How money flow statistics can be used to detect money laundering activity in graph-based financial crime detection.

Luca-Serban Ionescu
Supervisor: Kubilay Atasu[1]
Responsible professor: Zeki Erkin[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Luca Ionescu
Final project course: CSE3000 Research Project
Thesis committee: Kubilay Atasu, Zeki Erkin, Megha Khosla

Abstract

Financial crime represents a growing issue which contemporary society is facing, especially in the form of money laundering, which aims to conceal the origin of illicit funds through a network of intermediate transactions. State of the art solutions for detection of money laundering in graph representations of financial data include supervised techniques such as Graph Neural Networks. Although provably efficient, their main limitation stands in the fact that they require a dataset of correctly and completely labeled transactions, which is often unfeasible to obtain. This work aims to explore money flow statistics as an unsupervised approach to money laundering detection, through computing statistics of accounts based on the amount of money received and sent in a certain time frame or network flow analysis using maximum flow algorithms. Therefore, this paper aims to answer two questions, namely *What are the existing solutions using money flow statistics?* and *How would these money flow statistics methods perform on a realistic dataset of transactions?*. The analysis benchmarks the identified algorithms on a realistic dataset of financial transactions in order to observe their limitations and suggest further research into how these limitations can be overcome in order to make money flow statistics methods a feasible solution to money laundering detection.

1 Introduction

Financial crime refers to all illicit activities that take place in the financial sector and involve multiple accounts which are tied through money transfers, having the purpose of illegally acquiring or hiding the origin of money. Financial crime can take multiple forms, such as payments fraud, financial scams, account frauds and money laundering. The global financial crime volume has been estimated around \$1.4-\$3.5 trillion per year according to the latest industry reports [15], which places the identification and mitigation of financial crime as a top priority of contemporary society. This paper will be focusing mainly on money laundering, which has the aim of concealing the origin of funds generated by criminal activity, to make it appear as the funds have originated from legitimate sources [10].

Existing literature proposes two data representations for financial information, which can be used for analysis and identification of financial crime. One representation is the tabular format, where rows represent transactions and columns represent features. Although this format is structured, it does not provide much insight into patterns that might be formed through repeated transactions between accounts. This is why a graph representation is often preferred, as it enables analysts to uncover insights through subgraph patterns [4].

Documented algorithms for money laundering detection in graph-based settings are split into two categories: supervised, which require labeled data in order to accurately detect money laundering activities and unsupervised which do not require labels and compute their result based on the structure of the financial graph.

Unsupervised methods include dense-subgraph algorithms which aim to detect fraudulent patterns in financial graphs. These algorithms often focus on the analysis of dense subgraph components, which are networks of accounts that are strongly connected between each other, but loosely connected to the rest of the graph. The existence of these patterns is a valid indicator for money laundering, as it indicates that money might be circulating through a series of accounts and returning to the same initial account, meaning that the process is trying to hide the real provenience of the cash inflow. However, fraudsters have adapted to this technique and have exploited a vulnerability caused by the fact that dense subgraphs represent single-step transfers. Therefore, by performing a multi-step transfer through a series of intermediary accounts, also known as camouflage, they can avoid detection by such algorithms. [10] [12]. One other unsupervised approach is graph mining, which analyzes sub-graph patterns and detects money laundering based on known laundering patterns, exemplified in Figure 4. Similarly to dense-subgraph approaches, these methods can be overcome by fraudsters through adaptation, creating complex and dynamic patterns [10].

Supervised methods include graph neural networks which aim to capture complex relational patterns and dynamics within financial networks [7]. Although superior to unsupervised techniques in terms of adaptability to complex and evolving patterns, the main limitation of supervised techniques is that they require accurately labeled data with a large number of both positive and negative labels, which is often unfeasible to acquire [9].

One solution that is proposed by current literature which allows for unsupervised detection of money laundering in multi-step transfers, without relying on pre-defined subgraph patterns is flow statistic analysis. This technique analyzes the flow of money in and out of accounts rather than the density of subgraphs, therefore being able to detect large flows of money between accounts even when camouflaged with intermediate vertices. By analyzing the flow of money in and out of an account, a custom anomalousness metric can be defined for each individual account, which can then be used in an anomaly detection task in order to retrieve the subgraphs with the highest probability of facilitating money laundering. [12].

The aim of this work is to investigate existing solutions utilizing money flow statistics for detecting money laundering in graph-based financial crime detection and to benchmark these solutions on a dataset of realistic synthetic financial transactions with complex laundering patterns in order to determine whether the analysis of money flow statistics is a suitable solution to the limitations of both supervised and unsupervised techniques and identify potential limitations of existing flow-based solutions.

The paper is structured as follows: Section 2 will explore key concepts for understanding money flow statistics methods, while Section 3 will detail the existing solutions that use money flow statistics. Section 4 will detail the conditions in which the experiments have been performed and Section 5 will look at the results obtained, while Section 6 will consider the ethical implications of the experiment. Finally, Section 7 will discuss the findings and Section 8 will draw conclusions and suggest further research based on the discussion.

2 Preliminaries

2.1 Directed multigraphs

A directed multigraph is a directed graph which allows multiple edges between the same two nodes and loops within the graph. This data structure is often used in the context of financial data, as, through its directed nature, it can highlight both the source and the destination of a transaction. Additionally, allowing loops in the graphs encapsulates real-world behavior such as money retrieved from deposit accounts (self-edges) or money circulating and returning to the sender (cycles). By allowing multiple edges between accounts, the multigraph can represent multiple transactions which can be sent at different timestamps or with different amounts, often represented as weights to the edges, as seen in Figure 1.

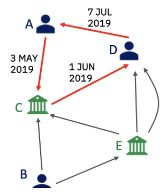

Figure 1: Directed financial multigraph [2].

2.2 Multipartite graphs

A graph is named k-partite when its vertex set can be partitioned into k subsets such that no edge has both ends in one single subset [5]. Multipartite graphs can be used in financial crime detection to outline layers of camouflage for money laundering. By formatting the transaction graph as a k-partite graph, where the first layer represents suspected sources of money laundering and the last layer as suspected destinations of the laundered money, analysis could reveal the intermediate accounts used to camouflage money laundering activities. One example of such graph is exemplified in Figure 2.

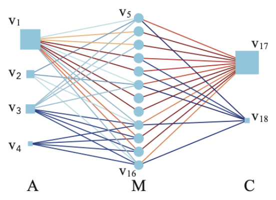

Figure 2: Tripartite financial graph [12].

2.3 Tensors

Tensors are N-dimensional arrays which are able to express multiple relations of any order as a higher-dimensional tensor [14]. Their application in financial crime detection comes as a solution to the limitation of traditional graphs, which are only able to encode edges in a two-dimensional format, considering their source and destination. Tensors are capable of encoding edges based on other multi-dimensional attributes such as time, which allows visualizing patterns in transactions not only based on source and destination, but also based on the timestamp at which the transaction occurred. Figure 3 shows a pair of coupled tensors, which represents two tensors connected by common values in one of the dimensions.

