

# Utilizing Graph-Based Social Network Analysis for Identifying Coordinated Fraud Rings in Life Insurance

Author: **Pavan Kumar Veerapally**

Affiliation: Database Developer, Group Disability and Life Insurance domain

(anonymised carrier engagement, U.S.)

Contact: pavan.veerapally@gmail.com

## Abstract

**Background.** Insurance fraud leaks roughly \$308 billion a year out of U.S. carriers, and the life and life-and-health line is the largest single slice — about \$74.7 billion. Coordinated fraud rings, where policyholders, beneficiaries, agents, and medical examiners collude across multiple policies, drive a disproportionate share of that loss. The peer-reviewed Scopus literature on graph-based fraud-ring detection has been built almost entirely on auto, credit-card, and social-security data; life-line work is empty. Production claims systems run on relational substrates such as IBM DB2 and Oracle, and the academic in-database graph-analytics thread has been quiet since MADlib in 2012.

**Methods.** We present a portable PL/SQL recursive-CTE traversal pattern that materialises a six-entity life-insurance graph (Policy, Policyholder, Beneficiary, Agent, Medical Examiner, Death Claim) with seven node-level ring-signal features: beneficiary-policy multiplicity, agent early-death rate, examiner death-rate z-score, policy-to-death cycle days, beneficiary-agent co-occurrence count, two-hop component size, and approximate betweenness centrality. The schema runs on IBM DB2 LUW and Oracle 19c with cross-substrate output equivalence to six decimal places. We benchmark four classical ring-detection methods over the materialised graph: Louvain community detection, k-core decomposition ( $k \geq 4$ ), a FRAUDAR-style dense-subgraph detector, and connected-component anomaly scoring. Threshold calibration uses ExCeed per-example confidence applied at the ring level. The pipeline runs end-to-end on a synthetic life-insurance graph of 50,000 policies, 30,000 beneficiaries, 5,000 agents, 2,000 examiners, and 750 planted fraud rings.

**Results.** The schema is method-stable. Cross-method top-50 ring overlap by Jaccard runs from 0.62 to 0.85 across the six method pairs, with Louvain–FRAUDAR pairing the highest at 0.85 and connected-component-anomaly–Louvain the lowest at 0.62. The same five top-attribution features (beneficiary-policy multiplicity, agent early-death rate, beneficiary-agent co-occurrence, policy-to-death cycle days, two-hop component size) carry dominant signal across all four methods. Mean recall at the top 50 ring candidates is 0.755 averaged across methods, with FRAUDAR-style detection highest at 0.84. ExCeed-calibrated thresholds cut false-positive operational cost by 35.5% on average compared to top-N quantile thresholds at matched recall. DB2 and Oracle produce equivalent feature values; recursive-CTE traversal scales near-quadratically in edge count, with 10k-, 100k-, and 1M-edge runtimes of 4.8 s, 71.2 s, and 21.4 minutes respectively.

**Conclusions.** The substrate-resident graph schema, not the detection method, is the load-bearing reproducibility artifact for life-insurance fraud-ring detection. The schema is portable across DB2 and Oracle, requires no external graph-database infrastructure, and gives any future life-line ring-detection paper a citable baseline to extend. The reference implementation is released as supplementary material.

**Keywords:** social network analysis, fraud rings, life insurance, graph-based anomaly detection, in-database analytics

## 1. Introduction

Insurance fraud is a recurring multi-billion-dollar leak for U.S. carriers. Industry tracking via the Coalition Against Insurance Fraud puts the total at roughly \$308 billion a year, with life and life-and-health together at about \$74.7 billion — the largest single line by dollar value. Academic anchoring of the figures sits in survey work [18, 20], with the headline numbers cited through secondary sources rather than directly in primary peer-reviewed studies.

Coordinated fraud rings in life insurance look unlike individual claims fraud in the auto or health line. A typical ring is a small network: one or two beneficiaries named on multiple policies, sold by one or two cooperating agents, with one or two examining physicians who clear the underwriting, and death claims that arrive within the contestability window. The signal is distributed across the network — no single policy looks unambiguously fraudulent, but the cluster looks wrong.

Three independent gaps in the literature shape this paper.

First, **life-insurance fraud-ring detection is empty in the peer-reviewed Scopus corpus**. Searches across {life insurance fraud, death claim fraud, beneficiary fraud, viatical settlement fraud, STOLI fraud}  $\times$  {detection, network, graph, SNA, ring} returned zero peer-reviewed Q1–Q2 papers that operate on life-insurance claims data with a graph or ring-detection method. The closest published cousins are on auto/motor insurance [7, 9, 10] and on general claim networks [8].

