

Received March 20, 2022, accepted April 21, 2022, date of publication April 25, 2022, date of current version May 6, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3170467

# Anomaly Detection in Graphs of Bank Transactions for Anti Money Laundering Applications

BOGDAN DUMITRESCU<sup>1</sup> , (Member, IEEE), ANDRA BĂLTOIU<sup>1</sup>, AND ȘTEFANIA BUDULAN<sup>2</sup>

<sup>1</sup>Department of Automatic Control and Computers, University Politehnica of Bucharest, 060042 Bucharest, Romania

<sup>2</sup>Tremend Software Consulting, 030134 Bucharest, Romania

Corresponding author: Bogdan Dumitrescu (bogdan.dumitrescu@upb.ro)

This work was supported by the Grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI-UEFISCDI, through the PNCDI III under Project PN-III-P2-2.1-PED-2019-3248 (Graphomaly).


**ABSTRACT** Our aim in this paper is to detect bank clients involved in suspicious activities related to money laundering, using the graph of transactions of the bank. Although we have a labeled real dataset, our target is not only to obtain relevant results on it, but also on random graphs in which typical anomaly patterns have been injected. So, we want simultaneously adequacy to the real data and robustness. Our method is based on designing new features; the most important are those resulting from the reduced egonet, which is the subgraph that remains from an egonet after eliminating the nodes connected with a single edge to the center; another feature is built by appealing to random walks and serves as indicator of circular flows. Our features are added to usual egonet features and a general anomaly detection algorithm, in our case Isolation Forest, serves to detect the anomalies. Experiments on the real data and a comprehensive set of synthetic data show that our approach is adequate, robust and better than some previous methods.

**INDEX TERMS** Anomaly detection, bank transactions, money laundering, graphs, egonet, random walk.

## I. INTRODUCTION

Money laundering is an activity through which illegally obtained money are introduced and circulated as apparently legitimate transactions, such that their source is difficult to track. The amount of money subject to laundering has been estimated at between 2% and 5% of the global Gross Domestic Product (GDP), although, as argued in [1], a significant part of this amount is hard to trace and does not even enter the banking system. In any case, only a very small amount of the laundered money, estimated at 1%, is seized. Since the stakes are so high, states have institutions and banks have divisions dedicated to Anti Money Laundering (AML) activities.

There are many AML methods, our concern being only in data analysis, specifically by looking at the graph representation of bank transactions. Our angle is that of anomaly detection (AD) in the graph, thus using a machine learning approach.

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad Ayoub Khan .

## A. THE PROBLEM

The input data is the transaction list of a bank during a certain time window. A directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is extracted from the transaction list; the nodes (vertices) from  $\mathcal{V}$  are accounts; the edges from  $\mathcal{E}$  represent transfers between two accounts; the transferred sums are the edges weights. There is a unique edge  $(k, \ell) \in \mathcal{E}$  between nodes  $k, \ell \in \mathcal{V}$ , with only two attributes: the total sum transferred from  $k$  to  $\ell$  during the time window, denoted  $s_{k\ell}$ , and the number of transactions,  $n_{k\ell}$ ; the money amounts are converted to a single currency. At least one of the two accounts involved in a transaction is a client of that bank; the other can be a client of another bank. When several accounts can be identified as belonging to the same client, they are aggregated in a single node. The purpose is to find anomalous nodes, in a broad AML sense.

This setup is a simplified view of the bank transactions, ignoring some information that may be useful, like the currency, the exact time of the transaction, the modality in which the transfer was ordered, the amount available in the source and/or destination accounts. However, it is highly informative on the network of relations between accounts.

Most banks have rule-based warnings that point to suspicious transfers, based on regulations, experience and own understanding of illegal financial activities. For example, a large transfer to/from a fiscal paradise is a certain trigger for further verification. Rule-based AML tools can be very effective in spotting isolated dubious transactions or even small fraudulent networks, but may fail in the face of an intricate money laundering scheme.

The graph-based approach is not seen as a replacement of the rule-based system, but as a complementary tool that can detect more complex or new fraud schemes. We also note that AML is more difficult for a single bank, due to the intrinsic partial access to the operations of a criminal group.

## B. OUR CONTRIBUTION

There are many methods for anomaly detection in graphs and they cover many facets of the AML problem as posed above. An overview will be presented in Section II. However, the AML task is simply too complex to be definitively solved. Scarcity of public data contributes to the difficulty of assessing which approaches and algorithms are the most appropriate.

We propose a method shaped in a standard format. First, we design and compute node features that capture relevant information from the graph. Then, an AD algorithm is run on the computed feature values to detect the abnormal nodes.

Specifically designed features have the advantage that they can be based on direct insight on money laundering schemes and can more easily be accepted by banking experts, although machine learning algorithms still have an important role. Graph features or statistical scores have been extracted from graphs in several previous works, notably [2] (egonet features), [3] (local connectivity features, like the shortest distance between the endpoints of a transaction, or global features like the page rank), [4] (features related to spectral localization, community properties, node connectivity, NetEMD [5] measures of the difference between an expected network and the network at hand).

Our contribution consists of proposing new features resulted from *reduced egonets*, i.e., egonets from which the nodes connected with a single edge to the center are removed. Their differences with respect to the egonets enhance the peculiarities of fraud patterns, thus making easier the work of AD methods in isolating the true anomalies. We also introduce features derived from *random walks*, especially related to the amount of money that return to a node through a cycle.

We are in the good position of having access to real bank transactions, labeled by AML specialists. However, since this is still a partial view, *our goal is to provide graph anomaly detection methods that are suited for this real dataset as well as for random graphs in which typical fraud patterns were injected*. So, we simultaneously want adequacy to real data, which are in limited supply due to the privacy constraints of the banks, and robustness by addressing generic structures. Like in [3], we aim to complement the existing systems based on rules or analytics, not to replace them. Also, we want not to

excessively tune the methods to the specifics of a single bank, but to leave place for generality. Comprehensive experiments show that our goal is attained in good measure.

## C. CONTENT OF THE PAPER

Our paper is organized as follows. We start in Section II with an overview of previous work, trying to outline the existing main ideas. Section III describes the characteristics of our datasets; understanding the data structures is central in graph analysis, due to the complexity of the possible abnormal patterns. Section III-A presents a real graph of transactions provided by a bank, with two levels of annotations for suspicious transactions. Synthetic data are the subject of Section III-B; they consist of random graphs in which typical anomaly patterns have been injected. Section IV presents the proposed features, derived from reduced egonets (Section IV-A) and random walks (Section IV-B). Section V gives the feature sets computed for our data, on which anomaly detection methods can be run. Section VI contains the results of our methods and a few well known ones, expressed especially in terms of the true positive rate. Our methods offer a good mix of results on real and synthetic data; the reduced egonet features appear to be the most robust. We publish on our website the real data and programs for generating the synthetic data, as well as the Python implementation of our methods.

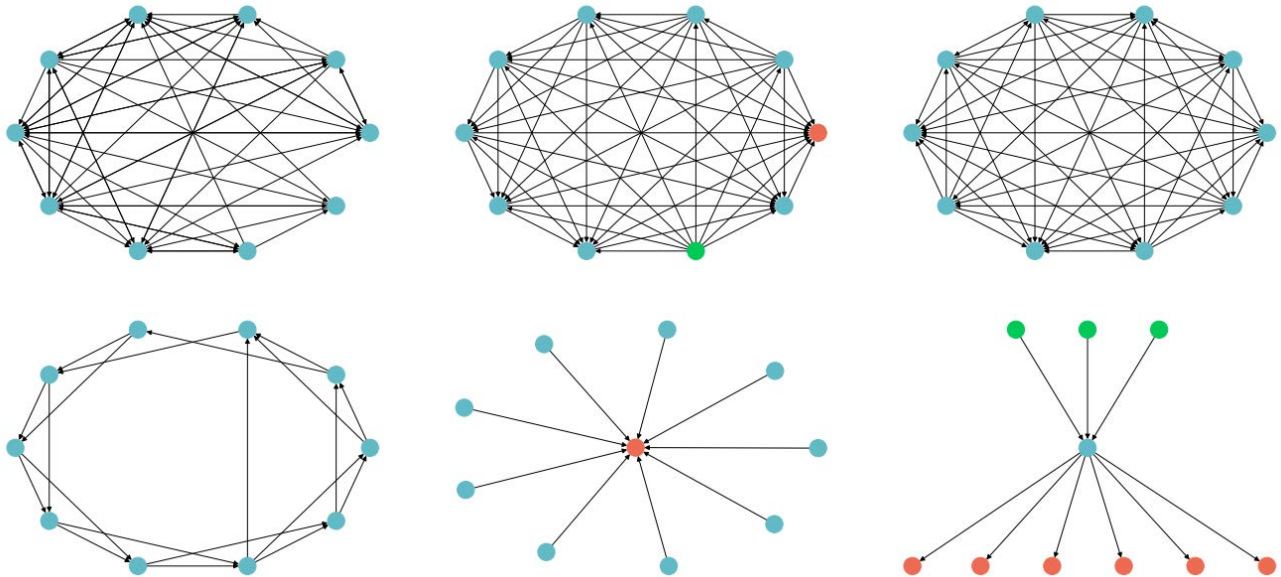
## II. BACKGROUND AND RELATED WORK

Anomaly detection in graphs is a very active topic, with many applications, most of the research taking place in the latest ten years. We enumerate here the most important lines of attack, many of them directly related to AML. We also describe the position of our approach with respect to them.

### A. DETECTING KNOWN PATTERNS

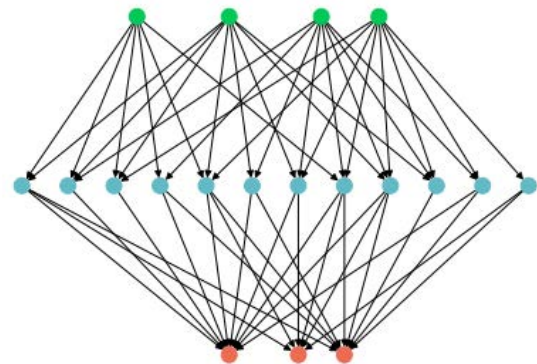
Money laundering often generates subgraphs with special topologies. Several such patterns are illustrated in Figure 1. Even though the real instances of these patterns have not exactly the shown ideal form, detection methods inspired by them have good behavior. Although indirectly, the special subgraphs discussed below have inspired our feature design, see especially the examples from Section IV-A.