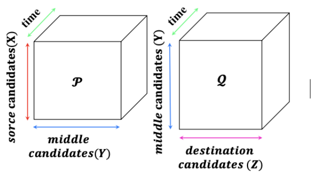

Figure 3: Pair of coupled tensors [18].

2.4 Money Flow Statistics

In the context of money laundering detection, money flow statistics refers to the information obtained from the analysis of the flow of funds in and out of accounts in a transaction graph. The inflow into *i* is defined as total amount of money received by account *i*, noted as $inflow(i)$ and, conversely, the outflow of node *i* is the total amount of money sent by account *i*, noted as $outflow(i)$. Flow statistics are used in financial graphs in order to identify suspicious accounts based on patterns observed in the inflow and outflow of funds. *Sun et al.* propose in [18] three main characteristics of money laundering when analyzed through flow statistics:

- **Density:** Due to the limited amount of middle accounts available, fraudsters choose to launder large amounts of money through a relatively low number of middle accounts, resulting in a very dense flow of money in and out of middle accounts.
- **Zero-Out:** Middle accounts used in money laundering usually transfer out most of the inflow of money they receive, as residual money left in the account usually represents a loss for the launderers.
- **Fast-In/Fast-Out:** Transfers through middle accounts usually happen quickly in order to reduce the time of being intercepted, meaning that the inflow and outflow of a middle account happen within a short time frame.

2.5 Network Flow

A network flow model represents the transfer of some commodity–such as money, material, or information–across a network from sources to destinations, obeying capacity constraints on the edges and conservation of flow at intermediate nodes. In a flow network, nodes are split into sources (s), which are nodes with zero inflow, sinks (t), which have zero outflow and intermediate nodes, which are the rest. Therefore, if f is a feasible flow in a network G, the value $f(G)$ of the flow is the outflow from the source, which must reach the destination. A feasible flow in a capacitated network such that the value of the flow is as large as possible is called a maximum flow in the network. The problem of finding a feasible flow in a network such that its flow value is maximum is known as the maximum flow problem [3]. Thus, for a capacitated network $G = (N, A)$ with a nonnegative capacity u_{ij} and weight x_{ij} associated with each ark $(i, j) \in A$ and flow value v, the maximum flow problem from s to t can be formalized as maximizing v in the equation presented in Figure 6 [1].

3 Background

3.1 FlowScope

Li et al. describe in [12] FlowScope, a flow statistics-based solution to the limitations of dense subgraph money laundering detection. The algorithm is designed to operate on a k-partite graph, where k is an arbitrary length given as a parameter, identifying the most suspicious subgraphs based on a metric defined in the paper. FlowScope defines the anomalousness of an account i in subset S based on the difference between the minimum flow going through i, $f_i(S) = min\{inflow(i), outflow(i)\}$, and the maximum flow, $q_i(S) = max\{inflow(i), outflow(i)\}$, scaled by a factor λ, as seen in the following formula, where λ is a hyperparameter resembling a residual factor, indicating the importance of residual money for the launderers, $d_i(S)$ is the degree of node i in subset S, $A \cup C$ are the first and last layer of the K-partite graph and M_l are intermediate nodes, as seen in Figure 2:

$$w_i(\mathcal{S}) = \begin{cases} f_i(\mathcal{S}) - \dfrac{\lambda}{1+\lambda} q_i(\mathcal{S}), & \text{if } v_i \in \mathcal{M}_l \\ d_i(\mathcal{S}), & \text{if } v_i \in \mathcal{A} \cup \mathcal{C} \end{cases}$$

Therefore, an intermediate node is considered highly suspicious when the difference between the minimum flow passing through it, $f_i(S)$, and the retention of money inside the account (computed as the difference between the maximum flow, $q_i(S)$, and minimum flow, $f_i(S)$ through the account) is high. The high difference shows that the retention is minimal when compared to the flow, satisfying the Zero-Out property of money laundering. When computing the suspiciousness, the residual is multiplied by a residual factor λ which resembles the importance of the leftover money for the launderers.

After the suspiciousness feature is computed for each account, FlowScope uses an anomaly detection algorithm which greedily removes the nodes with the lowest weight from one of the sets (A, M_l or C), until one of the sets is empty. Then, it returns the subgraph with the highest anomalousness computed as an average over the suspiciousness values of all nodes, as seen in the following formula, where $g^k(S)$ represents the anomalousness score of subset S, in a k-partite graph:

$$g^k(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{l=1}^{k-2} \sum_{v_i \in \mathcal{M}_l} f_i(\mathcal{S}) - \lambda(q_i(\mathcal{S}) - f_i(\mathcal{S}))$$

FlowScope has been tested on two dataset, namely **Czech Financial Dataset (CFD)**, a dataset of anonymous transactions from the Czech Bank and **CBank**, a dataset of real transactions from an anonymous bank, with a labeled group of money laundering. For evaluating the performance of FlowScope, laundering was injected randomly to nodes within the dataset, with camouflage edges being created based on a distribution, aiming to keep retention in middle accounts at low levels.

3.2 CubeFlow

In [18], *Sun et al.* propose CubeFlow, a flow statistics-based approach to money laundering detection which aims to identify laundering blocks in pairs of coupled tensors, such as the ones in Figure 3. The paper identifies a limitation of FlowScope, which is its inability to consider transaction time while computing anomalous blocks, therefore completely ignoring the Fast-In/Fast-Out property of money laundering. To overcome this limitation, CubeFlow operates on an input of coupled tensors, which hold an additional dimension, time. CubeFlow uses a similar metric to FlowScope, considering the suspiciousness metric of a node as the difference between the flow going through the account and the residual value, however, the algorithm unfolds the tensor based on the middle accounts, such that it can compute anomalousness values for each middle account, at each time bin. This means that the flow is not considered using all transactions in the graph, which happens in the case of FlowScope, but just transactions in the same time bin, therefore fulfilling the Fast-In/Fast-Out property. Similarly to FlowScope, CubeFlow greedily removes nodes with the lowest score until one of the sets (A, M_l, C) is empty and returns the dense block with the highest overall score, computed as an average over the suspiciousness score of all nodes in the block.

The evaluation methodology is similar to FlowScope, money laundering accounts and edges are randomly selected and the amounts of money laundered are chosen such that the residual amount is almost zero.

3.3 WeirdFlows

WeirdFlows is defined in [6] as a top-down search pipeline rather than an anomaly detection algorithm, as it aims to provide an interface that allows investigators to analyze all possible flows of money between entities in a financial graph, that can be built at different levels of time and spatial aggregation by using flow statistics.

