# Data Mining Assignment 3

Zihao Xu zihxu@kth.se

Minchong Li mincli@kth.se

*Date: November 22, 2023*

## 1 Introduction

In this assignment we choose the paper TRIÈST and implement 2 algorithms: the BASE and IMPROVED algorithm.

## 2 Datasets

We use the Social circles: Facebook dataset. The basic information of this dataset is shown in 1.

| Dataset statistics | |
|---|---|
| Nodes | 4039 |
| Edges | 88234 |
| Nodes in largest WCC | 4039 (1.000) |
| Edges in largest WCC | 88234 (1.000) |
| Nodes in largest SCC | 4039 (1.000) |
| Edges in largest SCC | 88234 (1.000) |
| Average clustering coefficient | 0.6055 |
| Number of triangles | 1612010 |
| Fraction of closed triangles | 0.2647 |
| Diameter (longest shortest path) | 8 |
| 90-percentile effective diameter | 4.7 |

**Figure 1:** Info about Facebook dataset

# 3 Reservoir sampling

We implement the reservoir sampling in `triest.sample_edge()`. When the edge number of current subgraph $S$ is lower than $M$, we directly insert the upcoming edge into the $S$. And when the edge number is not lower than $M$, we do a biased coin flip where there's a probability of $\frac{M}{t}$ we sample a random edge from $S$ with an uniform distribution, then remove it and add the upcoming edge.

# 4 TRIÈST Implementation

To implement the 2 TRIÈST algorithms, we generally follow the pseudo code of Algorithm 1 in the paper. We create a class `graph.TRIESTGraph` to store the edge and neighbors, and implement all the algorithms in `triest.py`.

# 5 How to Build and Run

Firstly, download the dataset and unzip it into the `assignment3_root/dataset`.

Secondly, install the requirements:

```
pip install -r requirements.txt
```

Then, use the following command to run the program:

```
cd ./src/
python main.py ../dataset/facebook_combined.txt --M 5000 --model base
```

Arguments explanation:

- required positional argument: path of dataset.
- `--M`: Reservoir size.
- `--model`: Model type: 'base' or 'improved'

# 6 Results

In this part, we compare the performance of Base Triest algorithm and Improved Triest algorithm by plotting out the estimated number of triangles vs. size of reservoir.

From the figures above, we found:

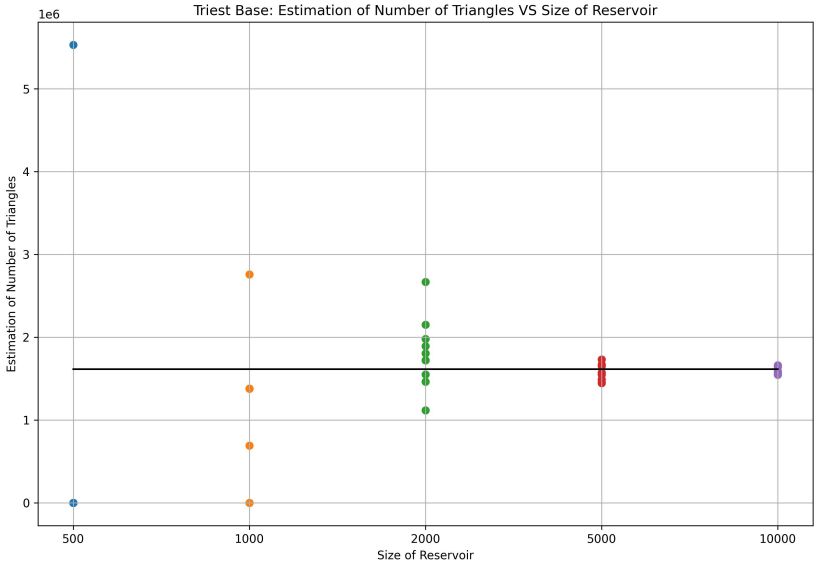


**Figure 2:** Triest Base: Estimation of Number of Triangles VS Size of Reservoir

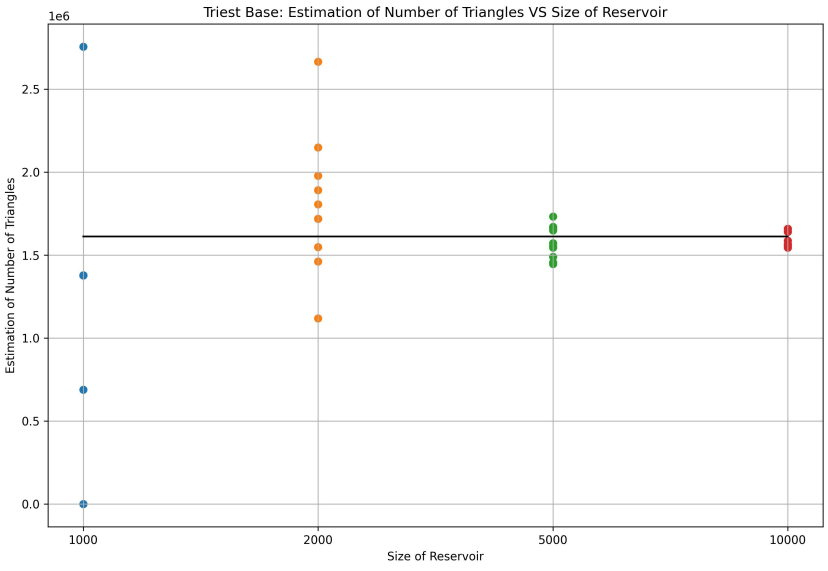


**Figure 3:** Zoomed: Triest Base: Estimation of Number of Triangles VS Size of Reservoir

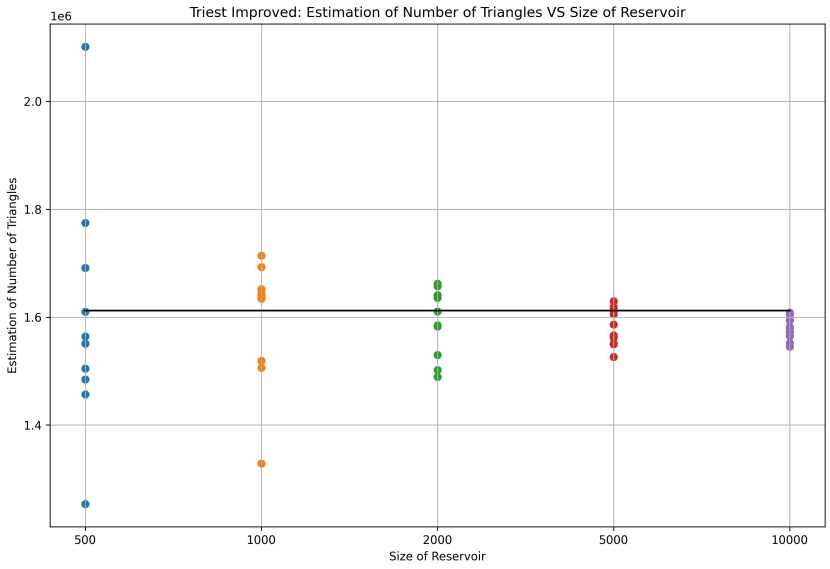


**Figure 4:** Triest Improved: Estimation of Number of Triangles VS Size of Reservoir

1. As the size of Reservoir increases, the standard deviation of estimation decreases.
2. Triest Improved has lower standard deviation on estimation than Triest Base, which proves it to be better algorithm.
3. Intuitively, it's seen from the figures that the estimation is unbiased.

# 7 Bonus Points

## 7.1 What were the challenges you faced when implementing the algorithm?

One challenge we encountered is the algorithm's dependency on the properties of the graph. The algorithm is designed for undirected graphs, so special attention is needed when selecting the dataset. Initially, we chose a different directed graph dataset, resulting in frustrating outcomes. Another noteworthy aspect pertains to the features of Python. We utilized tuples (u, v) to store edges. In Python, tuples (u, v) and (v, u) are not considered equal. To address this issue, we ensured that the smaller of the two values in u and v becomes the first element of the tuple when implementing the algorithm.

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

TRIÈST cannot be easily parallelized, as this algorithm receives data stream as input, which makes it stateful. During the process of the algorithm, the graph changes dynamically, which increases the difficulty of parallelization.

## 7.3 Does the algorithm work for unbounded graph streams? Explain.

Yes, TRIÈST work for unbounded graph streams. This algorithm uses reservoir sampling to avoid OOM error, thus ensuring it works for unbounded streams.

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

The BASE and IMPROVED TRIÈST algorithms do not support edge deletions. To make it able to handle edge deletions, we can use TRIÈST-FD. TRIÈST-FD can handle edge deletions using random pairing (RP). In TRIÈST-FD, edge deletions seen on the stream will be compensated by future edge insertions. In this algorithm, a counter is used to record the number of uncompensated edge deletions.