- Nodes with high out or in traffic, named volcanoes and blackholes in [6], or stars in other works, may indicate the beginning or the end of a flow of money with dubious source. Flowscope [7] targets a more evolved scheme, where money flows from a few sources to a few destinations through middle accounts that serve only as buffers, as illustrated in Figure 2. This pattern corresponds to what in the AML community is known as the placement-layering-integration model [8], with each activity being carried out by the source, middle and destination nodes respectively. Intermediate accounts are also considered in [9], which uses a tensor approach to modelling tripartite patterns relevant to money laundering.



**FIGURE 1.** Anomaly structures: clique (top left), clique directed (top middle), clique random (top right), ring (bottom left), star (bottom middle), star directed (bottom right).

- A cycle is a classic fraud pattern, especially if the amount transferred over it is nearly constant; in [10], cycles with various lengths are detected in real time, using a hot point index to speed up the search.
- Cliques or dense subgraphs may show attempts to mask the money flow and make analysis difficult. Despite the impression that cliques should be conspicuous, it was proved in [11] that finding the densest subgraph of a certain size is NP-hard. **OddBall** [2] is a very successful method with this purpose, using simple statistical measures. In [12] high density subgraphs are found by efficiently solving a nonconvex quadratic programming problem. A **spectral perspective is considered in [13]**, where the adjacency characteristics of dense subgraphs are analyzed.
- All the above patterns and others (for example, heavy paths, which represent successive transfers of a large sum between several accounts) are targeted in [4], where more than one hundred graph related statistics are proposed, directly giving anomaly scores by their departure from normality, where “normal” behavior (the null model) is ingeniously sampled from the whole graph. CoDetect [14] targets also several possibly fraudulent patterns (outlier point, merge, ring) and models the weighted graph similarity matrix and the node feature matrix with low-rank decompositions.
- Although with no application to AML, but mostly for the discovery of fake reviews, FRAUDAR [15] detects dense subgraphs in bipartite graphs optimizing a suspiciousness function. EigenPulse [16] deals with a similar problem, but on streaming graphs.



**FIGURE 2.** Tripartite graph.

## B. STATISTICAL APPROACHES

Instead of searching (almost) fixed patterns, one may examine how normal nodes and their neighborhoods are, based on the distribution of certain relevant features or scores. Some of the approaches above may be seen as such, for example [2], which is based on deviations from a power-law distribution, and [4]. In [17], a normality measure is introduced to characterize neighborhoods of attributed networks. Earlier, in [18], a greedy beam search and the Minimum Description Length principle were used starting from the idea that normal graph structures are those leading to best compression. In [3], deviation from normality is defined by transactions outside the local community of a node, using easy-to-compute features that place or not a transaction within such a community; the main purpose is real-time card fraud detection; only amount

and time stamp are used as edge attributes. Indications of anomalous activity can also be found in the evolution of the graph in time. Dynamic features derived from this time-series approach can therefore also be considered, as done in [19], in order to distinguish bursts of activity from typical transaction regularity. MonLAD [20] considers a series of money laundering traits that describe network activity specific to fraudulent schemes by also taking temporal patterns into account. For example, star-like patterns are considered suspect if, in addition, the time between transactions is short. The authors define statistical features that summarize account activity based on these fraudulency assumptions and set up corresponding thresholds with respect to generalized Pareto distributions. The thresholds are subsequently used to compute anomaly scores and label transactions.

In a general sense, since we compute features, our approach has a statistical flavor. However, we do not explicitly compute statistical scores, but leave the task of discovering the anomalies in the ensemble of features to general purpose AD methods.

### C. LEARNING APPROACHES

More abstract approaches are based on learning using a global objective function. Our method has no resemblance with them.

A first example is that of building embeddings. Low-dimensional real vectors are associated with nodes, attempting to associate relations in the graph with distances between the vectors. For example, in *node2vec* [21], neighborhoods (sampled with random walks) are preserved in the low-dimensional space; anomaly detection is explicitly the goal in [22], with emphasis on nodes that are connected with many communities. *Structural deep network embedding* (SDNE) [23] preserves distances also in second-order proximities. *Netwalk* [24] proposed clique embedding via a deep *autoencoder neural network*, minimizing pairwise distance among vertices on random walks while encouraging sparsity; anomalies are nodes not belonging to clusters (have large distance to cluster centers).

Other learning approaches optimize a global objective function that characterizes normality or suspiciousness in a broad sense. Usually the optimization structure belongs to deep learning and the form is that of an autoencoder. The result consists of scores that are associated to nodes. For example, in [25], [26], the input is an attributed network, and the autoencoder works with both graph structure and node attributes. In [27], two decoders with different purposes, generative and contrastive, are used on the output of a single encoder to provide anomaly scores. Specifically oriented to financial fraud detection is [28], proposing deep learning using stacked auto-encoders and restricted Boltzmann machines.

### D. OTHER VIEWPOINTS

Besides the approaches described above for fraud detection in financial applications, there are others that do not fit our

classification. For example, social network statistics are used in [29] to assess risk profiles of the clients of a factoring company.

Our view of the previous work is certainly incomplete and subjective. A more general picture can be found in the review articles [30]–[35], the latter on methods using deep learning. A related field is that of network anomaly detection [36], [37], where the purpose is to find malicious actions over a computer network; although information flow is different from that of bank transfers, graph methods can be useful. An example is [38], where sudden changes in dynamic graphs are detected via changes in PageRank scores, with application to network intrusion detection.

Another classification may be made based on the issue of anomaly detection itself. Most of the methods propose scores that can be used directly to decide which nodes are more likely to be the outliers. Other methods, including ours and those producing embeddings, only build a set of features, which are then fed to an anomaly detection algorithm, like those from PyOD [39].

An important issue is that of complexity and scalability, since graphs involved in AML operations are usually large. Here the spectrum is quite large, from algorithms that work in real time to others that need huge resources and are impractical for very large graphs, with millions or even only hundreds of thousands nodes. Our method is placed somewhere in the middle, in the sense that it can be applied to large graphs, but does not operate in real time.

## III. DATASETS

Finding anomalies in a transaction graph entails having access to (ideally large) datasets. However, most of the industries for which this analysis would be a great fit, such as financial services, refrain from publicating datasets for the research community. Besides the information that can be anonymized, such as the source and destination IDs, other important features may be more difficult to conceal. Another impediment when dealing with real data comes from the transactions that are made to/from the outside environment, where we have little to no information about their corresponding destination/source. The alternative is to generate synthetic datasets, with the ideal of maintaining a natural anomalies-to-regular entries ratio, as well as inserting anomalies that would be found in a real-life scenario.

Credit Card Fraud Detection dataset [40], one of the few real-world datasets, contains transactions made by European cardholders, through their credit cards in 2013. This dataset contains 284,807 transactions out of which 492 transactions (0.173%) are fraudulent. Besides the amount and time information, the other public numerical values are obtained via Principal Component Analysis (PCA). In [3] real banking data also oriented to card fraud are used, but not made public.

IEEE-CIS Fraud Detection dataset [41] was collected and published by IEEE Computational Intelligence Society (IEEE-CIS) in collaboration with Vesta company; it contains real e-commerce transactions with a little over 590K



instances, of which 3.5% are fraudulent. Even though several anomaly detection methods can be applied here, a transaction graph cannot be associated with this type of data, given that the transactional interaction between users is not captured.

The Bitcoin blockchain is public and large, but the number of accounts is also very large and it is very difficult to group accounts by owner. Some AML advances are presented in [42].

Going now to synthetic datasets, PaySim [43] is a simulator that can generate datasets of mobile money transactions that are similar to real transactions, using **agent based modeling**. Frauds are modeled according to different scenarios by setting some parameters outside the normal interval. BankSim [44] is a similar solution for constructing bank transactions datasets and is suited for simulating payment frauds such as card theft or unauthorized internet purchases; however, it has no money laundering models. An example dataset has been made public [45], however it does not include all supported fraud types and further development has shifted to commercial settings [46]. A similar commercial agent-based solution is [47].

In this landscape, where data for testing AML algorithms are so scarce, we have the advantage of a real dataset, presented next.

#### A. LIBRA BANK DATASET

Libra Internet Bank, a Romanian bank, provided a complete list of transactions over three months,<sup>1</sup> from which we have extracted the graph corresponding to transfers between accounts.

It is worth mentioning that several steps of preprocessing have been performed, before extracting the transaction graph. These steps involved the alignment of labeled transactions with the transaction dataset, merging possibly duplicated information when both the originator and beneficiary entities of the transaction had accounts within the bank, or leaving aside other transactions like card payments. Notably, sometimes the lack of information from the outside environment may lead to imperfect or incorrect graph construction, for example by creating multiple nodes instead of a single one, in the case of multiple accounts belonging to the same external entity.

Here are a few characteristics of the Libra dataset, after preprocessing (the amounts are converted to local currency):

- number of transactions: 4558805
- mean transaction value: 3264
- median transaction value: 152
- max transaction value: 42 million
- min transaction value: 0.01

Figure 3 shows the distribution of the transferred amounts. We note that it can be approximated with a log-normal distribution, although having a somewhat heavier tail.

<sup>1</sup>In compliance with Regulation EU 679/2016 (“GDPR”) and other relevant legislation, no personal identifiable information (personal data) has been disclosed during the development of the present paper or of any associated work that has been done in relation to the present paper.

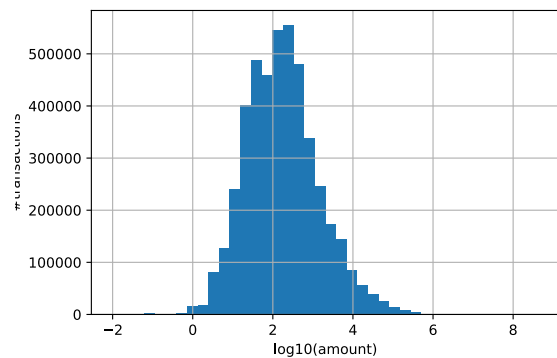


FIGURE 3. Histogram of transaction amounts for the libra dataset.

