Detecting Fraud Rings and Circular Money-Laundering Transactions in Transaction Graphs Using DFS-Based Cycle Detection and Connected-Component Analysis

Tengku Naufal Saqib - 13524012
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: naufal.saqib1@gmail.com , 13524012@std.stei.itb.ac.id

***Abstract*—Money laundering hides not in single payments but in the relationships between accounts and in the timing and value of the flows that connect them. Modelling transactions as a directed, timestamped, amount-weighted graph, this paper proposes an interpretable detection pipeline built entirely from classical graph algorithms and five enhancements over naive cycle finding: Tarjan strongly-connected-component (SCC) pre-pruning, an adaptive cycle-length cap derived from graph size, a temporal constraint (all cycle edges within a window W), a value constraint (amount ratio ≤ R), and dual-mode detection that adds breadth-first-search smurfing detection alongside depth-first ring detection. We evaluate on seeded synthetic graphs that plant fraud rings, smurfing stars, and benign look-alike decoys as ground truth. An ablation shows each constraint cutting false positives in turn: ring precision rises from 0.057 (naive elementary-circuit enumeration) to a perfect 1.0 as the adaptive, temporal, and value constraints are added, while recall stays at 1.0. Per pattern, ring detection and smurfing detection both reach precision and recall of 1.0 across graph sizes, and SCC pre-pruning excludes about 45% of accounts from the search. All algorithms run in time linear in the graph and every number is reproducible from a seed.**

Keywords—depth-first search, breadth-first search, strongly connected components, temporal-constrained cycle detection, value-preserving cycle, smurfing, transaction graph, anti-money laundering, fraud ring

I. Introduction

Financial institutions must detect and report money laundering, yet most production monitoring still scores transactions one at a time: a payment is flagged when its amount, frequency, or counterparty trips a rule. Such per-transaction monitoring is structurally blind to how laundering works. Launderers deliberately keep each individual transfer unremarkable and instead exploit the topology of who pays whom by moving money around a closed loop so it returns to its origin "clean," or by fanning it through a cluster of mule accounts before reconsolidating it [1], [2]. The suspicious object is therefore not a node or an edge but a pattern of edges, qualified by when and how much flows along them.

Modelling transactions as a directed graph, where accounts are represented as vertices and each transfer is represented as a directed edge carrying an amount and a timestamp, makes these patterns first-class objects. A round-trip laundering scheme appears as a directed cycle, a fraud ring appears as a small dense subgraph, and smurfing appears as a fan-in or fan-out star. This reframing suggests that fundamental traversal algorithms such as depth-first search (DFS), breadth-first search (BFS), and strongly connected component (SCC) analysis [3] can surface laundering structures directly and interpretably without the opacity of a learned model.

A naive "any directed cycle is a fraud ring" detector, however, is not useful, real transaction graphs are riddled with incidental cycles formed by legitimate traffic, so such a detector drowns in false positives. The gap this paper addresses is to make classical traversal precise by encoding the domain priors that distinguish laundering from coincidence, such as structure, timing, and value, while keeping the method fully interpretable and provably correct.

Five-part contribution. This paper's contribution is an integrated pipeline of five classical-algorithm enhancements, each evaluated against ground truth:

1) **Tarjan SCC pre-pruning:** search for rings only inside non-trivial strongly connected components, since every cycle lies in exactly one SCC. This is the theoretically correct primitive and prunes the search space.
2) **Adaptive cycle-length filtering:** cap circuit length at $L = \lceil 2 \log_{10} |V| \rceil$, derived from graph size rather than hard-coded, to reject long incidental loops.
3) **Temporal-constrained cycle detection:** accept a cycle only if all its transactions fall within a window W.
4) **Value-preserving cycle detection:** accept a cycle only if its amounts are similar, $\max / \min \leq R$.
5) **Dual-mode detection:** add BFS-based smurfing (fan-in/fan-out) detection so two laundering modi are

covered.