The application of WeirdFlows involves two steps:

- **Structural adaptation:** A transaction network is created using financial transaction data.
- **Path identification on temporal network:** Multiple directed temporal networks are created within a specific time aggregation. The algorithm then computes all paths from a specific node, within each time bin.

A transaction through n intermediaries in a transaction network forms a path of length $n + 1$. The WeirdFlows pipeline aims to identify the transaction flow from a given input node x and verify the maximum possible amount of money sent from x to other nodes within a maximum distance n. The maximum amount of money from node x to node y is defined as the sum of the minimum weights of each path from x to y.

Therefore, the algorithm computes, for a selected time aggregation, each n-length path from every node in the network, along with its maximum flow of money, which it stores in a table, and creates a search pipeline which allows filtering the information based on source, destination node and time bin. An analyst could uncover money laundering behavior by analyzing the time series of the weight of a flow and identifying spike values, which denote a larger than usual flow of money on a specific route in the network.

3.4 DenseFlow

In [13], *Lin et al.* identify two main challenges of money laundering detection in blockchain networks, specifically focusing on Ethereum transactions:

- **Massive and dense gangs of money laundering accounts:** In blockchain networks, criminals can generate numerous disposable accounts, as the cost of creating an account is close to free. This helps them evade being frozen by centralized cryptocurrency service providers. (**C1**)
- **Complex multi-layered money laundering pathways:** To disperse and transfer illegal funds as quickly as possible, hackers create multi-layered money laundering pathways across numerous accounts. (**C2**)

To address these two challenges, the paper proposes DenseFlow, an algorithm designed for money laundering detection on directed multigraphs resembling an Ethereum transaction network. In order to tackle challenge **C1**, DenseFlow identifies accounts engaged in extensive, high-frequency asset transfer by detecting dense subgraphs and considering characteristics of money laundering transactions. To tackle challenge **C2**, the algorithm traces the flow of funds from money laundering sources to accounts of fraudsters along the money laundering pathway using an algorithm that solves the maximum flow problem.

The paper proposes a joint suspiciousness metric, composed on three metrics:

- **Topological:** An account becomes more suspicious when it primarily establishes transactions with suspicious accounts and has fewer connections to others, computed based on the following formula, where, given a suspicious subset S, the edge set of the graph, E and the node set of the graph, N, $\alpha_i(S)$ represents the topological suspiciousness of node i in subset S and e_{xy} represents the amount of money transferred in transaction e, from account x to account y:

$$\alpha_i(S) = \frac{\sum_{(j,i)\in E \wedge j \in S} e_{ji}}{\sum_{(k,i)\in E \wedge k \in N} e_{ki}} \tag{1}$$

- **Temporal:** The node's participation level in transaction surges within the subset S, compared to the entire node set V. The temporal suspiciousness of node i in subset S is defined as $\beta_i(S)$ in the following formula, where V is the node set of the graph, $T_i(S)$ denotes the transaction timestamp set of node i connecting within nodes of the subset S. Φ represents the participation level of node i within subset S:

$$\beta_i(S) = \frac{\Phi[T_i(S)]}{\Phi[T_i(V)]}, \tag{2}$$

- **Monetary:** The suspicious rating of a transaction is computed as the ratio of total transaction amount within a timeframe to the total amount over the entire duration. The suspicious rating between nodes i and j at timestamp t with transaction amount a is defined in Equation 5a, where $A(\tau(t))$ represents the sum of the amounts for all transactions within the timestamp set τ, at time slot t. The monetary suspiciousness of the node i in subset S, $\gamma_i(S)$, is computed in Equation 5d, as the Kullback-Leibler divergence of the suspicious rating of node i in subset S, compared to the rest of the node set V, multiplied by a balancing factor, defined in Equation 5c.

Finally, the suspiciousness $f_i(S)$ of node i in subset S is computed as the fusion of the three suspiciousness metrics, as described in the following equation, where b is a hyperparameter:

$$f_i(S) = \sum_{(j,i)\in E \wedge j \in S} e_{ji} \cdot b_i^{\alpha_i(S)} + \beta_i(S) + \gamma_i(S) - 3 \tag{3}$$

The goal of the algorithm is to find a subset S^* that maximizes the suspiciousness, with the suspiciousness of a subset computed as the average metric of its nodes.

After identifying a dense subgraph S^*, DenseFlow constructs a flow network over the transaction graph, where edge weights correspond to transaction amounts and act as capacities. It then introduces a synthetic sink node t, connected with infinite-capacity edges to all nodes in S^* and, using the maximum flow algorithm proposed in [8], computes the maximum flow of money from laundering source s to destination t. The nodes and edges with positive flow represent the most plausible laundering pathways and are added to the suspicious set F. Finally, the algorithm outputs the union of the dense subgraph and the accounts through which the maximum flow is carried, $S^* \cup F$.

Analysis on the algorithm has been performed on four different money laundering datasets involving Ethereum laundering with clear paths from the source of the launderings to the dense blocks of fraudulent accounts.

3.5 SMoTeF

Shadrooh et al. propose in [16] Smurf Money Laundering Detection Using Temporal Order and Flow Analysis (**SMoTeF**), an algorithm which aims to detect smurfing laundering patterns which are temporally feasible. In the context of money laundering detection, the term *smurfing* refers to the laundering technique of breaking down the amount of money to launder into smaller amounts spread throughout various financial entities known as smurfs [17].

The paper aims to combine maximum money flow with temporal order such that it can avoid false positives in the form of temporally infeasible flows. This could be a flow $A \to B \to C$ with transaction $B \to C$ happening before transaction $A \to B$. Although the flow might fit the conditions of suspiciousness, it is a false positive, as the money reaching account C could not have originated from account A.

The initial step of the algorithm processes the input as a temporal directed multigraph and applies a smurf pattern detection algorithm. The paper uses existing solutions for smurf pattern detection, such as AutoAudit [11], which retrieve a set of subgraphs containing smurf money laundering patterns.

The next step uses a temporal constraint checking algorithm in order to discard patterns which do not satisfy natural time precedence conditions of transactions. For each journey j starting from the source of the smurf pattern v_s, passing through middle accounts v_m and reaching the destination v_d, the algorithm constructs the fan-in set from v_s to v_m, l^+ and the fan-out set from v_m to v_d, l^-. It uses the sets to verify that each fan-out edge in l^- is reachable in time by some fan-in edge in l^+. It uses the temporal pairs extracted from l^+ and l^- to construct two cut-sets, C_{Tsd} containing fan-in edges with a temporally corresponding fan-out edge and C'_{Tsd} containing fan-out edges with a corresponding fan-in. Finally, from the set of patterns extracted in the first step, the algorithm discards all patterns where the number of valid journeys from v_s to v_d is smaller than T_s, which is a predefined threshold for the minimum number of valid smurfs in the pattern.