The associated graph of transactions, denoted  $\mathcal{G}_{\text{Libra}}$  has the following characteristics:

- number of nodes (distinct client IDs): 385100
- number of edges (recall that there is a single edge between two nodes, with cumulated amounts over all transfers between those nodes): 597165
- average in/out degree: 1.55

We see that the graph is sparse. Most of the clients are involved in a single transaction during the chosen 3-month time window.

The transactions have been labeled by members of the AML division of the bank. There are two kinds of labels. The first is named *alert* and is generated by an undisclosed set of rules; an alert means that the respective transaction has to be investigated by specialized personnel. The second kind of label is called *report* and marks a transaction whose level of suspicion is high enough to be reported for further examination to the state authorities that investigate money laundering. (So, note that a report does not necessarily indicate crime.) Obviously, a transactions labeled 'report' is also labeled 'alert'. Some information on alerts and reports is as follows:

- transactions with alerts: 517
- transactions with reports: 11
- distinct nodes involved in transactions with alerts: 600
- distinct nodes involved in transactions with reports: 15

We associate with each node anomaly weights for alerts and reports, defined as the number of transactions labeled as alerts or reports, respectively, to which the node participates. For example, the maximum number of alerts associated with a node is 22 and the maximum number of reports is 3. In general terms, we will also name anomaly an alert or a report. Some examples of neighborhoods of anomalous nodes will be shown in Section IV-A. The graph can be downloaded in csv format from <http://graphomaly.upb.ro/>.

#### B. SYNTHETIC DATASETS

Following previous work and notably [4], we have generated synthetic test graphs that have a general 'normal' structure

TABLE 1. Synthetic graphs properties.

Connection Probability	$\mathcal{G}_7$	$\mathcal{G}_{12}$	$\mathcal{G}_{17}$	$\mathcal{G}_{22}$
Min in-block	0.0003	0.0003	0.0003	0.0003
Max in-block	0.005	0.007	0.008	0.01
Min inter-block	0.0001	0.0001	0.0001	0.0001
Max inter-block	0.0002	0.0004	0.0006	0.0008
No. edges	437334	726825	878293	112866
No. connected pairs	377059	628017	877756	112814
Density	7.54	12.56	17.55	22.56

on which are injected anomalies suited for investigating fraud and particularly money laundering in bank transactions.

### 1) UNDERLYING GRAPH

The graph of normal (legitimate) transactions is generated using a **stochastic block model** (SBM) consisting of 50000 nodes and 5000 modules. The intuition behind the choice is that, in real financial networks, communities (formalized here as blocks) tend to be rather on the small side, with few actors maintaining regular activity among themselves.

The size of each module is randomly set such that each block contains between 0.01% and 0.9% of the total number of nodes. We start by constructing a directed graph based on connectivity probabilities for the nodes and, after building the underlying network structure, we add multiple edges between already connected nodes, in order to form the multidirected graph. More precisely, the SBM is defined in terms of edge probabilities either between nodes belonging to the same block or between two nodes from different blocks. In order to obtain graphs with different edge densities, we vary these probabilities.

We generate four types of graphs, named according to their density:  $\mathcal{G}_7$ ,  $\mathcal{G}_{12}$ ,  $\mathcal{G}_{17}$ ,  $\mathcal{G}_{22}$ , the index being the average degree of a node (in fact, the average degrees are slightly higher, see Table 1). Note that the degree is the sum of indegree and outdegree.

With the underlying graphs constructed as such, we further add edges between connected nodes. In order not to create artificially dense structures within the graph, we condition this addition on the existing local neighborhood connectivity of each node. As such, we first randomly choose an upper limit for the number of outgoing edges and check whether the threshold is already met by the existing connections of each node. Otherwise, we iterate through the neighborhood of the node and further add outgoing edges until the limit is reached. Thus, generally, graphs that are already dense have fewer multiple edges between two nodes, while less dense graphs present more multiple edges. In choosing the threshold, we take the maximum number of outgoing edges from a node to nodes belonging to different blocks to be 3, in all experiments. The number of outgoing edges from a node to other nodes in the same block, the maximum is 6, but for  $\mathcal{G}_{12}$ , where it is set to 10.

Table 1 presents the stochastic block model connectivity probabilities for each graph, together with the resulting

number of edges and average density. The row 'number of edges' corresponds to the multigraph. The row below it shows the number of edges in the compacted oriented graph, where there is a single edge between a pair of nodes, with cumulated amounts of all transactions between those nodes. The edge density is computed with the latter number of edges, as relevant for our approach. The number of edges is that of the graph containing anomalies, which are added as described next. All the reported values are means over three different instances of the same graph configuration.

### 2) ANOMALIES

We experiment with different structured anomalies that are typical to financial frauds (see again Section II for an introduction to the problem). These are cliques, stars and rings, shown in Figure 1.

For cliques, we devise three variants, which we call regular, directed and random. Regular cliques are constructed by first generating edges in both directions between every two nodes and then removing a fraction of these edges at random. As such, the resulting structure is not a complete clique. The fraction is set to 40% in all experiments. In directed cliques, there exists an edge between every two nodes, however the direction is kept the same, such that it mimics a flow of money that goes in one direction only; in Figure 1 (top middle), the orange node only receives (outdegree 0), the green node only sends (outdegree 9) and the blue nodes have outdegrees with all values from 1 to 8 (the clique has 10 nodes). Lastly, random cliques are fully connected cliques in which the direction of edges is assigned at random. As a result, it may happen that one node only has incoming edges (thus acting as a sink), or it may only have outgoing edges (thus acting as a source), but typically there is no consistent flow direction.

In order to obtain realistic rings, we employ a **Watts-Strogatz model** [48] with mean degree  $K = 4$  and rewiring parameter  $\beta = 0.2$ . So, the ring of full length is not necessarily complete and also there are shortcuts. The edges are oriented in the same direction, anti-clockwise in Figure 1 (bottom left).

We experiment with two types of star structures: regular and directed. Regular stars contain one node that only has incoming edges, while the others only have outgoing edges. In directed stars, the flow passes from nodes that only have outgoing edges, through a single node that collects these edges and is in turn connected to nodes that only have ingoing edges. Figure 1 (bottom middle and right) shows typical examples of each of these structures.

All anomalies contain 10 nodes. Each graph contains 5 rings, 15 cliques and 5 stars, regardless of their type. These anomalies are implanted in the underlying transaction graph by randomly selecting the nodes and adding the corresponding edges. We do not restrict nodes to belonging to only one anomaly; however, superpositions are rare: the number of anomalous nodes is usually 249 or 250. So, the percentage of anomalous nodes is 0.5%.

For each of the four types of graphs defined in Section III-B1, we experiment with different types of anomalies. We update our naming convention by adding a letter that shows the type of clique anomalies: no letter for regular clique, 'd' for directed clique, 'r' for random clique.

### 3) EDGE ATTRIBUTES

In order to obtain a realistic setup, transaction amounts follow a base-10 log-normal distribution, which resembles the observed real distribution in the Libra dataset, as shown in Figure 3. The location and scale of the distribution are set to 3 and 1 respectively. We then prune the values to the  $[1, 10^6]$  range; this operation decreases the mean to about 12000 from its theoretical value 14167; the median value of 1000 is only slightly affected. For each anomaly, when attributing the transaction amounts, we generate a normal distribution of mean and variance chosen uniformly at random from the following ranges:  $[10^4, 10^5]$  for the mean and  $[1000, 3000]$  for the variance.

## IV. NODE FEATURES

As mentioned in Section I-B, we propose sets of features that capture essential properties of the transaction graph. Then, a **general AD algorithm** is run on the computed feature values. The resulted scores are used to build a list in which the nodes are ordered decreasingly by abnormality likelihood.

Some of the features are basic information for a node  $k$  of the (directed) transaction graph  $\mathcal{G}$ . We denote

$$\mathcal{N}_i(k) = \{\ell \in \mathcal{V} \mid (\ell, k) \in \mathcal{E}\}$$

the set of in-neighbors of node  $k$ . Similarly,

$$\mathcal{N}_o(k) = \{\ell \in \mathcal{V} \mid (k, \ell) \in \mathcal{E}\}$$

is the set of out-neighbors. The set of *basic* features is

$$\begin{aligned} \mathcal{F}_{\text{basic}} = & \{\text{in/out degree}\} \\ & \cup \{\text{total in/out amount}\} \\ & \cup \{\text{average in/out amount}\} \end{aligned} \quad (1)$$

where the features are defined as follows:

- *in/out degree*: the numbers  $d_i(k) = |\mathcal{N}_i(k)|$  and  $d_o(k) = |\mathcal{N}_o(k)|$  of edges that enter or exit node  $k$ ;
- *total in/out amount*, the total amount of money that enter/exit node  $k$ :

$$ta_i(k) = \sum_{\ell \in \mathcal{N}_i(k)} s_{\ell k}, \quad ta_o(k) = \sum_{\ell \in \mathcal{N}_o(k)} s_{k\ell};$$

- *average in/out amount*, which is simply the total amount divided by the total number of transactions involving node  $k$  as destination/source; for example, the *average in amount* is

$$aa_i(k) = \frac{ta_i(k)}{\sum_{\ell \in \mathcal{N}_i(k)} n_{\ell k}}.$$

Since money laundering is efficient only if large amounts of money are manipulated (in any case clearly larger than

usual transfers), we expect the total and average amounts to contain relevant information. The degree is also relevant for the connectivity of a node.

The *egonet* has been shown [2] to be a good tool for the detection of some fraud structures, especially cliques and stars. The *egonet*  $E(k)$  (of radius 1) of node  $k$  is the subgraph of  $\mathcal{G}$  generated by node  $k$  and all its direct neighbors. So, its nodes are

$$k \cup \mathcal{N}_i(k) \cup \mathcal{N}_o(k) \quad (2)$$

and its edges are all edges in  $\mathcal{E}$  between these nodes. Similarly, one can define the in- and out-egonets, but we will use only the full *egonet*. Also, *egonets* taking into account extended neighborhoods could be considered, but their size can become impractical for large graphs. Moreover, from the perspective of a single bank, the *egonets* of radius 2 give a more distorted view than those of radius 1, since the connections between the clients of other banks are not available.

