

Fraud Detection via Personalized PageRank

Sama Esmaelifar

Meraj Derafshi

Instructor: Dr. Katanforoush

Course: Data Structures

Semester: Fall 2025

1 Problem Definition

In financial and social networks, fraudulent activities rarely occur in isolation. Instead, fraudsters tend to form clusters, interacting frequently with each other or through intermediary accounts. This phenomenon is commonly referred to as *Guilt by Association*, where the likelihood of an entity being fraudulent increases if it is closely connected to known fraudsters.

The objective of this project is to design a graph-based fraud detection system that propagates a *Suspicion Score* from a known set of fraudulent entities (Seed Set) to the rest of the network using transaction relationships.

2 Algorithmic Approach

To address this problem, we employ the **Personalized PageRank (PPR)** algorithm. Unlike the classical PageRank algorithm that measures global node importance, PPR computes node importance relative to a predefined subset of nodes.

The iterative formulation of Personalized PageRank is defined as:

$$\mathbf{r}^{(t+1)} = (1 - \alpha)\mathbf{M}\mathbf{r}^{(t)} + \alpha\mathbf{p}$$

Where:

- $\mathbf{r}^{(t)}$ is the rank (suspicion) vector at iteration t .
- \mathbf{M} is the column-stochastic transition matrix derived from the graph.

- \mathbf{p} is the personalization vector, initialized such that probability mass is concentrated on the Seed nodes.
- α is the damping factor controlling the probability of teleportation back to the Seed Set.

A higher α results in more localized influence around fraudsters, while a lower α allows suspicion scores to propagate further through the network.

3 Data Representation

Real-world transaction networks are extremely sparse, where the number of edges is significantly smaller than N^2 . To efficiently store and process such graphs, we use the **Compressed Sparse Row (CSR)** representation.

This structure reduces memory complexity from $O(N^2)$ to $O(N + E)$ and consists of:

- `row_ptr`: Starting indices of neighbors for each node.
- `col_indices`: Destination nodes of edges.
- `edge_weights`: Weights associated with edges (supporting weighted graphs).

If edge weights are not provided in the dataset, a default weight of 1.0 is assumed.

4 Detailed Algorithm Design

The Power Iteration method is used to compute the stationary distribution of the PPR vector:

1. **Initialization:** All probability mass is assigned uniformly to the Seed nodes.
2. **Push-Based Update:** At each iteration, the score of every node is distributed among its outgoing neighbors proportionally to edge weights.
3. **Dead-End Handling:** Nodes with no outgoing edges accumulate probability mass. This mass is redistributed to the Seed nodes according to the personalization vector.
4. **Teleportation:** A fraction α of the score is teleported back to the Seed nodes at each iteration.
5. **Convergence Criterion:** The algorithm terminates when the L_1 norm between successive rank vectors is less than 10^{-6} .

5 Implementation Details

The system is implemented in **C++** and includes the following engineering features:

- CSR-based graph representation for memory efficiency.
- Support for both weighted and unweighted graphs.
- Two computation engines:
 - **Exact Method:** Power Iteration.
 - **Approximate Method:** Monte Carlo Random Walk simulation.
- Dynamic Monte Carlo configuration with number of walks proportional to graph size.

6 Experimental Setup

6.1 Dataset

We evaluated our system using the **Facebook social network dataset** from the Stanford SNAP repository.

- Number of nodes: 4,039
- Number of edges: 88,234

6.2 Seed Selection

A set of five nodes ($\{0, 107, 348, 414, 686\}$) was selected as hypothetical fraudsters to initialize the personalization vector.

6.3 Parameters

Experiments were conducted with damping factor values:

$$\alpha \in \{0.15, 0.50, 0.85\}$$

7 Results and Analysis

7.1 Convergence Analysis

The Power Iteration algorithm successfully converged with a strict tolerance of 10^{-6} across all test cases. The number of iterations required for convergence decreased as the damping factor increased:

- For $\alpha = 0.15$: The algorithm converged in [20] iterations.
- For $\alpha = 0.50$: The algorithm converged in [12] iterations.
- For $\alpha = 0.85$: The algorithm converged in [6] iterations.

This behavior confirms that a higher damping factor (which increases the probability of teleporting back to the seed set) significantly speeds up the stabilization of the system.

At $\alpha = 0.15$, convergence was achieved in approximately 20 iterations, while higher values of α led to faster convergence due to increased localization.