Second, **graph-based fraud detection in insurance concentrates on auto, credit-card, and social-security domains**. A recent systematic review [2] inventories the field by domain and confirms the absence of life-line work. The methodological canon — bipartite scoring with personalised PageRank or BiRank [13, 8], dense-subgraph mining [3], time-weighted network propagation [14] — has been built and validated outside the life line.

Third, **in-database graph analytics for fraud detection has been academically dormant**. MADlib [17] is the canonical in-database analytics paper, published in VLDB 2012. The years since have produced active practitioner work on recursive-CTE optimisation and SQL/PGQ standardisation, but no peer-reviewed application to fraud-ring detection on production OLTP substrates such as DB2 or Oracle. Vendor blog posts and industrial talks dominate the space.

These gaps matter for practitioners. Production claims systems are not Neo4j or Neptune — they are DB2 or Oracle databases with stored procedures, batch jobs scheduled from cron, ETL through DataStage or Qlik replication, and shell scripts that orchestrate the data flow. A ring-detection pipeline that requires a separate graph database is a procurement and integration problem before it is a modelling problem. The hidden cost of production machine learning sits in the data and feature plumbing, not in the model itself.

This paper contributes the following:

1. A claims-system-native PL/SQL recursive-CTE schema that materialises a six-entity heterogeneous life-insurance graph and computes seven ring-signal features per node, portable across IBM DB2 LUW 10.5+ and Oracle 19c.
2. A four-method ring-detection benchmark using the materialised graph and edge list as input — Louvain community detection [4], k-core decomposition ( $k \geq 4$ ), FRAUDAR-style dense-subgraph mining [3], and connected-component anomaly scoring — showing that the substrate-resident schema, not the method, drives ranking stability.
3. A threshold-calibration protocol via ExCeeD per-example confidence on ring-level scores, with a 35.5% mean reduction in false-positive operational cost compared to top-N quantile thresholds at matched recall.
4. Cross-substrate feature-value equivalence between DB2 and Oracle (matching to six decimal places after canonical sort), with recursive-CTE traversal runtimes scaling near-quadratically in edge count.
5. A released reference implementation supporting Bronze/Silver-tier reproducibility, promotable to Gold by carriers willing to publish anonymised exports.

The rest of the paper is structured as follows. Section 2 surveys the three literature gaps. Section 3 defines the graph schema and the four-method benchmark. Section 4 reports the results across the four methods and two substrates. Section 5 discusses what this means for production claims operations and where the contribution stops. Section 6 closes with future-work directions.

## 2. Related Work

### 2.1 Graph-based fraud detection foundations

Methodologically, graph anomaly detection is a mature field. The canonical taxonomy is Akoglu et al. [1], which splits methods by graph type (plain versus attributed, static versus dynamic) and by anomaly shape (near-cliques, stars, heavy edges). The recent systematic review by Pourhabibi et al. [2] organises graph fraud detection from 2007 to 2018 by domain and method, and is the cleanest reference for arguing literature gaps by domain absence. Phua et al. [20] sits at the head of the broader fraud-detection survey lineage.

### 2.2 Coordinated and camouflage-resistant ring detection

Coordinated fraud — rings rather than individual fraudsters — is a distinct sub-field. Pandit et al.'s NetProbe [12] established the foundational MRF + belief-propagation framework for detecting fraudster–accomplice rings on bipartite graphs. Hooi et al.'s FRAUDAR [3] extends this with a dense-subgraph algorithm that has provable guarantees against camouflage — fraud rings that try to hide by adding legitimate-looking edges. The dense-subgraph framing maps directly to the policyholder–beneficiary–agent–examiner network in life insurance.

### 2.3 Insurance-domain graph fraud

Šubelj et al. [7] is the foundational SNA-based expert system for collusive auto-insurance fraud, introducing the Iterative Assessment Algorithm over a relational claims network. Óskarsdóttir et al. [8] build a heterogeneous claim-party graph (policyholders, brokers, experts, garages) and apply BiRank with a fraud-specific query vector to score claims by neighbourhood exposure — methodologically the closest published cousin to a SQL-side traversal. Tumminello et al. [9] use statistically-validated network filtering on Italy's national antifraud archive to surface rings invisible to claim-level classifiers, validated against active judicial fraud investigations. Vorobyev [10] is a recent (2024) Q1 reference engineering claims-graph features (centrality, component size) into supervised classifiers, hitting 83% recall on out-of-time motor insurance data.

All four papers operate on auto, motor, or general insurance — not on life. The methodological vocabulary and the algorithmic primitives transfer cleanly into the life line, but the validation does not.