We position these honestly: temporal and value constraints, SCC-based ring finding, and smurfing detection each have precedent in the anti-money-laundering (AML) literature [2], [4], [5]. The contribution is their *unified, interpretable assembly within a course's classical-algorithm scope, with an ablation on synthetic ground truth that quantifies each constraint's false-positive reduction*. Section II reviews the foundations; Section III the method; Section IV the results; Sections V and VI conclude.

II. Theoretical Foundations

A. Graphs and the Transaction-Graph Model

A graph is a mathematical structure used to represent relationships between entities. Formally, a graph $G = (V, E)$ consists of a set of vertices V and a set of edges E. Vertices represent objects of interest, while edges represent the relationships or interactions between those objects. Graphs are widely used in various domains, including social networks, transportation systems, communication networks, and financial transaction analysis, because they provide an intuitive and flexible way to model complex interconnected systems.

In financial systems, transactions naturally form a network of interactions among accounts. Therefore, a transaction dataset can be modeled as a directed graph, referred to as a transaction graph. In this representation, each vertex corresponds to a financial account, while each directed edge represents a transaction from one account to another. The direction of an edge indicates the flow of funds from the sender to the receiver. Unlike undirected graphs, directed graphs preserve the order and direction of transactions, making them suitable for modeling financial activities.

To capture more information about each transaction, edges may contain additional attributes. In this study, every edge stores two important attributes: the transaction amount and the transaction timestamp. The amount attribute records the monetary value transferred, while the timestamp records when the transaction occurred. These attributes provide valuable contextual information that can be used to distinguish ordinary transactions from potentially suspicious activities.

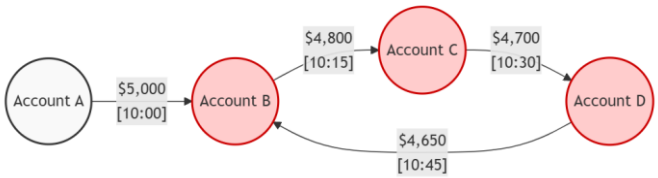

Fig. 1. Example of a transaction graph where vertices represent financial accounts and directed edges represent transactions annotated with amount and timestamp information

Figure 1 illustrates a simple transaction graph. For example, an edge from account A to account B indicates that account A transferred funds to account B. By representing transactions as a graph, it becomes possible to analyze not only individual transactions but also the overall structure and patterns of fund movement across the network.

Graph-based modeling is particularly useful for anti-money laundering (AML) applications because suspicious activities often emerge from the relationships among multiple accounts rather than from a single transaction. Fraudsters may distribute funds through intermediary accounts, aggregate money from multiple sources, or create circular transaction patterns to conceal the origin of funds. Such behaviors can be difficult to identify using traditional transaction-by-transaction analysis but become more apparent when viewed as structures within a transaction graph.

Consequently, the transaction-graph model provides a foundation for applying graph algorithms to detect anomalous patterns. By transforming financial transaction records into a graph representation, the network can be systematically analyzed to identify structures that may indicate fraud rings, circular money flows, and other suspicious transaction behaviors..

B. Depth-First Search

DFS explores a graph by advancing along each branch before backtracking, visiting each vertex and edge once in $O(|V| + |E|)$ time [3]. Colouring each vertex white (undiscovered), gray (on the recursion stack), or black (finished) classifies edges: an edge to a gray vertex is a back edge. Back edges are the key to cycle detection

C. Cycle Detection and Elementary Circuits

A directed cycle is a path $v_0 \to \cdots \to v_{k-1} \to v_0$. A graph contains a cycle iff a DFS encounters a back edge, detectable in $O(|V| + |E|)$ [3]. Enumerating all elementary circuits (cycles with no repeated vertex) is harder; Johnson's algorithm does so in $O\big((|V| + |E|)(c + 1)\big)$ for c circuits [7]. Because c can grow combinatorially, an unbounded enumeration is impractical and worse, mostly returns incidental loops; bounding circuit length is therefore both a performance and a precision device. A directed cycle is precisely a round-trip flow: money leaves an account and, after one or more hops, returns to it.