Finally, SMoTeF performs a network flow analysis on each of the remaining patterns in the set in order to compute how much money is transferred from source v_s to destination v_d through middle nodes v_m. This is done to eliminate patterns that are temporally valid, but involve non-suspicious amounts of money, by checking that the maximum flow exceeds a predefined threshold T_f. The analysis is also used to identify suspicious time periods for audit and investigation.

For each valid journey j of the form $v_s \rightarrow v_m \rightarrow v_s$ in the subgraphs extracted, SMoTeF creates a time-ordered list of edge occurrences, $tol(j)$. It maintains a residual capacity counter l^+ for all fan-in edges and, for each edge in $tol(j)$, accumulates the incoming flow if the edge is a fan-in, or computes the maximum temporal flow as $Mtf(s,d) = min(capacity(l^+), w_e)$, subtracting this flow from the fan-in capacity. If the total flow across all journeys in a pattern $Mtf(G) < T_f$, the pattern is discarded.

In order to evaluate the algorithm on a financial dataset, the Czech Financial Dataset (**CFD**) was chosen. Synthetic smurfing patterns were injected in the dataset by randomly selecting three sets of nodes corresponding to v_s, v_m and v_d, with a varying number of middle nodes between 5 and 50 for each pattern. Half of the journeys in all patterns have been injected in order to satisfy all temporal conditions and thus have proper minimum temporal cuts. For the remaining half, it was imposed that they do not adhere to the time precedence constraints.

3.6 Limitations of Analysis

Analyzing the evaluation process of each algorithm described in this section, it can be observed that the analysis for most of these algorithms has been performed through injecting synthetic laundering transactions in real transaction datasets, which perfectly fit the evaluation criteria of the algorithm, such as low retention in middle accounts or high flow density. Although suitable for a theoretical analysis of the correctness of the algorithms, these transactions are highly unrealistic and do not reflect the actual performance that the algorithms would yield in a realistic setting with complex laundering patterns. Therefore, the purpose of the experiment performed in the following sections is to observe how these flow statistics solutions to money laundering detection perform in realistic settings and identify and understand their weaknesses such that improvements can be suggested.

4 Methodology

4.1 Approach

The following sections aim to benchmark four flow-based money laundering detection algorithms, namely FlowScope, CubeFlow, DenseFlow and SMoTeF (WeirdFlows is excluded from this analysis as it does not perform an identification of laundering patterns by itself) on the realistic synthetic financial dataset proposed by *Altman et al.* in [2]. The accuracy of these algorithms will be measured using a minority-class F1 score. If the algorithms achieve a high accuracy, they will be tested on larger versions of the dataset and on versions with different illicit densities in order to determine their strengths and weaknesses depending on the dataset. Otherwise, the analysis will dive deeper into the reason for the low accuracy to identify and understand the limitations of the algorithms, such that improvements can be suggested.

4.2 Dataset

To address the limitation in analysis, the dataset proposed for this experiment is AMLWorld, a realistic synthetic dataset proposed by *Altman et al.* in [2]. The dataset proposes a record of synthetic transactions spanning over multiple banks and currencies. The data is obtained using a generator that creates a multi-agent virtual world in which some of the agents are criminals with illicit money to launder. Agents can resemble banks, individuals and companies, with realistic transaction modeling such as individuals and companies buying items and making bank transfers to get supplies or pay salaries and pensions. The criminals launder their money through both the patterns presented in Figure 4 and other natural contexts such as paying salaries and buying supplies. The strong realism of the dataset comes from its complexity in terms of inter-agent relations, such as individuals which are able to be employed and purchase supplies from companies, but also own shares and receive interest payments. This unique degree of resemblance to real-world transactions has motivated the use of the AMLWorld dataset for determining the performance of flow statistic based money laundering detection in a realistic setting.

The dataset offers six variations, with varying illicit transaction ratio and number of transactions. The variation chosen for this experiment is the HI-Small, with a high illicit ration (1 in 807 transactions) and small size ($\sim$5M total transactions).

4.3 Data pre-processing

FlowScope

When using the AMLWorld dataset to benchmark FlowScope, the main challenge comes from the discrepancy between the graph formats of AMLWorld and the FlowScope input. While AMLWorld provides a directed multigraph of transactions, FlowScope expects a multipartite graph input format. Therefore, in order to create a multipartite graph preserving the connections between the AMLWorld entities, a series of pre-processing transformations have been performed on the raw dataset:

- **Transaction normalization:** AMLWorld transactions span across fifteen different currencies. Because FlowScope relies on the difference between minimum and maximum flow through nodes in order to determine residual amounts, all transaction amounts must be expressed in the same currency. Therefore, all amounts have been converted to EUR, using the publicly available exchange rates.
- **Parallel edge aggregation:** The multipartite format required by FlowScope does not support parallel edges. Therefore, in order to preserve the $inflow(i)$ and $outflow(i)$ of any node i, parallel edges have been aggregated by summing up their transaction amounts.
- **Conversion to multipartite format:** FlowScope operates on k-partite graphs represented through k-1 files, with $file_i$ containing edges from one layer i to layer $i+1$, along with the transaction amount. Nodes in the files are represented with contiguous integers starting from 0.

In order to represent the AMLWorld data in the required k-partite format, a unique string $Node\_Id$ was constructed for each node by concatenating their $Bank\_id$ and $Account\_id$. Parallel transactions were aggregated based on their source and destination $Node\_id$ by summing up their amounts, resulting in a dataframe containing the source and destination ids and the amount of each transaction.

Using a Depth-First Search, all paths of length $k-1$ have been extracted from the resulting graph and separated in transactions. As a result path $A \to B \to C$ was split into $A \to B$ and $B \to C$. The $Node_id$s of each transaction have been encoded to contiguous integers, identical nodes receiving the same encoding in the same layer, but different encodings in different layers. This condition ensures that:

- There are no edges between nodes of the same layer, as vertices of a self edge of the form $A \to A$ would be encoded differently in the first and second layer.
- There are no backwards edges, as in cycles such as $A \to B \to A$, the A in the third layer would receive a different encoding that the one in the first layer.

To achieve this, the nodes in each layer S have been encoded as integers in the interval $[offset, offset + unique(S)]$ where $unique(S)$ represents the number of unique ids in layer S and $offset$ represents the number of unique ids in the previous layers. Therefore, in each file $file_i$ an edge $A \xrightarrow{amount} B$ is encoded as $Encode_i(A), Encode_{i+1}(B), amount$ where $Encode_i(X)$ represents the encoding corresponding to layer i of $Node\_id$ X.

To compute the evaluation metrics on the data returned by FlowScope, the same encodings are used in reverse in order to obtain the original $Node_id$s.