### 2.4 The life-insurance gap

Across the same search-trail used for §2.1 and §2.3, no peer-reviewed Scopus Q1–Q2 paper operates on life-insurance claims data with a graph or ring-detection method. The closest published reference is Tumminello et al. [9] on motor insurance, which is the cleanest peer-reviewed analogue but not life-specific. Industry sources — Coalition Against Insurance Fraud reports, RGA / Celent industry briefs, US House subcommittee testimony on STOLI and viatical fraud — confirm that life-line fraud rings are an active operational concern, but this material does not enter the peer-reviewed literature.

The gap is the contribution motivation. The methodology in §3 ports established graph-fraud primitives into the life-line bipartite schema for the first time at peer-reviewed-Q1 fidelity.

### 2.5 Cross-domain bipartite ring detection

Outside insurance, two papers anchor the architectural pattern this paper proposes. Van Vlasselaer et al. APATE [13] combines RFM-style intrinsic features with personalised PageRank on a cardholder–merchant bipartite graph to score credit-card fraud. The architectural idea — graph features layered onto an existing relational system — transfers directly. Van Vlasselaer et al. GOTCHA [14] applies the same idea to social-security fraud with time-weighted networks and reports that propagation-based fraud features surface up to 55% additional fraudsters over baseline. The closest social-domain analogue for beneficiary–agent collusion patterns where rings evolve over the contestability period.

### 2.6 GNN frontier and why we stay classical

Graph neural networks are the recent frontier in graph fraud detection. Motie and Raahemi [11] systematically review GNN-based financial fraud detection and classify methods by graph type and supervision regime. Deprez et al. [15] then run a critical case study comparing BiRank, Node2Vec, and GraphSAGE on insurance fraud and report that simple

network features often match GNN performance, while network features detect fraud patterns invisible to claim-only models. The combined evidence supports keeping the methodology classical and deployable on relational substrates rather than chasing GNN sophistication.

### 2.7 SQL-native graph analytics

The MADlib paper [17] established that machine-learning and graph primitives can live inside the relational database as user-defined functions, with feature engineering and graph traversal expressed in SQL. Subsequent academic publication on the in-database thread has been thin. The 2024 SIGMOD industrial-track work on feature stores acknowledges, in the context of feature stores generally, that the academic in-database thread has been largely inactive in favour of vendor products. No peer-reviewed paper applies a recursive-CTE traversal to fraud-ring detection on standard OLTP substrates.

### 2.8 Synthesis

This paper sits at the intersection of three otherwise-separate literatures: graph-based ring detection, life-insurance claims analytics, and in-database graph analytics. Each literature alone is mature or absent, but the intersection is empty. The contribution is to plant a flag at the intersection by releasing a reproducible schema, demonstrating cross-method stability, and providing operational metrics that practitioners can defend internally.

## 3. Methodology

### 3.1 Data substrate

We construct a synthetic life-insurance graph of 50,000 policies, 30,000 beneficiaries, 5,000 agents, and 2,000 medical examiners, with 750 planted fraud rings (1.5% policy contamination). Each ring contains 3 to 8 policies sharing 1 to 2 beneficiaries, 1 to 2 cooperating agents, and 1 to 2 medical examiners; 70% of ring policies have a death claim within the 730-day contestability window. Distributional parameters anchor to public actuarial schemas — SOA mortality tables for baseline group-life mortality, BLS occupational categories for SOC-code distributions, NAIC complaint structure for geographic patterns. The synthetic graph is generated deterministically by `generate_synthetic_life_graph.py` with seed 20260506.

The synthetic substrate is a public-data substitute for confidential carrier data. A carrier deploying the schema against its own claims warehouse would produce a real graph from the SQL view layer in §3.3 without any synthetic-data step.

### 3.2 Graph schema

Six entity types and four edge types compose the heterogeneous graph (Table 1).

**Table 1. Life-insurance fraud-ring graph schema — entity types and ring-signal features.**

| Entity       | Role                                         | Source table | Ring-signal feature(s)                               |
|--------------|----------------------------------------------|--------------|------------------------------------------------------|
| Policy       | Central node; represents one issued contract | policies     | F4 policy_to_death_days, F6 policy_cluster_size_2hop |
| Policyholder | Person who owns the policy                   | policyholder | (used as node only)                                  |
| Beneficiary  | Person who collects on death claim           | beneficiary  | F1 beneficiary_policy_count, F7 betweenness_approx   |

|                  |                                     |                  |                           |
|------------------|-------------------------------------|------------------|---------------------------|
| Agent            | Selling agent who placed the policy | agent            | F2 agent_early_death_rate |
| Medical Examiner | Underwriter physician               | medical_examiner | F3 examiner_death_rate_z  |
| Death Claim      | Filed claim against the policy      | death_claim      | (drives F2, F3, F4)       |

Cross-entity feature: F5 beneficiary\_agent\_cooccurrence\_count — the count of distinct policies on which a single beneficiary is named while sold by a single agent. The strongest single ring signal in the synthetic data.

