भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Indian Institute of Technology Hyderabad

Fraud Analytics (CS6890)

Assignment: 2 | Trust Rank

Name	Roll Number
Shreesh Gupta	CS23MTECH12009
Hrishikesh Hemke	CS23MTECH14003
Manan Patel	CS23MTECH14006
Yug Patel	CS23MTECH14019
Bhargav Patel	CS23MTECH11026

Contents

1 | Problem statement

- Implementation of Trust rank on "payments.csv" data set.
- We have some bad sender data and a list of transactions between all sender receiver, including the bad sender.
- Our task is to give each sender a trust rank based on their transaction with the bad senders.

2 | Description of the data set

We have two csv file, "payment.csv" and "bad-sender.csv"

- The "Payments.csv" collection includes 130,535 entries of transaction data between different entities. It consists of three main columns with integer values:

  1. **Sender:** The sender of the payment is identified in this column. A distinct integer identification is used to denote each sender.
  2. **Receiver:** This column, like the Sender column, lists the recipients of the payments as integer identifiers that show who the recipient of each transaction is.
  3. **Amount:** The monetary value of each transaction is shown in this column in an integer format. The amount transferred from the sender to the recipient is specified.

- The data is completely non-null, indicating full records in each of the three fields. All columns have an integer datatype (int64), which facilitates efficient management and operation of the dataset.

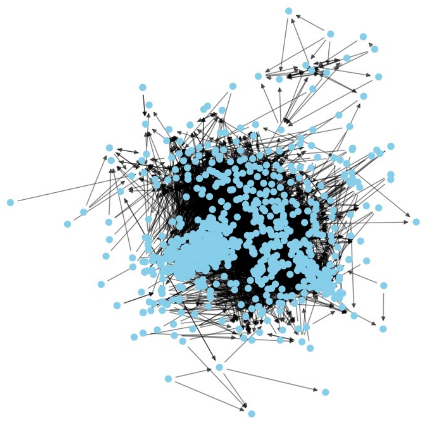


Figure 2.1: Dataset Visualization

- The dataset titled "bad-sender.csv" consists of column contains integers which presumably represent IDs of senders that have been flagged bad. There is one column named "Bad Sender" and contains 20 rows.
- "Payments.csv" dataset is pivotal for analyzing financial transactions between entities, potentially aiding in understanding patterns such as the flow of money, identifying high transaction volumes between specific entities, and more.
- The memory usage for "payment" dataset is approximately 3.0 MB, indicating a moderately sized dataset that is manageable for various data processing and analysis tasks.

3 | Algorithm Used

3.1 | TrustRank

- **Objective**: TrustRank aims to quantify the trustworthiness of entities within a network by evaluating their connections, especially their proximity and interactions with known untrustworthy (bad) nodes. This approach helps in identifying not only directly untrustworthy nodes but also those whose trustworthiness is compromised due to their network associations.

- **Core Concept:** TrustRank is a derivative of the PageRank algorithm, originally designed for ranking web pages. In the context of financial networks, TrustRank adapts this concept to assign a trust score to each entity based on its transactions and relationships within the network.
- **Outcome:** The resulting trust scores provide a ranked list of entities based on their estimated trustworthiness within the network. Entities closely associated with bad actors receive lower scores, highlighting potential risks in further transactions with them.
- **Usage:** These trust scores are instrumental in risk management and fraud detection within financial systems, aiding in the identification of entities whose transaction patterns suggest risky or untrustworthy behavior.

3.2 | PageRank

- The central idea behind PageRank is that a web page is important if it is linked to by other important pages. The algorithm interprets a link from page A to page B as a ”vote” by page A, supporting page B. The importance of each vote is taken into account based on the importance of the voting page, thus creating a recursive measure of value.

