International Journal of Scientific Research in Engineering & Technology
Volume6, Issue6 (November- December 2026), PP: 44-49.
https://www.doi.org/10.59256/ijsreat.20260606007
www.ijsreat.com
ISSN No: 2583-1240

Anti-Money Laundering Tracker Using Graph-AI

Hima J Ajjapla[1], Manasa M[2], Megha M[3], Nanditha N[4], Dr. Arun Kumar G. H[5]
[1, 2, 3, 4]Department of Information Science and Engineering, Bapuji Institute of Engineering and Technology, Davangere, Karnataka, India.
[5]Assistant Professor, Department of Information Science and Engineering, Bapuji Institute of Engineering and Technology, Davangere, Karnataka, India.

To Cite this Article: *Hima J Ajjapla[1], Manasa M[2], Megha M[3], Nanditha N[4], Dr. Arun Kumar G. H[5], " Anti-Money Laundering Tracker Using Graph-Ai ", International Journal of Scientific Research in Engineering & Technology, Volume 06, Issue 06, November-December 2026, PP:44-49.*

Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution License; Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract: *Money laundering schemes can involve transfers through several connected accounts, so suspicious activity may be difficult to recognize by examining each transaction separately. This paper presents an Anti-Money Laundering (AML) Tracker that combines graph-pattern analysis with graph representation and machine-learning methods to support the examination of potentially suspicious transaction networks. Accounts are represented as nodes and transfers as directed relationships in Neo4j. The project uses the IBM AMLSim synthetic dataset, with approximately 9,945 accounts and 68,289 transactions reported in the project materials. Following data validation and preprocessing, account and transaction records are loaded into the graph database. Cypher queries are used to explore circular transfers, layering-like chains, fan-in and fan-out structures, and multi-hop paths. The described graph-AI pipeline includes FastRP embeddings, Random Forest classification, and GraphSAGE-based neighbourhood learning. Graph visualization helps users inspect connected accounts and candidate suspicious paths. The functional checks documented for the project show that selected graph operations and pattern queries produced their expected outputs. These checks do not demonstrate predictive accuracy; no verified numerical model-evaluation results are available for reporting. The system is intended to assist investigators, not to determine criminal activity automatically. Future evaluation should document model settings, use held-out data, measure false positives, and assess whether findings from synthetic data generalize to real financial transactions.*

Keywords: *Anti-money laundering, graph analytics, Neo4j, AMLSim, graph learning.*

I. INTRODUCTION

Financial institutions handle large numbers of transfers between accounts. Money laundering may involve a sequence of transactions designed to hide where funds originated or where they are going. When transactions are reviewed one at a time, connections among accounts can be overlooked. Looking at transaction paths and groups of connected accounts can help reveal circular flows, chains of transfers, and repeated relationships. For this reason, transaction monitoring can benefit from examining both individual transaction details and the wider network.

A graph represents entities and the relationships between them, making it suitable for examining connected financial activity. In this project, each account is modelled as a node and each transfer as a directed relationship. Neo4j stores the transaction network, and Cypher queries are used to explore paths and structural patterns. Graph-analysis and machine-learning methods are intended to complement these queries by representing account relationships and identifying patterns associated with potential risk.

The project uses the IBM AMLSim synthetic dataset. The project records describe approximately 9,945 accounts and 68,289 transaction relationships. The aim is to build a graph-based tracker that highlights potentially suspicious transaction structures and presents them in a form that can be inspected. An alert is an indicator for further investigation, not proof of money laundering; the surrounding context and evidence must be reviewed by a person.

II. RELATED WORK

Research on anti-money-laundering detection has examined graph learning at scale, including work using graph-based methods to study financial transaction networks [10]. Graph convolutional networks have also been investigated for financial forensics in Bitcoin transaction data [11]. These studies support the use of relationships between entities as a source of information for detecting suspicious activity. The IBM AMLSim project provides synthetic transaction data for developing and testing AML approaches [8].

Neo4j supports graph storage and traversal, while Cypher provides a query language for working with nodes, relationships, and paths [3], [4]. The Neo4j Graph Data Science library offers graph algorithms and machine-learning functions [5], and PyTorch Geometric provides tools for developing graph-learning models [6]. Other studies have considered graph neural networks and methods for handling imbalanced financial-fraud data [2], [7], [9]. Apache Kafka can be used to process event streams, although its presence in an architecture does not by itself demonstrate that live transaction streaming

has been implemented [1].

