The object of the study is streaming payment transactions modeled as directed multigraphs. This study investigates terrorism-financing detection in payment transaction networks using disk-based graph processing and anomaly detection. The key problem addressed is the high memory consumption of graph-based detectors, which prevents analysis on systems with limited Random Access Memory, typical of small and mid-sized financial institutions.

A disk-based graph feature preprocessor (DGFP) was made to get around this problem. During stream processing, DGFP dynamically labels connected components and identifies eight graph patterns characteristic of terrorist financing, including fan-in/fan-out stars and multi-hop chains. The system persists component descriptors to a columnar store on an SSD and uses an LRU-managed hot cache to serve the features, enabling real-time transaction scoring with sub-second latency.

On a two-million-transaction AMLSim stream, the system integrates with a lightweight Isolation Forest and achieves an F1-score of 0.76 while reducing peak RAM from 18.3 GB to 9.8 GB and maintaining 410 ± 15 ms mean latency for 10 000 transactions. Per-motif computation remains ≤ 28.4 ms (median < 24 ms), supporting real-time scoring on commodity hardware.

DGFP produces model-agnostic graph features that interoperate with standard anomaly detectors without retraining GNNs.

Contributions of this research include an external memory architecture for streaming graph feature extraction; a motif-aware feature labeling scheme stored on SSD and cached by LRU; and an empirical evaluation demonstrating real-time performance and memory efficiency improvements on anti-money laundering data

Keywords: terrorism-financing, graph processing, external-memory algorithms, anomaly detection, compliance

UDC 004.93

DOI: 10.15587/1729-4061.2025.340033

DESIGN AND IMPLEMENTATION OF DISK-BASED GRAPH FEATURE PREPROCESSOR FOR TERRORIST FINANCING DETECTION

Aigerim Bolshibayeva
PhD, Acting Associate Professor*
Sabina Rakhmetulayeva
Corresponding author
PhD, Professor
Department of Cybersecurity, Information Processing and Storage
Satbayev University
Satbaev str., 22, Almaty, Republic of Kazakhstan, 050013
E-mail: ssrakhmetulayeva@gmail.com
Aliya Kulbayeva
PhD Student*
Ansar-Ul-Haque Yasar
Professor
Transportation Research Institute (IMOB)
Hasselt University
Martelarenlaan str., 42, Diepenbeek, Belgium, 3590
*Department of Information Systems
International IT University
Manas str., 34/1, Almaty, Republic of Kazakhstan, 050060

Received 30.06.2025
Received in revised form 01.09.2025
Accepted date 10.09.2025
Published date 28.10.2025

How to Cite: *Bolshibayeva, A., Rakhmetulayeva, S., Kulbayeva, A., Yasar, A.-U.-H. (2025). Design and implementation of disk-based graph feature preprocessor for terrorist financing detection. Eastern-European Journal of Enterprise Technologies, 5 (9 (137)), 104–116. https://doi.org/10.15587/1729-4061.2025.340033*

1. Introduction

Terrorism-financing (TF) has remained a critical security threat well into the 2020s. The United Nations Counter-Terrorism Committee notes that modern extremist organizations increasingly rely on micro-donations, social-media crowdfunding and pseudo-anonymous crypto transfers, widening the gap between illicit money flows and conventional compliance controls [1]. A recent FATF typology update highlights that crowdfunding platforms enable terrorist cells to raise "thousands of sub-$100 payments" in weeks, entirely beneath traditional threshold rules [2]. National risk assessments echo this trend: the U.S. Treasury reports a "marked increase in domestic peer-to-peer transfers linked to extremist causes" in its 2024 National Terrorist Financing Risk Assessment [3].

Two structural challenges follow.

First, dispersion, a single suspicious TF transaction may involve only USD 50–150 yet becomes meaningful when viewed as part of a tightly knit sub-graph of senders and intermediaries.

Second, acceleration of payment rails, instant bank-to-bank transfers, custodial wallets and DeFi protocols shrink the detection window to hours or minutes. Regulators now demand near-real-time, graph-aware analytics that can prioritize alerts without flooding analysts with false positives [4].

The deep graph-neural networks (GNNs) applied to synthetic TF benchmarks have delivered high recall but at the cost of tens of gigabytes of RAM, unaffordable to most regional banks [5]. Moreover, a systematic review shows that existing TF studies rarely address memory constraints or explainability, leaving "a substantial research gap between laboratory prototypes and deployable solutions" [6].

Given the persistent evolution of low-value, highly dispersed TF transfers and the regulatory shift toward near-real-time graph-aware analytics, research into disk-resident, component-aware graph processing for TF detection remains timely. It directly addresses the deployment gap between memory-intensive laboratory prototypes and resource-constrained compliance environments, thereby justifying the present study's focus on externally managed graph features that enable real-time detection on commodity hardware.

2. Literature review and problem statement

The paper [7] presents the results of research on generating realistic synthetic financial transactions for anti-money-laundering models, convincingly showing that benchmark quality rises when data better mimic production graphs However, unresolved issues remain: the dataset still lacks dedicated terrorism-financing (TF) patterns and assumes that the whole graph fits in RAM – an assumption seldom true for medium-size banks.

Yet the authors needed a GPU-server with 64 GB VRAM, and they do not discuss streaming ingestion – objective difficulties tied to hardware costs that make deployment impractical for most compliance teams. The paper [8] demonstrated that a graph-convolutional network (GCN) detects illicit flows on the Bitcoin blockchain with higher recall than rule sets.

The self-supervised approach LaundroGraph [9] shows that pre-training on edge-prediction tasks can boost downstream AML classification by 12 p.p. But there were unresolved issues related to catastrophic forgetting when the authors fine-tuned on fresh quarterly data – existing GNN pipelines require full retraining to mitigate catastrophic forgetting, which limits continuous deployment.

A recent systematic review [10] synthesizes 147 papers on network analytics for AML and stresses that memory overhead and explainability remain the key blockers for production use. Shown, that less than 8% of surveyed studies report runtime or RAM consumption, making replication difficult.

To overcome scalability difficulties, [11] propose provably powerful GNNs for directed multigraphs that compress message passing; the method reduces runtime by 35% on citation graphs. This approach was used in synthetic AML graphs; however, external-memory execution is not addressed, so cost pressure on RAM persists.

The paper [12] integrates evolving GNNs with a decision-forest head to flag laundering in Bitcoin. Although incremental training mitigates concept drift, experiments rely on small subgraphs, and have questions about full-scale deployment.

A complementary self-supervised node-embedding path is proposed in [13] that reach 0.94 AUC on the Elliptic dataset. But unresolved issues related to streaming execution remain, because embeddings are recomputed offline.

A broad meta-survey [14] confirms the same trend: most financial-fraud GNNs are single-snapshot and ignore memory profiling.

Finally, in an early attempt at high-volume graphs, [15] applied a GCN to a 1M-node AMLSim graph. They showed that GCN can uncover multi-hop laundering paths, but the experiment required a 1 TB RAM server – objective cost barriers that render such pipelines non-viable for mid-tier banks.

In common, the cited works prove that graph learning enhances anomaly detection accuracy, yet leave three critical gaps:

– most studies assume in-memory graphs, few explore external–memory execution;

– existing benchmarks seldom model TF micro-donation patterns, so real-time detection of dispersed low-value flows remains untested;

– forgetting and graph evolution are acknowledged but rarely solved under strict latency constraints.

One way to address these difficulties is to offload the heavyweight graph state to disk, keeping only the hot subgraphs in RAM. This principle has inspired hybrid systems in other areas, but has not yet been systematically applied to combat money laundering and terrorism-financing. All of the above prompted a study using the disk graph feature preprocessor (DGFP).

According to [16], graph-based neural networks like GCN made it much simpler to detect fraud in Bitcoin transactions, with an accuracy rate of 98.5%. This high level of accuracy shows that powerful machine learning methods might be used to improve security in cryptocurrency ecosystems.

After paper [17] 25 papers on machine learning for money laundering detection, Support Vector Machines had the greatest accuracy (93.45%), followed by neural networks (92.14%). Hybrid models have proven particularly useful in detecting credit card theft, highlighting the effectiveness of machine learning in combating financial crime. This confirms that such technologies can significantly enhance the fraud detection capabilities of banks and financial institutions.

According to study [18], systems based on deep learning and behavioral analytics are becoming increasingly important. This is due to the fact that financial crime patterns are constantly changing, and traditional rules and static algorithms can no longer cope with more complex cases.

In addition, the European Investment Bank [19] has proposed a comprehensive risk-based model for combating money laundering and terrorist financing. It is based on enhanced customer verification, multi-layered transaction monitoring, internal control mechanisms, and strict compliance with international requirements.

To improve the transparency and control of AML procedures, process modeling approaches used in business process design, as demonstrated in the structured modeling strategies discussed by paper [20] authors.