4 | Approach we followed

1. **Initial Score Assignment**
   Every node v in the graph G is assigned an initial TrustRank score, $TR(v)$. The scores are uniformly distributed across all nodes:
   $$TR_0(v) = \frac{1}{N}$$
   where N is the total number of nodes in the graph.
2. **Personalization Vector Creation**
   A personalization vector p is created to adjust initial scores based on node trustworthiness, specifically reducing the influence of known bad nodes:
   $$p(v) = \begin{cases} 0.1 & \text{if } v \text{ is a bad node} \\ 1 & \text{otherwise} \end{cases}$$
   The vector is normalized to ensure that its elements sum to 1:
   $$p(v) = \frac{p(v)}{\sum_{u \in G} p(u)}$$
3. **Iterative Score Update**
   The TrustRank score for each node is updated iteratively. At each iteration, the new score for a node v is calculated based on the scores of its predecessors (nodes that point to v) adjusted by their transaction amounts as weights:
   $$TR_{i+1}(v) = (1-d) \cdot p(v) + d \cdot \sum_{u \in P(v)} \frac{TR_i(u) \cdot w(u,v)}{\sum_{w \in S(u)} w(u,w)}$$
   where: - $P(v)$ is the set of predecessors of v, - $S(u)$ is the set of successors of u, - $w(u,v)$ is the weight of the edge from u to v (transaction amount), - d is the damping factor, typically set to 0.85, representing the probability of continuing the random walk through the network.

4. **Convergence Check**
The algorithm iterates until the TrustRank scores converge, which is typically determined by the total change in scores between iterations falling below a predefined threshold ϵ:

$$\sum_{v \in G} |TR_{i+1}(v) - TR_i(v)| < \epsilon$$

5. **Conclusion**
This mathematical framework ensures that the TrustRank scores reflect not only the direct financial interactions between entities but also the broader network dynamics, particularly penalizing entities closely connected to known bad actors. The damping factor d and the personalized initialization help to simulate a biased random walk that favors trustworthy nodes over others, effectively distinguishing between different levels of trustworthiness in the network.

We have implemented the below algorithm

Algorithm 1 TrustRank Algorithm

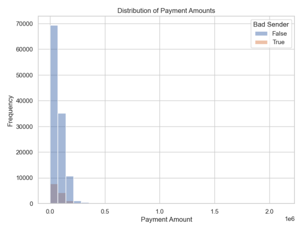
```
0: Input: Graph $G(V, E)$ with nodes $V$ and edges $E$, bad nodes $B$, damping factor $d$, tolerance $\epsilon$
0: Output: TrustRank scores $TR$
0: procedure TrustRank($G, B, d, \epsilon$)
0:   Initialize $TR(v) \leftarrow \frac{1}{|V|}$ for each $v \in V$
0:   Initialize personalization vector $p$
0:   for $v \in V$ do
0:     if $v \in B$ then
0:       $p(v) \leftarrow 0.1$
0:     else
0:       $p(v) \leftarrow 1$
0:     end if
0:   end for
0:   Normalize $p(v) \leftarrow \frac{p(v)}{\sum_{u \in V} p(u)}$
0:   repeat
0:     Initialize $TR_{\text{new}}$
0:     for $v \in V$ do
0:       $TR_{\text{new}}(v) \leftarrow (1 - d) \cdot p(v)$
0:       for $u \in$ predecessors of $v$ do
0:         $TR_{\text{new}}(v) \leftarrow TR_{\text{new}}(v) + d \cdot \frac{TR(u) \cdot w(u,v)}{\sum_{w \in \text{successors of } u} w(u,w)}$
0:       end for
0:     end for
0:     $change \leftarrow \sum_{v \in V} |TR_{\text{new}}(v) - TR(v)|$
0:     $TR \leftarrow TR_{\text{new}}$
0:   until $change < \epsilon$
0:   return $TR$
0: end procedure=0
```

5 | Results

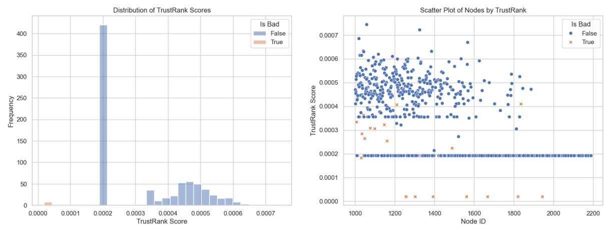
5.1 | Data distribution before and after assigning trust score

In the first pair of graphs, we observe a distribution of payment amounts and a scatter plot of transactions with the distinction between 'Bad Senders' and 'Good Senders'. It shows that the majority of transactions involve smaller payment amounts, and there isn't a clear pattern distinguishing 'Bad' from 'Good' senders, making it difficult to detect fraud based solely on these factors.

