



KUNGLIGA TEKNISKA HÖGSKOLAN

REPORT

---

# Homework 3 : Mining Data Streams

---

***Author :***

Massimo Perini  
Samuel Leonardo Gracio

***Professor :***

November 25, 2018

## 1 Short explanation of the program

Our program is coded in python. For this lab, we could choose the document we wanted to use. We worked with the second document, which is the following :

*"L. De Stefani, A. Epasto, M. Riondato, and E. Upfal, TRIEST: Counting Local and Global Triangles in Fully-Dynamic Streams with Fixed Memory, KDD'16."*

This document is a paper which gives different version of the algorithm Triest, «A suite of one-pass streaming algorithms to compute unbiased, low-variance, high-quality approximations of the global and local (i.e., incident to each vertex) number of triangles in a fully-dynamic graph represented as an adversarial stream of edge insertions and deletions. Our algorithms use reservoir sampling and its variants.»

Our program is an implementation of the two first version of the TRIEST algorithm, which are respectively designated by "Triest\_base" and "Tries Impr" for the improved version. The program is split into classes : the first one is the class Edges which creates the different edges and define several methods for them and the second is the class for the algorithm.

## 2 Instructions to test the program

In order to run the code, you have different parameters :

- The dataset you use. We used one dataset from the following website, which has many graphs datasets : <http://konect.uni-koblenz.de/networks/>.

- The set limit and the sample-size. To avoid huge computing time, we put it to 200 for the set limit and 1000 for the sample-size.

Then, you only need to run the code by calling the different classes. The results will be in the following form : "With 1000 samples the expected value is 800.7167120000001 . The true value is 651 . Error: 149.71671200000014 triangles".

## 3 Results

For the results, we present you the approximation made by the two different version of the algorithm :

- For the first one, i.e the Triest\_base algorithm, we've got : "Base: With 1500 samples the expected value is 425.4166822741641 . The true value is 651.0 . Error: 225.5833177258359 triangles"

- For the second one, i.e the Triest\_impr algorithm, we've got :  
"Improved: With 1500 samples the expected value is 630.5582373333335 . The true value is 651 . Error: 20.441762666666477 triangles"

We found better results with the improved version which was what we expected.

## 4 Optional task for extra bonus

### Q1

The biggest challenge has been the choice of the data structure able to keep the counters.

### Q2

The algorithm can be parallelized. A random sample can be generated as follows: associate a random numbers id with each element and pick the elements with the k largest ids (hashing the numbers). The elements associated with the k largest ids form a random subset. The counters can be updated in parallel to, since sum and subtraction are associative and commutative operations (for example using Spark accumulators).

### Q3

Yes, when queried at time  $t$ , triest base returns  $\epsilon \cdot \tau$  (and  $\tau$  is updated for every element of the stream). Therefore we can query the algorithm while the computation is running on a not bounded stream. The result will be the estimated number of triangles in the stream.

### Q4

No, triest base doesn't support edge deletions. We can adopt triest-FD if we want to support edge deletions. It is based on random pairing (RP), a sampling scheme that extends reservoir sampling and can handle deletions. The idea behind RP scheme is that edge deletions seen on the stream will be "compensated" by future edge insertions.