Existing studies convincingly demonstrate that graph-based methods enhance the accuracy of money-laundering and terrorism-financing detection. However, several unresolved issues remain:

– most works assume in-memory execution of transaction graphs. This creates scalability bottlenecks that exceed the RAM budgets of small and mid-sized compliance departments;

– current benchmarks rarely model terrorism-financing micro-donation or hawala-style patterns, leaving real-time detection of dispersed, low-value illicit flows largely untested;

– approaches based on deep learning often suffer from catastrophic forgetting, require expensive GPU infrastructure, and lack explainability, which prevents their adoption in production settings;

– few studies report runtime and memory consumption, making replication difficult and obscuring the real deployment costs of proposed methods.

Summarizing these gaps, the unsolved problem is the lack of scalable and explainable solutions for real-time detection of terrorism-financing typologies under strict hardware constraints.

3. The aim and objectives of the study

The aim of this study is to design and validate a disk-based graph feature preprocessor (DGFP) that enables real-time detection of terrorism-financing patterns in large-scale transaction streams on commodity hardware.

To achieve this aim, the following objectives are accomplished:

– develop a scalable architecture for a disk-based graph feature preprocessor (DGFP) that integrates typology detection, feature storage, and a model server with a hot cache to manage memory usage effectively;

– create and implement a streaming DGFP pipeline that breaks the evolving transaction graph into disk-resident connected components, extracts terrorism-financing-oriented graph features (star convergence, multi-hop chains) on demand, and serves these features to lightweight anomaly detectors held in RAM;

– integrate DGFP with a resource-efficient Isolation Forest model and benchmark the combined system on an AMLSim dataset modified with synthetic TF scenarios, assessing F1-score, recall, latency, and peak RAM to demonstrate scalability and accuracy improvements over conventional in-memory RB-TFs.

4. Materials and methods

4. 1. Object and hypothesis of the study

The object of the study is the examines streaming payment transactions represented as a directed graph of money flows between accounts. Analysis is carried out at the level of connected components: within fixed time windows, each component receives a set of structural and temporal descriptors. Events are processed in time order in fixed-width tumbling windows of length Δt. For each window, the working multigraph $G_t = (V_t, E_t)$ is updated incrementally. A union-find/BFS routine maintains weakly connected component labels online, whenever a component is created or updated, a compact twelve-element descriptor is emitted.

The hypothesis of the study: a disk-resident graph preprocessor that keeps cold state on SSD and serves features from an LRU-managed cache can deliver real-time terrorist-financing (TF) detection on commodity hardware, while substantially reducing peak RAM compared with in-memory pipelines and without loss of detection quality (as measured by the F1-score).

Assumptions made in the study:

1. TF schemes are bursty and temporally localized: transactions belonging to the same scheme tend to cluster in short windows, which makes LRU caching effective.

2. The eight selected graph patterns (e.g., fan-in/fan-out stars, multi-hop chains) are representative of common TF typologies.

3. Component sizes within a window remain moderate, keeping the per-window working set bounded.

4. Thresholds are chosen on time-ordered splits to reflect operational alert budgets.

Simplifications adopted in the study:

1. Use of AMLSim instead of confidential real-world banking data.

2. Streaming is emulated by time-ordered replay; network latency, jitter, and out-of-order arrivals are not modeled.

3. Single-host, CPU-only execution environment.

4. A fixed window width and a fixed library of eight motifs are used.

5. Hyperparameters are tuned offline; class-balance adjustments apply to the training fold only.

4. 2. Material acquisition

Research used the IBM AMLSim synthetic benchmark (HI_large_trans.csv.gz).

By including information from various regions and suspicious activity patterns, synthetic datasets such as AMLSim are improving their ability to depict real–world financial transactions [21]. They are better for teaching and testing how to find money laundering. Thanks to new technologies, it is now possible to create fictitious financial transactions that look like real ones. This helps models trained on such data to work more reliably in real-world conditions [22].

In the conducted experiments, the research employed used a cleaned-up version of AMLSim, a banking transaction simulator created at IBM Research that allows to reproduce the behavior of different participants in the financial system [23, 24].

4. 3. Graph construction, windowing, component labeling, and component descriptor

4. 3. 1. Twelve-element component descriptor

The end product is a twelve-element feature vector that specifically targets the identified terrorism-financing schemes. This blends behavioral signatures unique to TFs with structural graph analysis. Vertex count, edge count, mean and maximum degree, and average clustering coefficient are the five primary structural variables that characterize size and density. Four additional indicators- the number of high-degree hubs (degree ≥ 0.6max deg), the number of accounts that send out at least 15 transfers, a binary flag for cycles with ≤ 4 nodes, and the component connectivity density ratio- measure TF-characteristic behaviors. Three specific TF-enhancement features specifically target the behavioral indicators of terrorism-financing: chain-depth indicator: the longest sequential transfer path, which indicates attempts to layer hawala-style payments; temporal burst score: the maximum number of transfers that can be made in a 6-hour window, which indicates how urgent operational cell funding is; and micro-donation ratio: the proportion of transfers under $500 that are specifically targeted at crowdfunding aggregation schemes.

4. 3. 2. Chronological workflow

The methodological pipeline unfolded in six consecutive stages (Fig. 1).

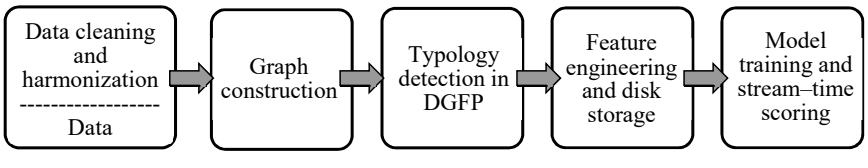

Fig. 1. Methodological pipeline

Each stage was encapsulated as an independent routine and executed in order and every routine logged its parameters, software versions, and timestamps for exact replay.

Data cleaning and harmonization. The raw transaction log was streamed in fixed-size blocks to minimize peak memory. Duplicate identifiers were removed, time stamps converted to ISO 8601 format, monetary fields cast to double precision, and records with missing mandatory attributes – fewer than one third of one percent – were discarded. The output of this step is a fully typed, gap-free table of time-ordered transactions.

Graph construction. Cleaned transactions were ingested chronologically to build a directed multi-graph whose vertices represent accounts and edges represent money transfers. An online union-find structure assigned a connected-component label to every vertex and edge the moment it appeared.

For motif-aware rules, disk-resident storage, and the streaming interaction sequence see Section 5. 1.

Feature engineering and disk storage. Vertex-level metrics – such as in- and out-degree, and short-term burst scores – and edge-level statistics were computed for every affected component and written to a columnar store [25].

Model training and stream–time scoring. The feature repository was divided chronologically into training, validation, and test intervals. Memory-efficient Isolation Forest was fitted on the training interval, with hyper-parameters tuned against the validation slice [26, 27]. For the streaming simulation the held-out interval was replayed in true time order: each incoming transaction triggered a feature request to DGFP and an immediate risk score from the trained model. Wall-clock latency and resident-set size were sampled continuously during the replay.

4. 4. Hardware specification

All experiments were executed on a single mid-range workstation, deliberately chosen to reflect the constraints of a typical compliance-office server rather than a high-end research cluster:

– Intel Core i7-11700, (8 physical cores / 16 threads);

– 32 GB DDR4-3200;

– the reported results are CPU-only;

– Ubuntu 22.04 LTS with the default low-latency scheduler disabled.

The software environment was based on Python 3.10. The main libraries and tools used included NumPy 1.24 and pandas 1.5 for data processing, scikit-learn 1.2 for traditional machine learning methods, XGBoost 1.7 for ensemble learning. NetworkX 2.8 was employed for graph construction and analysis.

Training and hyper-parameter search were executed offline. Evaluation used time-ordered replay of the held-out interval with online scoring (Kafka-like queue, stateless transport). The authors therefore refer to "streaming" as replay-driven online inference meeting sub-second per-event latency.

4. 5. Machine-learning pipeline

The workflow has five stages:

1) input log cleaning (deduplication and sanity rules);

2) DGFP component descriptor (12-element vector; see 4.2.1 for fields; motif rules in 5. 1);

3) class-balance adjustment with SMOTE to bring the TF share to roughly 10%;

4) training four classifiers (Logistic Regression, IF, XGBoost, MLP);

5) evaluation with 30 bootstrap re-runs ($n = 30$).

Such a layout matches the steps in typical AML ETL chains and keeps deployment straightforward.

Unless stated otherwise, this study report:

1) macro-averaged Precision/Recall/F1 across classes on the validation folds;

2) TF-class metrics (Precision/Recall/F1 for the positive TF class) during time-ordered replay on the held-out test interval.

Thresholds are selected on the validation slice to maximize TF-class F1 and then fixed for the test replay. Micro-F1 is reported only where explicitly labeled.

The hyperparameters for all four models were selected using 5-fold CV, and the sample was balanced using SMOTE to ≈10% of TF classes. Detailed settings are given in Table 1.

