

Assignment 3: Mining data streams

Abdullah Abdullah | Aksel Uhr

aabdull@kth.se | auhr@kth.se

GROUP 41

1. Introduction

In this assignment, we created a program that takes a fixed sample of edges of a data stream and estimates local and global triangle counts. This estimation has been proven to be accurate to the true number of triangles in a streaming graph and is hence an optimization solution for large graph streams that likely will not fit in memory. Two algorithms were constructed, TRIEST-BASE and TRIEST-IMPR. Both algorithms were presented by De Stefani et al., (2016) in their paper. During development, a mock-up graph was used with 14 edges and three triangles. For testing, we used the stanford webgraph containing 1469679 triangles.

Task 1.

The base algorithm and the improved algorithm uses reservoir sampling technique which was implemented after we read the data. During runtime, the idea is to insert edges into sample S when $t \leq M$, where M is the size of the sample and t is a counter. Once the sample is filled, we start replacing elements in the sample with the probability M / t . This means that as the number of edges increases (t increases), the probability of replacing elements increases. Doing this, we keep the sample graph updated with regards to all edges in the stream, such that the sample is a fair representation of the entire stream.

Task 2.

TRIEST-BASE

After the reservoir sampling technique is done and if a new insertion to the sample has happened, the counters are updated since there is a new edge and hence a new neighborhood to consider. All counter values are then used for the estimation of the global and local number of triangles.

The algorithm takes the edge samples and an incoming edge that in turn has two nodes. Two sets are instantiated, one for node 1 and the second one for node 2. Then, we iterate through the sample edges and try to find matching values. For any match either in the incoming edge between node 1 or node 2, we append it to the corresponding set (s_1 for node 1 and s_2 for node 2).

Now that we know that there are neighbors in our sample for node 1 and node 2, we find the shared neighborhood for both nodes together (i.e. if they are in s_1 and s_2). This means that we have two out of three sides of a triangle from a node that are in s_1 and s_2 , and based on

what operation we are doing (add or remove), we update the global and local counts (increment or decrement by 1). This count is then being multiplied with the estimation, which ultimately will give us the global triangle count. Notheworthy, we do not estimate the local triangles in our solution, only the global ones.

TRIEST-IMPROVED

The improved version is mostly similar to the base version but varies in a few points which is discussed accordingly.

In the improved version the counters are updated before the if statement regardless of whether the current edge is to be added or dropped. In the base version estimation is calculated and multiplied with global counters to estimate the triangle counts but in the improved version the weight is calculated at the update counters function and is added to the global counter value.

2. How to build and run

We recommend setting up a virtual environment inside the project folder and installing the packages there, instead of a global install of the packages on the machine. In the project root folder:

Unix/macOS:

```
$ python3 -m venv venv
```

```
$ source venv/bin/activate
```

```
(venv) $ python -m pip install requirements.txt
```

Windows (PS):

```
PS> python -m venv venv
```

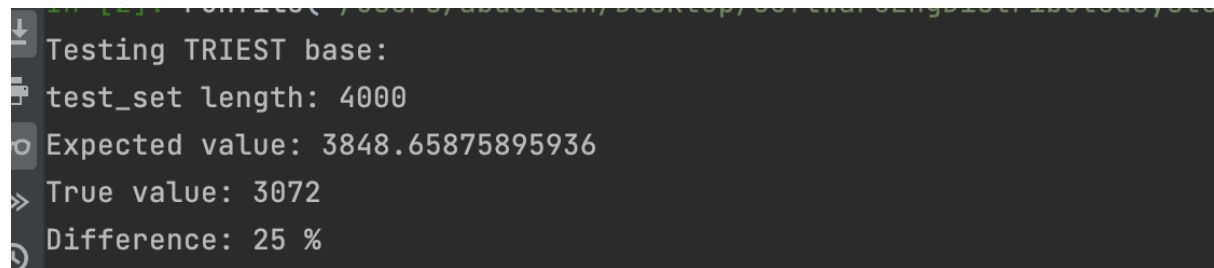
```
PS> venv\Scripts\activate
```

```
(venv) PS> python -m pip install requirements.txt
```

Now, run main.py and you should see some output if the packages were installed correctly!

3. Evaluation / Results

3.1 TRIEST-BASE

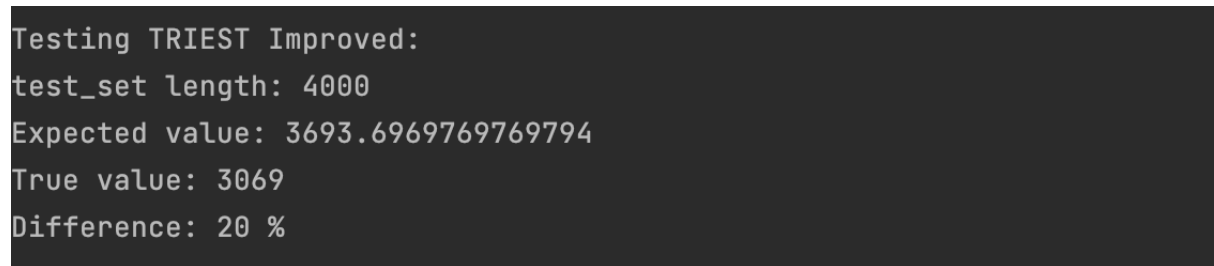


```
Testing TRIEST base:  
test_set length: 4000  
Expected value: 3848.65875895936  
True value: 3072  
Difference: 25 %
```

Fig 1. Triest-Baset.

Figure one shows the output for TRIEST-BASE algorithm. The dataset used was too large to handle on our machine so we just went with 4000 sample of the data for output purpose. The difference between true and expected value is about 25% for the given sample size.

3.1 TRIEST-IMPR



```
Testing TRIEST Improved:  
test_set length: 4000  
Expected value: 3693.6969769769794  
True value: 3069  
Difference: 20 %
```

Fig 1. Triest-Improved.

Figure 2 shows the output for TRIEST-IMPROVED version. The difference is reduced to 20% between expected and true value.

Reference

De Stefani, A. Epasto, M. Riondator, and E. Upfal. 2016. *TRIEST: Counting local and global triangles in fully-dynamic streams with fixed memory size.*