CS6890: Fraud Analytics - Assignment 1
Computing Trust Rank

Akshay Santoshi - CS21BTECH11012
Nitya Bhamidipaty - CS21BTECH11041

March 2025

1 Approach

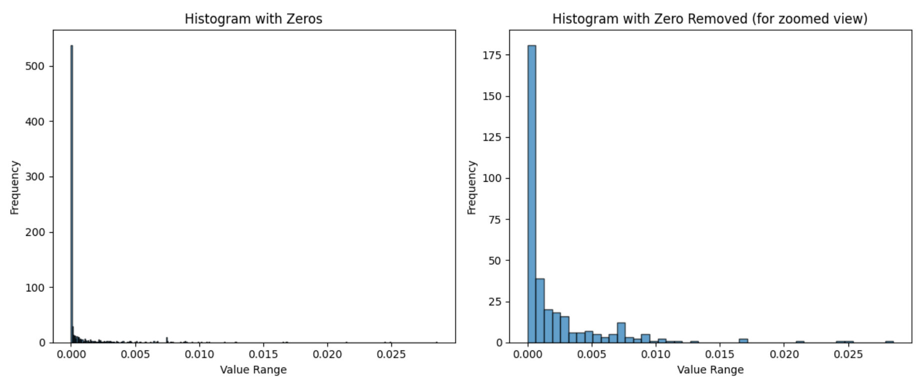
- The given dataset consists of two Excel files. 'Payments.xlsx' contains 130535 transactions, with each row consisting of a 'Sender', 'Receiver' and 'Amount'. 'bad_sender.xlsx' contains a list of 20 bad senders. (Should be in the same folder as code.py for running).
- We model these transactions as a directed graph, with the nodes representing the sender or receiver and the weighted directed edge as the amount of transaction from the sender to receiver.
- The trust rank algorithm is implemented using the Pregel framework.
- We initialize a variable 'outgoing_edges which maps each sender to a list of (receiver, amount) pairs.
- Multi-edges are reduced to a single one with its weight as the sum of all the edge weights. This will reduce the number of messages propagated (between different nodes/threads) without changing the final result.
- We defined a 'TrustRankVertex' class which has initialization method and an update method.
- The initializaton method makes use of Vertex class from Pregel to set up the out vertices, vertex id, and trust value initialization.
- The trust values for the given bad senders are initialized to $\frac{1}{\text{no. of bad senders}}$, rest of the nodes are initialized to zero. So the fraud-score propagates from the bad senders to the rest of the graph.
- So a higher trust score indicates a higher chance that a bank is fraudulent.

- The transition matrix T is slightly modified for weighted edges. The out-degree of a node is the sum of the weights of the out-edges.

$$T(p,q) = \begin{cases} 0, & \text{if } (q,p) \notin E, \\ \frac{\text{weight}(q,p)}{\text{out-deg}(q)}, & \text{if } (q,p) \in E. \end{cases}$$

- The update method iteratively updates the trust rank where the trust propagates from bad senders to their neighbours.
- We compute the new trust rank based on incoming messages and distribute trust to the outgoing edges such that it is proportional to the transaction amount.
- Finally, we create the TrustRankVertex object for each node and run Pregel on this graph. It updates the trust rank by making use of parallel threads and after the max_iterations are reached at each vertex, it prints the trust rank for each node.

2 Results



- Results are for $\alpha = 0.85$ (dampening factor) and 50 iterations.
- On running `code.py`, the plot is saved as `Histogram.png` and results are stored in `TrustValuesResult.xlsx` (also printed to stdout).
- On the left the histogram of all the trust values is plotted.
- Most of the banks are good i.e not fraudulent, hence trust value 0 has a large number of nodes ≈ 600.
- The right plot is only to view values other than zero i.e for a zoomed view.

```
    Bad Sender Trust Values
------------------------------
```

```
1303 0.0075000000000000015 1259 0.0075000000000000015
1562 0.0075000000000000015 1147 0.010154988749819452
1393 0.0075000000000000015 1031 0.0075000000000000015
1210 0.021492828473745243  1042 0.01679580245638882
1048 0.01080176166320239   1256 0.0075000000000000015
1668 0.0075000000000000015 1161 0.007731886578835433
1007 0.028530440071183258  1034 0.012843563739028587
1836 0.007531698596705518  1099 0.010594430450201161
1489 0.007564768191146335  1821 0.0075000000000000015
1076 0.011995184717713013  1944 0.0075000000000000015
```

- Note that bad senders are getting higher fraud value. So the results are align with the actual data.
- Some accounts with many transactions from fraud accounts also have a higher trust value.