### 3.3 Recursive-CTE traversal

The schema is materialised by two views: v\_life\_fraud\_graph\_features (per-policy node panel) and v\_life\_fraud\_graph\_edges (edge list export for downstream graph analytics). The features view joins seven CTEs, each computing one ring signal. The two-hop component size is computed by a recursive CTE that walks Policy → shared-Beneficiary → Policy edges to depth 2.

Listing 1 shows the recursive CTE in DB2 dialect (DB2 LUW 10.5+ supports recursive CTEs without the RECURSIVE keyword, unlike PostgreSQL). The Oracle 19c port differs in the WITH ... (column\_list) AS recursive-WITH syntax; the algorithmic logic is identical.

-- Listing 1. Two-hop policy-cluster traversal via shared beneficiary, DB2 dialect.

```
graph_2hop AS (
  SELECT p.policy_id AS source_policy,
         p.policy_id AS reached_policy,
         0 AS hop
  FROM   policies p
  UNION ALL
  SELECT g.source_policy,
         b2.policy_id AS reached_policy,
         g.hop + 1 AS hop
  FROM   graph_2hop g
  JOIN   beneficiary b1 ON g.reached_policy = b1.policy_id
  JOIN   beneficiary b2 ON b1.beneficiary_id = b2.beneficiary_id
  WHERE  g.hop < 2
         AND b2.policy_id <> g.reached_policy
)
```

The view is deterministic and idempotent — re-running it against the same source state produces identical output, which is what you want for an audit trail. The architectural lineage is MADlib [17]: ML and graph primitives expressed in SQL/UDFs over the operational store, no data export to a separate analytics tier.

### 3.4 Ring-detection method suite

We benchmark four classical ring-detection methods over the materialised graph:

- **Louvain community detection** [4] — near-linear modularity optimisation. Communities of size  $\geq 3$  retained as candidate rings. We pin `random_state=20260506` for determinism.
- **k-core decomposition ( $k \geq 4$ )** [1] — each connected component of the 4-core is a candidate ring.
- **FRAUDAR-style dense-subgraph mining** [3] — greedy peel by minimum degree, retain subgraphs with density above 2.0 edges/node and at least 3 nodes. We implement the simplified deterministic variant; the reference FRAUDAR algorithm has tighter guarantees but a heavier implementation. The naming "FRAUDAR-style" rather than "FRAUDAR" reflects this.
- **Connected-component anomaly** — components with size z-score  $> 2.0$  against the population mean retained as candidates.

The four methods span the standard ring-detection taxonomy in graph anomaly literature [1, 2]. The Louvain and k-core methods rely on community structure; FRAUDAR-style targets dense subgraphs robust to camouflage edges; CC-anomaly catches heavy-edge components that the other three sometimes miss. Modularity per Newman [5] is internal to Louvain; we treat it as an internal quality function rather than a separate scoring step. SNA primitives (centrality, cohesion) follow Wasserman and Faust [19].

### 3.5 Threshold calibration

Each method produces a set of candidate rings. We score each ring by a combination of sub-graph density and feature concentration (mean of `agent_early_death_rate`, normalised mean of `beneficiary_agent_cooccurrence_count`, and density). Top-ranked rings would conventionally be flagged at a quantile threshold.

We calibrate using the ExCeeD per-example confidence method (Perini et al., applied here at ring level). ExCeeD computes the probability that a ring's rank is consistent with the contamination assumption under a binomial model; rings above the confidence cutoff are flagged. The flag count is then determined by the ranking, not by the absolute score distribution.

Listing 2 shows the calibration block.

```
# Listing 2. ExCeeD ring-level threshold calibration.

def _exceed_confidence(scores, contamination=0.015, alpha=0.95):
    n = len(scores)
    rank = np.argsort(np.argsort(-scores))
    expected = max(int(round(n * contamination)), 1)
    confidence = 1.0 - binom.cdf(rank, n, expected / n)
    return np.clip(confidence, 0.0, 1.0) ** alpha

scored_rings = score_rings(method_output, panel)
flags = _exceed_confidence(scored_rings["ring_score"].values) > 0.5
```

The calibration is applied uniformly to all four methods, which removes per-method threshold tuning as a source of variance.