Graph-pattern rules are relatively easy to inspect, but legitimate activity can produce similar structures and therefore trigger false alerts. Learned models may identify additional patterns, but their results are meaningful only when the training setup and evaluation are clearly documented. This project brings together graph-pattern queries, graph-analysis methods, and visualization to support investigation. The results reported here focus on functional behaviour rather than unverified numerical model performance.

III. PROBLEM STATEMENT AND OBJECTIVES

This project addresses the challenge of identifying potentially suspicious activity when relevant evidence is spread across transactions involving several connected accounts. Reviewing transfers individually can make longer paths and relationships difficult to recognize. The system therefore aims to prepare transaction data for graph analysis, identify selected network patterns, and present candidate findings for human review.

- Prepare account and transaction records for graph-based analysis.
- Represent accounts as nodes and transfers as directed relationships in Neo4j.
- Use Cypher queries to search for circular paths, multi-level chains, fan-in structures, and fan-out structures.
- Apply the graph-analysis and machine-learning components described in the implementation, including FastRP, Random Forest classification, and GraphSAGE-based neighbourhood learning.
- Visualize connected accounts and candidate suspicious transaction paths for review.
- Present findings as indicators for investigation rather than automatic conclusions of criminal activity.

IV. METHODOLOGY

Dataset and preprocessing

The project uses the IBM AMLSim synthetic dataset, with approximately 9,945 account records and 68,289 transactions reported in the project materials. The data are provided in CSV files, including accounts.csv and transactions.csv. Preprocessing checks the required columns and data types, validates account identifiers and transaction fields, and handles missing, invalid, or duplicate records where appropriate before the data are imported into Neo4j. The final record counts should be checked against the output of the team's executed preprocessing pipeline before submission.

Graph construction

Each bank account is represented as an Account node. Each transaction is stored as a directed relationship from the sending account to the receiving account. Transaction attributes, such as the amount and any available fraud indicators, are retained as properties. This graph structure allows the system to trace transfers across multiple steps and examine relationships that are not apparent from individual records.

Cypher-based pattern detection

The Cypher queries examine circular transfers, layering-like chains, fan-in, fan-out, and multi-hop paths. Circular-path queries look for routes that return to an earlier account. Layering-like analysis follows sequences of transfers across connected accounts. Fan-in describes an account receiving funds from multiple senders, while fan-out describes an account sending funds to multiple destinations. Multi-hop queries trace transfers across several relationships. These structures may warrant review, but none is sufficient on its own to establish illegal activity.

Graph-AI and machine-learning analysis

The implemented graph-AI workflow uses FastRP to generate embeddings that represent structural information about account relationships. A Random Forest classifier uses the features defined in the project pipeline to produce predictions or scores. GraphSAGE learns node representations by aggregating information from neighbouring nodes. These methods have different roles: FastRP generates graph embeddings, Random Forest performs feature-based classification, and GraphSAGE learns neighbourhood-aware representations. For reproducibility, the paper should report the exact model configuration, target-label definition, training and test split, hyperparameters, and evaluation metrics used by the team.

Visualization and investigation

Neo4j visualization displays account nodes, directed transactions, and connected paths. These views allow an investigator to follow the movement of funds and inspect groups of accounts flagged by the queries or models. Human review is needed to distinguish suspicious activity from legitimate transactions that happen to have similar patterns.

The system is organized into five stages: (1) account and transaction data input; (2) validation and preprocessing; (3) graph storage in Neo4j; (4) Cypher-based pattern detection and graph-AI processing; and (5) visualization and investigation. The main modules cover data preparation, graph-database management, suspicious-pattern detection, graph-AI analysis, and presentation of results. This separation allows data preparation, graph operations, model processing, and visualization to be handled as distinct tasks.

Apache Kafka is relevant to a message-streaming design for transaction data. However, the workflow described and evidenced in this project is based on batch ingestion from CSV files. The system should therefore be described as real-time only if a separate streaming workflow has been implemented and tested.

Implementation

The data-preparation module reads account identifiers, sender and receiver identifiers, transaction amounts, and available fraud indicators from the AMLSim files. Neo4j stores account nodes and transaction relationships. The detection module runs Cypher queries to retrieve circular paths, layering-like chains, fan-in, fan-out, and multi-hop paths. The implemented graph-AI components include FastRP embeddings, Random Forest classification, and GraphSAGE-based neighbourhood learning. The visualization module presents the resulting graph structures and selected candidate paths.

