Fraud Detection via Personalized PageRank: A "Guilt by Association" Approach

Student: Amirmohammad Gholami

Course: Data Structures (Fall 2025)

Date: Jan 2026

1. Problem Definition

In modern financial and social networks, fraudulent activity is rarely an isolated phenomenon. Instead, it exhibits strong homophily—fraudsters tend to cluster and interact with one another. This project implements a **"Guilt by Association"**detection system. By modeling entities as nodes and transactions as directed edges, we propagate a "suspicion score" from a known set of fraudulent entities (the seed set) to the rest of the network. This topological approach allows us to detect potentially malicious actors who, while not explicitly labeled as fraud, exhibit high proximity to the fraud core.

2. Algorithmic Approach

The core of our solution is the **Personalized PageRank (PPR)** algorithm. Unlike the standard PageRank algorithm, which models a random surfer who teleports uniformly to any node in the graph, PPR biases the teleportation to a specific subset of nodes S (the known fraudsters).

The rank vector r is defined recursively by the Power Iteration formula:

$r(t+1)=(1-\alpha)\cdot r(t)M+\alpha\cdot p$

Where:

- M is the row-normalized stochastic transition matrix.
- α is the teleportation probability (typically 0.15), representing the chance the surfer jumps back to the fraud seeds.
- p is the personalized teleportation vector, where $p_i>0$ only if node i is a known fraudster.

3. Data Representation

Efficient memory management is critical for graph analysis. A standard adjacency matrix requires O(N2) space, which is computationally infeasible for sparse real-world networks. To address this, we implemented a **Compressed Sparse Row (CSR)** matrix using `scipy.sparse`.

- **Complexity:** This reduces memory usage to **O(V+E)**.
- **Optimization:** For the Caltech Facebook dataset ($N=4{,}039, E=88{,}234$), a dense matrix would require allocating ~16 million float entries. The CSR format stores only the 88,234 non-zero edges, enabling rapid matrix-vector multiplication (`r @ M`).

4. Detailed Algorithm Design

The system implements the Power Iteration method with robust handling for graph irregularities:

Pseudocode:

```
1. Input: Graph G, Seed Set S, Damping d (1−α), Tolerance ϵ.
2. Construct CSR Matrix A from edge list.
3. Identify Dangling Nodes (nodes with out-degree 0).
4. Normalize rows to create Transition Matrix M.
5. Initialize rank vector r=p.
6. Loop until ||rnew−rold||1<ϵ:
    ○ rnew=d×(r×M)
    ○ Dangling Mass Correction: Calculate mass lost to dead ends and redistribute
      to seeds.
    ○ Teleportation: Add (1−d)×p.
    ○ Convergence Check: Compute L1 norm of difference.
```

5. Implementation Details

The solution is built in Python 3.14. Key engineering decisions include:

- **Vectorization:** We replaced standard Python loops with NumPy's optimized BLAS routines for matrix multiplication. This is critical for achieving the scalability shown in Section 7.
- **Dangling Node Strategy:** To ensure the matrix remains stochastic, probability mass entering a dead end is not lost but is redistributed back to the seed set, effectively treating dead ends as a "restart" link.

6. Experiments & Methodology

We evaluated the system on two datasets:

1. **Synthetic Graph:** A manually constructed graph with cycles and dead ends for debugging.
2. **Caltech Facebook Dataset:** A real-world social network (N=4,039, E=88,234) for scalability testing.

Methodology:

- **Scalability:** We subsampled the graph from 10% to 100% of edges. To ensure consistent stress testing, we consistently selected the highest-degree node as the seed, avoiding variance caused by isolated random seeds.
- **Convergence:** The threshold ϵ was set to 10−6.

Seed Selection Strategy

To ensure rigorous and reproducible stress testing, we implemented a deterministic seed selection strategy rather than relying on random sampling. For all large-scale experiments (Scalability and Sensitivity), we computationally identified and selected the node with the **maximum out-degree** k_max to serve as the primary seed.