CubeFlow

In order to adapt the AMLWorld data to the coupled tensor format required by CubeFlow, two encodings were considered during the experiments. It is worth noting that both methods differ from that of FlowScope by the initial temporal aggregation performed. Based on the timestamp of each transaction, time bins have been created for windows of different lengths (30min, 1h, 6h, 12h, 14h, 48h, 72h) and thus the parallel edge aggregation was performed not only considering source and destination $Node_id$s, but also the time bin of the transaction.

- The first encoding considered is similar to that of FlowScope, assuming the format of the two coupled tensors as a tripartite graph with an additional dimension representing the time bin of the transaction. Since CubeFlow is limited to two coupled tensors, the FlowScope preprocessing is modified to only consider tripartite graphs (k=3). Files are also modified to store, aside from source, destination and amount, the time bins of the transactions.
- The second encoding strategy was inspired by the experimental setting described in the paper, which involves splitting the node set of the graph into sources, where money transferring out is much larger than money transferring in and destinations, where money flowing in is much larger than money flowing out. The rest of the nodes are considered middle nodes.

The second encoding strategy was implemented by iterating over the nodes of the account and splitting nodes into source set X, where $outflow(x) > t * inflow(x)$ and destination set Z where $inflow(z) > t * outflow(z)$, with t being a hyperparameter for which multiple values have been tried (5, 10, 15, 20). The rest of the nodes were attributed to the middle set Y. Y was filtered to contain only middle nodes that connect nodes from X to nodes from Z in order to result in a pair of correctly coupled tensors, with edges between X and Y representing tensor $\mathcal{P}$ and edges between Y and Z representing tensor $\mathcal{Q}$, as observable in Figure 3.

DenseFlow

DenseFlow operates on temporal directed multigraphs, which aligns with the format provided in the AMLWorld dataset. The algorithm requires two files, one containing the transactions of the multigraph and the other containing the source accounts of laundering patterns, which it uses for tracing the flow towards the laundering groups.

The transaction file was obtained by creating $Node_id$s for each transaction as before and time binning within a 24h interval. The temporal component of DenseFlow was configured to consider temporal anomalies with regard to the previous and next time bin. Edges have not been aggregated as done for FlowScope and CubeFlow, as the multigraph required by DenseFlow allows parallel edges between nodes. Therefore, the transaction file contains the $Node_id$s of the source and destination of each account, the transaction amount and the time bin.

The source accounts of laundering patterns are extracted from the file containing the laundering patterns, offered by the dataset. The source is determined, for each pattern, as the node s with $inflow(s) = 0$ or the source of the first transaction, in the case of cycle patterns. The $Node_id$s of these accounts are stored in the source account file.

SMoTeF

SMoTef takes as input a directed multigraph of financial data with nodes encoded as contiguous integers starting from zero, along with the transaction amount and the time bin. In order to bring the AMLWorld dataset to this format, accounts were identified with a $Node\_id$ composed of the bank and account id, similarly to previous algorithms, with each unique $Node\_id$ receiving an integer encoding in the interval $[0, unique(S)]$, where $unique(S)$ represents the number of unique $Node_id$s. Parallel edges have not been aggregated, as the directed multigraph format allows for these edges. Time binning has been performed at the lowest level (1 minute), as SMoTeF uses it to determine temporal precedence of transactions, rather than time between transfers.

4.4 Evaluation Metric

The evaluation metric chosen for the experiment is the minority class F1 score. Considering that the purpose of the experiment is the binary classification between fraudulent and non-fraudulent transaction, with a minority in fraudulent transactions, the minority class F1 provides an estimate of the accuracy by providing the harmonic mean (Equation 6a) between how many of the flagged transactions are actually laundering (Precision) (Equation 6b) and how many of the entire set of laundering transactions have been correctly identified (Recall) (Equation 6c).

5 Results

5.1 Hypothesis

Comparing the large level of complexity that the AMLWorld dataset brings with the relative simplicity of the injected synthetic patterns that the flow statistics-based algorithm have been tested on, it is highly likely that these algorithms will not achieve a high accuracy due to their lack of flexibility to complex patterns. Additionally, the dataset was designed for Graph Neural Networks which, through being trained on labeled data, are more flexible to the complex patterns in the dataset.

Thus, the purpose of the experiment is not to prove that the flow statistics-based algorithms are a feasible solution in realistic settings, but to observe their behavior and understand their limitations. By drawing these conclusions, further research can be suggested in order to overcome their limitations and incorporate them in more flexible solutions.

5.2 Experimental results

FlowScope

When running FlowScope, the parameters which have to be pre-defined are the number of partitions of the multipartite graph, k, the number of suspicious subgraphs that should be extracted, n and the residual factor λ. Because the number of partitions has to be specified, it is deduced that, in one run, FlowScope can only determine laundering patterns of length $k-1$. By experimenting with multiple values of k, it was determined that for any $k > 6$, the resulting graph becomes too sparse for FlowScope to identify suspicious subgraphs and thus discards all middle nodes as they yield negative suspiciousness because of low connectivity. Therefore, in order to compute the accuracy for a wider number of pattern lengths, the F1 score was computed over a number of five runs, with $k \in [2, 6]$. Parameter n was selected as 3 and the residual factor $\lambda = 4$ was selected according to the recommendations of the paper.

The F1 score achieved after the five runs is $F_1 \approx 19.88\%$, which is significantly lower than the Graph Neural Network solutions presented in [2], which achieve up to 63% accuracy. Diving deeper into the subgraphs extracted, it is observable that FlowScope manages to identify some laundering edges with high density and relatively low amounts of money left in middle accounts. However, most real laundering transactions are overshadowed by very large transfers (such as those in BTC), which are not always laundering.

One additional experiment that was performed was removing percentages of the non-laundering edges and observing the evolution of the accuracy, which is visible in Table 1. Therefore, it is visible that as the number of non-fraudulent transactions of the original dataset ($\sim$ 5M transactions, out of which 3209 fraudulent) is decreased, the F1 score rises.

Non-Fraudulent Data (%)	F1 Score
100%	19.88%
50%	23.30%
25%	34.34%
10%	45.60%

Table 1: F1 scores achieved by FlowScope for varying amounts of non-fraudulent data with $k \in [2, 6]$

CubeFlow

During the experiment, CubeFlow was benchmarked using both pre-processing techniques described in Section 4.3, along with multiple time aggregations mentioned in the same section. In all settings, the algorithm has achieved an F1 score $F_1 = 0.00\%$. This result comes in close relation with CubeFlow's main limitation when compared to FlowScope, which is its inability to analyze chains of coupled tensors larger than two. This means that the algorithm is, by design, unable to detect laundering patterns with more than one middle layer, which is very common in the laundering patterns of AMLWorld. The low F1 score also comes related to the Fast-In/Fast-Out check performed by CubeFlow. Analyzing the laundering patterns provided, it is revealed that some patterns can span throughout up to six days out of the 18 defined in the dataset.