The project environment uses Neo4j Desktop and Cypher for graph storage and queries, along with Python-based data processing and machine-learning tools. The implemented graph-AI components include FastRP, Random Forest, and GraphSAGE; the project materials also reference Neo4j Graph Data Science, scikit-learn, and PyTorch/PyTorch Geometric. Apache Kafka should be described as part of the implemented system only if the team used and tested it in the transaction workflow. The final submission should state the actual software versions and each tool's role.

V. RESULTS AND DISCUSSION

Functional checks were recorded for dataset loading, graph creation, transaction-relationship creation, amount-property storage, fraud-indicator filtering, multi-level path retrieval, circular-transaction detection, layering-pattern analysis, fan-in, fan-out, and graph analysis. The project records mark the listed checks as passing, indicating that the tested graph operations and selected pattern queries returned their expected outputs. These checks demonstrate selected functional behaviour; they do not, by themselves, measure the predictive quality of a trained model.

The project also includes visualizations of the imported dataset, fan-out and fan-in structures, a fraud-account network, and a multi-hop transaction chain. These views make it easier to inspect connected transaction structures and follow paths between accounts. However, graph visualizations and successful functional tests do not establish model accuracy. Metrics such as precision, recall, F1-score, a confusion matrix, false-positive rate, and runtime should be reported only after they have been calculated from a documented evaluation run.

A useful next step is to evaluate the graph-pattern queries and each trained model on the same held-out dataset. The evaluation should define the target labels and decision thresholds, describe the train/test split, and use consistent metrics for each method. Since AMLSim is synthetic, the results should be interpreted as findings on that dataset and should not automatically be generalized to real banking transactions.

Project Figures and Visual Results

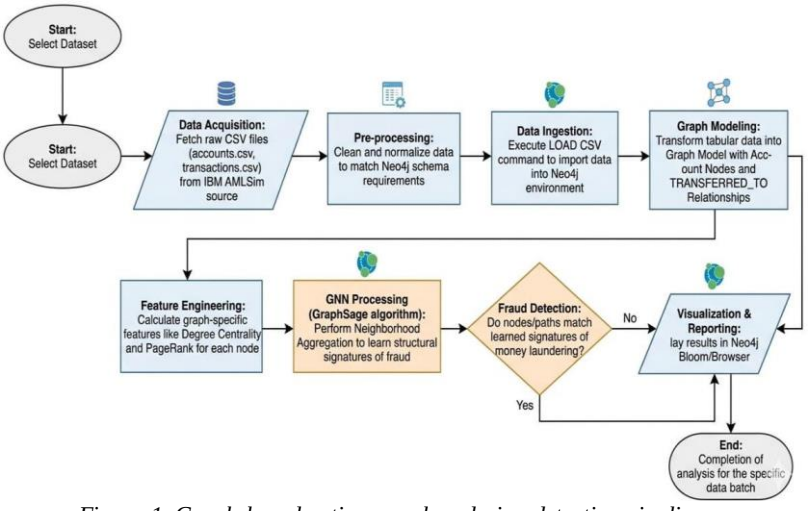


Figure 1. Graph-based anti-money laundering detection pipeline.

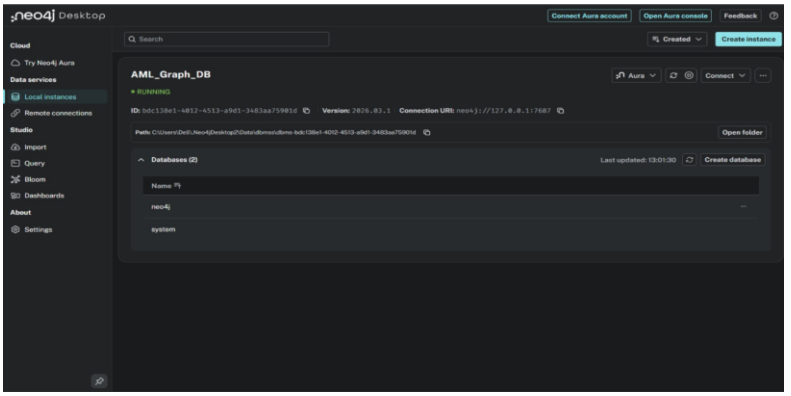


Figure 2. Neo4j environment used for graph database operations.