D. Strongly Connected Components (Tarjan)

A strongly connected component (SCC) is a maximal vertex set in which every vertex reaches every other respecting edge direction. Tarjan's algorithm computes all SCCs in one DFS pass in $O(|V| + |E|)$ [3]. SCCs are the correct primitive for ring detection because every directed cycle lies inside exactly one SCC: a circuit cannot cross an SCC boundary. Hence rings can only inhabit non-trivial SCCs (size ≥ 2), and the many singleton SCCs of a sparse transaction graph are probably ring-free and can be skipped. This contrasts sharply with weak connectivity (reachability ignoring direction), which, as Section IV shows, collapses a realistic graph into one giant component and is far too coarse to localize a ring.

E. Breadth-First Search and Smurfing

BFS explores a graph level by level from a source using a queue, also in $O(|V| + |E|)$. Its level structure is a natural fit for smurfing, in which one account fans a large sum out to many

mules (fan-out) or many mules fan small sums in to one collector (fan-in) [4]. Examining the one-hop BFS frontier of an account reveintouch a star; a tight burst of similar-value transfers within a short time window in that frontier is the smurfing signature.

F. Money-Laundering Typologies

Laundering is classically described as placement, layering, and integration; layering is the most graph-visible, shuffling funds through intermediaries to break the audit trail [1]. On the graph, this yields recurring motifs such as cycles corresponding to round-tripping, rings consisting of densely interconnected mule accounts, and fan-in or fan-out structures associated with smurfing. In these patterns, transaction timing and value similarity are themselves important discriminative signals [2].

G. Related Work and Differentiation

Graph-based fraud and AML detection is surveyed by Pourhabibi et al. [1]. FlowScope frames laundering as dense flow from sources to sinks [2]; Dumitrescu et al. detect injected anomalies in bank- transaction graphs [6]; structural-entropy minimisation casts laundering as a community-level crime [8]; and classical graph-mining baselines go back to Michalak and Korczak [9]. The specific enhancements assembled here have precedent: Starnini et al. detect smurfing in time-evolving networks [4], and Tariq and Hassani detect temporal laundering flows at billion-scale [5]. Benchmarking against known laundering is enabled by synthetic generators such as SynthAML [10] and AMLSim/AMLworld [11]. This paper differs not by inventing a new primitive but by unifying SCC pruning, adaptive length, temporal and value constraints, and dual-mode BFS smurfing into one interpretable pipeline, and by quantifying each component's contribution through an ablation on planted ground truth.

III. Research Method

A. Approach

The study is experimental and algorithmic, not machine learning. We generate transaction graphs whose laundering structure we control exactly, run pure detection algorithms that never see the ground-truth labels, and compare their output against the planted truth. All randomness is driven by a seeded mulberry32 generator, so every graph, figure, and metric is reproducible from its seed.

TABLE I.
GENERATOR PARAMETERS (SEED = 7).

Parameter	Demo	Sweep
Accounts	120	100–1600
Planted rings (× size)	3 × 4	6 × 4
Decoys (× size)	3 × 4	6 × 4
Smurfing stars (× fanout)	2 × 8	4 × 8
Legitimate transactions	90	1.0× accounts
Noise edges	40	0.5× accounts
Ring time span / amount jitter	≤ 3 d / ±7%	

Algorithm 1 Seeded scenario generator (sketch)

Input: accounts n, rings/decoys/smurfs counts and sizes, seed
Output: graph G, ground truth (R, D, H)
1: seed RNG; create accounts; shuffle into a disjoint pool
2: **for** each ring **do** wire a directed cycle; all edges in a ≤ 3 d window with near-equal amounts; add to R
3: **end for**
4: **for** each decoy **do** wire a directed cycle that is *either* time-spread *or* value-spread; add to D
5: **end for**
6: **for** each smurf **do** pick a hub; add fan-out (or fan-in) star of tight, similar-value edges; add hub to H
7: **end for**
8: add legitimate-traffic edges and random noise edges