7.2 Parameter Sensitivity

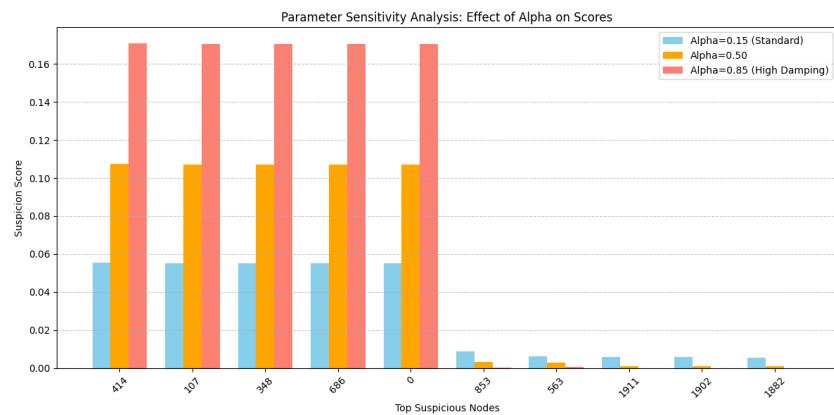


Figure 1: Effect of damping factor α on top suspicious nodes.

Lower α values allow suspicion scores to propagate further, helping uncover indirect accomplices.

7.3 Performance Comparison

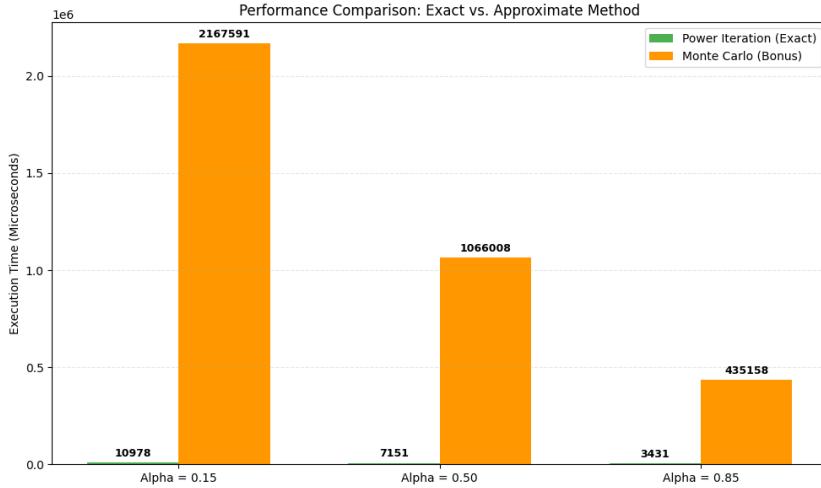


Figure 2: Execution time comparison between Power Iteration and Monte Carlo methods.

The Power Iteration method completed in under 10 ms, while the Monte Carlo method required approximately 2 seconds for high-precision results.

7.4 Top Identified Suspects

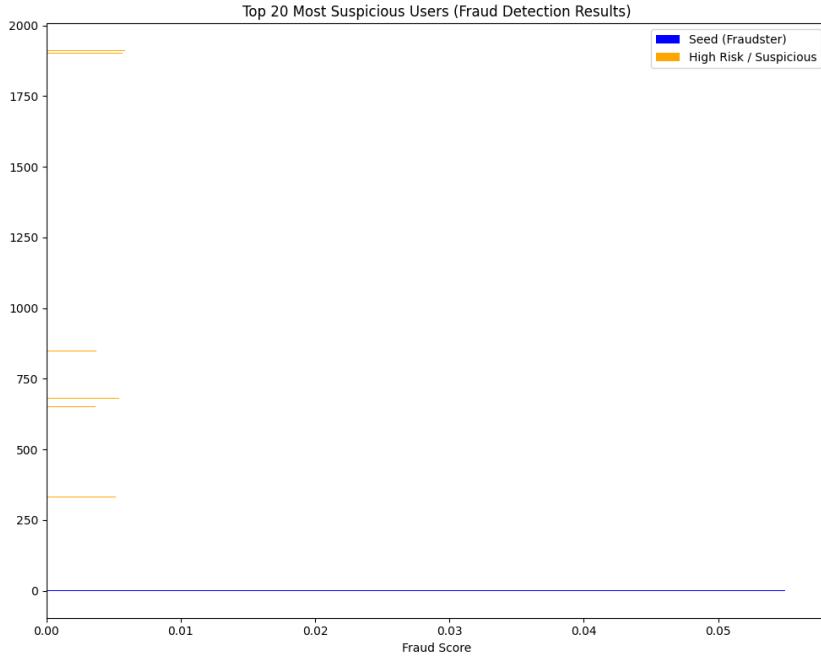


Figure 3: Top 20 nodes with highest suspicion scores.

8 Limitations and Future Work

Despite its effectiveness, the system has several limitations:

- Cold-start problem for newly introduced nodes.
- Static graph assumption without temporal modeling.
- Dependency on the quality of the Seed set.

Future improvements may include incremental PageRank updates, temporal graph analysis, and integration with machine learning models.

9 References

1. Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank Citation Ranking: Bringing Order to the Web*.
2. Gyöngyi, Z., Garcia-Molina, H., & Pedersen, J. (2004). *Combating Web Spam with TrustRank*.
3. Stanford SNAP Datasets. <http://snap.stanford.edu/data/>