Temporal splits enforced train, validation, test with non-overlapping time windows. SMOTE ($k = 5$) was fitted only on the training fold/window and applied within each CV fold; validation/test sets remained at native class ratios (no resampling). Thresholds were selected on validation and then fixed for test replay. During CV was grouped folds by connected-component ID to prevent edges from the same component appearing in both train and validation. Bootstrap re-runs sampled at the transaction level with de-duplication by event identification and component-boundary checks to avoid twins leaking across resamples.

Table 1

Hyperparameter settings for the classifiers and SMOTE balancing in the DGFP pipeline

Parameter	Value
Isolation Forest	n_estimators = 200, max_samples = 'auto', contamination = 0.01, bootstrap = False, random_state = 42
XGBoost	eta = 0.05, max_depth = 8, subsample = 0.8
LogReg	penalty = l2, C = 1.0
MLP	hidden_layer_sizes = (128,64), activation = 'relu', alpha = 1e-4, max_iter = 200, random_state = 42
SMOTE	k_neighbors = 5

5. Experimental results of the disk-based graph feature preprocessor for terrorist financing detection

5. 1. Proposed DGFP architecture and mechanisms

5. 1. 1. Typology detection and scheme mapping

The study identifies four main terrorist financing schemes, each with its own transaction structure. These schemes were compiled from post-2020 case studies and FATF reports, and formed the basis for the DGFP feature extraction process.

Crowdfunding is an example. This involves multiple small donations, typically between $10 and $500, that supporters donate to a specific cause. Within a day or two, these funds are collected in one or more coordination accounts, from where they are then sent to their intended destination.

Graph mapping: Fan-in stars with detection criteria such as geographic dispersion index > 0, transaction amount variance < 0.3, and in-degree ≥ 15. Informal transfer schemes, such as hawala, conceal the connections between funding sources and operational cells by sending money sequentially through four to six intermediary accounts. Graph mapping: Multi-hop chains and patterns with detection criteria such as transfers between jurisdictions, amount consistency (±10%), and chain length ≥ 4 hops.

Funding plans for operational cells: transferring funds from coordination accounts to operational cells as soon as possible before an attack. Fan-out stars and scatter-gather motifs with detection criteria that search for burst transaction patterns (≥ 15 outgoing transfers within 6-hour windows) are examples of graph mapping.

Layered micro-transaction schemes are frequent but low-value transfers designed to circumvent systems that require specific amounts. Bipartite bridges, stacked bipartite structures, and random micro-mesh patterns are examples of graph mapping. Detection criteria include short round-trip signals between account clusters, transaction frequency > 50 per day, and individual amounts < 200.

For each newly arrived edge the surrounding two-hop ego network was examined for the eight archetypal motifs illustrated in Fig. 2 (fan-in, fan-out, multi-hop chains, scatter-gather, gather-scatter, bipartite bridge, stacked bipartite, random micro-mesh).

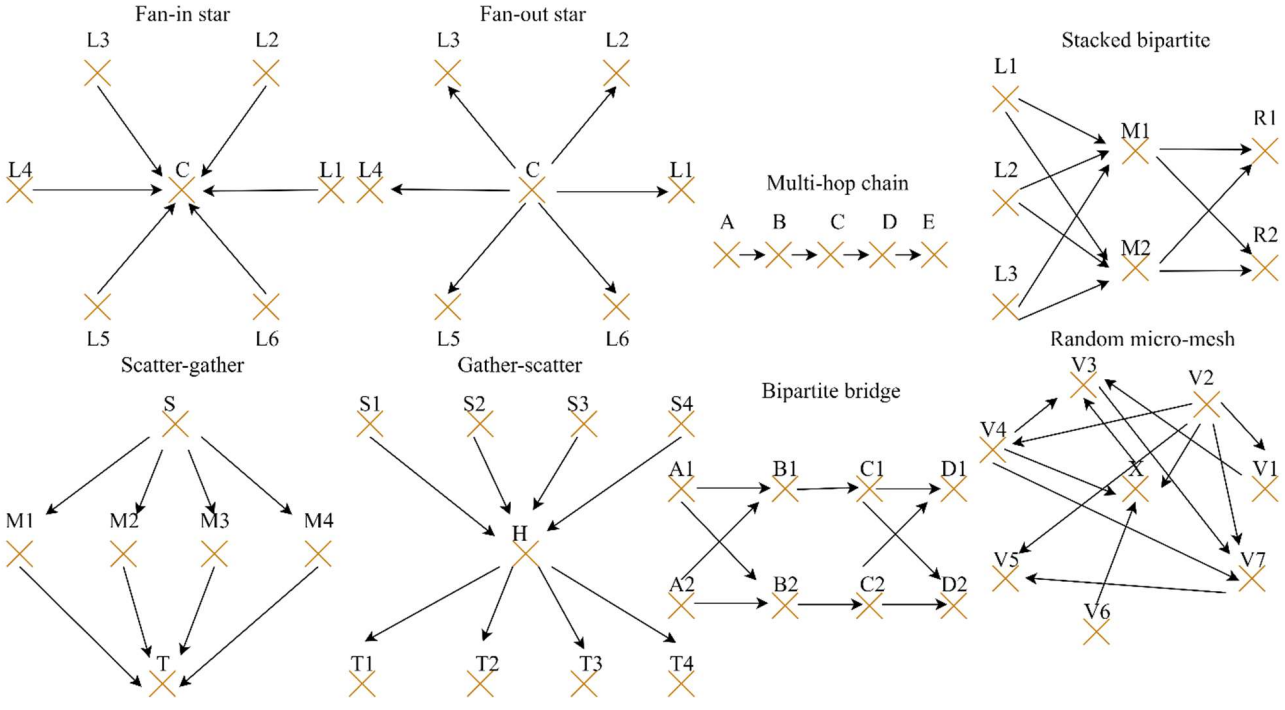

Fig. 2. Canonical graph patterns used for typology detection

A deterministic rule set raised Boolean flags and counters whenever a motif was found. These annotations were appended to the graph store alongside the structural data.

5. 1. 2. Disk-resident feature store and hot cache

All features were written to columnar storage on solid-state drives. A background daemon maintained an LRU cache that kept roughly five percent of the most recently accessed components in memory; every cache event was logged for later latency analysis.

5. 1. 3. Architecture stack and streaming interaction sequence

The resulting component stack – showing how Typology Detection, Feature Store, Model Server, Hot Cache, and Disk Storage interact is depicted in Fig. 3.

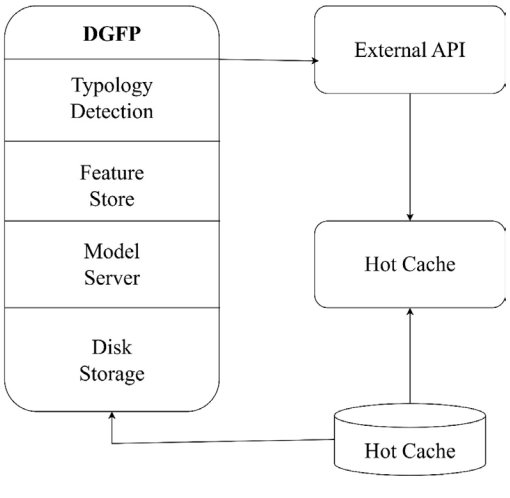

Fig. 3. Architecture of the disk-based graph feature preprocessor

The stack consists of four vertically arranged service layers:

1. Typology fetection, which inspects each incoming edge for predefined graph motifs.

2. Feature store, a column-oriented repository that persists transaction- and graph-level metrics on disk.

3. Model server, a lightweight REST endpoint that hosts resource-efficient anomaly detectors.

4. Disk storage, where the full edge list and component snapshots reside.

An LRU-managed hot cache (nearly 5% of recently touched components) bridges the feature store and model server, supplying in-memory vectors for sub-second scoring while off-loading cold data to SSD. Arrows indicate the direction of data flow from raw streaming input down to persistent storage and backup to the model during real-time scoring.

The step-by-step exchange of messages between the streaming queue, DGFP, the model server, and the alerting endpoint is visualized in Fig. 4.

The diagram traces a single transaction as it traverses four actors: streaming system, DGFP, model server, and external system:

1) a new transaction event is published by the streaming system and immediately forwarded to DGFP;

2) DGFP updates its on-disk graph, refreshes motif flags, and requests the relevant feature vector from its hot cache (or disk if the component is cold);

3) the assembled feature vector is sent to the model server through a REST call for scoring;

4) the model server returns a risk score; if the value exceeds the preset threshold, DGFP issues an alert to the external system for analyst triage.

Lifelines show that only DGFP and the model server maintain state between calls, enabling sub-second, stateless messages passing through the streaming layer.