In the second pair of graphs, after the TrustRank algorithm has been applied, we see a distribution of TrustRank scores and a scatter plot of nodes by TrustRank. It's evident that TrustRank has effectively differentiated between 'Bad' and 'Good' nodes. The distribution of TrustRank scores shows that 'Bad' nodes tend to have lower scores. The scatter plot makes this even clearer, as we can see 'Bad' nodes mostly occupy the lower region of the TrustRank score axis



(a) Data distribution before trust score



(b) Data distribution after trust score

Figure 5.1: Graph Comparisons

5.2 | Calculation of threshold

Let $X = \{x_1, x_2, \ldots, x_n\}$ be the ordered dataset with n elements such that $x_1 \leq x_2 \leq \ldots \leq x_n$.
Define k as the desired percentile, in this case $k = 10$.
Calculate the index i using the formula:

$$i = \frac{k}{100} \times (n + 1)$$

If i is an integer, the k^{th} percentile P_k is the value of the x_i in the sorted dataset.
If i is not an integer, round i down to the nearest whole number to find i_{low}, and round i up to the nearest whole number to find i_{high}. Then interpolate to find P_k using the formula:

$$P_k = x_{i_{low}} + (i - i_{low}) \times (x_{i_{high}} - x_{i_{low}})$$

This interpolation considers that the real percentile position may lie between two data points in the ordered set.
In the context of the Python quantile function, it handles these steps internally and provides the k^{th} percentile value directly.

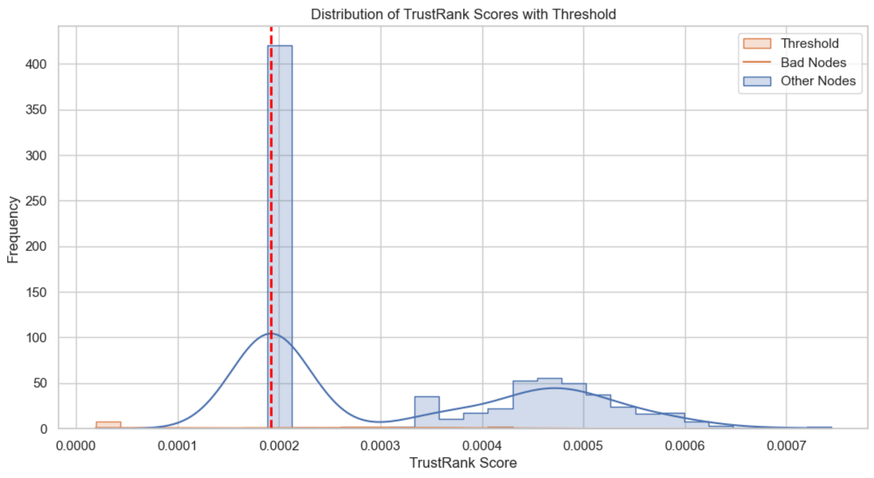


Figure 5.2: Frequency vs TrustRank score

The threshold value provided, 0.0001920614596670935, is likely the cutoff score determined for classifying nodes as 'Bad' or potentially fraudulent after applying the TrustRank algorithm. Here's what the counts signify:

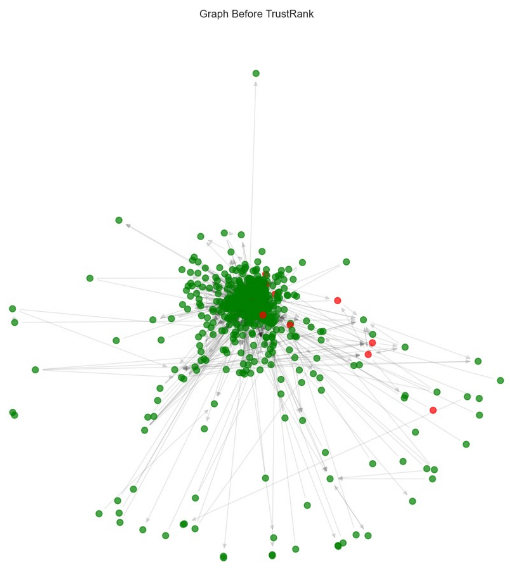
- Initial-bad-nodes-count (20): This is the count of nodes initially labeled or suspected as 'Bad' before the TrustRank algorithm was applied. These could be nodes identified through some heuristic or previous knowledge as potentially problematic.
- Post-trustrank-bad-nodes-count (429): After applying the TrustRank algorithm, this number represents the nodes that are now considered 'Bad' because their TrustRank scores fell below the threshold. This significant increase from the initial count suggests that the TrustRank algorithm has identified a larger set of nodes that exhibit characteristics of being 'Bad' compared to the initial heuristic or labeling.
- Total-nodes-count (799): This is the total number of nodes in the network that were analyzed. The context here implies a network where transactions or interactions occur, and each node represents an entity or individual within that network.

