Report 3: Mining Data Streams

Group 150: Lorenzo Deflorian, Riccardo Fragale

November 21, 2025

1 Introduction

2 Our Methods

A first important thing to notice is the selection of the data structures where to store all the edges of the stream contained in the input file. The choice for us seemed to be between a set of tuples and a set of frozensets. It was not easy to define which of the two was better in terms of speed and efficiency but we ended up choosing the frozensets solution. Since this algorithm is explicitly meant for undirected graphs there was no real reason in using tuples as without the direction we would have to store two tuples for each edge (definitely not the be3st in terms of memory consumption). Considering also that both the Base and the Improved algorithm doesn't support edge deletion, using a set of frozensets was the best choice in our opinion.

2.1 Triest-Base Implementation

2.2 Triest-Improved

2.3 Testing

3 Results

4 Extra questions

What were the challenges you faced when implementing the algorithm? The algorithm was explained very well, especially in terms of pseudocode. Moreover, the procedure was quite straightforward and there were only minor differences between the Base and the Improved version. We found a bit hard to correctly identify the data structures to correctly store the set of edges of the graph. We describe our choice in the previous section related to the methods we used.

Can the algorithm be easily parallelized? We believe that introducing data parallelism in the algorithm doesn't really make a sense since the data is flowing into the algorithm through a data stream and that the operations on the counters are blocking. Moreover, the operations on th counters rely on the current state of the sampleset, which can be modified dynamically.

Still to improve

Does the algorithm work for unbounded graph streams? The algorithm works on data streams, that are by definition unbounded. In addition, it can be queried anytime and it gives you the current estimation for the number of global triangles. As visible in the experiments, the algorithm only needs the size of the sampleset, a counter of the triangles and the number of elements of the stream already stream.

Does the algorithm support edge deletions? At least considering only the algorithms that we implemented (TriestBase and TriestImrpoved) we have no real support for edge deletion. This is is a way a limit of the algorithms. Said that, in the paper that we were given there is also a description of *Triest-FD* which is a further improvement that considers also a fully-dynamical stream in which we have both edge insertion and edge deletion. **Triest-FD** is built upon the concept of Random Pairing (RP) by keeping track of the edges deleted from the sampleset due to deletion in the stream and the overall number of deletions. This information influences the insertion of a new edge in the sampleset and the improved formula for the estimation of the number of triangles.