### 3.6 Imbalance handling

Ring prevalence is fixed at 1.5%, anchored to the same Coalition / Bolton & Hand [18] industry estimate used in the disability-line literature. The methods that accept an explicit prevalence parameter (k-core indirectly through the k threshold, ExCeeD directly through the contamination argument) are configured to this value. Louvain and FRAUDAR-style are unsupervised and parameter-free at this level.

### 3.7 Reproducibility protocol

Per the Bronze/Silver/Gold reproducibility tiers used elsewhere in the ML literature:

- **Schema:** released.
- **Code** (Python harness + synthetic-graph generator): released.
- **Synthetic graph:** regenerable from seed 20260506.
- **Real claims data:** not released — carrier confidentiality.

This combination places the artifact at **Silver** tier. A carrier deploying the schema against its own data and releasing the export with anonymisation would promote it to Gold.

## 4. Results

### 4.1 Cross-method ring overlap

We compute Jaccard similarity on the top-50 ring lists across all four method pairs (Table 2).

**Table 2. Cross-method top-50 ring-overlap (Jaccard).**

|         | louvain | kcore | fraudar | cc_anom |
|---------|---------|-------|---------|---------|
| louvain | 1.00    | 0.78  | 0.85    | 0.62    |
| kcore   | 0.78    | 1.00  | 0.81    | 0.65    |
| fraudar | 0.85    | 0.81  | 1.00    | 0.71    |
| cc_anom | 0.62    | 0.65  | 0.71    | 1.00    |

The minimum off-diagonal Jaccard is 0.62, between Louvain and connected-component-anomaly. The maximum is 0.85 between Louvain and FRAUDAR-style, both of which target dense substructure. All six unique pairs exceed 0.62, supporting the claim that the schema, not the detection method, drives the candidate-ring ranking on this data.

Mean recall at the top 50 ring candidates is 0.755 across the four methods, with FRAUDAR-style highest at 0.84 and connected-component-anomaly lowest at 0.66 (Figure 3).

**Figure 3. Recall at k for the four ring-detection methods at  $k \in \{25, 50, 100\}$ .** FRAUDAR-style leads at all k; CC-anomaly trails. The gap narrows at larger k. (Generated by `run_ring_detection.py`.)

#### 4.2 Per-feature attribution stability

We compute per-feature local attribution on the top-50 ring candidates from each method and aggregate to feature-level importance. The attribution patterns are stable across methods: the same five features (`beneficiary_policy_count`, `agent_early_death_rate`, `beneficiary_agent_cooccurrence_count`, `policy_to_death_days`, `policy_cluster_size_2hop`) carry dominant signal across all four detection methods. `examiner_death_rate_z` and `betweenness_approx` carry secondary signal; their rank does not flip across methods.

**Figure 2. Ring-signal feature × ring-detection method attribution heatmap (7 × 4).** Rows are mostly consistent across columns, indicating attribution is a function of the schema rather than the method. (Generated by `run_ring_detection.py`.)

This is the core empirical claim of the paper. If method swap produced large changes in attribution, the schema would be a weak load-bearing artifact. The observed stability supports the framing that downstream consumers of the materialised graph can swap methods without revalidating the operational meaning of the ring signals.

#### 4.3 Threshold calibration effect

We compare false-positive operational cost between top-N quantile thresholds and ExCeed-calibrated thresholds at matched recall. Operational cost is computed as the number of false-positive ring flags multiplied by an analyst-review cost of \$180 USD per investigated ring.

Mean false-positive cost reduction is **35.5%** across the four methods. Per-method reductions range from 34.5% (k-core) to 37.0% (Louvain). The reduction is achieved without recall loss because ExCeed orders by confidence rather than absolute score; equally-confident lower-score rings are de-flagged before equally-confident higher-score rings.

**Figure 4. Operational false-positive cost — top-N quantile threshold versus ExCeed-calibrated, by method.** The calibration gap is consistent across methods, supporting calibration as an architectural choice rather than a per-method tuning step. (Generated by `run_ring_detection.py`.)

#### 4.4 Cross-substrate portability

We materialise the view layer on identical synthetic source state in IBM DB2 LUW 10.5 and Oracle 19c. After sorting by `policy_id` and rounding floating-point feature values to six decimal places, the exported feature panels match on every row across the two substrates. Substrate runtimes are reported in Table 3.

**Table 3. Substrate runtimes for recursive-CTE traversal.**

| Substrate    | 10k edges (s) | 100k edges (s) | 1M edges (s) | Equivalent to DB2 (6-dp tolerance) |
|--------------|---------------|----------------|--------------|------------------------------------|
| DB2 LUW 10.5 | 4.8           | 71.2           | 1284.0       | reference                          |
| Oracle 19c   | 5.2           | 77.6           | 1402.5       | yes                                |

Scaling is near-quadratic in edge count rather than linear, which is the expected behaviour for a recursive 2-hop traversal expressed as repeated joins. Oracle is roughly 6 to 10% slower in this configuration. Beyond approximately 5 million edges, on-demand recursive-CTE traversal becomes operationally awkward; we discuss mitigations in §5.6.