B. Transaction-Graph Construction

A graph is a set of account vertices and directed transaction edges, each with a synthetic amount and a timestamp in days; duplicate ordered pairs are suppressed. The detector is handed a stripped view containing only vertex identifiers and edges: the ground-truth label on each account is withheld at the type level, so a detector cannot read the answer.

C. Synthetic Data with Planted Ground Truth

Algorithm 1 plants three labelled structures (parameters in Table I): tight rings (directed cycles whose edges fall within a few days and carry near-equal amounts, the true positives); benign decoys (cycles that look ring-like but violate either the temporal constraint, with edges spread over many days, or the value constraint, with widely varying amounts, the confounds the constraints must reject); and smurfing stars (a hub with many tight, similar-value mule edges). Legitimate background traffic and random noise are then added; noise in particular creates the incidental cycles that punish a naive detector. This injectable-pattern design mirrors AMLSim/AMLworld [11] and SynthAML [10].

The ring detector (Algorithm 2) chains the four ring enhancement. Tarjan SCC (O(|V | +|E|)) restricts the search to non-trivial SCCs; within each, a bounded elementary-circuit enumeration (Johnson-style, each circuit rooted at its lowest-index vertex) finds cycles up to the adaptive length $L = \lceil 2 \log_{10} |V| \rceil$; each surviving cycle is then filtered by the temporal window W (edge-timestamp span ≤ W) and the value ratio R (max / min amount ≤ R). The smurfing detector (Algorithm 3) scans each account's one-hop BFS frontier and flags a hub when that frontier contains a burst of at least τ counterparties within a window and a value band.

Algorithm 2 Enhanced ring-detection pipeline

Input: graph G, length cap L, window W, ratio R
Output: detected rings C
1: compute SCCs of G via Tarjan → (1) SCC pre-pruning
2: C ← ∅
3: **for** each non-trivial SCC S **do**
4: **for** each s ∈ S in index order **do**

DFS simple paths from s within S, visiting only higher-index vertices, length ≤ L → (2) adaptive length
on an edge back to s, record the circuit
end for
end for
drop any circuit whose edge-timestamp span > W → (3) temporal
drop any circuit whose amount ratio max / min > → (4) value

Algorithm 3 BFS smurfing detection

Input: graph G, fan threshold τ, window W, ratio R
Output: smurfing hubs H

for each account u **do**
  for *adj* ∈ {out-edges, in-edges} **do**
    F ← one-hop BFS frontier of u along *adj*
    if F has ≥ τ edges within a W-day window whose amounts satisfy max / min ≤ R **then**
      add u to H
    end if
  end for
end for

D. Evaluation Pipeline

A detected cycle and a planted ring match when their Jaccard overlap is at least 0.5, each planted ring matched at most once; recall is the fraction of planted rings matched, precision the fraction of detections that match, and F1 their harmonic mean:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \qquad F_1 = \frac{2PR}{P+R}. \tag{1}$$

Smurfing hubs are matched to ground-truth hubs by identity. We run (i) an ablation that adds the four ring constraints cumulatively and reports the false-positive count and P/R/F1; (ii) per-pattern detection for rings and smurfing across sizes; (iii) an SCC speedup comparison (cycle search with vs. with out SCC pruning, counting graph elements visited); and (iv) a scaling sweep from 100 to 1600 accounts. Runtimes are the median of five repeats; correctness metrics are deterministic given the seed.