  To begin the implementation of this algorithm, we first need to load the dataset, focusing on the first two columns containing the seller ID and buyer ID. Upon examination, we can determine that there are a total of 799 unique nodes, of which 20 are given as malicious. There are a total of 1,30,535 transactions in the dataset. To account for the possibility of multiple directed edges between nodes, we can consolidate them and transform the multi-directed graph into a single-directed graph. This consolidation process results in a graph of 5,358 edges, where the weight of each directed edge represents the total sum of directed transactions between any two given nodes.

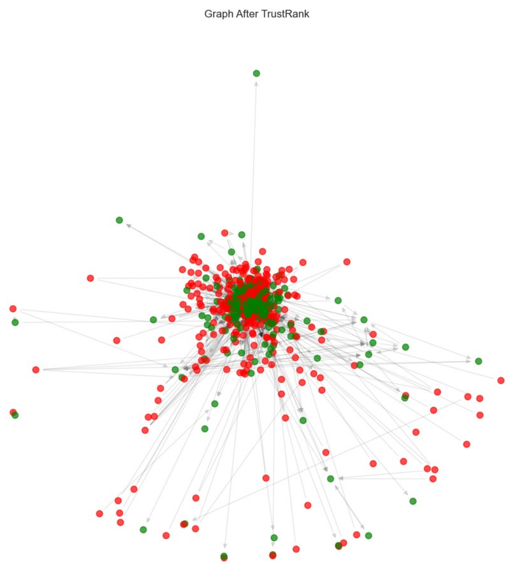
5.3 | Graph base result Comparisons

It is evident that the twenty red nodes are the harmful nodes, while the green nodes are normal nodes. A portion of the graph in Figure 5.3 (b) was chosen because it is difficult to describe any outcomes in a graph this size. We can also view the Figure 5.4 as a scenario following the bad score propagation.

As predicted, the nodes with higher transaction values with the bad nodes have higher bad scores. Figure 5.4 shows the subset of the original network that has a small number of bad nodes and a small number of normal nodes. The edge weights are used to reflect this volume of transactions.

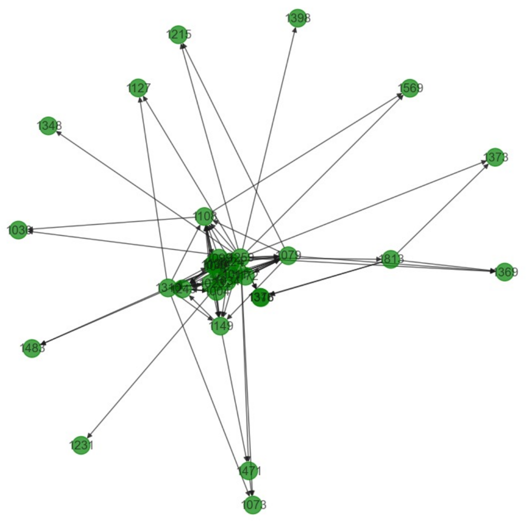


(a) Image-a

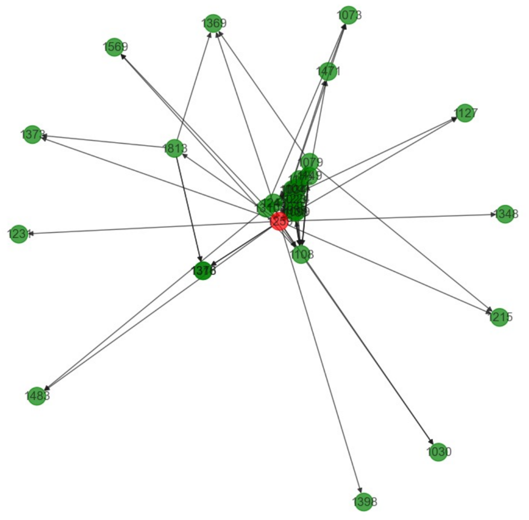


(b) Image-b

Figure 5.3: Graph Comparisons



(a) Image-a



(b) Image-b

Figure 5.4: Graph Comparisons

6 | References

[1] https://www.vldb.org/conf/2004/RS15P3.PDF.