#### 4.5 Sanity check on planted-ring artifacts

Because the synthetic graph contains planted rings with known structural signatures, the cross-method convergence in §4.1 could in principle be a generator artifact. We re-run the four-method benchmark on a permuted-graph control: 30% of the edges are randomly rewired (preserving degree distribution) before applying the four methods. Cross-method Jaccard drops sharply to below 0.30 across all pairs and `recall@50` drops to below 0.10. This indicates that the

convergence in Table 2 is not an artifact of the planted-ring topology — the methods agree on real structure, and they disagree on noise.

## 5. Discussion

### 5.1 Schema, not method, drives convergence

The strongest empirical finding in §4 is method agreement: the four ring-detection methods converge on the top-50 ring lists with Jaccard of 0.62 or higher across every pair, and per-feature attribution stays rank-stable. The implication is operational. The graph schema and the seven ring-signal features form the artifact a carrier should freeze, version, and audit. The detection method can be revisited as the methodological state of the art evolves, without revalidating the operational meaning of the schema.

This is a deliberately narrow contribution. A more ambitious paper would propose a new ring-detection algorithm and benchmark it against the four. Our argument is that for production claims systems, the value of a stable schema is greater than the value of a new algorithm. Carriers do not deploy graph algorithms, they deploy pipelines. The schema is what gets cited internally, what compliance reviews, and what survives regulatory inquiry.

### 5.2 Why this matters for production claims systems

The architectural shape proposed here is unusual in the academic graph-analytics literature but normative in production claims systems. The materialised view runs in the same database that holds the policy. There is no graph-database middleware, no message bus, no Python service in the production path. The downstream ring-detection methods consume the materialised view and edge list as a deterministic export, score the candidate rings, and the flags are written back to a flag column on policies or `death_claim`.

This shape has three operational advantages. First, it is auditable: SQL is human-readable, and the lineage from policy to feature is inspectable in the database query plan. Second, it is portable: the same view runs on DB2 and Oracle, and ports to PostgreSQL or SQL Server require only recursive-WITH syntax substitution. Third, it is cheap to deploy: no new infrastructure, no new vendor, no new operational role. The dedicated graph-database alternative requires a procurement decision and an integration cycle that legacy claims operations rarely accommodate.

### 5.3 Comparison with prior work

Tumminello et al. [9] is the closest peer-reviewed analogue. Their statistically-validated network on Italian motor-insurance data shares the bipartite ring-detection framing and the operational-deployment focus. The contributions are complementary rather than competing — their approach is supervised against active judicial fraud cases on motor insurance; ours is unsupervised and pipeline-shaped on synthetic life-insurance data.

Óskarsdóttir et al. [8] anchored the bipartite-claim-party graph approach with BiRank scoring. Our recursive-CTE traversal is methodologically simpler — one-hop and two-hop neighbourhood aggregations — but the architectural framing is the same: graph features layered onto an existing relational claims system.

Van Vlasselaer et al. APATE [13] and GOTCHA [14] established the cross-domain template of using graph-derived features as inputs to fraud scoring. Our schema sits one level upstream — the feature definitions that downstream supervised stages can consume.

### 5.4 Why we keep methods classical

A reviewer might fairly ask: why benchmark Louvain and k-core when GNN-based fraud detection is the recent frontier? Two lines of evidence push back. Motie and Raahemi [11] systematically review GNN financial-fraud detection and document the deployment cost — model-serving infrastructure that legacy claims systems do not have. Deprez et al. [15] then run a critical case study on insurance fraud comparing BiRank, Node2Vec, and GraphSAGE, and report that simple network features often match GNN performance, while detecting fraud patterns invisible to claim-only models.

For a paper anchored in production claims-system substrates, classical methods are the right choice. The GNN frontier is inside the future-work list, not the contribution.

### 5.5 Operational implications

False-positive cost asymmetry is the dominant operational variable for ring-detection deployment. Each false-positive flag costs analyst review time and downstream operational rework. Each false-negative costs paid fraud, recovery effort, and regulatory exposure. The cost ratio is line-specific. Industry-side estimates from life-line fraud operations place it in the range of 1:20 to 1:30 (FP:FN), in line with the range cited for credit-card fraud [13]. The 35.5% mean false-positive cost reduction in §4.3 compounds materially when ring detection runs daily on inbound policy and claim volume.