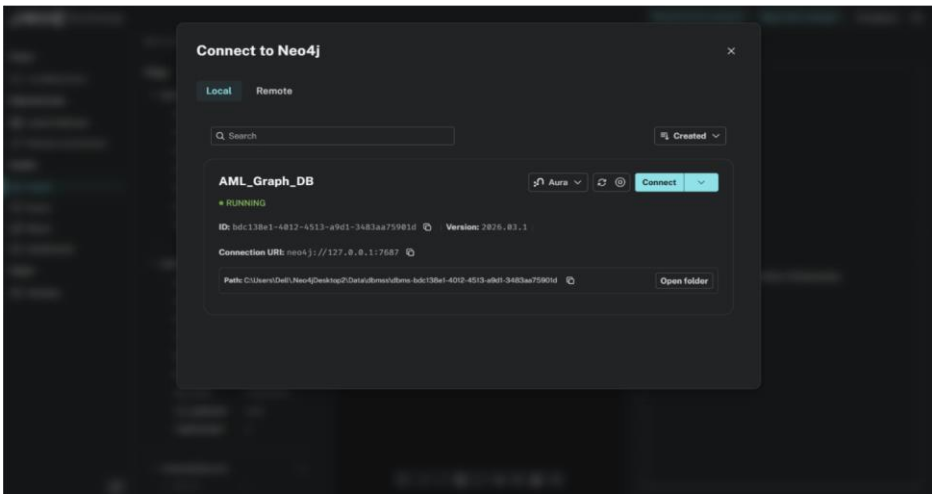


Figure 3. Transaction data configuration in the project environment.

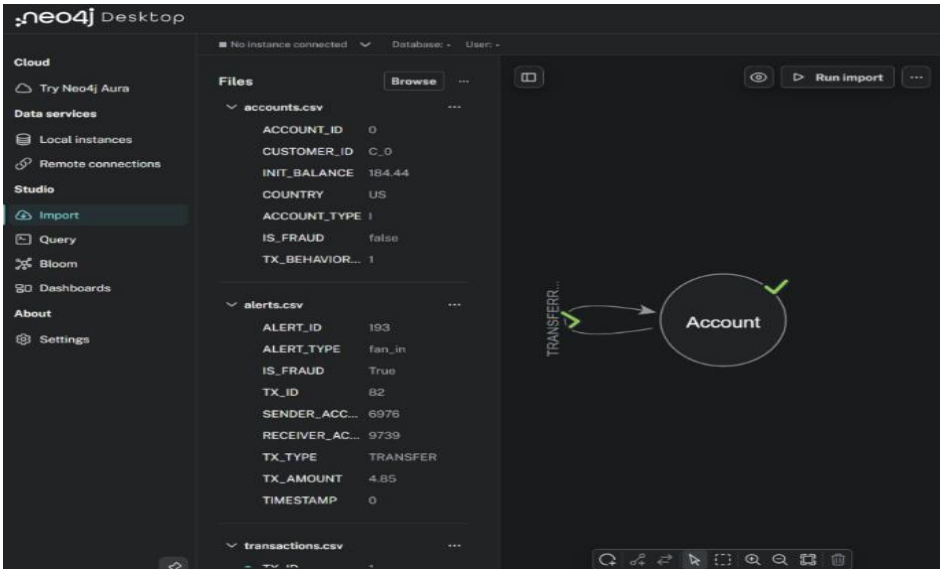


Figure 4. Account and transaction graph stored in Neo4j.