Additionally, similarly to FlowScope, by analyzing the dense blocks identified by CubeFlow, it is clear that the real laundering patterns have been overshadowed by very large volume transactions. When filtering out the top 10% of transaction values, the algorithm fails to detect any suspicious middle nodes, meaning that the graph becomes too sparse.

DenseFlow

The F1 score achieved by the algorithm is $F_1 \approx 0.73\%$. Analyzing the number of accounts extracted from each set in the final result, it is observable that, for most laundering patterns, there are no nodes coming from the maximum flow subset.

This comes into connection with the fact that 36% of the laundering sources have less than 15 other nodes reachable, which greatly affects the probability of a laundering patterns having a connecting path to the dense subgraph identified, therefore making the maximum flow computation impossible.

The dense subset misses most of the suspicious nodes of AMLWorld, which is connected to laundering nodes not adhering to the three components of the DenseFlow suspiciousness metric. This could mean low topological suspiciousness for sparse structures such as Fan-In, Fan-Out or Stack or a low number of temporal laundering spikes, as observable in Figure 5. The reasons for this inaccuracy will be discussed more in-depth in Section 7.

SMoTeF

SMoTeF achieves an accuracy of $F1 \approx 19.05\%$, with $Precision = 1$ and $Recall = 0.105$. Diving deeper into the patterns identified by the algorithm, it was observed that all suspicious transactions identified construct Scatter-Gather patterns in the AMLWorld graph, according to Figure 4. Moreover, the F1 score was also measured with regard to just the subset of Scatter-Gather patterns, instead of the entirety of the laundering transaction dataset, achieving a $F1 \approx 68.40\%$.

6 Responsible Research

Reproducibility

The implementation of all algorithms which were used in this work is described in a reproducible manner in their corresponding papers, which are referenced accordingly. Moreover, the implementations used in the experiment performed by this paper are extracted from the public GitHub pages referenced in the related papers, without any alterations other than those discussed in the Methodology section. Thus, the use of the algorithms is reproducible by accessing the same implementations that are provided by the authors.

The dataset used, presented in [2], is publicly available at the Kaggle page referenced by the authors of the paper, along with a file containing the laundering patterns existent in the dataset. The modifications made to the original dataset in order to transform it into a suitable format for each of the algorithms presented in this paper is described in the Section 4.3. Therefore, by following the same pre-processing steps on the available dataset, the same data should be obtained. By running the publicly available implementation of each algorithm on the data obtained, the same results should be achieved.

Additionally, the code used to benchmark these algorithms, along with all pre and post-processing logic is submitted alongside this paper [1], to ensure full transparency of the implementation used. This project also includes all intermediate data files, such as transactions with normalized payment amounts and extraction of specific laundering patterns.

Integrity and Ethical Considerations

All sources, for datasets, algorithms and concepts used in this paper have been cited in the Bibliography. The results are the genuine evaluations of the algorithms, which have not been manipulated in the benchmarking process. Evaluation metrics were chosen to reflect the real-world challenges of financial crime detection, particularly class imbalance. The minority-class F1 score was used as the primary metric to avoid misleading results caused by high accuracy on the majority (non-laundering) class.

One ethical consideration of this work is the possible privacy breach caused by the use of financial data in the experiment. However, the dataset chosen provides a fully synthetic set of data, with transactions and entities created by a generator described in [2]. Therefore, with no use of real financial or personal data, the experiment does not raise any privacy concerns.

One other ethical consideration of this work is the possibility of exposure of the money laundering detection techniques to launderers. Knowing the limitations of laundering detection algorithms would allow the launderers to exploit them. However, this paper does not highlight any confidential algorithm or any technique used in practice by banks to detect money laundering activity. This means that the limitations presented in this paper are derived from public information and do not endanger any laundering detection mechanism currently in place.

Finally, the societal relevance of the discoveries made in this paper stands in the improvement in the performance of money laundering detection algorithms that can be brought by integrating money flow statistics-based methods into already existing solutions. By understanding and addressing the limitations presented in this paper in further research, money flow statistics can be used to enhance existing techniques or aid the development of new ones.

7 Discussion

While looking at the results obtained by FlowScope, two main limitations of the algorithm can be derived. Firstly, the algorithm is designed to detect linear laundering chains with low retention. However, this is not the case in many of the patterns identified in Figure 4, as they are low-volume and dispersed, rather than high-volume and linear. The two laundering patterns that fit the description of FlowScope are the Random and the simple cycle. However, these patterns do no hold the densest flow in the transaction graph, meaning that they are overshadowed by very large transaction chains. According to FlowScope's suspiciousness formula, flows with very large amounts where the residual is small compared to the flow seem more suitable.

Secondly, the algorithm is clearly affected by the large amount of noise caused by the low illicit ratio, which causes laundering patterns to be lost among the large pool of transactions. This is proven by the data in Table 1, where, as the noise minimizes, the algorithm becomes progressively more accurate.

As mentioned in the previous sections, the main limitation that leads to the poor performance of CubeFlow is that it is unable to detect laundering paths longer than 2. Considering that many of the cycles and random patterns are large chains of transactions, it is natural that the algorithm will not be able to detect these chains.

One additional limitation that might break CubeFlow's as-

[1]https://github.com/Luca-Ionescu/cse3000-money-flow-statistics

sumption for Fast-In/Fast-Out transfers is that of the time distribution of laundering transactions. While CubeFlow expects transactions of a laundering chain to fall in the same time bins, the patterns in AMLWorld are split throughout up to six days, which represents $1/3$ of the total timeframe of the dataset. Setting the time aggregation too high, although bringing transactions to the same bins, will result in very noisy subsets (which was proven to decrease accuracy in the case of FlowScope), while smaller time bins might spread laundering patterns across too many timestamps. The preferred time aggregation for this analysis has been chosen as 24h, as it provides 18 time bins throughout the dataset, being loose enough to capture a certain degree of the patterns in adjacent bins, but tight enough to create a sensible time separation.

In the case of DenseFlow, there are two issues corresponding to the two subsets identified by the algorithm. Firstly, for most patterns, there is no maximum flow path identified from the source to the dense laundering group. This means that laundering sources are isolated, which is supported by the 36% of sources with less than 15 reachable nodes. Fundamentally, the laundering patterns of AMLWorld are not designed to reach larger groups, which breaks the assumption of the algorithm.