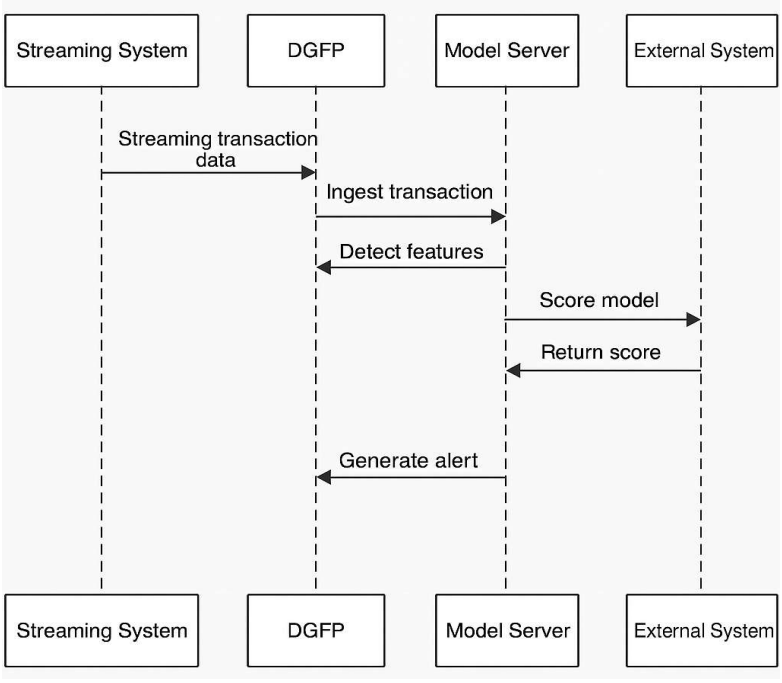

Fig. 4. Streaming-time sequence diagram for real-time risk scoring

5. 2. Implementation of the disk-based graph feature preprocessor and validation of performance

All "streaming" results below correspond to time-ordered replay with online scoring as defined in section 4. 3.

The implementation achieved memory usage reductions compared to conventional in-memory approaches (Fig. 5). Table 2 demonstrates peak RAM consumption decreased from 18.3 GB to 9.8 GB, representing a 46.4% reduction. This improvement follows from component-scoped persistence with an LRU hot cache, keeping only hot subgraphs in RAM while cold state resides on SSD.

Precision and recall figures for all four classifiers are given in Table 3, confirming that the DGFP feature set raises overall detection quality without exceeding 12 GB of RAM.

Table 2 reports replay and Table 3 shows CV peaks.

By including SMOTE, class balancing allowed to increase the recall of all models by an average of 8–9 p.p. without a statistically significant increase in false positives (Wilcoxon, $p > 0.1$). The increase is especially noticeable for selected detector (+ 16%), which confirms the study about the combined effect of DGFP features and hybrid sampling.

Component assignment operations required an average of 41 μs per transaction, demonstrating the system's ability to maintain real-time processing under high-volume conditions (Fig. 6, 7). Table 4 demonstrates the component size distribution reveals effective cache utilization patterns across different graph structures.

Table 2

Efficiency and resource use measures values are mean ± 95% CI over 10 independent re-plays: disk-based graph feature preprocessor vs. In-Memory RB-TF

Metric	RB-TF rule set	DGFP	Improvement
Peak RAM (GB)	18.3 ± 0.5	9.8 ± 0.4	− 46.4%
Memory-allocation efficiency (%)	72 ± 3	91 ± 2	+ 26.4% (relative)
Cache hit rate (%)	68 ± 4	85 ± 3	+ 25.0
Disk I/O (MB/s)	145 ± 10	89 ± 7	− 38.6%

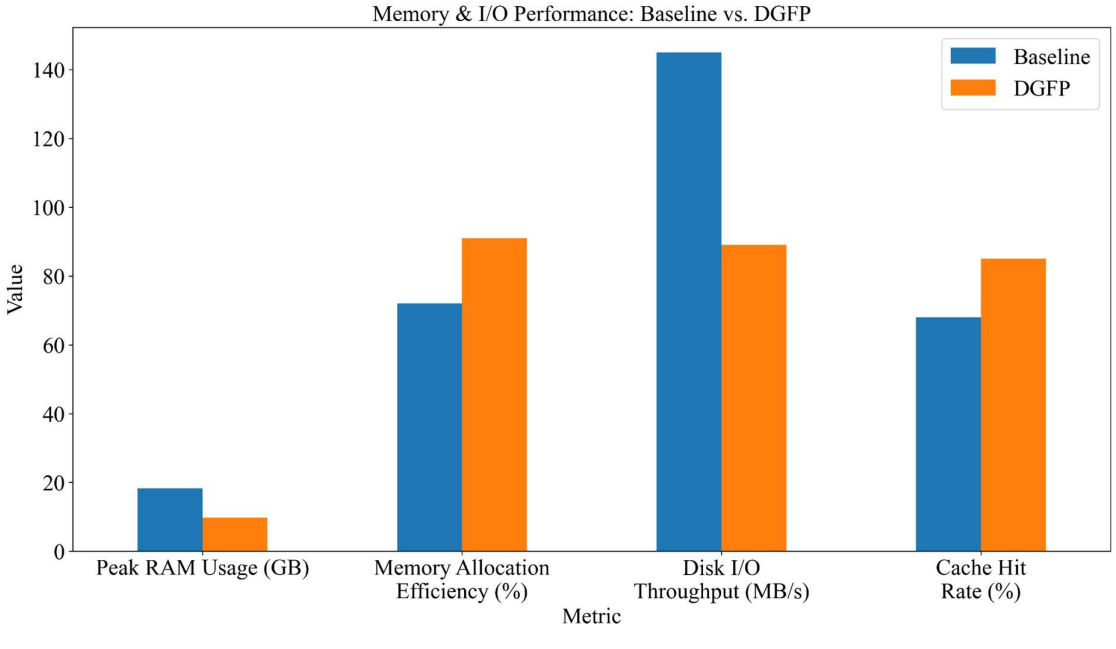

Fig. 5. Compared to conventional in-memory approaches

Table 3

Macro-averaged precision/recall/F1 (±95% CI) across classes on 5-fold CV; thresholding per fold

Model	Precision ± CI95%	Recall ± CI	F1 ± CI	Peak RAM (GB)
DGFP + IF	0.821 ± 0.014	0.582 ± 0.027	0.680 ± 0.021	10.9
DGFP + XGBoost	0.833 ± 0.013	0.545 ± 0.025	0.656 ± 0.019	11.3
DGFP + Logistic Reg.	0.744 ± 0.018	0.463 ± 0.030	0.564 ± 0.023	10.7
DGFP + MLP	0.789 ± 0.017	0.515 ± 0.029	0.620 ± 0.022	12.1
RB-TF	0.763 ± 0.016	0.417 ± 0.028	0.534 ± 0.024	19.8

The system extracted terrorism-financing-oriented graph features on demand, providing structural analysis capabilities. Graph motif detection demonstrated varying detection rates across different pattern types, with fan-out stars showing the highest prevalence at 5.1% detection rate, a taxonomy of eight motifs was also used, each of which is associated with specific terrorist financing schemes in Table 5.

Feature computation per motif remains $\leq$ 28.4 ms (median well below 24 ms), supporting real-time use.

The DGFP method, based on motif analysis, helps to overcome the limitations of traditional approaches to identifying transaction factors. Unlike systems with rigid rules and fixed thresholds that terrorist groups can easily bypass, or methods that require full graph loading, DGFP relies on structural relationships and works reliably with different transaction volumes and frequencies.