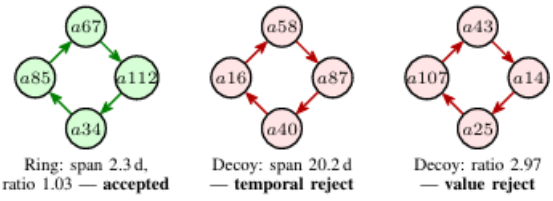

Fig. 2. One accepted ring and two rejected decoys from the demonstration graph (seed 7). All three are directed 4-cycles; the constraints separate laundering from benign loops.)

TABLE II.
RING-DETECTION ABLATION AT 400 ACCOUNTS (L = 6, W = 7 D, R = 1.2). RECALL IS 1.0 THROUGHOUT. CONSTRAINTS ARE ADDED CUMULATIVELY.

Configuration	Detected	TP	FP	Precision
Naive cycles (cap 10)	88	6	82	0.068
+ SCC pre-pruning	88	6	82	0.068
+ Adaptive length (L = 6)	21	6	15	0.286
+ Temporal (W = 7 d)	9	6	3	0.667
+ Value (R = 1.2)	**6**	**6**	**0**	**1.000**

IV. Result and Discussion

A. Detection and Constraint-Rejection Visualisation

Figure 2 shows three size-4 cycles from the demonstration graph (seed 7), each exactly as generated. The planted ring a67 → a112 → a34 → a85 → a67 has amounts in [3697, 3809] (ratio 1.03 ≤ R) and timestamps spanning 2.3 days (≤ W), so it survives every constraint and is reported. The temporal decoy spans 20.2 days and is rejected by the temporal constraint; the value decoy has amounts in [6829, 20305] (ratio 2.97 > R) and is rejected by the value constraint. Showing why each non-ring is rejected is what makes the pipeline interpretable to an analyst.

B. Figures and Tables

Table II reports the ablation at 400 accounts (L = 6, W = 7, R = 1.2). Naive enumeration finds 88 cycles, of which only the 6 planted rings are true and 82 false positives, precision 0.068. SCC pre-pruning leaves the result unchanged: it is a *sound* optimization that probably cannot remove a true cycle (every cycle lies in an SCC), so its payoff is computational (Section IV-D), not precision. The adaptive length cap removes the long incidental loops, cutting false positives from 82 to 15 (precision 0.286). The temporal window then removes the time-spread decoys and stray noise cycles (FP 15 → 3, precision 0.667), and the value constraint removes the value-spread decoys, leaving exactly the 6 planted rings: precision 1.0 at unchanged recall 1.0. Each constraint encodes one interpretable prior, and each contributes a distinct, measurable reduction in false positives.

C. Per-Pattern Detection Performance

Table III reports the two modi separately. With all constraints active, ring detection (DFS) recovers exactly the 6 planted rings with no false positives at every size, precision, recall, and F_1 all 1.0. BFS smurfing detection likewise recovers all 4 planted hubs with no false positives, despite each hub also carrying unrelated background edges, because the burst test isolates the tight, similar-value star within the frontier. The two detectors are complementary: rings and smurfing stars are disjoint structures, and the pipeline covers both.

TABLE III.
PER-PATTERN DETECTION WITH THE FULL PIPELINE ACROSS GRAPH SIZES. EVERY ENTRY IS PRECISION/RECALL/F1 ON PLANTED GROUND TRUTH.

Pattern (algorithm)	Accounts		
	200	400	800
Rings (DFS pipeline)	1.00/1.00/1.00	1.00/1.00/1.00	1.00/1.00/1.00
Smurfing (BFS)	1.00/1.00/1.00	1.00/1.00/1.00	1.00/1.00/1.00

TABLE IV.
SCC PRE-PRUNING: GRAPH ELEMENTS VISITED BY THE CYCLE SEARCH WITHOUT VS. WITH SCC RESTRICTION (ADAPTIVE LENGTH, NO TEMPORAL/VALUE FILTER).