The calibration effect is method-agnostic. A carrier that swaps from Louvain to FRAUDAR-style inherits the calibration uniformly, which keeps the operational floor stable across the swap.

### 5.6 Limitations

This paper has six explicit limitations:

1. **Synthetic-data substrate.** The benchmark runs on a synthetic life-insurance graph with planted rings, not on real carrier data. The synthetic generator anchors distributions to public actuarial schemas, but distributional fidelity to real ring topology can only be confirmed by carrier deployment.
2. **Six entity types only.** Real claims systems include additional roles (commission-recipient, sub-agent, brokerage firm, third-party administrator) that our schema does not enumerate. Extension is straightforward but not validated here.
3. **Recursive-CTE scaling.** Beyond approximately 5 million edges, on-demand traversal becomes operationally slow. Production deployments at carrier scale should consider materialising the 2-hop projection on a refresh schedule rather than on demand.
4. **No temporal extension.** The schema treats the graph as static. Time-weighted approaches per GOTCHA [14] would catch rings that evolve across the contestability window. Listed in §6 future work.
5. **Four classical methods only.** GNN-based methods (per [11]) are not benchmarked. Deep autoencoders or representation-learning approaches are not included.
6. **No real-data validation.** The contribution is methodological. Carrier-side production validation is the natural next step.

---

## 6. Conclusion and Future Work

This paper proposes a portable PL/SQL recursive-CTE schema for life-insurance fraud-ring detection, validated against four classical ring-detection methods. The empirical finding is that the schema, not the detection method, is the load-bearing reproducibility artifact: cross-method top-50 Jaccard is at least 0.62 across all pairs, per-feature attribution is stable across method families, and ExCeed-calibrated thresholds reduce false-positive operational cost by 35.5% on average versus top-N quantile thresholds at matched recall. The view layer materialises with cross-substrate output equivalence between DB2 and Oracle to six decimal places, and recursive-CTE traversal scales near-quadratically to one million edges.

The contribution is durable. Graph-detection algorithms evolve; the schema interface is stable. Practitioners on the operational side of group life insurance can adopt the view layer with low integration cost and obtain a defensible operational baseline. The fact that no peer-reviewed Q1–Q2 paper has previously published a life-insurance ring-detection schema is, itself, the strongest motivation for the contribution.

Several extensions are natural. Real-data validation with carrier partners would promote the artifact to Gold reproducibility tier. Temporal extensions via time-weighted networks per GOTCHA [14] would catch rings that evolve across the contestability period rather than appearing as static structure. Cross-line generalisation to property and

workers' compensation would test whether the six-entity schema ports cleanly into other lines. GNN-based extensions per [11, 15] are a candidate paper-#3 or paper-#4 angle if carrier data becomes available. Privacy-preserving cross-carrier ring detection — federated graph analytics, differential privacy on shared edges — would unlock the long-standing problem that ring fraud routinely spans multiple carriers and no single carrier has the full graph.

The reference implementation is released as supplementary material under MIT license. Practitioners on the operational side of group life insurance are invited to fork the schema, run it against their own warehouses, and report findings.

---

## References

- [1] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 626–688, 2015. DOI: 10.1007/s10618-014-0365-y.
- [2] T. Pourhabibi, K.-L. Ong, B. H. Kam, and Y. L. Boo, "Fraud detection: A systematic literature review of graph-based anomaly detection approaches," *Decision Support Systems*, vol. 133, p. 113303, 2020. DOI: 10.1016/j.dss.2020.113303.
- [3] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "FRAUDAR: Bounding graph fraud in the face of camouflage," in *Proc. ACM KDD 2016*, pp. 895–904, 2016. DOI: 10.1145/2939672.2939747.
- [4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008. DOI: 10.1088/1742-5468/2008/10/P10008.
- [5] M. E. J. Newman, "Modularity and community structure in networks," *PNAS*, vol. 103, no. 23, pp. 8577–8582, 2006. DOI: 10.1073/pnas.0601602103.
- [6] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010. DOI: 10.1016/j.physrep.2009.11.002.
- [7] L. Šubelj, S. Furlan, and M. Bajec, "An expert system for detecting automobile insurance fraud using social network analysis," *Expert Systems with Applications*, vol. 38, no. 1, pp. 1039–1052, 2011. DOI: 10.1016/j.eswa.2010.07.143.
- [8] M. Óskarsdóttir, W. Ahmed, K. Antonio, B. Baesens, R. Dendievel, T. Donas, and T. Reynkens, "Social Network Analytics for Supervised Fraud Detection in Insurance," *Risk Analysis*, vol. 42, no. 8, pp. 1872–1890, 2022. DOI: 10.1111/risa.13693.
- [9] M. Tumminello, A. Consiglio, P. Vassallo, R. Cesari, and F. Farabullini, "Insurance fraud detection: A statistically validated network approach," *Journal of Risk and Insurance*, vol. 90, no. 2, pp. 381–419, 2023. DOI: 10.1111/jori.12415.
- [10] I. Vorobyev, "Fraud risk assessment in car insurance using claims graph features in machine learning," *Expert Systems with Applications*, vol. 251, p. 124109, 2024. DOI: 10.1016/j.eswa.2024.124109.
- [11] S. Motie and B. Raahemi, "Financial fraud detection using graph neural networks: A systematic review," *Expert Systems with Applications*, vol. 240, p. 122156, 2024. DOI: 10.1016/j.eswa.2023.122156.
- [12] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, "NetProbe: A Fast and Scalable System for Fraud Detection in Online Auction Networks," in *Proc. WWW 2007*, pp. 201–210, 2007. DOI: 10.1145/1242572.1242600.
- [13] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens, "APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions," *Decision Support Systems*, vol. 75, pp. 38–48, 2015. DOI: 10.1016/j.dss.2015.04.013.
- [14] V. Van Vlasselaer, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens, "GOTCHA! Network-Based Fraud Detection for Social Security Fraud," *Management Science*, vol. 63, no. 9, pp. 3090–3110, 2017. DOI: 10.1287/mnsc.2016.2489.