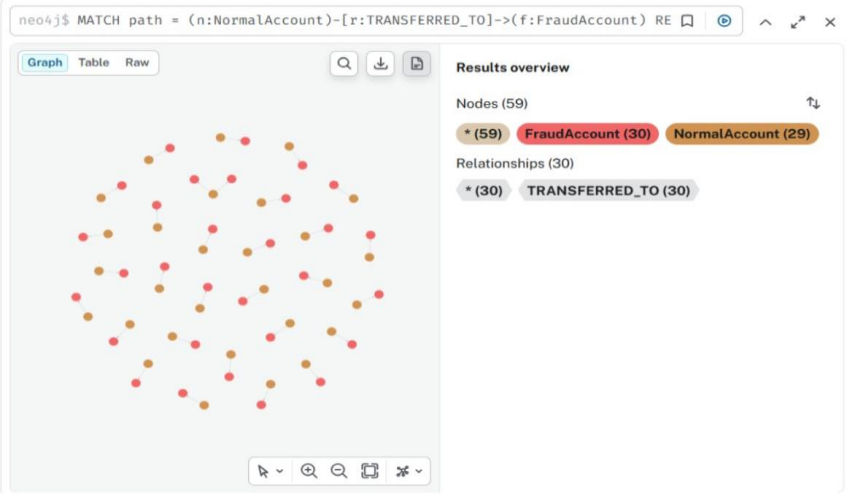


Figure 5. Fan-out pattern: one account transfers funds to multiple accounts.

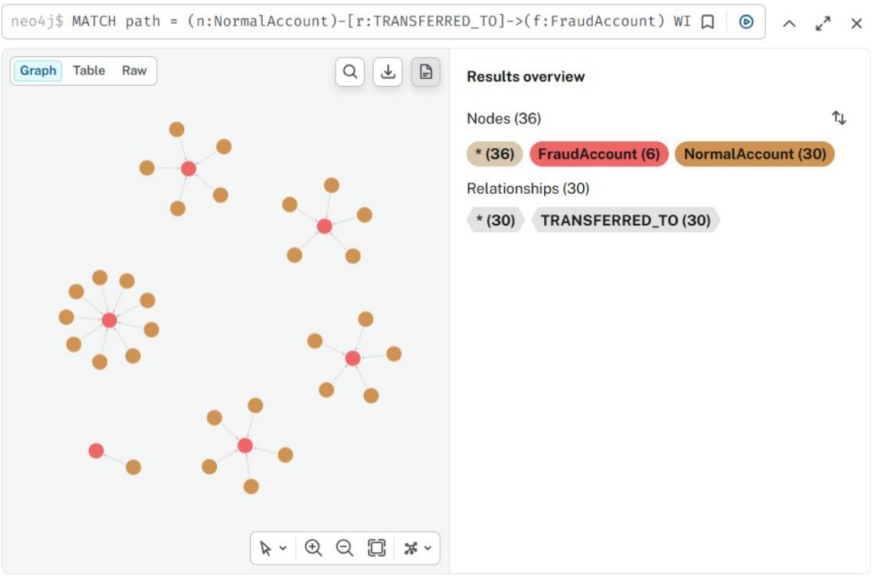


Figure 6. Fan-in pattern: multiple accounts transfer funds to one account.

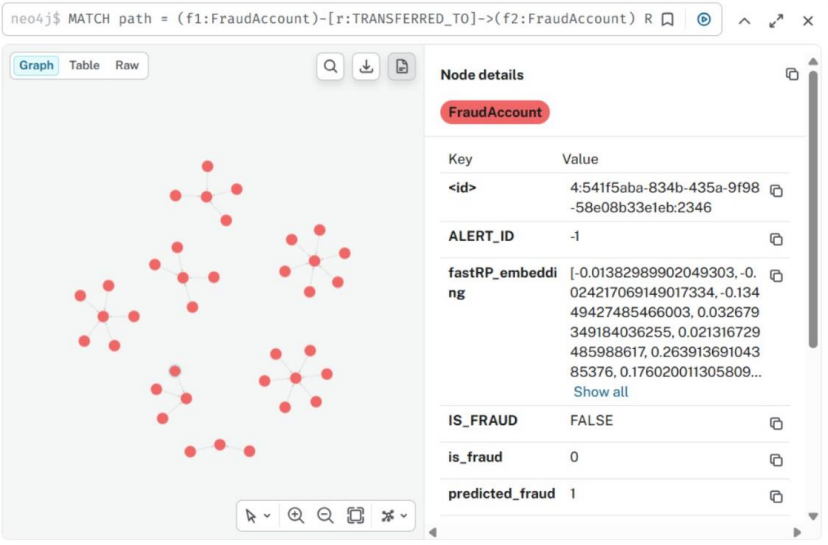


Figure 7. Connected network of accounts marked as potentially fraudulent in the dataset.