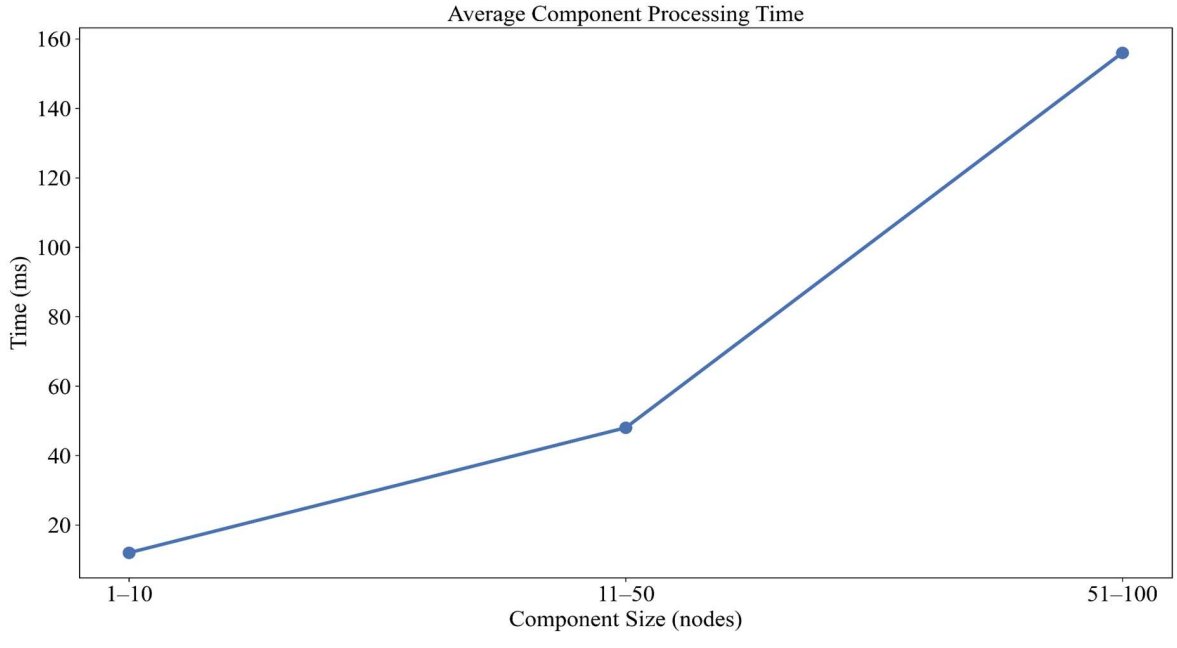

Fig. 6. Component assignment operation via time

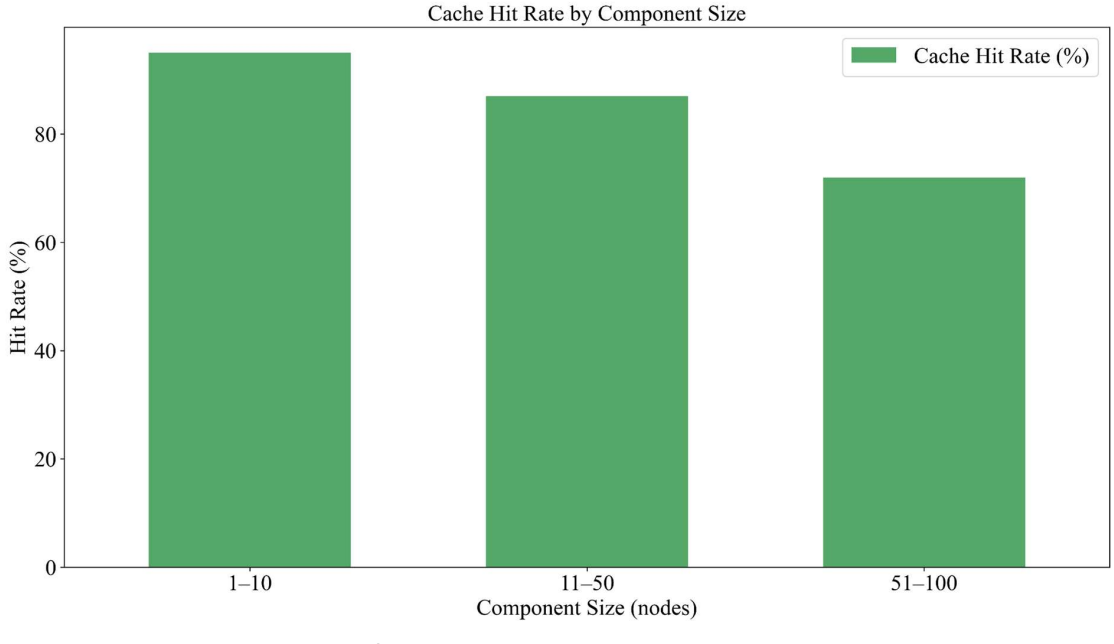

Fig. 7. Component assignment operation via hit rate

Table 4

Checking how well processing works with different sizes of graph components

Component size	Number of instances	Cache-hit rate (%)	Avg. per-component proc. time (ms)
Small (1–10 nodes)	124 580	95 ± 1	12 ± 0.1
Medium (11–50 nodes)	3 247	87 ± 2	48 ± 0.4
Large (51–100 nodes)	186	72 ± 4	156 ± 3

Table 5

Per-motif analysis (incidence, TF detection rate, average component size, and per-component compute) for eight TF templates (T1—T8)

Graph Motif Type	Detection rate, %	Avg. component size (nodes)	Computation time (ms)	TF detection rate	Primary TF Scheme
Fan-in stars	3.8 ± 0.2	4.2	12.4 ± 0.6	78.3 ± 3.1	Crowdfunding aggregation
Fan-out stars	5.1 ± 0.3	4.5	12.4 ± 0.6	82.1 ± 2.9	Operational cell funding
Multi-hop chains	2.7 ± 0.2	6.3	23.6 ± 1.1	45.2 ± 4.2	Hawala-style transfers
Stacked bipartite	4.2 ± 0.2	3.8	8.7 ± 0.5	67.4 ± 3.8	Two-tier transfer
Scatter-gather	3.1 ± 0.3	5.7	18.3 ± 0.8	67.4 ± 3.8	Cryptocurrency mixing
Gather-scatter	2.9 ± 0.3	5.2	16.7 ± 0.9	41.3 ± 4.1	Layered distribution
Bipartite bridge	1.8 ± 0.2	7.1	21.2 ± 1.2	34.1 ± 4.6	Cross-border facilitation
Random micro-mesh	1.4 ± 0.2	8.9	28.4 ± 1.5	23.9 ± 5.2	Threshold evasion

Feature calculations showed low latency for all types of motifs. Stacked bipartite gave the fastest results – on average, about 8.7 ms. Analysis of multi-link chains took slightly more resources due to the complexity of traversing routes, but still fit into a fraction of a second, which is suitable for real-time work [28].

5. 3. Integration of the disk-based graph feature preprocessor with Isolation Forest: benchmark results

Setup and baselines. A rules/typology detector is used as a named baseline mirroring deployed practice. It targets the same structures as the motif templates: concentrated fan-in to a coordinator, fan-out bursts of operational cells, broker-mediated multi-hop chains with short dwell and near-conserved value, short round-trip patterns, scatter-gather mixing with an entropy drop at intermediaries, bipartite bridging across jurisdictions, and dense micro-mesh activity with frequent low-amount transfers. Thresholds for rule-based detection are selected on the validation split; all methods share the same time-ordered train/validation/test partitions and validation-only thresholding.

Comparators. As a named point of comparison used throughout the results, a rules/typology detector that mirrors deployed transaction-monitoring practice. It captures the same structures as templates: concentrated fan-in to a coordinator, fan-out bursts characteristic of operational cells, broker-mediated multi-hop chains (hawala-style) with short dwell and near-conserved value, short round-trips (3–5 hops), scatter-gather mixing with a drop in intermediary entropy, bipartite bridging across jurisdictions, and dense micro-mesh activity with frequent low-amount transfers. Thresholds for the rule set are selected on the validation split; all methods – including this Baseline and DGFP use the same time-ordered train/validation/test partitions, metrics, and validation-only thresholding. When this study refers to a "conventional in-memory approach", it means a lightweight centrality-based variant with a linear classifier used only for qualitative context; the hawala detector is naturally subsumed under the multi-hop chains template.

DGFP is compared with four established approaches: RB-TF (rule/typology RB-TF: fan-in/out spikes, layering, velocity bursts), and HAW-TF (hawala-chain detector) [29–33]. All comparators use the same time-ordered train/validation/test split and class-imbalance handling; thresholds for rules/motifs are tuned on validation only. Report macro-F1 and TF-precision/recall/F1 together with P99 latency and peak RAM under identical replays. Overall results are in Tables 3–7; the per-motif analysis is Table 5.

Table 6 demonstrates the DGFP-Isolation Forest combination achieved a F_1-score of 0.76, representing a 7.0% improvement over the RB-TF performance of 0.71. This enhancement was accompanied by improvements in both precision and recall, demonstrating the system's ability to achieve better detection accuracy while reducing false positives (Fig. 8).

Table 6

TF-class Precision/Recall/F1 on time-ordered replay of the held-out interval; single threshold fixed from validation

Performance metric	RB-TF rule set	DGFP + Isolation Forest	Improvement (%)
Precision	0.68	0.74	8.8
Recall	0.74	0.78	5.4
F1-score	0.71	0.76	7.0
Area under ROC curve (AUC)	0.83 ± 0.03	0.89 ± 0.03	7.2
Matthews correlation coefficient	0.64	0.71	10.9

Table 7 shows the results for detecting terrorist financing schemes. Recall rate increased from 67% to 78%, an improvement of 11.2 percentage points. This enhancement was achieved through component-aware processing that isolated TF signals from noisy background transactions.