We note that the features from  $\mathcal{F}_{\text{basic}}$  are obviously also *egonet* features. A specific *egonet* feature is

- *egonet edge density*, which is the ratio between the number of edges of the *egonet* and its number of nodes.

Other features can be considered, like the *egonet weight*, which is the sum of amounts on all its edges, as well as other derivated features.

All these features have been used in a form or another in previous research. Besides them, we propose some new ones, as described below.

### A. REDUCED EGONET

**Definition 1:** The *reduced egonet* (or *egored*)  $E_{\text{red}}(k)$  of node  $k \in \mathcal{V}$  is its *egonet*  $E(k)$  from which we remove the nodes that are connected only with  $k$ , with a single edge, no matter its direction. ■

**Example 1:** We give here of two examples from the Libra dataset, built around nodes with reports. Figure 4 shows an *egonet* and the corresponding *egored*. The colors are: red for the central node, orange for other nodes with reports, blue for the normal nodes that have at least two edges (including the case, visible in the figure, where both edges are with the center node, in both directions) and green for the normal nodes with a single edge. The *egonet* has 24 nodes and the *egored* only 9 nodes. Figure 5 shows another example, now containing three nodes with reports; the *egonet* has 26 nodes and the *egored* has 19. ■

The basic features can be immediately redefined for the *egored*. For example, the *egored indegree* is the number of edges from  $E_{\text{red}}(k)$  that enter node  $k$ . Obviously, the relation *egored indegree*  $\leq$  *indegree* holds. So, we add to our list of possible features the *egored* total amounts, average amounts and edge density.

Moreover, if we want a normalization, we can define, for example, the *egored relative indegree*, which is the ratio between the *egored indegree* and the *indegree*. Hence, this is a number between 0 and 1. Relative values can be defined sim-

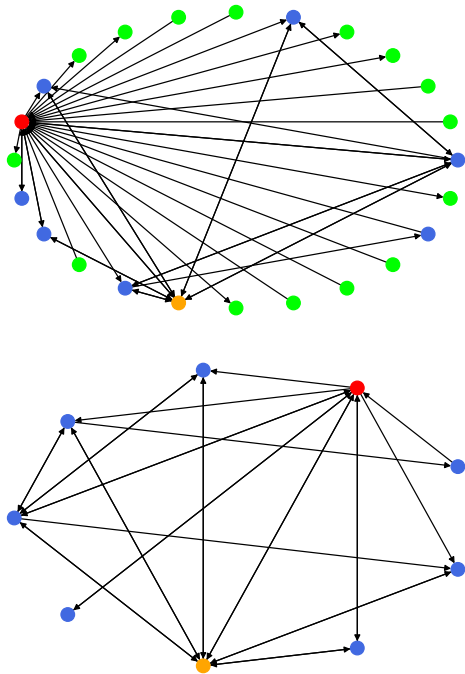


FIGURE 4. Egonet (top) and egored (bottom) of an anomaly from the Libra dataset.

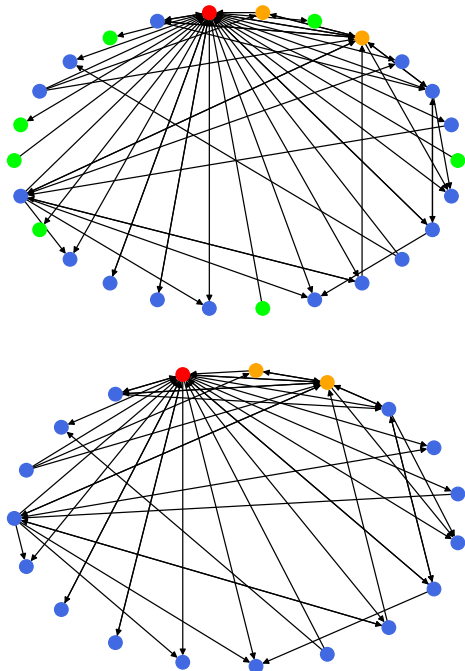


FIGURE 5. Egonet (top) and egored (bottom) of another anomaly from the Libra dataset.

ilarly for the other egored features. Note that while the *total amount* cannot be larger in the egored, the *egored average*

*amount* may be in any relation with the egonet *average amount*.

The benefits of egoreds can be seen when looking at the differences between the egonet and the egored features. They are biased towards the extreme in some fraud patterns compared with a normal node.

- Nodes that are tightly connected (near-cliques) with possibly heavy transactions, but that have also legitimate activities with other accounts, are more visible in the egored because the legitimate transactions go often through isolated nodes. Hence, the egored amounts are nearly equal to the egonet amounts, but the egored density is clearly higher and also the average amounts can be higher.
- For a pure star pattern, the egored of the center node contains only that node. For a near-star pattern, the egored is still almost depleted of nodes and the amounts are much smaller than in the egonet.
- In a tripartite graph, like the FlowScope [7] AML model, made of sources, intermediaries and destinations (see Figure 2), the egored has clearly distinguishing features. The sources/destinations send/receive large amounts to/from intermediaries and have relatively few interactions with other accounts; they are similar to volcano/blackhole (star) graphs [6]. The intermediaries have balanced and fairly large in-out amounts. In all cases, their egoreds are almost empty and show only small amounts, hence they are very different from the egonets. It is also quite unlikely that sources-intermediaries-destinations are all at the same bank, so it is hard to see the full tripartite graph, but only some of the accounts and the transactions. However, the relative aspects of the egored and egonet are the same even if only part of the graph is accessible.

**Example 2:** Coming back to Figure 4 we see that for real data the anomaly patterns are not necessarily so obvious. However, the average in/out degree, which is equal to the edge density, is 1.79 for the egonet and 3.11 for the egored, which is a quite significant change. Similarly, in Figure 5, the edge density is 2.12 for the egonet and 2.53 for the egored; the growth is smaller but still present. We note that, in both cases, the anomalous nodes are present in the egored. The opposite situation can be seen in Figure 6, also built from the Libra dataset, where the egonet is a directed star, which is a textbook pattern; only the egonet is shown, since the egored is the center node alone. Note, however, that only two transactions (and three nodes) are labeled as reports. ■

**Example 3:** Figure 7 shows the distributions of the *outdegree* and *egored outdegree* for all nodes of an instance of the synthetic graph  $\mathcal{G}_{22}$  and also for the anomalies in the same graph. Although the distribution of the *outdegree* for the anomalies has clearly a larger mean, its shape is not so different from that of normal nodes; the proportion of anomalies for a given value of the degree does not vary significantly. The *outdegree* does not give sufficient information



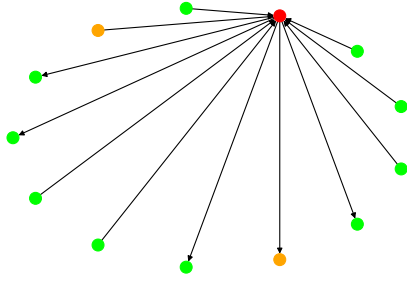


FIGURE 6. A directed star egonet around an anomaly of the Libra dataset.

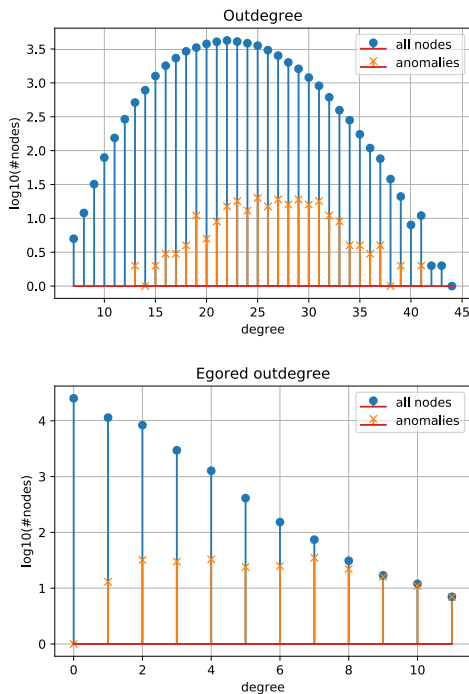


FIGURE 7. Outdegree (top) and egored outdegree (bottom) distribution for a synthetic graph  $\mathcal{G}_{22}$ , for all nodes and anomalies.

for distinguishing anomalies. On the contrary, the *egored outdegree* of the anomalies is clearly informative; larger values of this feature are much more likely for anomalies than for normal nodes. ■

For the Libra dataset the direct benefit of each egored feature is not that obvious. However, several egored features together appear to give benefits, as shown in Section VI.

## B. RANDOM WALK FEATURES

Random walk (RW) is a simple tool for graph analysis (and not only) that can be used in many circumstances. Anomaly detection in graphs has seen several ways of putting RW to work, a direction initiated in [49]. In [50], RWs use transition probabilities based on a local information graph; the purpose is to detect less connected points, associated with anomalies, a situation rather opposite to ours. In some neural network approaches, for example [24], [27], RWs are directly used as input for the network, as a sampling method for the input

graph. We use RWs to extract information on the amounts circulated over long paths, especially in the case of cycles (rings).

For a node  $k$ , we generate random walks with given maximum length  $r$ , starting or ending in  $k$ . (The backward RWs, ending in  $k$ , are generated forward on the reversed graph.) Denote  $k_0 = k, k_1, \dots, k_r$  the nodes of a RW. On each RW, we compute the amount transferred from start to end:

$$\min_{i=1:r} s_{k_{i-1}, k_i}. \quad (3)$$

Normally, this amount is small, as there is little correlation between the transactions of the nodes. When the amount is large, it may show money deliberately circulated through many accounts in an attempt to cover their origin. Especially important is the case when the RW returns to the start node  $k_0$ . If that happens, we cut the RW to the current length.

**Definition 2:** The feature *RW ring max* associated with node  $k$  is the maximum of the amount (3) over all the RWs of length at most  $r$  starting or ending in node  $k$ , for which there is an  $i \leq r$  such that  $k = k_0 = k_i$  in (3); if there is no such RW, the feature value is zero. ■

In other words, *RW ring max* is the maximum amount that is transferred over a cycle (or ring) containing the node. We have computed and used other features, like the average or the maximum amount over a RW, but we will not report their results, as they seem to bring no extra benefits for the data considered in this study.