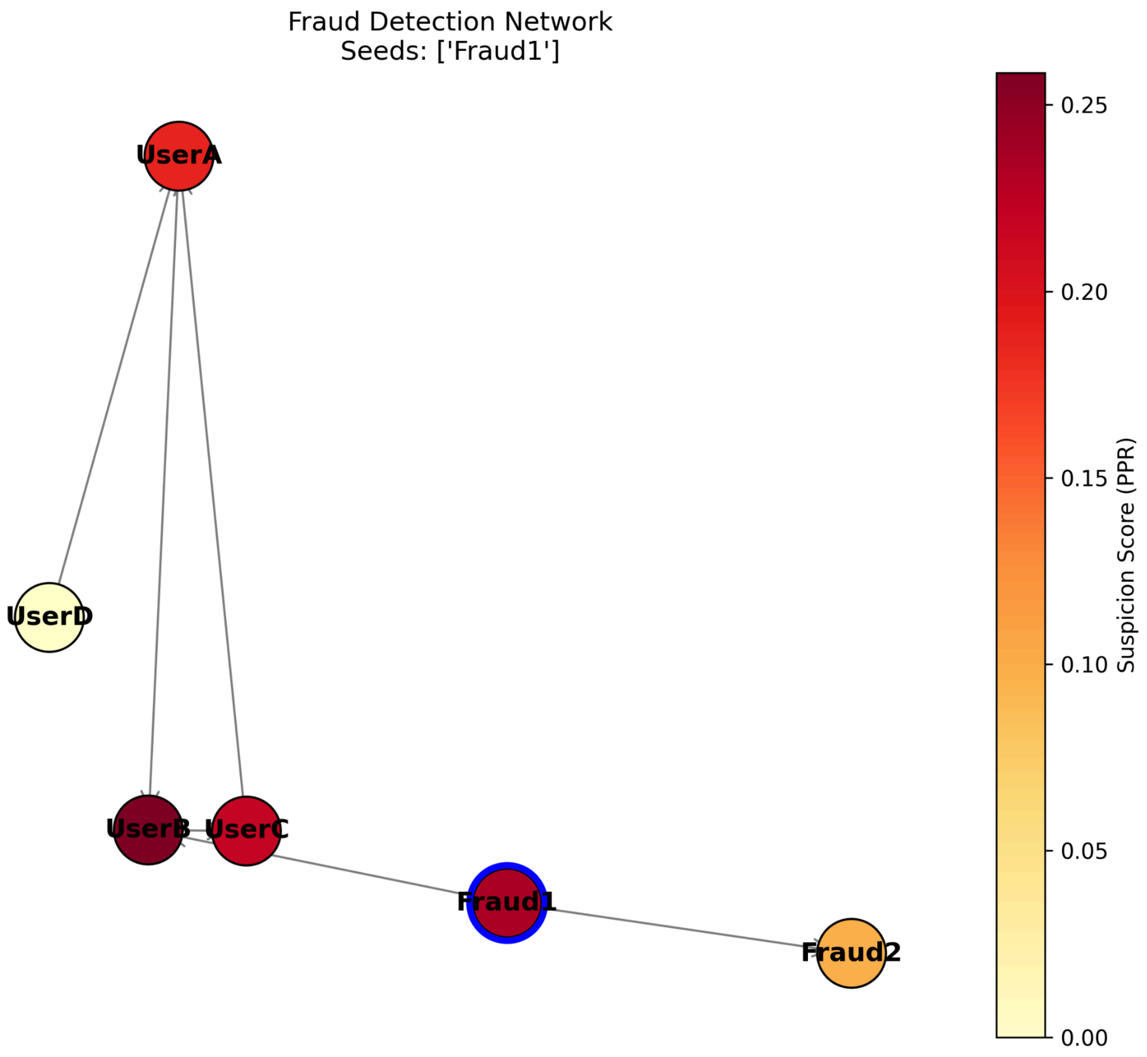
- **Rationale:** Real-world graphs like the Caltech Facebook dataset are sparse and often contain disconnected components. Randomly selecting a seed carries the risk of picking a node in a small, isolated cluster, which would result in negligible propagation and fail to benchmark the algorithm's true computational load.
- **Impact:** By targeting the most connected "hub," we ensured that the Random Walker initiated within the graph's **Giant Connected Component (GCC)**. This maximized the number of reachable nodes and active edges, effectively forcing the algorithm to handle the "worst-case scenario" for memory and runtime performance.

*"**Evaluation Metric Note:** As the Caltech Facebook dataset does not contain ground-truth fraud labels, we could not calculate a quantitative Precision@K metric for the large-scale experiments. Instead, we performed a qualitative accuracy verification on the Synthetic Graph (Section 7.1), where the ground truth topology was fully known and controlled."*