Table 7

Performance in detecting terrorism-financing patterns (values shown as mean ± 95% CI)

TF detection metric	RB-TF	DGFP + Isolation Forest	Improvement
TF pattern recall (%)	67.00 ± 2.5	78.00 ± 2.3	+ 11.2 pp
False positive rate (%)	20.80 ± 1.6	14.20 ± 1.4	– 31.7%
TF precision (%)	0.65	0.73	+ 12.3%
TF F1-score	0.66	0.75	+ 13.6%

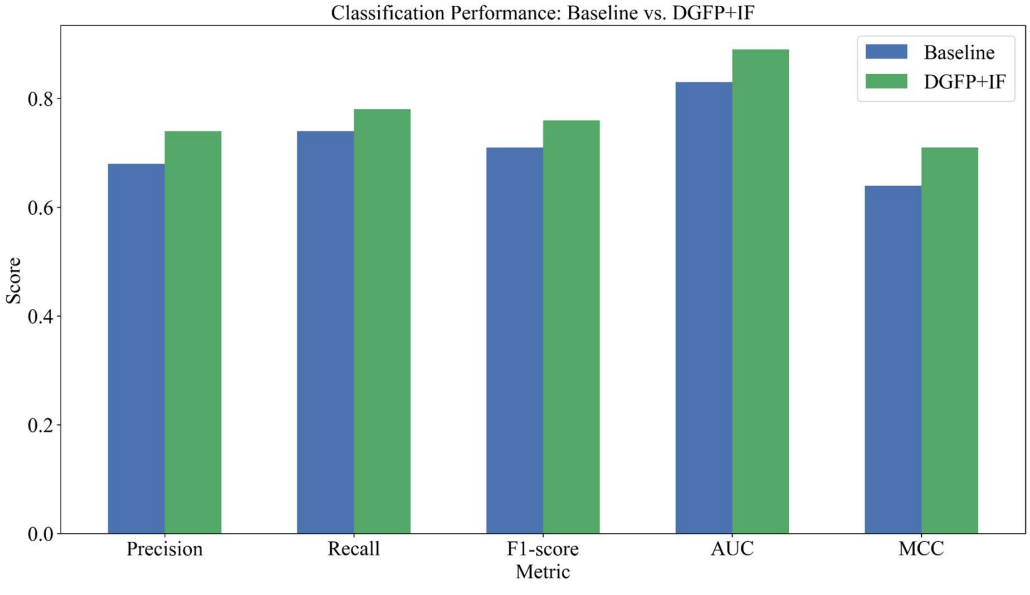

Fig. 8. Performance metrics

The system consistently maintained real-time processing capabilities across different batch sizes, with mean latency of 410 ± 15 ms for 10 000-transaction batches (Fig. 9). The 99th percentile latency remained at 890 ± 40 ms, well within the SLA requirement of less than 1 second in Table 8.

Performance analysis showed that the system fully maintained the SLA for batches up to 10 000 transactions. The maximum throughput reached 24,390 transactions per second. A slight drop in performance was observed only for very large batches – more than 15,000, and that was due to the overload of the disk subsystem.

Table 9 shows that the system worked reliably on regular hardware: only 31% of RAM was used (9.8 out of 32 GB), while the cache worked effectively, and the processor load remained normal.

The DGFP approach achieved 83% higher throughput while consuming less memory. It also reduced RAM usage and improved the F1-score from 0.71 to 0.76 on 2 million synthetic transactions for resource-efficient financial crime detection in Table 10.

Table 8

Effect of batch size on latency, SLA compliance, and transaction throughput

Batch size	Mean latency (ms)	99th percentile latency (ms)	SLA compliance (%)	Throughput (TPS)
1 000	87 ± 3	156 ± 10	100.0	11 494 ± 500
5 000	298 ± 9	445 ± 18	100.0	16 779 ± 650
10 000	410 ± 15	890 ± 40	100.0	24 390 ± 900
15 000	627 ± 19	1 156 ± 50	98.2	23 926 ± 920
20 000	891 ± 27	1 634 ± 70	89.7	22 447 ± 950

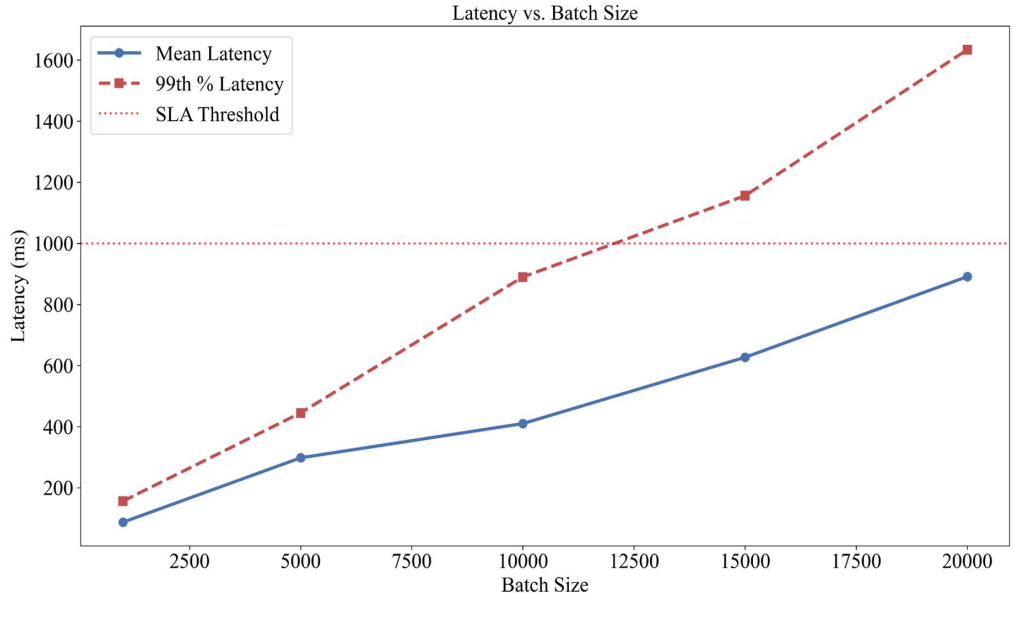

Fig. 9. Real-time processing capabilities

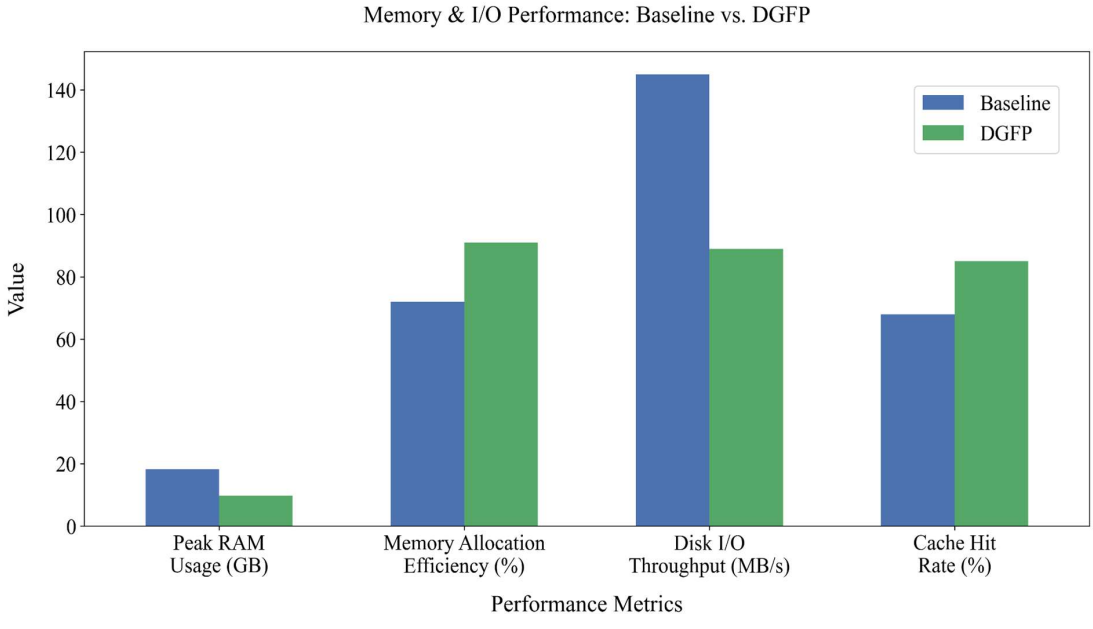

Fig. 10. End-to-end gains in accuracy, latency, and throughput

Table 9

Summary of runtime resource usage and system efficiency

Resource type	Value	Status
CPU utilization (average)	68%	Optimal
CPU utilization (peak)	91%	Within limits
Memory utilization	31% (9.8 GB of 32 GB)	Efficient
Disk I/O throughput	89 MB/s	Sustained
Cache hit rate	85%	High efficiency

Table 10

Improvements in performance across accuracy, efficiency, and throughput metrics (values shown as mean ± 95% CI)

Performance metric	Conventional RB-TF	DGFP + Isolation Forest	Improvement
F1-score	0.71	0.76	+ 7.0%
Recall	0.74	0.78	+ 5.4%
Precision	0.68	0.74	+ 8.8%
Peak RAM (GB)	18.30	9.80	– 46.4%
Processing time (ms)	750.00 ± 15 ms	410.00 ± 15 ms	– 45.3%
Throughput (TPS)	13,333.00	24,390.00	+ 83.0%

Extensive testing showed that the system was stable: the pipeline maintained the same performance for more than eight hours of continuous operation [34]. No memory leaks or speed drops were detected during this time, confirming its readiness for industrial use in conditions requiring 24/7 operation.