An important RW issue is that of the modality for choosing, once in a node  $k$ , the next node from  $\mathcal{N}_o(k)$  (when generating a forward RW) or  $\mathcal{N}_i(k)$  (for a backward RW). We adopted two methods to choose the transition probability:

- equal probability; this method was used with the Libra data, since the graph is sparse and the likelihood of finding relevant paths is large;
- probability proportional with the amount; so, the probability of choosing node  $\ell \in \mathcal{N}_o(k)$  in a forward RW is

$$\frac{s_{k\ell}}{\sum_{j \in \mathcal{N}_o(k)} s_{kj}};$$

this method was used with the synthetic data, where the edge density is large and so random walks must be helped to “follow the money”.

Of course, the second method is slower, since we have to read an edge attribute for each neighbor.

## V. ANOMALY DETECTION ALGORITHMS

We combine now the features defined in Section IV in sets of features to be used for anomaly detection. Only the feature sets that proved successful or are natural building stones for our construction will be presented, although we have tried many other combinations.

The egonet related feature set is

$$\mathcal{F}_{\text{egonet}} = \mathcal{F}_{\text{basic}} \cup \{\text{egonet edge density}\}, \quad (4)$$

where the basic features are defined in (1).

We extend this set with the similar egored features and obtain

$$\begin{aligned}\mathcal{F}_{\text{egored}} = & \mathcal{F}_{\text{egonet}} \cup \{\text{egored relative in/out degree}\} \\ & \cup \{\text{egored relative total in/out amount}\} \\ & \cup \{\text{egored average in/out amount}\} \\ & \cup \{\text{egored edge density}\}\end{aligned}\quad (5)$$

Although egonet and egored features have equal share, hereafter we name the above set *egored* feature set (and only sometimes *egonet+egored*). Note that the absolute value of the egored average amount is used rather than the relative one, as it gives better results, although in principle there should be no significant difference.

Finally, we make use of a single random walk feature, added to the egonet features:

$$\mathcal{F}_{\text{egonet+RW}} = \mathcal{F}_{\text{egonet}} \cup \{\text{RW ring max}\}\quad (6)$$

In the synthetic datasets, the number of transactions between two nodes is small and also has small variation, hence we remove in/out average amounts from (1) and (5) and obtain features sets denoted  $\mathcal{F}'_{\text{egonet}}$ ,  $\mathcal{F}'_{\text{egored}}$  and  $\mathcal{F}'_{\text{egonet+RW}}$  as variations of those defined in (4), (5) and (6), respectively. The reason is that average amounts essentially duplicate the total amount information (unlike in the Libra case, where average and total amounts have quite a different distribution), thus giving it too much weight with respect to the topology information carried by the other features.

*Remark 1:* The Python implementation for extracting the above features is relatively straightforward. We note a somewhat unexpected fact. The egonet of a node can be computed using the Networkx function `ego_graph`; since we want the full egonet, we need to first convert the oriented graph to an unoriented one; surprisingly, the function `ego_graph` is quite slow in computing an egonet of radius 1. In our experience, it is much faster to simply identify the in/out neighbors using the functions `predecessors/successors` and then extract the subgraph corresponding to the nodes from (2) using the `subgraph` method.

After computing the features, the anomaly detection is made with Isolation Forest (IF) [51], using the implementation from PyOD [39]. Although any other AD method could be used, we selected IF due to the good performance with our data and reasonable execution time.

## VI. EXPERIMENTAL RESULTS

We report here results of our method described in Section V, with feature sets egonet (4), egored (or rather egonet+egored) (5), or egonet+RW (6), and also results of other methods that have proved successful in graph anomaly detection problems with known patterns. OddBall [2] is extremely successful in detecting near-cliques. The method from [4] can detect several types of patterns; unfortunately, its complexity forbids the computation of many of the proposed statistics for a graph the size of the Libra one. We selected a set of simple statistics, namely those using the Geometric

Average of Weights (GAW, GAW10, GAW20) and the standardised version of the node degree; this is called the basic module in [4, Sec.3.1]; here, we call it GAW; the result is directly a score, hence no AD method is required.

The Python implementations of our algorithms are available at <http://graphomaly.upb.ro/>.<sup>2</sup> We have used the OddBall implementation given by the authors at <https://www.andrew.cmu.edu/user/lakoglu/tools/Oddball-lite.zip>. The GAW implementations have been taken from the sources published by the authors at <https://sites.google.com/site/elliottande/anomalydetection>. The tests have been made on a laptop with an i7 processor with 6 cores and 16 GB memory. Isolation Forest was taken from PyOD [39]; we have run it with the default parameters, with the exception of the number of trees, which was taken equal to 200 instead of the default 100 value.

The true positive rate (TPR) or recall is the ratio of true positives and total number of anomalies. Since for both Libra and synthetic data a node can be in several suspicious transactions, we associate to node  $k$  the anomaly weight  $w_k$ , which is the number of these transactions; a normal node has  $w_k = 0$ . Denoting  $W = \sum_{k \in \mathcal{Y}} w_k$  the sum of all weights and assuming the nodes are ordered by an AD method in decreasing order of perceived abnormality, the TPR is

$$\text{TPR}(k) = \frac{1}{W} \sum_{i=1}^k w_k. \quad (7)$$

We note the  $W$  is twice the number of suspicious transactions; for Libra data,  $W = 1034$  for alerts and  $W = 22$  for reports; for the synthetic graphs,  $W = 1910$  for the graphs containing regular cliques and  $W = 1640$  for the graphs containing directed or random cliques.

Since the number of nodes is large and checking the fraudulent nature of the transactions involving an account requires the work of a human specialist, we are interested in finding the relevant anomalies in the first few outliers predicted by the AD method. We focus on the TPR for the first 0.1%, 0.2%, 0.5% and 1% of the nodes, with special emphasis on the first mark; note that 0.1% means 385 nodes for the Libra data, which is not a small number.

As a global measure, we also compute the TPR area under curve, defined as

$$\text{TPR AUC}(k) = \frac{1}{k} \sum_{i=1}^k \text{TPR}(i). \quad (8)$$

We report the TPR AUC for the first 1% of nodes and denote it  $\text{AUC}_{1\%}$ . For the sake of completeness, we also give the overall TPR AUC, computed over all nodes of the graph.

We note that TPR AUC is more relevant than receiver operating characteristic (ROC) AUC when anomalies are weighted, especially when the AUC is partial, only on 1% of nodes in our case. If two anomalies with different weights

<sup>2</sup>A more comprehensive software is the library Graphomaly, freshly released at <https://gitlab.com/unibuc/graphomaly/graphomaly>

**TABLE 2.** TPR results for several methods on the Libra graph. The best result on each column is in bold.

Method	Alerts					Reports				
	AUC_1%	TPR 0.1%	TPR 0.2%	TPR 0.5%	TPR 1%	AUC_1%	TPR 0.1%	TPR 0.2%	TPR 0.5%	TPR 1%
$\mathcal{F}_{\text{egonet}}$	0.5990	0.2873	0.4365	<b>0.6607</b>	<b>0.8149</b>	0.5747	0.1455	0.3364	0.6727	<b>0.8500</b>
$\mathcal{F}_{\text{egored}}$	<b>0.6042</b>	<b>0.4045</b>	<b>0.5143</b>	0.6596	0.7446	<b>0.6032</b>	<b>0.3545</b>	<b>0.4364</b>	<b>0.6955</b>	0.7727
$\mathcal{F}_{\text{egonet+RW}}$	0.5991	0.3322	0.4652	0.6605	0.7855	0.5945	0.2818	0.3636	0.6818	0.8364
OddBall	0.2504	0.0445	0.0948	0.2514	0.4671	0.4183	0.1364	0.2727	0.5454	0.5909
GAW	0.5561	0.1982	0.4265	0.6286	0.7843	0.4250	0.1362	0.3636	0.5000	0.5909

have successive positions in the score-ordered list, ROC AUC is the same no matter the relative order of the two anomalies, while TPR AUC is larger if the anomaly with larger weight is first. On the other side, the perfect value for ROC AUC is 1, while for TPR AUC it must be computed from case to case; for our data, where the number of anomalies is very small, the perfect TPR AUC value is close to 1. For example, for the alerts in Libra data, TPR AUC is 0.99946; for the synthetic graph  $\mathcal{G}_{22}$ , it is 0.9983.

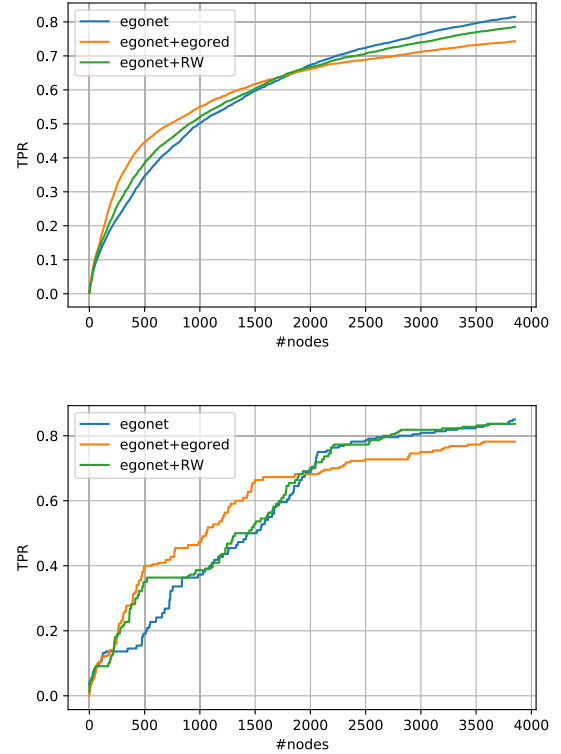
### A. RESULTS ON LIBRA DATASET

The results on the Libra graph of the methods listed in the beginning of the section are shown in Table 2, separately for alerts and reports. For our methods, the results are averaged over 10 IF runs. Figure 8 gives the TPR evolution on the first 1% predicted anomalies, for our methods.

The RW features were computed using RWs of length  $r = 5$  and 100 repetitions for each starting/ending node. OddBall gives anomaly scores for nine pairs of statistics. The results given in Table 2 are the best over all pairs; they are obtained by “egonet number of nodes vs egonet number of edges” for reports, and “egonet-in-degree vs egonet-in-weight” for alerts. The other statistics give rather poor results. We report here the results for the case where the (unique) weight on edge  $(k, \ell)$  used by OddBall was the total amount  $s_{k\ell}$ ; the results for average amounts are similar.

