CS6890: Assignment 1
Computation of Trust Rank using Pregel Framework

Team Members:
CS22BTECH11043 : Nethi Keerthana
CS22BTECH11012 : Bolla Nehasree
ES22BTECH11025 : N. Krishna Chaitanya

1 Problem Statement

Given a seed set of bad nodes and payment transaction details between nodes representing weights of edges, we need to identify bad nodes. To make it scalable, we use the Pregel framework. TrustRank is used to find bad nodes, starting with a set of oracle-provided bad nodes as a seed group. This method results in higher ranks for bad nodes and lower ranks for good ones, functioning like a DistrustRank.

2 Description of Dataset

The dataset consists of two files:

- **Payments.xlsx**: Contains transaction information between senders and receivers.
  - **Sender**: Represents the seller in a transaction.
  - **Receiver**: Represents the buyer in a transaction.
  - **Amount**: Indicates the amount of the transaction.
- **bad_sender.xlsx**: Contains a list of bad nodes.
  - **Bad Sender**: IDs of nodes identified as bad.

Transactions between the same sender-receiver pairs are summed up. Nodes with no outgoing links are linked to all bad nodes with equal edge weights.

3 Algorithm

TrustRank is an algorithm similar to PageRank but designed to identify trustworthy and malicious entities in a network, useful for fraud detection in financial transactions.

3.1 Graph Construction

The function `TrustRankVertex.create_vertices(payments, bad_senders)` builds the transaction graph:

1. Extract unique senders, receivers, and known bad nodes.
2. Initialize vertices:
   - Higher initial trust for bad nodes.
   - Outgoing edges assigned transaction amounts.
   - Normalize edge weights to sum to 1.
   - Ensure nodes without outgoing edges link to bad nodes.
3. Run Pregel's TrustRank computation to obtain final scores.

3.2 TrustRank Algorithm

Vertex Properties:

- **value**: The trust score (initially high for bad nodes).
- **out_vertices**: List of connected nodes.
- **edge_weights**: Stores transaction amounts as edge weights.
- **damping_factor**: Controls probability of trusting bad actors.
- **num_supersteps**: Defines number of iterations.

Update Logic:

- Each node aggregates incoming trust scores.
- Computes new trust score using the TrustRank formula:

$$\text{new\_trust} = \alpha \sum \text{incoming trust scores} + (1-\alpha) \times \text{initial probability of bad nodes} \tag{1}$$

- Sends updated trust score weighted by outgoing edges.
- Terminates after reaching the maximum iterations.

3.3 Pregel Framework (Graph Processing Model)

A Pregel-like system consists of:

- **Vertex Class**: Represents each node in the graph.
- **Pregel Class**: Manages supersteps and distributes work.
- **Worker Class**: Simulates parallel execution with multiple threads.

Each vertex stores its ID, trust score, outgoing edges, processes messages, and updates values based on incoming trust scores.

4 Results

4.1 Histogram of TrustRank Values

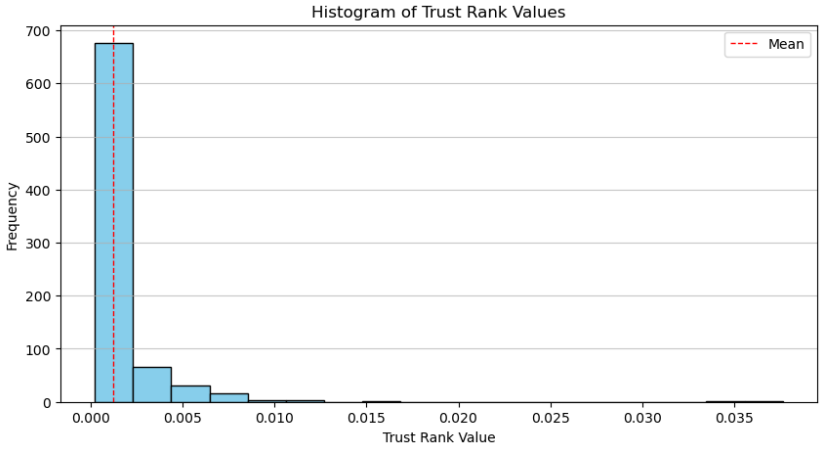

Figure 1: Histogram of Trust Rank Values

Most nodes have distrust ranks between 0.00 and 0.01, while a few have higher ranks, indicating that the majority of nodes are good, except for a few.

4.2 Distrust vs Node ID

This plot shows how distrust values vary across nodes.

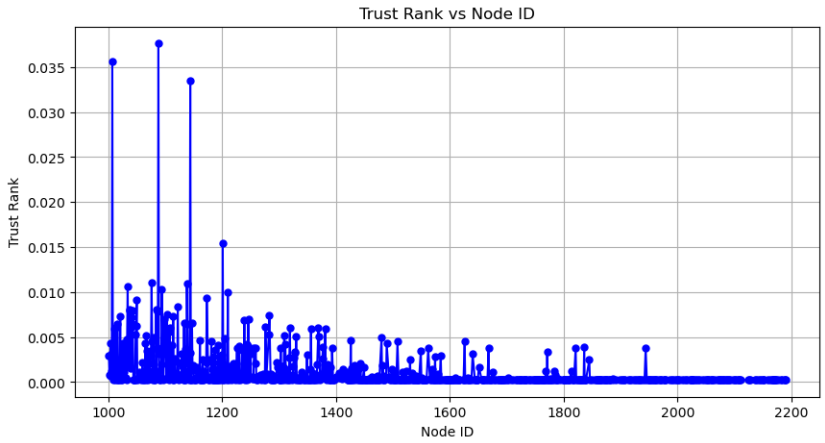

Figure 2: Distrust vs Node ID

5 Analysis

1. A small number of nodes have high DisTrustRank scores, indicating they are likely bad nodes.

2. Most nodes have close-to-zero DisTrustRank scores, suggesting they are trustworthy.