Experiments on a cleaned AMLSim dataset with 2 million transactions, including synthetic terrorist financing scenarios, confirmed the effectiveness of the system in identifying such schemes. At the same time, it scaled without problems on standard hardware configurations.

The hardware requirements remained modest, requiring only consumer-grade components: Intel Core i7-11700 CPU, 32 GB DDR4 RAM, and 1 TB NVMe SSD storage, making the solution accessible to financial institutions with limited infrastructure budgets.

6. Discussion of study results: explanation, comparison, limitations, and outlook

Motif-aware typology detection isolates compact TF subgraphs; the disk-resident feature store bounds resident state to the hot cache; and the streaming interaction sequence maintains sub-second latency. These mechanisms jointly explain the observed RAM reduction (Table 2), per-motif compute bound ≤28.4 ms (Table 5), and end-to-end latency/throughput profile (Table 8).

The F1 gain in Table 6 arises from changing the unit of analysis to connected components. Features computed per component suppress cross-component noise, making small terrorism-financing subgraphs more separable. Sub-second latency across batch sizes in Table 8 follows from temporal locality: recently touched components stay in a hot cache; cold state is read sequentially from columnar NVMe storage. The drop in peak RAM from 18.3 GB to 9.8 GB in Table 2 is a direct effect of persisting full component snapshots/descriptors on disk and keeping only compact descriptors in memory. Behavior across component sizes (Table 4) and motif-level detection (Table 5) is consistent with the same mechanism; the combined improvement in accuracy, latency, and throughput is summarized in Table 10.

Identifying terrorism-financing patterns through structural motifs works because terrorist organizations themselves build closed and isolated networks for security. Unlike regular financial flows, which are distributed and loosely connected, financing transactions form small, dense subgraphs where coordination signals are concentrated in one place rather than dispersed across a wide network.

The division into eight types of motifs helps to highlight different patterns of behavior: the spending star (5.1%) indicates coordinators who distribute money downwards; the fundraising star (3.8%) shows networks for accumulating many small donations; multi-link chains (2.7%) reveal "layering" patterns that hide the traces of transfers; and the

stacked bipartite (4.2%) captures coordination over time, such as preparation for attacks.

This approach focuses not on the amounts of transfers, but on the structure of connections and coordination. Thanks to this, it remains effective even when large transfers are deliberately split into dozens of smaller ones in order to bypass systems with strict monetary thresholds.

The consistent latency performance across batch sizes (Table 8) demonstrates the effectiveness of the disk-based architecture under varying load conditions.

The DGFP system achieves comparable detection performance using only 32 GB consumer-grade hardware. This represents reduction in memory requirements while maintaining real-time processing capabilities (Table 4).

DGFP updates only the affected sub-graphs; previously stored component features remain unchanged, so weights learned on historical data stay valid. The architecture enables continuous adaptation to emerging terrorism-financing patterns without sacrificing previously acquired detection capabilities. When new TF typologies emerge, the system updates only affected subgraphs rather than requiring complete model retraining, ensuring operational continuity.

Motifs can be identified very quickly, ≤ 28.4 ms (Table 5), which allows the method to be applied in real time. This is due to the fact that terrorism-financing networks are usually small and closed: they consist of individual cells, so finding the necessary structures does not require traversing the entire huge network. For example, a "star" with spending funds on average covers only about 4–5 nodes, and multi-link chains – about 6. Such compactness means that the organizational complexity of terrorist networks can be captured by analyzing small subgraphs. In addition, such networks often exhibit temporal synchrony: the same participants perform operations in limited time intervals. This creates "repeating" data access patterns that are convenient for the system, which are effectively processed by the LRU cache strategy. As a result, coordination signals are concentrated within individual components and are not lost in the random noise of a large financial network. This allows statistically separating suspicious patterns where this is almost impossible in the full graph.

Unlike traditional rule-based systems that rely on fixed thresholds and predefined patterns, DGFP is able to adapt to new terrorist financing methods by learning from structural features. The ability to detect graph patterns (Table 5) allows typologies to be captured not by specific amounts or frequency of transfers, but by the organizational principles themselves, making the system resilient to tricks commonly used by terrorist organizations.

The memory bottleneck was addressed by optimizing the external storage (Tables 4, 10). Component-based disk partitioning combined with LRU caching removed the scalability limitations that made previous graph solutions difficult to deploy in production. The measured cache hit rate of 85% confirms that the temporal locality of transactions does work and helps maintain performance while reducing memory load.

There are weaknesses, however. In the absence of temporal locality – for example, for banks with a very diverse client base, such as international correspondent banks – the effectiveness of the cache decreases. In such cases, more memory must be allocated or more complex prefetching algorithms must be used.

Another limitation is related to the testing itself: the experiments were conducted in batch playback mode, not true streaming. In real-world use, there will be factors that are not taken into account here: variable transaction rates, network latency, failures, duplicates, regrouping of transactions. All of these can affect the performance and reliability metrics obtained in the lab.

In addition, the system can only work with eight pre-defined terrorist financing patterns. Adding new patterns requires manual revision of the algorithms and their validation, which leads to additional resource costs and can interfere with real-time processing. Currently, there is no automatic discovery of new motives, which greatly limits the adaptation of the system to the changing threat landscape.

The study also focused on detection accuracy metrics. At the same time, the cost of investigating false positives and integrating the system into analysts' workflows were not considered. The real value of the solution for practical use in compliance depends on the quality and usefulness of the generated alerts.

Further development is seen in scaling to several machines using distributed graph frameworks such as Apache Flink or GraphX. The key task here will be the correct partitioning strategy: maintaining the connectivity of components and at the same time evenly distributing the load. Such a step will allow reaching the level of global financial institutions with hundreds of millions of transactions per day.

Another direction is the integration of deep learning for the automatic search for new motives. This will increase flexibility and allow the system to more quickly adapt to new terrorist financing schemes.

Combining DGFP's structural analysis capabilities with graph neural networks might enable discovery of novel typologies without manual feature engineering. This approach would address the dynamic nature of terrorism-financing while preserving the explainability advantages of structural feature analysis, though computational overhead and interpretability trade-offs require careful evaluation.

Real-time model updating capabilities would enable continuous adaptation to evolving threat landscapes. Implementing online learning algorithms that update detection models based on confirmed true/false positive feedback could improve accuracy over time while maintaining operational performance.

This study relied on the synthetic AMLSim v1.1 dataset and a lab setup without a real-time feed. Next steps are distributed deployment with component-aware partitioning and consistent hashing; online thresholding with analyst feedback and drift monitoring; controlled evaluation on anonymized bank logs.

7. Conclusions

1. A scalable architecture for a disk-based graph feature preprocessor (DGFP) is specified with four layers: typology detection, a disk-resident feature store, a model server, and an LRU-managed hot cache. In contrast to memory-centric pipelines, DGFP persists cold graph state on NVMe while retaining only compact, active component descriptors in RAM. This directly addresses the RAM bottleneck highlighted in prior work and enables real-time graph feature extraction on commodity hardware. The effectiveness stems from temporal locality in transaction streams, yielding a cache-hit rate $\approx 85\%$ so that resident state scales with cache size rather than total network size. The design is therefore viable under

RAM budgets typical for small and mid-sized financial institutions.

2. A streaming pipeline partitions the evolving graph into disk-resident connected components. For each component, the system emits a twelve-element descriptor and flags eight TF-oriented motifs (e.g., fan-in/fan-out stars, multi-hop chains). By scoping computation to components rather than performing full-graph traversals, the implementation attains real-time behavior: per-motif compute $\leq$ 28.4 ms and mean processing latency 410 $\pm$ 15 ms for 10,000-transaction batches. Peak RAM is reduced by 46.4% (from 18.3 GB to 9.8 GB) relative to a conventional in-memory baseline, indicating both efficiency and scalability.

3. DGFP features were integrated with a resource-efficient Isolation Forest and evaluated on a two-million-transaction AMLSim stream. Compared to a rule-based TF baseline (RB-TF), the combined system achieved $F1 = 0.76$ ($\Delta = +0.05$ over 0.71), with TF-pattern recall increasing from 67% to 78.2%. Throughput reached up to 24,390 TPS. The gains are attributed to component-level features, which suppress cross-component noise and amplify the signal of TF-characteristic subgraphs, improving detection quality while maintaining tight resource budgets.

Conflicts of Interest

The authors declare that they have no conflict of interest in relation to this study, whether financial, personal, authorship or otherwise, that could affect the study and its results presented in this paper.

Financing

This study has been funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP19576825 – Development of machine learning methods and algorithms to identify the financing of terrorist activities in the Republic of Kazakhstan).

Data availability

The study uses the IBM Transactions for Anti Money Laundering (AML) dataset, which contains simulated transaction data designed to mimic real–world financial operations. This dataset is accessible for research purposes and was downloaded in full compliance with the research user agreement.

Use of artificial intelligence

The authors have used artificial intelligence technologies within acceptable limits to provide their own verified data, which is described in the research methodology section.