The results from both Table 2 and Figure 8 show that the combination of egored and egonet features is clearly the best on the first marks (0.1% and 0.2% of the number of nodes) and marginally better on AUC\_1%; note that, for Libra data, the ideal alerts TPR value at 0.1% is 0.793, which shows that more than half of the alerts than that can be discovered are discovered. Egonet features alone are more successful in the long run, but clearly less effective in the top scores region. The single RW feature added to them appears to boost performance on the initial anomalies, while diminishing the long term effect; it is not so effective as egored features, but can serve as a compromise approach. GAW, despite its simplicity, has a reasonable performance, although clearly below that of egonet+RW. Finally, OddBall obtains low performance indicators, although it cannot be totally dismissed, the results on reports being near those of GAW.

The overall AUC results, shown in Table 3, confirm the trend that can be extrapolated from Figure 8. As we said, these results have smaller significance, since we do not want

**FIGURE 8.** TPR on first 1% predicted anomalies for the Libra dataset. Top: alerts. Bottom: reports.**TABLE 3.** TPR AUC for several methods on the libra graph.

Method	Alerts	Reports
$\mathcal{F}_{\text{egonet}}$	0.9943	0.9951
$\mathcal{F}_{\text{egored}}$	0.9840	0.9906
$\mathcal{F}_{\text{egonet+RW}}$	0.9939	0.9948
OddBall	0.6114	0.8139
GAW	0.9912	0.9779

to emulate exactly the rule-base AD in use, but to extend it in a robust manner to other possible anomalies.

The execution times for the extraction of the egored and RW features are about 35 and 30 minutes, respectively. The egonet features are extracted together with the egored ones, so we do not have a specific time for them; the most time consuming task, the extraction of the egonet from the graph, is common. Remind that the random walks are built with equal probabilities of the neighbors, see Section IV-B. An IF run takes at most 40 seconds. OddBall requires about two

**TABLE 4.** TPR AUC\_1% results for several methods on synthetic graphs.

Method	$\mathcal{G}_7$	$\mathcal{G}_{7d}$	$\mathcal{G}_{7r}$	$\mathcal{G}_{12}$	$\mathcal{G}_{12d}$	$\mathcal{G}_{12r}$	$\mathcal{G}_{17}$	$\mathcal{G}_{17d}$	$\mathcal{G}_{17r}$	$\mathcal{G}_{22}$	$\mathcal{G}_{22d}$	$\mathcal{G}_{22r}$
$\mathcal{F}'_{\text{egonet}}$	0.6546	0.6322	0.6568	0.6433	0.6208	0.6233	0.6062	0.5731	0.5813	0.5739	0.5721	0.5703
$\mathcal{F}'_{\text{egored}}$	0.8205	<b>0.8012</b>	0.7989	<b>0.8084</b>	<b>0.7917</b>	<b>0.7886</b>	<b>0.7918</b>	<b>0.7740</b>	0.7649	<b>0.7790</b>	<b>0.7476</b>	<b>0.7495</b>
$\mathcal{F}'_{\text{egonet+RW}}$	0.8011	0.6402	0.6429	0.7880	0.6046	0.5741	0.7662	0.5039	0.5031	0.7640	0.4593	0.4640
OddBall	<b>0.8214</b>	0.7348	<b>0.8049</b>	0.7769	0.7610	0.7571	0.7616	0.7132	<b>0.7670</b>	0.7733	0.7244	0.7424
GAW	0.2759	0.2172	0.2215	0.2454	0.1647	0.1477	0.2531	0.1538	0.1586	0.3062	0.1650	0.1912

**TABLE 5.** TPR AUC results for several methods on synthetic graphs.

Method	$\mathcal{G}_7$	$\mathcal{G}_{7d}$	$\mathcal{G}_{7r}$	$\mathcal{G}_{12}$	$\mathcal{G}_{12d}$	$\mathcal{G}_{12r}$	$\mathcal{G}_{17}$	$\mathcal{G}_{17d}$	$\mathcal{G}_{17r}$	$\mathcal{G}_{22}$	$\mathcal{G}_{22d}$	$\mathcal{G}_{22r}$
$\mathcal{F}'_{\text{egonet}}$	0.9892	0.9880	0.9889	0.9883	0.9853	0.9857	0.9831	0.9804	0.9774	0.9768	0.9716	0.9730
$\mathcal{F}'_{\text{egored}}$	<b>0.9978</b>	<b>0.9975</b>	<b>0.9978</b>	<b>0.9978</b>	<b>0.9974</b>	<b>0.9973</b>	<b>0.9957</b>	<b>0.9961</b>	<b>0.9943</b>	<b>0.9940</b>	<b>0.9925</b>	<b>0.9929</b>
$\mathcal{F}'_{\text{egonet+RW}}$	0.9950	0.9922	0.9923	0.9942	0.9900	0.9892	0.9894	0.9845	0.9793	0.9864	0.9761	0.9778
OddBall	0.9944	0.9799	0.9933	0.9874	0.9807	0.9580	0.9760	0.9450	0.9882	0.9814	0.9656	0.9813
GAW	0.9654	0.9584	0.9582	0.9495	0.9361	0.9317	0.9422	0.9163	0.9167	0.9408	0.9056	0.9178

hours for computing all its statistics. GAW is faster and needs only about 10 minutes.

Besides the methods whose results are given above, we have tried others, notably node2vec [21], with poor results. Corroborating with the OddBall results, we reach the natural conclusion that graph topology information alone, although important, is not sufficient for good performance without the information on the transaction amounts.

## B. RESULTS ON SYNTHETIC DATA

We run now the same methods on the synthetic graphs. We have generated three graphs from each of the 12 categories described in Section III-B: four graph densities (average degree 7, 12, 17, and 22) and three types of cliques (regular, directed, random). We remind that the other anomalies are the Watts-Strogatz ring and the star. Since the results with the directed star anomaly structure are similar to those with the regular star, they will not be reported.

For our methods, the parameters are the same as for the Libra data, with the exceptions listed below. As explained in Section V, all average amounts have been removed from the features. The RW length is  $r = 10$  and, during the walk, the probability of the next edge is proportional with the amount. The IF results are averaged over 5 runs.

For OddBall we report the results of “egonet number of nodes vs egonet number of edges”, which is consistently better than the other statistics.

The TPR AUC results are given in Tables 4 and 5 and consist of averages over the three instances of each graph. The AUC\_1% values from Table 4 show a few trends. The results are usually the best for the lowest graph density and worsen as the density increases; this is natural, since the anomalies have the same size and so the anomalous pattern is more difficult to find in a denser graph. For most methods, the results on the graphs containing directed cliques are the worst, followed closely by those containing random cliques; the difference is small for the egonet and egored features and larger for the other methods. For RW this is explainable by the inexistence of rings in the directed clique and the lower probability of finding rings in a random clique. For OddBall we do not have

an explanation; however, OddBall shows the largest variation of the results between instances of the same graph type, in contrast with our methods, which are quite consistent.

Egored features give the best results in most categories, followed closely by OddBall. Egonet+RW has clearly better results than egonet only for the graphs containing cliques, while for the other graphs the situation is somewhat reversed. GAW has poor results in the AUC\_1% category.

The results for the overall TPR AUC, shown in Table 5, give a slightly changed ranking. While egored is the clear winner, egonet+RW takes the second position, and OddBall falls behind and is beaten even by egonet in several cases. Since for the synthetic graphs the ground truth is totally available, the overall TPR AUC is certainly much more relevant than for the Libra data. So, we can say that the RW approach brings benefits to the egonet one, as its results are always better. Also, although GAW keeps the lowest ranking, its results are overall decent.

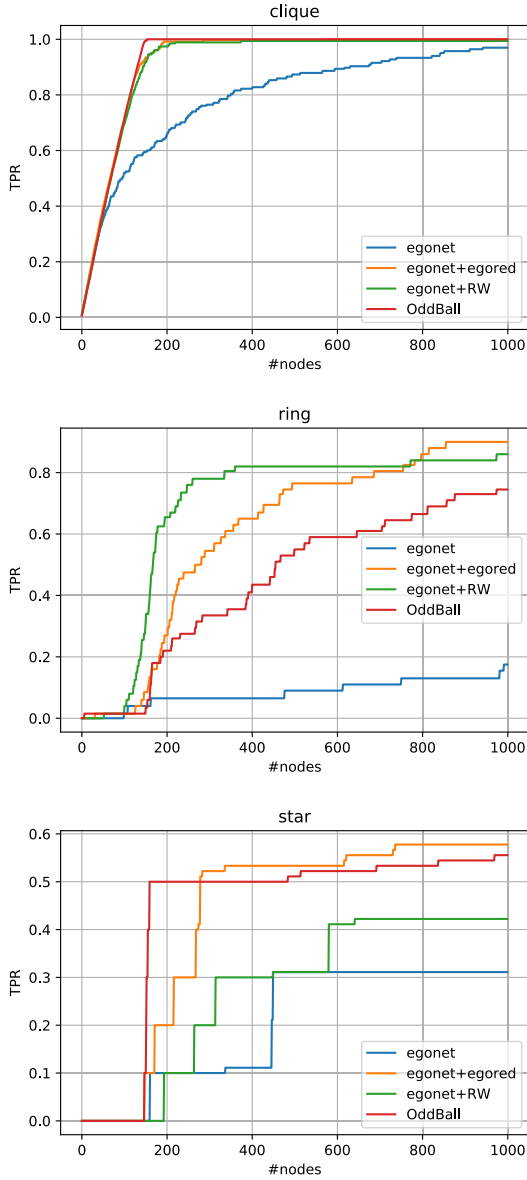
To get more insight on the behavior of the methods, we show in Figures 9–11 the TPR curves on the first 2% of the nodes, separately for each type of anomaly, on the densest graphs that we used. The curves are for a single graph in each category and, for our methods, for a single IF run; however, we chose a run whose results are close to the average.

As expected, OddBall has excellent results for all types of cliques, followed closely by the egored approach. Egonet and egonet+RW curves rise slower, but still reach 1 in the first 2% of the nodes or not much later.