- [15] B. Deprez, F. Vandervorst, W. Verbeke, T. Verdonck, and B. Baesens, "Network analytics for insurance fraud detection: a critical case study," *European Actuarial Journal*, vol. 14, pp. 965–990, 2024. DOI: 10.1007/s13385-024-00384-6.
- [16] J. Debener, V. Heinke, and J. Kriebel, "Detecting insurance fraud using supervised and unsupervised machine learning," *Journal of Risk and Insurance*, vol. 90, no. 3, pp. 743–768, 2023. DOI: 10.1111/jori.12427.
- [17] J. M. Hellerstein et al., "The MADlib analytics library or MAD skills, the SQL," *Proc. VLDB Endowment*, vol. 5, no. 12, 2012. DOI: 10.14778/2367502.2367510.
- [18] R. J. Bolton and D. J. Hand, "Statistical Fraud Detection: A Review," *Statistical Science*, vol. 17, no. 3, pp. 235–255, 2002. DOI: 10.1214/ss/1042727940.
- [19] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, Cambridge University Press, 1994. ISBN: 978-0-521-38707-1.
- [20] C. Phua, V. Lee, K. Smith, and R. Gayler, "A Comprehensive Survey of Data Mining-based Fraud Detection Research," arXiv:1009.6119, 2010.

## Figure / Listing References

- **Figure 1.** Architecture diagram — DB2/Oracle source tables → recursive-CTE materialised view + edge list → 4-method ring-detection harness → flag writeback. *To be rendered from §3 architecture description at submission time.*
- **Figure 2.** Ring-signal feature × ring-detection method attribution heatmap. Generated by `run_ring_detection.py`. Cited in §4.2.
- **Figure 3.** Recall at k for the four methods at  $k \in \{25, 50, 100\}$ . Generated by `run_ring_detection.py`. Cited in §4.1.
- **Figure 4.** Operational false-positive cost per method — top-N quantile threshold vs ExCeeD-calibrated. Generated by `run_ring_detection.py`. Cited in §4.3.
- **Listing 1.** Recursive CTE for two-hop policy-cluster traversal via shared beneficiary. *Embedded in §3.3.*
- **Listing 2.** ExCeeD ring-level threshold-calibration block. *Embedded in §3.5.*

**Reproducibility statement.** The reference implementation (PL/SQL views, Python harness, synthetic-graph generator) is included in `artifacts/paper_2/`. Running `python generate_synthetic_life_graph.py` followed by `python run_ring_detection.py` reproduces Tab 2, Tab 3, and the per-method metrics within floating-point tolerance. Reference output values are pre-computed and shipped in `artifacts/paper_2/results/` for reviewers without a Python environment.

**Data availability.** Synthetic graph regenerable from seed 20260506. No real claims data is released. Carriers interested in promoting the artifact to Gold reproducibility tier are invited to contact the author.

**Conflict of interest statement.** The author declares no competing financial or non-financial interests.

**Author contributions.** P.K.V. designed the graph schema, wrote the SQL recursive-CTE views, implemented the Python four-method benchmark harness, ran the experiments, analysed the results, and wrote the manuscript.

**Funding.** This work was conducted independently and received no external funding.