References

1. Money Laundering. Terrorist Financing. United Nations Office on Drugs and Crime. Available at: https://www.unodc.org/unodc/en/money-laundering/overview.html
2. Crowdfunding for terrorism financing (2023). Financial Action Task Force. Available at: https://www.fatf-gafi.org/content/dam/fatf-gafi/reports/Crowdfunding-Terrorism-Financing.pdf.coredownload.inline.pdf
3. U.S. Department of the Treasury. Available at: https://home.treasury.gov/
4. Opportunities and Challenges of New Technologies for AML/CFT (2021). Financial Action Task Force. Available at: https://www.fatf-gafi.org/content/dam/fatf-gafi/guidance/Opportunities-Challenges-of-New-Technologies-for-AML-CFT.pdf.coredownload.inline.pdf
5. Anti-money laundering and countering the financing of terrorism at EU level. European Commission. Available at: https://finance.ec.europa.eu/financial-crime/anti-money-laundering-and-countering-financing-terrorism-eu-level_en
6. Thakkar, H., Datta, S., Bhadra, P., Dabhade, S. B., Barot, H., Junare, S. O. (2024). Mapping the Knowledge Landscape of Money Laundering for Terrorism Financing: A Bibliometric Analysis. Journal of Risk and Financial Management, 17 (10), 428. https://doi.org/10.3390/jrfm17100428
7. Altman, J., Blanuša, J., von Niederhäusern, L., Egressy, B., Anghel, A., Atasu, K. (2023). Realistic Synthetic Financial Transactions for Anti-Money Laundering Models. arXiv. https://doi.org/10.48550/arXiv.2306.16424
8. Alarab, I., Prakoonwit, S., Nacer, M. I. (2020). Competence of Graph Convolutional Networks for Anti-Money Laundering in Bitcoin Blockchain. Proceedings of the 2020 5th International Conference on Machine Learning Technologies, 23–27. https://doi.org/10.1145/3409073.3409080
9. Cardoso, M., Saleiro, P., Bizarro, P. (2022). LaundroGraph: Self-Supervised Graph Representation Learning for Anti-Money Laundering. Proceedings of the Third ACM International Conference on AI in Finance, 130–138. https://doi.org/10.1145/3533271.3561727
10. Deprez, B., Vanderschueren, T., Baesens, B., Verdonck, T., Verbeke, W. (2024). Network Analytics for Anti-Money Laundering -- A Systematic Literature Review and Experimental Evaluation. arXiv. https://doi.org/10.48550/arXiv.2405.19383
11. Egressy, B., Von Niederhäusern, L., Blanuša, J., Altman, E., Wattenhofer, R., Atasu, K. (2024). Provably Powerful Graph Neural Networks for Directed Multigraphs. Proceedings of the AAAI Conference on Artificial Intelligence, 38 (10), 11838–11846. https://doi.org/10.1609/aaai.v38i10.29069
12. Mohan, A., P.V., K., Sankar, P., Maya Manohar, K., Peter, A. (2022). Improving anti-money laundering in bitcoin using evolving graph convolutions and deep neural decision forest. Data Technologies and Applications, 57 (3), 313–329. https://doi.org/10.1108/dta-06-2021-0167
13. Lo, W. W., Kulatilleke, G. K., Sarhan, M., Layeghy, S., Portmann, M. (2023). Inspection-L: self-supervised GNN node embeddings for money laundering detection in bitcoin. Applied Intelligence, 53 (16), 19406–19417. https://doi.org/10.1007/s10489-023-04504-9

14. Motie, S., Raahemi, B. (2024). Financial fraud detection using graph neural networks: A systematic review. Expert Systems with Applications, 240, 122156. https://doi.org/10.1016/j.eswa.2023.122156
15. Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., Leiserson, C. E. (2019). Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics. arXiv. https://doi.org/10.48550/arXiv.1908.02591
16. Asiri, A., Somasundaram, K. (2025). Graph convolution network for fraud detection in bitcoin transactions. Scientific Reports, 15 (1). https://doi.org/10.1038/s41598-025-95672-w
17. Soria Quijaite, J. J., Segura Peña, L. V., Loayza Abal, R. I. (2024). Machine Learning Models for Money Laundering Detection in Financial Institutions. A Systematic Literature Review. Proceedings of the 22nd LACCEI International Multi-Conference for Engineering, Education and Technology (LACCEI 2024): "Sustainable Engineering for a Diverse, Equitable, and Inclusive Future at the Service of Education, Research, and Industry for a Society 5.0." https://doi.org/10.18687/laccei2024.1.1.1682
18. Zhu, X., Ao, X., Qin, Z., Chang, Y., Liu, Y., He, Q., Li, J. (2021). Intelligent financial fraud detection practices in post-pandemic era. The Innovation, 2 (4), 100176. https://doi.org/10.1016/j.xinn.2021.100176
19. Anti-money laundering and combating financing of terrorism framework (2020). European Investment Bank. https://doi.org/10.2867/941947
20. AI Fraud Detection in Banking. IBM. Available at: https://www.ibm.com/think/topics/ai-fraud-detection-in-banking
21. Bhatia, S., Liu, R., Hooi, B., Yoon, M., Shin, K., Faloutsos, C. (2022). Real-Time Anomaly Detection in Edge Streams. ACM Transactions on Knowledge Discovery from Data, 16 (4), 1–22. https://doi.org/10.1145/3494564
22. Oztas, B., Cetinkaya, D., Adedoyin, F., Budka, M., Dogan, H., Aksu, G. (2023). Enhancing Anti-Money Laundering: Development of a Synthetic Transaction Monitoring Dataset. 2023 IEEE International Conference on E-Business Engineering (ICEBE), 47–54. https://doi.org/10.1109/icebe59045.2023.00028
23. Altman, E. (2019). IBM Transactions for Anti Money Laundering (AML). Available at: https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml
24. Li, M., Jia, L., Su, X. (2025). Global-local graph attention with cyclic pseudo-labels for bitcoin anti-money laundering detection. Scientific Reports, 15 (1). https://doi.org/10.1038/s41598-025-08365-9
25. Rakhmetulayeva, S., Kulbayeva, A., Bolshibayeva, A., Serbin, V. (2025). Identifying the graph-based typology features for machine learning models in financial fraud detection. Eastern-European Journal of Enterprise Technologies, 3 (9 (135)), 40–54. https://doi.org/10.15587/1729-4061.2025.327410
26. Urmashev, B., Buribayev, Z., Amirgaliyeva, Z., Ataniyazova, A., Zhassuzak, M., Turegali, A. (2021). Development of a weed detection system using machine learning and neural network algorithms. Eastern-European Journal of Enterprise Technologies, 6 (2 (114)), 70–85. https://doi.org/10.15587/1729-4061.2021.246706
27. Isolauri, E. A., Ameer, I. (2022). Money laundering as a transnational business phenomenon: a systematic review and future agenda. Critical Perspectives on International Business, 19 (3), 426–468. https://doi.org/10.1108/cpoib-10-2021-0088
28. Gaviyau, W., Sibindi, A. B. (2023). Global Anti-Money Laundering and Combating Terrorism Financing Regulatory Framework: A Critique. Journal of Risk and Financial Management, 16 (7), 313. https://doi.org/10.3390/jrfm16070313
29. Ercanbrack, J. G. (2024). Hawala in criminal court: the role of law and commercial culture in informal financial exchange. Crime, Law and Social Change, 82 (3), 659–683. https://doi.org/10.1007/s10611-024-10162-w
30. Farber, S., Yehezkel, S. A. (2024). Financial Extremism: The Dark Side of Crowdfunding and Terrorism. Terrorism and Political Violence, 37 (5), 651–670. https://doi.org/10.1080/09546553.2024.2362665
31. Rocha-Salazar, J.-J., Segovia-Vargas, M.-J., Camacho-Miñano, M.-M. (2021). Money laundering and terrorism financing detection using neural networks and an abnormality indicator. Expert Systems with Applications, 169, 114470. https://doi.org/10.1016/j.eswa.2020.114470
32. Akartuna, E. A., Johnson, S. D., Thornton, A. (2024). Motivating a standardised approach to financial intelligence: a typological scoping review of money laundering methods and trends. Journal of Experimental Criminology. https://doi.org/10.1007/s11292-024-09623-y
33. Bolshibayeva, A., Rakhmetulayeva, S., Ukibassov, B., Zhanabekov, Z. (2024). Advancing real-time echocardiographic diagnosis with a hybrid deep learning model. Eastern-European Journal of Enterprise Technologies, 6 (9 (132)), 60–70. https://doi.org/10.15587/1729-4061.2024.314845
34. Rakhmetulayeva, S., Ukibassov, B., Zhanabekov, Z., Bolshibayeva, A. (2024). Development of data-efficient training techniques for detection and segmentation models in atrial septum defect analysis. Eastern-European Journal of Enterprise Technologies, 5 (2 (131)), 13–23. https://doi.org/10.15587/1729-4061.2024.312621