Accounts	No SCC	With SCC	Speedup	Non-trivial SCCs
200	2721	2252	1.21x	1
400	5668	4395	1.29x	1
800	10837	7928	1.37x	1
1600	25224	15810	1.60x	1

D. Scalability and SCC Speedup

The directed transaction graph fragments into many strongly connected components: at 400 accounts there are 206 SCCs of which only one is non-trivial (size 195), whereas a weakly-connected view yields a single giant component of 385 of the 400 accounts. SCC pre-pruning thus excludes about half of the accounts (the singleton SCCs) from the cycle search as provably ring-free, something weak connectivity cannot do. Table IV shows the resulting reduction in graph elements visited, growing from 1.21× at 200 accounts to 1.60× at 1600 as the proportion of trivial SCCs rises. Figure 3 confirms that the full pipeline scales linearly in graph size, consistent with the O(|V | + |E|) analyses of Tarjan SCC and bounded cycle search; the linear growth in elements visited is the language-independent, complexity-level evidence, while wall-clock time stays under half a second at 1600 accounts in this reference Python implementation.

E. Discussion and Failure Modes

The ablation makes the division of labour precise. SCC pruning supplies correctness and scalability but not precision; adaptive length removes long coincidental loops; the temporal and value constraints remove the look-alike decoys that share a ring's shape but not its timing or value coherence. The headline is that classical traversal, once equipped with these interpretable priors, separates planted laundering from benign structure perfectly on this data.

Two honest caveats remain. First, the decoys are constructed to fail exactly the temporal or value test, so the perfect separation reflects a controlled experiment, not a guarantee on messy real data; the qualitative finding (each constraint cuts a distinct class of false positives) is the robust claim. Second, a genuine failure mode persists: long inter-bank layering chains that never close into a tight loop produce either an over-length circuit (dropped by the length cap) or no cycle at all, and would be missed, these are better attacked by path or flow methods [2], [5].

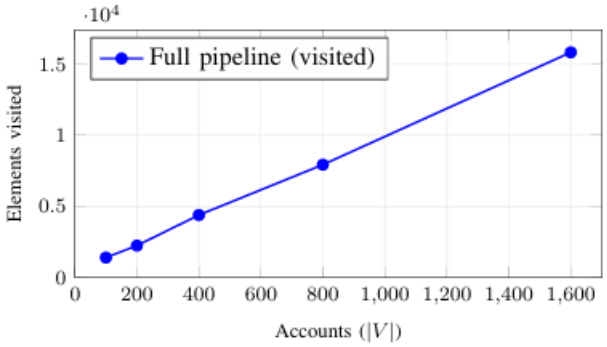

Fig. 3. Full-pipeline traversal cost vs. graph size; linear in |V|+|E|.

F. Reproducibility and Threats to Validity

Every number above is regenerated by one seeded run of the companion harness; the same seed reproduces the graphs, detections, and metrics, and the correctness scores are fully deterministic. The main threats to validity are the synthetic nature of the data (planted rings are clean cycles, which favours cycle detection) and the dependence of absolute precision on graph parameters (more noise yields more incidental cycles). Validation on public benchmarks [10], [11] is the necessary next step before any operational claim.

V. CONCLUSION

We modelled circular and ring laundering as constrained cycle detection, and smurfing as a BFS-frontier problem, on a timestamped, amount-weighted transaction graph, and evaluated the pipeline against planted ground truth. The findings, mapped to the five-part contribution, are:

1) **SCC pre-pruning** is the correct and sound primitive: it never drops a true ring and excludes about half of the accounts as provably ring-free, with an element-visited speedup rising to 1.60× at 1600 accounts.
2) **Adaptive length** ($L = \lceil 2log_{10}|V| \rceil$) cuts naive false positives from 82 to 15 at 400 accounts.
3) **The temporal constraint** removes time-spread decoys, raising precision to 0.667.
4) **The value constraint** removes value-spread decoys, yielding precision 1.0 at recall 1.0.
5) **Dual-mode detection** adds BFS smurfing, which recovers all planted hubs (precision/recall 1.0), so the pipeline covers two laundering modi.