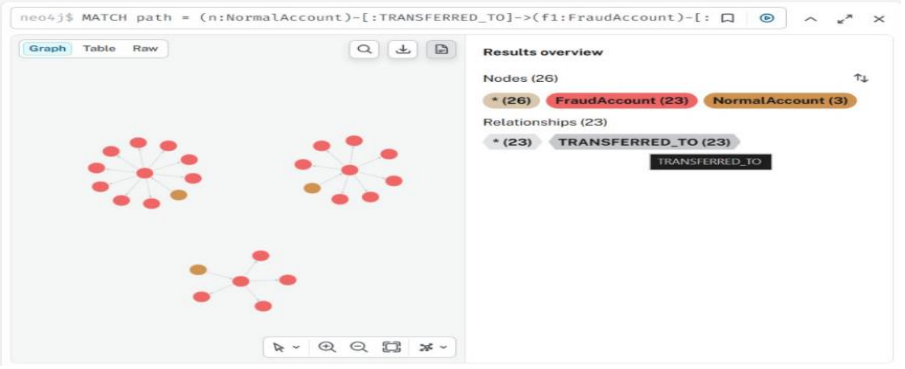


Figure 8. Multi-hop transaction chain from a normal account to an account marked as fraudulent in the dataset.

VI. LIMITATIONS

The system is evaluated using synthetic AMLSim data, which may not fully reflect real banking behaviour. Circular paths, fan-in, and fan-out structures can also occur in legitimate transactions, so the system may generate false positives. The available project results do not include a complete numerical model-evaluation table, and the paper therefore does not claim a specific accuracy or that one model performs better than another. Further work should evaluate each implemented AI component separately and clarify whether the deployed workflow uses batch ingestion, streaming ingestion, or both.

VII. CONCLUSION AND FUTURE WORK

The Anti-Money Laundering Tracker with Graph AI models accounts and transfers as a graph and uses Neo4j and Cypher to explore transaction relationships. The implemented approach examines circular transfers, layering-like chains, fan-in and fan-out structures, and multi-hop paths. It also includes FastRP graph embeddings, Random Forest classification, and GraphSAGE-based neighbourhood learning. Visualizations help investigators examine connected accounts and trace candidate suspicious paths that may be difficult to recognize from isolated transactions. The functional checks documented for the project indicate that selected graph operations and pattern queries produced their expected outputs.

The system is designed to support financial investigation, not replace it. Future work includes documenting the model configurations and data labels, evaluating the models on held-out data, measuring false-positive rates, comparing the methods with rule-based baselines, and testing the approach on additional datasets. Streaming performance should also be evaluated if real-time transaction ingestion is implemented.

REFERENCES

1. Apache Software Foundation. (n.d.). Apache Kafka documentation. https://kafka.apache.org/documentation/
2. Islam, S., Gupta, G. R., Chakraborty, A., Singh, S., Soni, A., et al. (2025). Detecting fraudulent transactions for different patterns in financial networks using Layer Weighted GCN. Human-Centric Intelligent Systems, 5, 181–195.
3. Neo4j. (n.d.). Cypher query language manual. https://neo4j.com/docs/cypher-manual/current/
4. Neo4j. (n.d.). Neo4j graph database documentation. https://neo4j.com/docs/
5. Neo4j. (n.d.). Neo4j Graph Data Science library manual. https://neo4j.com/docs/graph-data-science/current/
6. PyTorch Geometric. (n.d.). PyTorch Geometric documentation. https://pytorch-geometric.readthedocs.io/
7. Shi, Y., & Tang, M. (2026). Uncertainty-aware spatio-temporal contrastive graph neural networks for cyber financial fraud detection and risk management. Scientific Reports, 16, Article 25068.
8. Suzumura, T., & Kanezashi, H. (2021). Anti-money laundering datasets: InPlusLab Anti-Money Laundering Data. IBM AMLSim project. https://github.com/IBM/AMLSim
9. Tong, G., & Shen, J. (2023). Financial transaction fraud detector based on imbalance learning and graph neural network. Applied Soft Computing, 149, Article 110984.
10. Weber, M., Chen, J., Suzumura, T., Pareja, A., Ma, M., Kanezashi, H., Kaler, T., Leiserson, C. E., & Schardl, T. B. (2018). Scalable graph learning for anti-money laundering: A first look. arXiv:1812.00076. https://arxiv.org/abs/1812.00076
11. 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. KDD '19 Workshop on Anomaly Detection in Finance.