For rings, the situation is different. The RW approach has the best start, detecting earlier than other methods part of the anomalous nodes on rings; this is explained by the RW feature that is able to reveal rings with large transactions. The egored approach detects more anomalies, but slower. OddBall has inconsistent results, often reasonably good (especially in Figure 11) but sometimes quite poor, like in Figure 10. Egonet alone behaves rather poorly on all graphs.

Finally, most methods detect well enough star centers, especially egored and OddBall. Since the star centers are involved in many “fraudulent” transactions, their weight is much larger than the weight of the other nodes in the star;



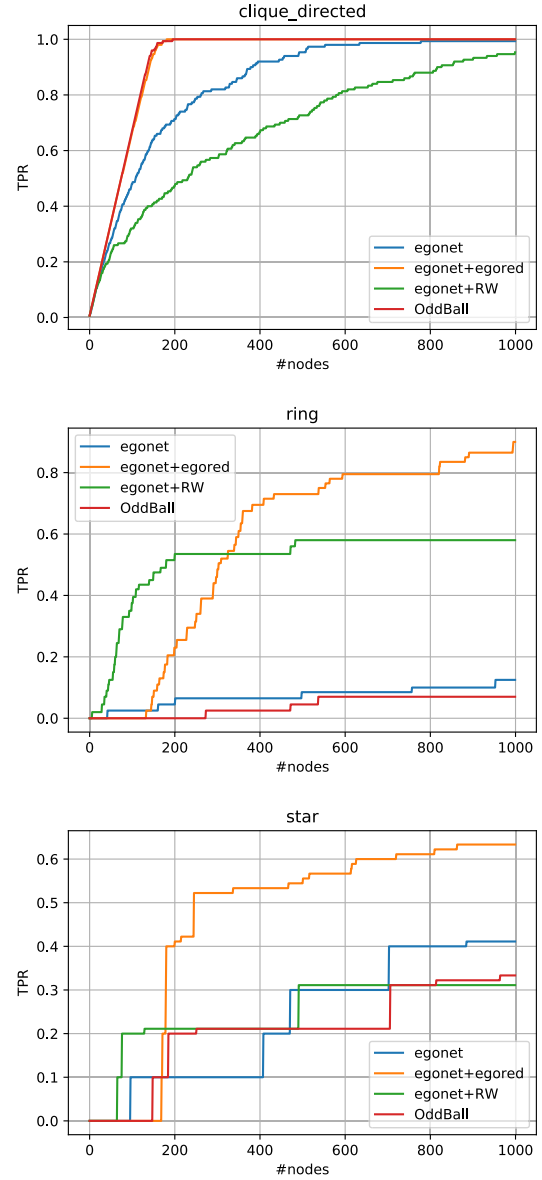


**FIGURE 9.** TPR for on first 1000 anomalies for a graph  $\mathcal{G}_{22}$ . Top: clique. Middle: ring. Bottom: star.

the detection of a star center leads to a large increase of the TPR curve, here precisely with 0.1 (since stars have 10 nodes, the weight of the center is 9, while the other nodes have weight 1; there are five stars and the centers gather half of the weights). The other nodes in the star are actually not easily discoverable by the studied methods; however, once the center is discovered, a targeted analysis of its neighborhood can reveal the star structure; this is beyond our purpose here.

So, the egored approach is nearly best for all types of anomalies, which shows its versatility and explains the overall best position among the studied methods.

The execution times of the methods depend on the density of the graph. The extraction of egored (and egonet) features

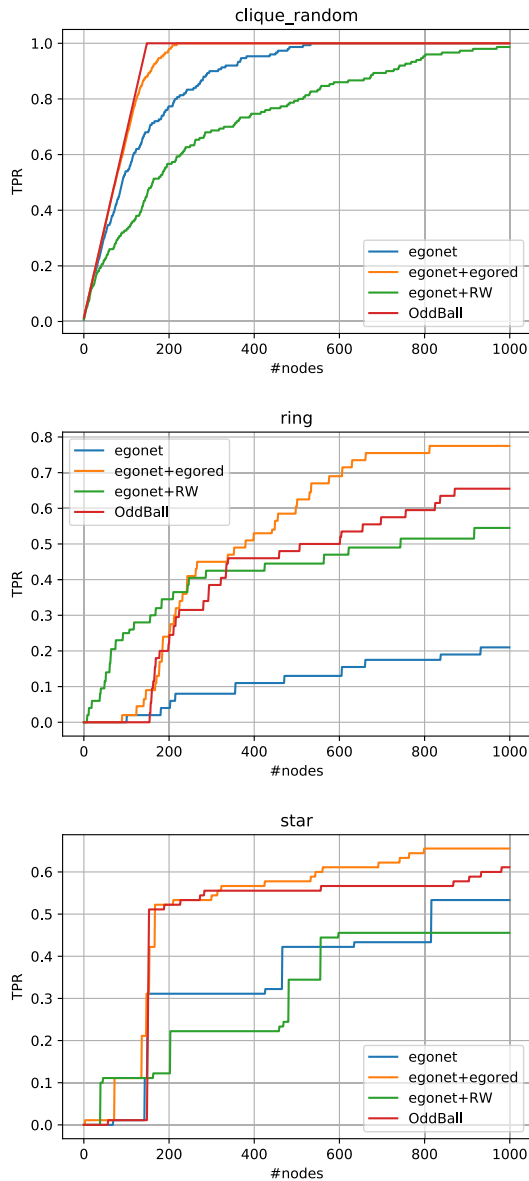


**FIGURE 10.** TPR for on first 1000 anomalies for a graph  $\mathcal{G}_{22d}$ . Top: clique directed. Middle: ring. Bottom: star.

takes from 38 seconds for  $\mathcal{G}_7$  to 140 seconds for  $\mathcal{G}_{22}$ . RW features need between 55 and 80 minutes; the increased time with respect to Libra data is explained by the larger RW length and the proportional probability computation. An IF run on the extracted features takes less than 5 seconds; here, the number of nodes is important, hence the smaller time compared with Libra data. OddBall needs between 250 and 575 seconds for computing all its statistics. Finally, GAW is again the fastest, with times between 44 and 58 seconds.

### C. DISCUSSION

We examine now the results of the proposed methods on both datasets, real and synthetic. The egored approach behaves very well in the early detection of a good amount of anomalies



**FIGURE 11.** TPR for on first 1000 anomalies for a graph  $\mathcal{G}_{22r}$ . Top: clique random. Middle: ring. Bottom: star.

(alerts and reports) from the Libra dataset. It also quickly detects most of the artificial anomalies from the synthetic graphs and has the best overall TPR AUC performance. So, we can say that putting together the egored and egonet features gives a robust graph AD method, covering well real and simulated situations.

The random walk approach is also promising. For Libra data, the addition of the RW feature to the egonet ones improves anomaly detection in the first 0.1% nodes. For synthetic data, it improves the overall behavior. It is also a method very suited for online implementation, as individual random walks can be quickly generated as the graph evolves and feature values can be permanently updated. In contrast, egonet and egored features are a coarser computation task.

Among the other approaches, egonet features give also balanced results on the real and synthetic data, but weaker than those of the proposed methods. OddBall discovers very well cliques and star centers, but has a weak performance on the Libra data. On the contrary, GAW has fairly good behavior on the Libra data (although below our methods), but is poor on the synthetic data.

## VII. CONCLUSION AND FURTHER WORK

We have introduced a method for anomaly detection in bank transactions, with the purpose of detecting money laundering activities. Our method can be summarized as follows:

- 1) Given a list of transactions, build the transaction graph, where nodes are accounts and edges have two attributes: the cumulated amount transferred from source to destination and the number of transactions.
- 2) Compute node features. Here is our main contribution. Most of these features are derived from the notion of reduced egonets (egoreds) and consist of in- and outdegrees, total and average amounts sent/received by a node, and edge density. These features are used together with the corresponding (standard) egonet features. We have also proposed new random walk features.
- 3) Run an anomaly detection algorithm, Isolation Forest in our case, on the feature values, to detect abnormal nodes (accounts).

We have obtained very good results with this method. Many of the labeled anomalies are recovered early in the Libra (real) dataset and also the behavior on synthetic graphs is excellent. So, our method appears to be adequate and robust.

Moreover, comparisons with some of the relevant existing methods, like OddBall or statistical approaches (GAW), are favorable; better true positive rate is obtained with comparable execution time. We have also shown that the addition of the newly proposed egored or random walk features to existing egonet features always improves the results.

We conclude that our method can bring clear benefits to AML activities, from the specific angle of the graph structure of the bank transactions. Certainly, other types of features can be added with the purpose of getting a more comprehensive approach.

Further work will be dedicated to the development of online algorithms and to validation on larger datasets, ideally on real datasets covering the transactions of several banks.

## ACKNOWLEDGMENT

The authors thank Libra Internet Bank for sharing anonymized transaction data for analysis and are grateful to their AML specialists for helpful comments and information.

## REFERENCES

- [1] M. Levi, P. Reuter, and T. Halliday, "Can the AML system be evaluated without better data?" *Crime, Law Social Change*, vol. 69, no. 2, pp. 307–328, Mar. 2018.