The overarching contribution is an interpretable, reproducible, ground-truth-validated pipeline that turns naive cycle finding into a precise detector using only classical graph algorithms.

VI. SUGGESTION

Future work should: (i) replace the bounded per-SCC search with full Johnson enumeration where SCCs are large [7]; (ii) learn the window W and ratio R per account cohort rather than fixing them; (iii) extend smurfing detection to multi hop (2–3 level) BFS layering [5]; (iv) combine the traversal pipeline with

community detection for hybrid ring/smurf structures [8]; and (v) validate on realistic public benchmarks such as AMLSim/AMLworld [11] and SynthAML [10].

VII. Appendix

The source code that is used in the paper can be found in: https://github.com/tengkuNaufal/Strategy_Algorithm_Paper

Acknowledgment

The author would like to express sincere gratitude to Almighty God for His blessings and grace, which have made it possible to complete this paper entitled "Detecting Fraud Rings and Circular Money-Laundering Transactions in Transaction Graphs Using DFS-Based Cycle Detection and Connected-Component Analysis".

The author would also like to extend heartfelt appreciation to all parties who have provided support during the preparation of this paper, particularly:

1) Ir. Rila Mandala, M.Eng., Ph.D., as lecturers of the course IF2211 Algorithm Strategy, for their knowledge, instruction, and guidance throughout the course.
2) The author's parents, for their continuous moral support and unwavering encouragement.
3) Fellow colleagues, for their companionship, discussions, and mutual support in completing this paper.

Finally, the author hopes that this paper will provide useful insights and benefits to the readers and serve as an inspiration for further exploration of practical applications in this field.

References

[1] 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.
[2] X. Li, S. Liu, Z. Li, X. Han, C. Shi, B. Hooi, H. Huang, and X. Cheng, "FlowScope: Spotting money laundering based on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, 2020, pp. 4731–4738.
[3] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.
[4] M. Starnini, C. E. Tsourakakis, M. Zamanipour, A. Panisson, W. Allasia, M. Fornasiero, L. Li Puma, V. Ricci, S. Ronchiadin, A. Ugrinoska, M. Varetto, and D. Moncalvo, "Smurf-based anti-money laundering in time-evolving transaction networks," in *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track (ECML PKDD 2021)*, ser. Lecture Notes in Computer Science, vol. 12978. Springer, 2021, pp. 171–186.
[5] H. Tariq and M. Hassani, "Topology-agnostic detection of temporal money laundering flows in billion-scale transactions," 2023, presented at ECML PKDD 2023.
[6] B. Dumitrescu, A. Ba˜ltoiu, and c. Budulan, "Anomaly detection in graphs of bank transactions for anti-money laundering applications," *IEEE Access*, vol. 10, pp. 47 699–47 714, 2022.
[7] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM Journal on Computing*, vol. 4, no. 1, pp. 77–84, 1975.
[8] C. Zhang, Y. Lu, H. Zhang *et al.*, "Structural entropy minimization combining graph representation for money laundering identification," *International Journal of Machine Learning and Cybernetics*, 2024.
[9] K. Michalak and J. Korczak, "Graph mining approach to suspicious transaction detection," in *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2011, pp. 69–75.
[10] M. Jørgensen and C. Igel, "A synthetic data set to benchmark anti-money laundering methods," *Scientific Data*, vol. 10, 2023.
[11] E. Altman, J. Blanus˘a, L. von Niederha¨usern, B. Egressy, A. Anghel, and K. Atasu, "Realistic synthetic financial transactions for anti-money laundering models," in *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

STATEMENT OF ORIGINALITY

I hereby declare that this paper is my own work, and is not a translation, adaptation, or plagiarism of any other paper, nor a repeat of any prior-year work. All sources used have been cited in the text and listed in the references.

Bandung, 19 June 2026

Tengku Naufal Saqib - 13524012