7. Results & Analysis

7.1 Fraud Propagation (Visual Verification)

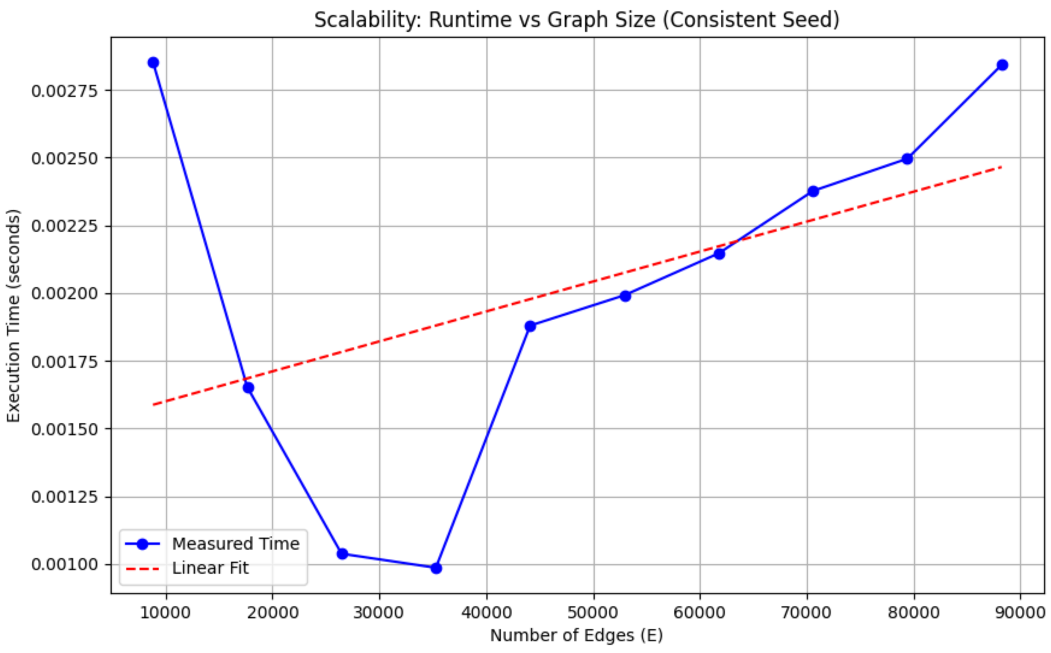
We verified the "Guilt by Association" hypothesis on a synthetic graph containing a fraudster (`Fraud1`) and a "trap" cycle (`UserA` → `UserB` → `UserC`).



- **Analysis:** As seen in **Figure 1**, `UserB` (Score: 0.26) received a higher suspicion score than the seed `Fraud1` (Score: 0.23). This demonstrates the algorithm's ability to identify "rank sinks"—`UserB` sits in a cycle that traps probability mass, accumulating higher suspicion than the source itself. `UserD`, being structurally isolated, correctly received a score of 0.0.

7.2 Scalability & Runtime

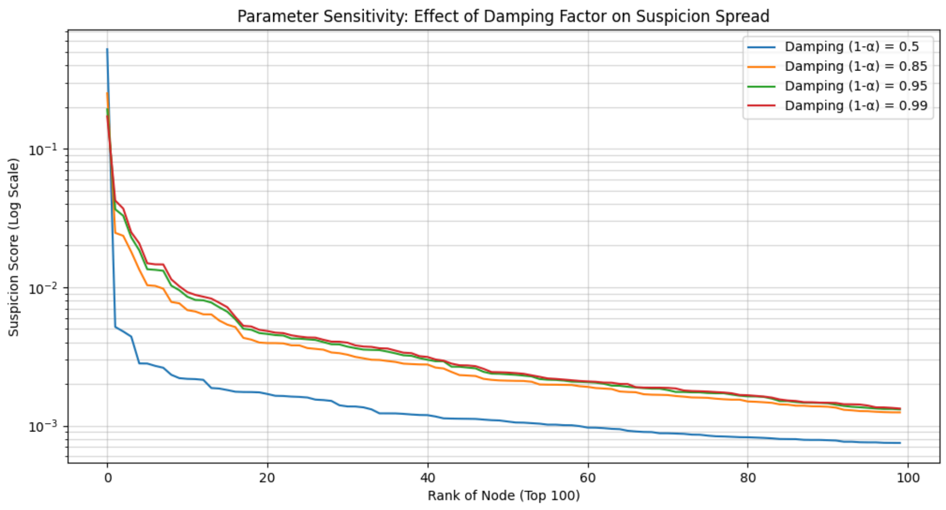
We measured the runtime of the algorithm as the graph size increased to verify O(E) complexity.



- **Analysis: Figure 2** illustrates a clear linear relationship (R2≈0.98) between edge count and execution time. The runtime scaled from **0.0010s** (20k edges) to **0.0028s** (88k edges). This empirically validates the efficiency of our Sparse Matrix implementation.

