2: **for each** element $(+, (u, v))$ from Σ **do**
3: $t \leftarrow t + 1$
4: **if** SAMPLEEDGE$((u, v), t)$ **then**
5: $\mathcal{S} \leftarrow \mathcal{S} \cup \{(u, v)\}$
6: UPDATECOUNTERS$(+, (u, v))$

7: **function** SAMPLEEDGE$((u, v), t)$
8: **if** $t \leq M$ **then**
9: **return** True
10: **else if** FLIPBIASEDCOIN$(\frac{M}{t}) =$ heads **then**
11: $(u', v') \leftarrow$ random edge from $\mathcal{S}$
12: $\mathcal{S} \leftarrow \mathcal{S} \setminus \{(u', v')\}$
13: UPDATECOUNTERS$(-, (u', v'))$
14: **return** True
15: **return** False

16: **function** UPDATECOUNTERS$((\bullet, (u, v)))$
17: $\mathcal{N}^{\mathcal{S}}_{u,v} \leftarrow \mathcal{N}^{\mathcal{S}}_{u} \cap \mathcal{N}^{\mathcal{S}}_{v}$
18: **for all** $c \in \mathcal{N}^{\mathcal{S}}_{u,v}$ **do**
19: $\tau \leftarrow \tau \bullet 1$
20: $\tau_c \leftarrow \tau_c \bullet 1$
21: $\tau_u \leftarrow \tau_u \bullet 1$
22: $\tau_v \leftarrow \tau_v \bullet 1$

fig 1. Shows the algorithm triest base which is an insertion-only algorithm that uses standard reservoir sampling. The estimation of triangle count for triest base is exact for t<=M if M is number of edges of the graph. In our data there are 107 edges and when M = 107 we get 57 triangle count which is the true value of triangle count of the graph.

1. UPDATECOUNTERS is called *unconditionally for each element on the stream*, before the algorithm decides whether or not to insert the edge into $\mathcal{S}$. W.r.t. the pseudocode in Alg. 1, this change corresponds to moving the call to UPDATECOUNTERS on line 6 to *before* the **if** block. MASCOT [27] uses a similar idea, but TRIÈST-IMPR is significantly different as TRIÈST-IMPR allows edges to be removed from the sample, since it uses reservoir sampling.
2. TRIÈST-IMPR *never* decrements the counters when an edge is removed from $\mathcal{S}$. W.r.t. the pseudocode in Alg. 1, we remove the call to UPDATECOUNTERS on line 13.
3. UPDATECOUNTERS performs a *weighted* increase of the counters using $\eta^{(t)} = \max\{1, (t-1)(t-2)/(M(M-1))\}$ as weight. W.r.t. the pseudocode in Alg. 1, we replace "1" with $\eta^{(t)}$ on lines 19–22 (given change 2 above, all the calls to UPDATECOUNTERS have $\bullet = +$).

fig 2. Shows the changes in algorithm 1 triest base implemented in order to compute triest impr.

Nora Al-Naami
Mohammadmehdi Satei

Result

As could be seen from fig 3 for triest base the curve converges to the number of true triangle count with bigger M since we are storing more edges.

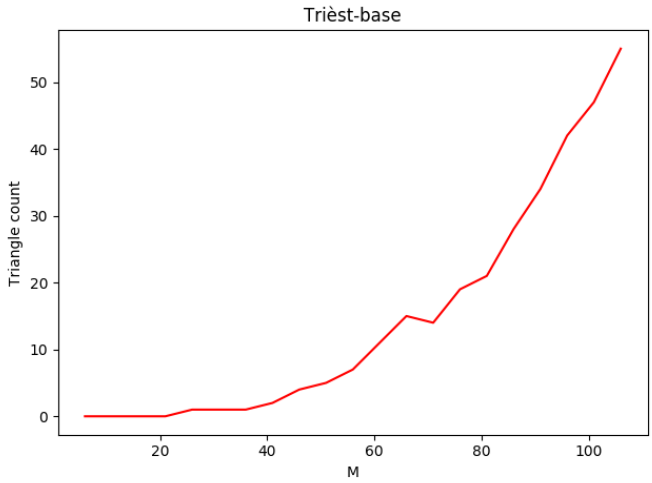

fig. 3 shows triest base algorithm for counting the triangles of a graph stream. The x-axis shows number of edges allowed to store and y-axis shows the number of triangles of the graph.

As could be seen from the graph in fig 4 Triest impr give better results than triest base since it has a lower variance. Triest impr starts to converge to the true traingle count around M = 50. While triest base have a very high variance increasing the triangle count based on number of M. Triest base did not have the true triangle count until M became 107 which is the number of edges of the graph meaning S is all the edges of the graph.

Nora Al-Naami
Mohammadmehdi Satei

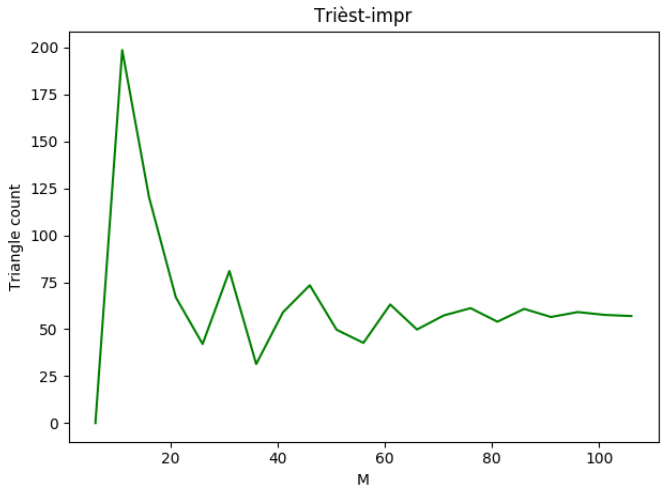

fig.4 shows triest impr algorithm for counting the triangles of a graph stream. The x-axis shows number of edges allowed to store and y-axis shows the number of triangles of the graph.

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

Undertsanding the meaning of neightbour and implementing it since the paper is not very clear about what it means and how one should implement it. Including, determining the value of M was difficult.

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

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

Yea, that's why we are using algorithms that use reservoir sampling since we have a graph stream which we don't know the size of or is unbounded.

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

The algorithms implemented in this do not support edge deletion since they are insertion-only algorithms. We need to use a scheme called random pairing that extends reservoir sampling to handle deletions. [2] triest-fd support edge deletions by keeping a counter di or d0 in order to keep track of the number of uncompensated edge deletions [2].

Run code
Python3 triest-base.py

Nora Al-Naami

Mohammadmehdi Satei

Python3 triest-impr.py

Citerade arbeten

[1] J. Vitter, “Random sampling with a reservoir,” ACM Transactions on Mathematical Software (TOMS), vol. 11, no. 1, pp. 37–57, 1985.

[2] M. Jha, C. Seshadhri, and A. Pinar, “A Space-Efficient Streaming Algorithm for Estimating Transitivity and Triangle Counts Using the Birthday Paradox,” ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 9, no. 3, pp. 1–21, 2015.