- [2] L. Akoglu, M. McGlohon, and C. Faloutsos, "oddball: Spotting anomalies in weighted graphs," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*. Berlin, Germany: Springer, 2010, pp. 410–421.
- [3] I. Molloy, S. Chari, U. Finkler, M. Wiggerman, C. Jonker, T. Habeck, Y. Park, F. Jordens, and R. van Schaik, "Graph analytics for real-time scoring of cross-channel transactional fraud," in *Proc. Int. Conf. Financial Cryptography Data Secur.* Berlin, Germany: Springer, 2016, pp. 22–40.
- [4] A. Elliott, M. Cucuringu, M. M. Luaces, P. Reidy, and G. Reinert, "Anomaly detection in networks with application to financial transaction networks," 2019, *arXiv:1901.00402*.
- [5] A. E. Wegner, L. Ospina-Forero, R. E. Gaunt, C. M. Deane, and G. Reinert, "Identifying networks with common organizational principles," *J. Complex Netw.*, vol. 6, no. 6, pp. 887–913, 2018.
- [6] Z. Li, H. Xiong, Y. Liu, and A. Zhou, "Detecting blackhole and volcano patterns in directed networks," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 294–303.
- [7] X. Li, S. Liu, Z. Li, X. Han, C. Shi, B. Hooi, H. Huang, and X. Cheng, "FlowScope: Spotting money laundering based on graphs," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 4731–4738.
- [8] *Money Laundering in Australia 2011*, Austral. Trans. Rep. Anal. Centre, National Circuit, ACT, Australia, 2011.
- [9] X. Sun, J. Zhang, Q. Zhao, S. Liu, J. Chen, R. Zhuang, H. Shen, and X. Cheng, "CubeFlow: Money laundering detection with coupled tensors," Mar. 2021, *arXiv:2103.12411*.
- [10] X. Qiu, W. Cen, Z. Qian, Y. Peng, Y. Zhang, X. Lin, and J. Zhou, "Real-time constrained cycle detection in large dynamic graphs," *Proc. VLDB Endowment*, vol. 11, no. 12, pp. 1876–1888, 2018.
- [11] S. Khuller and B. Saha, "On finding dense subgraphs," in *Proc. Int. Colloq. automata, Lang., Program.*, Rhodes, Greece, 2009, pp. 597–608.
- [12] S. Zhang, D. Zhou, M. Y. Yildirim, S. Alcorn, J. He, H. Davulcu, and H. Tong, "HiDDen: Hierarchical dense subgraph detection with application to financial fraud detection," in *Proc. SIAM Int. Conf. Data Mining*, 2017, pp. 570–578.
- [13] B. A. Miller, M. S. Beard, P. J. Wolfe, and N. T. Bliss, "A spectral framework for anomalous subgraph detection," *IEEE Trans. Signal Process.*, vol. 63, no. 16, pp. 4191–4206, May 2015.
- [14] D. Huang, D. Mu, L. Yang, and X. Cai, "CoDetect: Financial fraud detection with anomaly feature detection," *IEEE Access*, vol. 6, pp. 19161–19174, 2018.
- [15] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "FRAUDAR: Bounding graph fraud in the face of camouflage," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 895–904.
- [16] J. Zhang, S. Liu, W. Yu, W. Feng, and X. Cheng, "EigenPulse: Detecting surges in large streaming graphs with row augmentation," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2019, pp. 501–513.
- [17] B. Perozzi and L. Akoglu, "Scalable anomaly ranking of attributed neighborhoods," in *Proc. SIAM Int. Conf. Data Mining*, 2016, pp. 207–215.
- [18] W. Eberle, J. Graves, and L. Holder, "Insider threat detection using a graph-based approach," *J. Appl. Secur. Res.*, vol. 6, no. 1, pp. 32–81, 2011.
- [19] D. Savage, Q. Wang, P. Chou, X. Zhang, and X. Yu, "Detection of money laundering groups using supervised learning in networks," 2016, *arXiv:1608.00708*.
- [20] X. Sun, W. Feng, S. Liu, Y. Xie, S. Bhatia, B. Hooi, W. Wang, and X. Cheng, "MonLAD: Money laundering agents detection in transaction streams," in *Proc. 15th ACM Int. Conf. Web Search Data Mining (WSDM)*, New York, NY, USA, 2022, pp. 976–986.
- [21] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 855–864.
- [22] R. Hu, C. C. Aggarwal, S. Ma, and J. Huai, "An embedding approach to anomaly detection," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 385–396.
- [23] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1225–1234.
- [24] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "NetWalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 2672–2681.
- [25] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proc. SIAM Int. Conf. Data Mining*, 2019, pp. 594–602.
- [26] S. Bandyopadhyay, S. V. Vivek, and M. N. Murty, "Outlier resistant unsupervised deep architectures for attributed network embedding," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 25–33.
- [27] Y. Zheng, M. Jin, Y. Liu, L. Chi, K. T. Phan, and Y. P. P. Chen, "Generative and contrastive self-supervised learning for graph anomaly detection," *IEEE Trans. Knowl. Data Eng.*, early access, Oct. 12, 2021, doi: [10.1109/TKDE.2021.3119326](https://doi.org/10.1109/TKDE.2021.3119326).
- [28] A. M. Mubalalike and E. Adali, "Deep learning approach for intelligent financial fraud detection system," in *Proc. 3rd Int. Conf. Comput. Sci. Eng. (UBMK)*, Sep. 2018, pp. 598–603.
- [29] A. F. Colladon and E. Remondi, "Using social network analysis to prevent money laundering," *Expert Syst. Appl.*, vol. 67, pp. 49–58, Jan. 2017.
- [30] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," *Data Mining Knowl. Discovery*, vol. 29, no. 3, pp. 626–688, 2015.
- [31] Z. Chen, "Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: A review," *Knowl. Inf. Syst.*, vol. 57, no. 2, pp. 245–285, 2018.
- [32] P. Irofti, A. Pătrașcu, and A. Băltioiu, "Fraud detection in networks," in *Enabling AI Applications in Data Science*. Cham, Switzerland: Springer, 2021, pp. 517–536.
- [33] T. Pourhabibi, K. L. Ong, B. H. Kam, and Y. L. Boo, "Fraud detection: A systematic literature review of graph-based anomaly detection approaches," *Decis. Support Syst.*, vol. 133, Jun. 2020, Art. no. 113303.
- [34] W. Hilal, S. A. Gadsden, and J. Yawney, "Financial fraud: A review of anomaly detection techniques and recent advances," *Expert Syst. Appl.*, vol. 193, May 2022, Art. no. 116429.
- [35] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Trans. Knowl. Data Eng.*, early access, Oct. 8, 2021, doi: [10.1109/TKDE.2021.3118815](https://doi.org/10.1109/TKDE.2021.3118815).
- [36] G. Fernandes, J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommun. Syst.*, vol. 70, no. 3, pp. 447–489, 2019.
- [37] N. Mustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 128, pp. 33–55, Feb. 2019.
- [38] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos, "Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 647–657.
- [39] Y. Zhao, Z. Nasrullah, and Z. Li, "PyOD: A Python toolbox for scalable outlier detection," *J. Mach. Learn. Res.*, vol. 20, no. 96, pp. 1–7, Jan. 2019.
- [40] (2018). *Credit Card Fraud Detection*. [Online]. Available: <https://www.kaggle.com/mlg-ulb/creditcardfraud>
- [41] (2019). *IEEE-CIS Fraud Detection Competition*. [Online]. Available: <https://www.kaggle.com/c/ieee-fraud-detection/>
- [42] M. Weber, G. Domeniconi, J. Chen, D. Karl I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson, "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics," 2019, *arXiv:1908.02591*.
- [43] E. Lopez-Rojas, A. Elmir, and S. Axelsson, "PaySim: A financial mobile money simulator for fraud detection," in *Proc. 28th Eur. Model. Simulation Symp. (EMSS)*, Larnaca, Cyprus, 2016, pp. 249–255.
- [44] E. Lopez-Rojas and S. Axelsson, "BankSim: A bank payments simulator for fraud detection research," in *Proc. 26th Eur. Modeling Simulation Symp.*, 2014, pp. 144–152.
- [45] *Banksim*. Accessed: Apr. 28, 2022. [Online]. Available: <https://www.kaggle.com/ealaxi/banksim1>
- [46] *Synthesized Fraud Detection*. Accessed: Apr. 28, 2022. [Online]. Available: <https://www.synthesized.io/data-template-pages/fraud-detection>
- [47] *Uncovering Hidden Financial Crime Through Advanced Simulation*. Accessed: Apr. 28, 2022. [Online]. Available: <https://simudyne.com/wp-content/uploads/2019/10/Uncovering-Hidden-FinCrime-Whitepaper.pdf>
- [48] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
- [49] H. D. K. Moonesinghe and P.-N. Tan, "OutRank: A graph-based outlier detection framework using random walk," *Int. J. Artif. Intell. Tools*, vol. 17, no. 1, pp. 19–36, 2008.
- [50] C. Wang, H. Gao, Z. Liu, and Y. Fu, "A new outlier detection model using random walk on local information graph," *IEEE Access*, vol. 6, pp. 75531–75544, 2018.
- [51] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 413–422.



**BOGDAN DUMITRESCU** (Member, IEEE) was born in Bucharest, Romania, in 1962. He received the M.S. and Ph.D. degrees from the University Politehnica of Bucharest, Romania, in 1987 and 1993, respectively. He held several visiting research positions at the Tampere University of Technology, Finland, in particular that of FiDiPro Fellow (2010–2013). He is currently a Professor with the Department of Automatic Control and Computers, University Politehnica of Bucharest, where he has been working, since 1990. He authored the book *Positive Trigonometric Polynomials and Signal Processing Applications* and coauthored *Dictionary Learning Algorithms and Applications*. His research interests include optimization, numerical methods, and their applications to signal processing, especially using sparse representations. He was an Associate Editor (2008–2012) and an Area Editor (2010–2014) of the IEEE TRANSACTIONS ON SIGNAL PROCESSING.



**ANDRA BĂLTOIU** was born in 1985. She received the B.Sc., M.Sc., and Ph.D. degrees from the Department of Automatic Control and Computers, University Politehnica of Bucharest. Previously, she worked as a Research Scientist at the Institute of Space Science and the National Institute for Sport Research, with both positions involving data analysis and signal processing. Starting with 2018, she was a Research Assistant at the Research Institute of University of Bucharest. She is currently an Assistant Professor at the Department of Automatic Control and Computers, University Politehnica of Bucharest. She is also working on anomaly detection.



**ȘTEFANIA BUDULAN** received the M.Sc. degree in AI from the University Politehnica of Bucharest. She is an AI Software Engineer with more than six years of hands-on experience, developing AI-driven solutions for industries, such as financial services, media, telekom and the IoT, as well as a fundamental research advocate, especially in the field of natural language processing (NLP), as a Co-ordinator of the AI research development of several BSc. and MSc. students at the University Politehnica of Bucharest. She has a strong interest towards creating viable state-of-the-art solutions for industrial use, thus reducing the gap between academic research and the industry reach and potential. Being an AI Researcher and a Software Engineer, an AI Ethics, Policy and Governance Promoter, she thoroughly believes that data protection policies and model understanding play a key role in order to build safe AI products, escape human errors and data biases, increase AI explainability, and diminish the amount of misleading information.

...