Secondly, the suspicious accounts of AMLWorld do not fit the criteria of the joint suspiciousness defined by DenseFlow. In dispersed patterns, such as Fan-In, the source node has few incoming edges from other laundering nodes, while receiver nodes receive only one edge from a laundering node (which already has a low topological score), leading to a low topological score for the overall pattern. Additionally, the temporal component of the suspiciousness metric, assumes that the laundering patterns will create a temporal spike when compared to adjacent time bins. However, laundering patterns seem evenly distributed in time, as observable in Figure 5. This leads to a low temporal score. By having both a low temporal and topological score, the patterns are not selected by the dense subgraph component of DenseFlow.

The first observation towards the results presented by SMoTeF is the perfect precision. This means that all of the identified edges are actual laundering edges, with no false positives. This proves the efficiency of the two pruning techniques employed by the algorithm, considering temporal precedence and maximum temporal flow between the source and destination of the smurf patterns. Despite the perfect precision, the low F1 score when compared to Graph Neural Network solutions presented in [2] is caused by the low recall. This reveals the main limitation of the algorithm, which is that it is only able to consider smurf laundering patterns, as closer inspection of the identified patterns shows that 100% of them correspond to Scatter-Gather patterns in AMLWorld, as depicted in 4. Moreover, when computed against just the Scatter-Gather patterns in the dataset, the F1 score achieves a value comparable to the Graph Neural Network solutions, showing that SMoTeF is efficient in the context of smurfing patterns, but it lacks flexibility for different categories of patterns due to its graph mining approach to the extraction of smurfing patterns.

8 Conclusions and Future Work

This work has aimed to explore money flow statistics as an unsupervised approach to money laundering detection and answer two questions, namely *What are the existent solutions using money flow statistics?* and *How would these solutions perform on a realistic dataset of financial transactions?*. Coming as an answer to the first question, literature in the field proposes five algorithms, FlowScope, CubeFlow, WeirdFlows, DenseFlow and SMoTeF, out of which four have been chosen to be benchmarked on the AMLWorld dataset, which provides a realistically-generated dataset of synthetic financial transactions, by measuring the minority class F1 score achieved on recognizing laundering transactions. This benchmark has aimed to answer the second research question.

The results, although significantly lower than those of Graph Neural Networks, reveal several limitations of the flow statistics algorithms, which opens the doors for further research into how these limitations can be overcome such that flow statistics become a feasible solution to money laundering detection.

Firstly, as observed in the analysis of FlowScope and CubeFlow, the detection of laundering chains by chasing accounts with a high difference between flow and retention of money is overshadowed by very dense chains of transfers, which do not necessarily represent laundering transactions. Further research could explore how the the metric for identifying laundering chains with low middle account retention could be modified such that the Zero-Out property has more weight than the density of transfers. One suggestion for this matter could be using the percentage of the flow that money retention represents, instead of using their difference.

Secondly, one persisting issue in the analysis of the algorithms is the inability of the flow statistics algorithms to adapt to low-volume and dispersed patterns such as Fan-In, Fan-Out or Stack. Therefore, further research could explore how flow statistics can be used in combination with graph mining algorithms to identify complex patterns and filter out false positives using flow statistics. SMoTeF shows that maximum flow analysis manages to efficiently filter out all false positives from pre-identified Scatter-Gather patterns.

Finally, for algorithms considering the temporal difference between transactions in a laundering pattern in order to satisfy the Fast-In/Fast-Out property, one limitation is the inability to detect patterns loosely spread throughout time. Future research could explore how time binning could be performed dynamically in order to be able to detect patterns even if they are spread across a large part of the dataset timeframe.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications.* Prentice Hall, 1993.

[2] Erik R. Altman, Jovan Blanusa, Luc von Niederhäusern, Beni Egressy, Andreea Anghel, and Kubilay Atasu. Realistic synthetic financial transactions for anti-money laundering models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information*

Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023.

[3] V.K. Balakrishnan. *Schaum's Outline of Graph Theory: Including Hundreds of Solved Problems*. Schaum's outline series. McGraw-Hill Education, 1997.

[4] Jovan Blanusa, Maximo Cravero Baraja, Andreea Anghel, Luc von Niederhäusern, Erik R. Altman, Haris Pozidis, and Kubilay Atasu. Graph feature preprocessor: Real-time subgraph-based feature extraction for financial crime detection. In *Proceedings of the 5th ACM International Conference on AI in Finance, ICAIF 2024, Brooklyn, NY, USA, November 14-17, 2024*, pages 222–230. ACM, 2024.

[5] J. Adrian Bondy and Uppaluri S. R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2008.

[6] Arthur Capozzi, Salvatore Vilella, Dario Moncalvo, Marco Fornasiero, Valeria Ricci, Silvia Ronchiadin, and Giancarlo Ruffo. Flowseries: Anomaly detection in financial transaction flows. *CoRR*, abs/2503.15896, 2025.

[7] Dawei Cheng, Yao Zou, Sheng Xiang, and Changjun Jiang. Graph neural networks for financial fraud detection: A review. *CoRR*, abs/2411.05815, 2024.

[8] Andrew V. Goldberg and Robert Endre Tarjan. A new approach to the maximum flow problem. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 136–146. ACM, 1986.

[9] Prashank Kadam. Financial fraud detection using jump-attentive graph neural networks. *CoRR*, abs/2411.05857, 2024.

[10] Eren Kurshan, Honda Shen, and Haojie Yu. Financial crime & fraud detection using graph computing: Application considerations & outlook. *CoRR*, abs/2103.01854, 2021.

[11] Meng-Chieh Lee, Yue Zhao, Aluna Wang, Pierre Jinghong Liang, Leman Akoglu, Vincent S. Tseng, and Christos Faloutsos. Autoaudit: Mining accounting and time-evolving graphs. In Xintao Wu, Chris Jermaine, Li Xiong, Xiaohua Hu, Olivera Kotevska, Siyuan Lu, Weija Xu, Srinivas Aluru, Chengxiang Zhai, Eyhab Al-Masri, Zhiyuan Chen, and Jeff Saltz, editors, *2020 IEEE International Conference on Big Data (IEEE BigData 2020), Atlanta, GA, USA, December 10-13, 2020*, pages 950–956. IEEE, 2020.

[12] Xiangfeng Li, Shenghua Liu, Zifeng Li, Xiaotian Han, Chuan Shi, Bryan Hooi, He Huang, and Xueqi Cheng. Flowscope: Spotting money laundering based on graphs. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 4731–4738. AAAI Press, 2020.

[13] Dan Lin, Jiajing Wu, Yunmei Yu, Qishuang Fu, Zibin Zheng, and Changlin Yang. Denseflow: Spotting cryptocurrency money laundering in ethereum transaction graphs. In Tat-Seng Chua, Chong-Wah Ngo, Ravi Kumar, Hady W. Lauw, and Roy Ka-Wei Lee, editors, *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 4429–4438. ACM, 2024.

[14] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 809–816. Omnipress, 2011.