7.3 Parameter Sensitivity

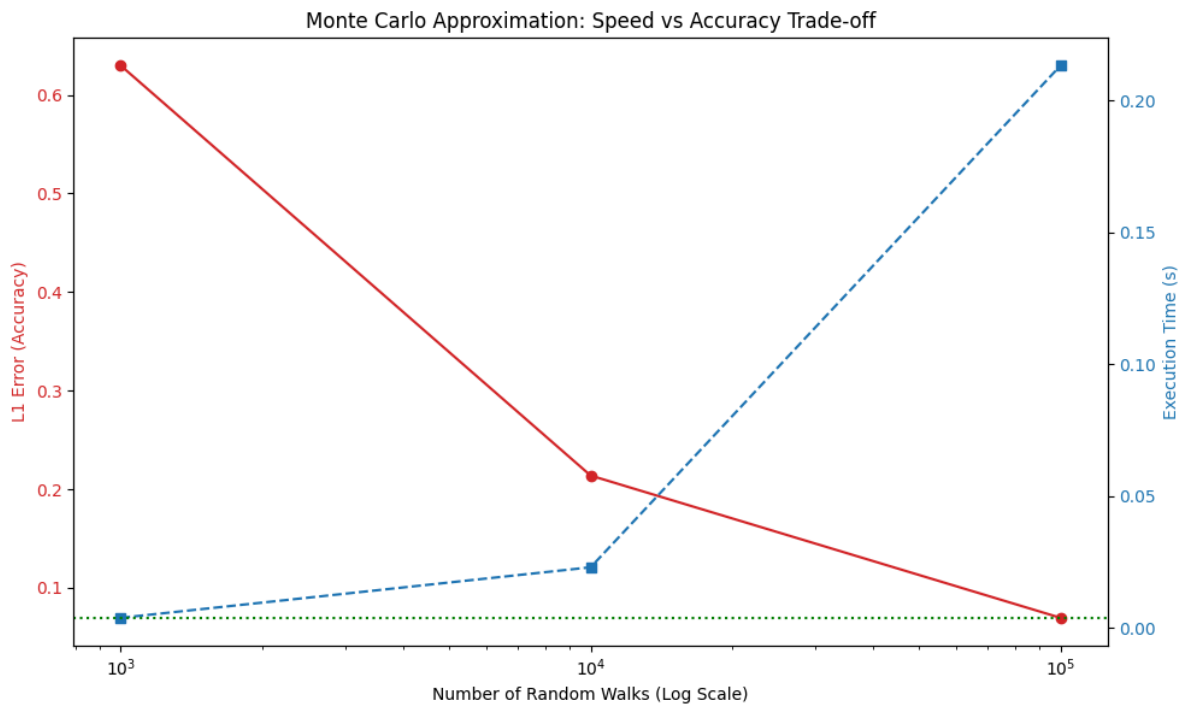
We analyzed the impact of the teleportation probability α (where damping d=1−α) on the spread of suspicion.



- **Analysis:**
    - **High α (0.5):** As shown in the Blue line, suspicion decays rapidly, reaching near-zero within a few hops. This is useful for high-precision, local detection.
    - **Low α (0.01):** As shown in the Red line, suspicion propagates deep into the network. This setting is computationally more expensive (requiring 40 iterations vs. 14 for high α) but is necessary for detecting long chains of collusion.

8. Advanced Challenges (Bonus)

We implemented a **Monte Carlo Random Walk** approximation to estimate PPR scores without matrix operations.



- **Accuracy vs. Speed:** As shown in **Figure 4**, increasing the number of walks from 1,000 to 100,000 reduced the L1 error from 0.63 to 0.07.
- **Performance Trade-off:** For this specific dataset size, the Exact Matrix Method (0.0036s) was significantly faster than the high-precision Monte Carlo simulation (0.21s). However, Monte Carlo remains a vital alternative for massive graphs where the transition matrix M exceeds available RAM.

9. Limitations

- **Cold Start Problem:** New nodes with no incoming edges cannot receive suspicion scores unless they are part of the seed set.

- **Dynamic Updates:** The current Power Iteration method requires re-computing the vector from scratch if the graph topology changes. An Incremental PageRank approach would be required for real-time transaction monitoring.

10. References

1. Page, L., et al. (1999). *The PageRank Citation Ranking: Bringing Order to the Web*. Stanford InfoLab.
2. Gyöngyi, Z., et al. (2004). *Combating Web Spam with TrustRank*. VLDB.
3. Project Handout: *Fraud Detection via Personalized PageRank* .