[15] Jason Piper and Alex Metcalfe. Economic crime in a digital age. Technical report, Association of Chartered Certified Accountants (ACCA) and Ernst & Young (EY), January 2020. Report published jointly by ACCA and EY.

[16] Shiva Shadrooh and Kjetil Nørvåg. Smotef: Smurf money laundering detection using temporal order and flow analysis. *Appl. Intell.*, 54(15-16):7461–7478, 2024.

[17] Michele Starnini, Charalampos E. Tsourakakis, Maryam Zamanipour, André Panisson, Walter Allasia, Marco Fornasiero, Laura Li Puma, Valeria Ricci, Silvia Ronchiadin, Angela Ugrinoska, Marco Varetto, and Dario Moncalvo. Smurf-based anti-money laundering in time-evolving transaction networks. In Yuxiao Dong, Nicolas Kourtellis, Barbara Hammer, and José Antonio Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track - European Conference, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part IV*, volume 12978 of *Lecture Notes in Computer Science*, pages 171–186. Springer, 2021.

[18] Xiaobing Sun, Jiabao Zhang, Qiming Zhao, Shenghua Liu, Jinglei Chen, Ruoyu Zhuang, Huawei Shen, and Xueqi Cheng. Cubeflow: Money laundering detection with coupled tensors. In Kamal Karlapalem, Hong Cheng, Naren Ramakrishnan, R. K. Agrawal, P. Krishna Reddy, Jaideep Srivastava, and Tanmoy Chakraborty, editors, *Advances in Knowledge Discovery and Data Mining - 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11-14, 2021, Proceedings, Part I*, volume 12712 of *Lecture Notes in Computer Science*, pages 78–90. Springer, 2021.

A Visualizations

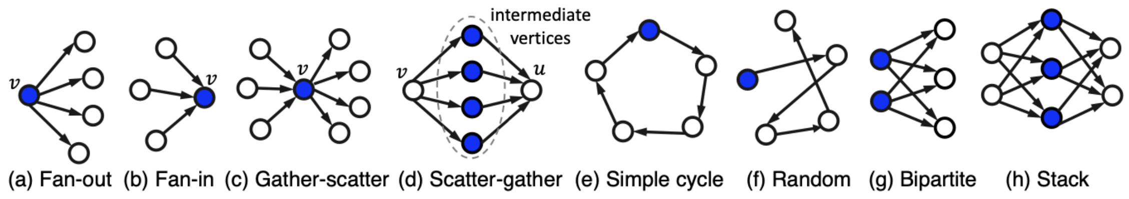

Figure 4: Money laundering patterns [2].

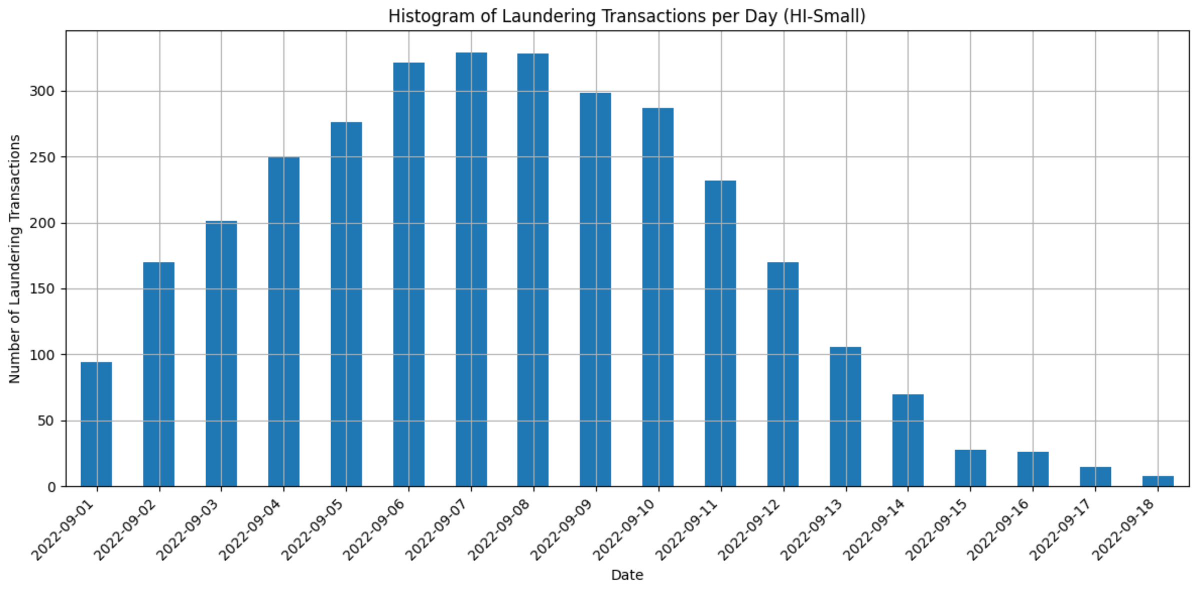

Figure 5: Time distribution of laundering transactions

B Formulas

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} v & \text{for } i = s, \\ 0 & \text{for all } i \in N \setminus \{s,t\}, \\ -v & \text{for } i = t \end{cases} \tag{4}$$
$$0 \le x_{ij} \le u_{ij} \quad \text{for each } (i,j) \in A.$$

Figure 6: Flow conservation and capacity constraints for the maximum flow problem. [1]

$$R(i,j,a,t) = \frac{A([t-d, t+d])}{A(\tau(t))} \tag{5a}$$

$$\eta_i(S) = \sum_{(j,i)\in E \wedge j \in S} e_{ji} \tag{5b}$$

$$\text{bal} = \min\left\{ \frac{\eta_i(S)}{\eta_i(V \setminus S)}, \frac{\eta_i(V \setminus S)}{\eta_i(S)} \right\} \tag{5c}$$

$$\gamma_i(S) = \text{bal} \times \text{KL}\left[R_i(S), R_i(V \setminus S)\right] \tag{5d}$$

Figure 7: Suspicious rating (a), balance factor (c) and monetary suspiciousness (d) for DenseFlow.

$$F_{1,\text{ minority}} = 2 \cdot \frac{\text{Precision}_{\text{minority}} \cdot \text{Recall}_{\text{minority}}}{\text{Precision}_{\text{minority}} + \text{Recall}_{\text{minority}}} \tag{6a}$$

$$\text{Precision}_{\text{minority}} = \frac{\text{TP}_{\text{minority}}}{\text{TP}_{\text{minority}} + \text{FP}_{\text{minority}}} \tag{6b}$$

$$\text{Recall}_{\text{minority}} = \frac{\text{TP}_{\text{minority}}}{\text{TP}_{\text{minority}} + \text{FN}_{\text{minority}}} \tag{6c}$$

Figure 8: F1-score (a), Precision (b) and Recall (c) formulas specific to the minority class.