Detecting Complex Money Laundering Patterns with Incremental and Distributed Graph Modeling

Haseeb Tariq
ING Bank
Amsterdam, The Netherlands
Eindhoven University of Technology
Eindhoven, The Netherlands
m.h.tariq@tue.nl

Alen Kaja
Identity & Risk Intelligence
Adyen
Amsterdam, The Netherlands
alen.kaja@adyen.com

Marwan Hassani
Mathematics & Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
m.hassani@tue.nl

***Abstract*—Money launderers take advantage of limitations in existing detection approaches by hiding their financial footprints in a deceitful manner. They manage this by replicating transaction patterns that the monitoring systems cannot easily distinguish. As a result, criminally gained assets are pushed into legitimate financial channels without drawing attention. Algorithms developed to monitor money flows often struggle with scale and complexity. The difficulty of identifying such activities is further intensified by the (persistent) inability of current solutions to control the excessive number of false positive signals produced by rigid, risk-based rules systems. We propose a framework called `ReDiRect` (Reduce, Distribute, and Rectify), specifically designed to overcome these challenges. The primary contribution of our work is a novel framing of this problem in an unsupervised setting; where a large transaction graph is *fuzzily* partitioned into smaller, manageable components to enable fast processing in a distributed manner. In addition, we define a refined evaluation metric that better captures the effectiveness of exposed money laundering patterns. Through comprehensive experimentation, we demonstrate that our framework achieves superior performance compared to existing and state-of-the-art techniques, particularly in terms of efficiency and real-world applicability. For validation, we used the *real* (open source) Libra dataset and the recently released synthetic datasets by IBM Watson. Our code and datasets are available at https://github.com/mhaseebtariq/redirect.**

***Index Terms*—Fuzzy communities, graph modeling, anti-money laundering, anomaly detection.**

I. Introduction

The United Nations Office on Drugs and Crime (UNODC) has estimated that approximately 2 to 5% of global GDP is laundered annually, which translates to between 715 billion and 1.87 trillion Euros each year [33]. Quantifying the toll of human suffering is even more complex. Current financial monitoring systems have struggled to detect these laundering activities effectively, leading to significant penalties being imposed on major banks by the regulators. A key issue for banks is that they can only observe transactions within their own systems, while criminals typically move funds across multiple banks in different jurisdictions to conceal the complete trail of money.

The money laundering process is typically divided into three phases: Placement, Layering, and Integration [33]. In the placement phase, criminals introduce illicit funds into the financial system, often by depositing them in a manner that does not raise suspicion. In the subsequent layering phase, they convolute the money trail by dispersing funds across various intermediaries. Finally, in the integration phase, the money is withdrawn from accounts that appear to be operating under legitimate conditions. The methods criminals employ in these phases are referred to as *typologies*. There are numerous well-documented typologies, such as *smurfing* (or *structuring*) [1], where large sums of illicit funds are broken into smaller and less noticeable deposits in several *mule accounts* [25].

Criminals continuously develop new techniques (or typologies) to circumvent financial monitoring systems. This constant innovation makes it challenging for *Anti-Money Laundering (AML)* experts to keep up with emerging risks. Financial institutions use a combination of *Transaction Monitoring (TM)* systems and *Know Your Customer (KYC)* protocols to prevent money laundering and terrorist financing. They also implement frameworks such *as Customer Due Diligence (CDD)* and *Ultimate Beneficial Owner (UBO)* identification, in addition to rules-based frameworks, to identify high-risk transactions. However, a significant proportion of money laundering activities remain undetected because experts spend an overwhelming amount of time sifting through and discarding a massive number of *false positive alerts*. In practice, this represents the most substantial challenge for financial institutions. Banks spend tens of millions of Euros annually on processing these alerts. Otherwise, due to lack of regulatory compliance, fines can amount to hundreds of millions of Euros [29]. Reports on *AML compliance costs* [24] provide staggering figures on the monetary burden faced by financial institutions.

Our proposed framework addresses these challenges by focusing on building more concise, fuzzy, and overlapping communities with the purpose of reducing the cognitive load for AML analysts. Consequently, (also) minimizing the *false positive rate*. Our key contributions are as follows:

- A novel and scalable way to formulate the AML modeling problem
- A data scope reduction technique that, in addition to scalability, also improves the model accuracy; and usefulness by *drastically* reducing the ILT (Investigation Lead Time) of the (outputted) alerts

- A novel metric to measure the holistic context of AML anomalies

Section II discusses existing work. Sections III & IV introduce the datasets, definitions, and ReDiRect. Section V details the experimental evaluation; and Section VI concludes the paper.

II. Related Work

AML modeling and fraud detection have several similarities; and face some common challenges. However, in the context of our research, we want to distinguish the two problem spaces. The focus of our research is strictly on detecting complex money laundering networks. More precisely, we want to identify how the anomalous behaviors of a money launderer are manifested as their transaction activities. We will briefly review some of the previous research on the detection of money laundering networks (and patterns) backed by graph algorithms.

To our knowledge, with the exception of FaSTM∀N [32], there is no existing work that focuses on *first* detecting communities (built on a transaction graph) with the purpose of identifying suspicious money laundering patterns in them. For example, FlowScope [17] and CubeFlow [30] *directly* identify suspicious flows in a global transaction graph; Graphomaly [19] and MonLad [20] identify suspicious *nodes* by constructing graph features; and GraphFeatureProcessor [23] introduces a library to construct graph features with the purpose of identifying anomalous *edges* (or transactions). In ExSTraQt [31], a similar but much more effective framework is proposed.

The modeling of transaction data as graphs is a popular choice [27] for building AML models. [15] presents a comparative study on some of the well-known community detection methods. The Louvain algorithm [4] is one of the most widely used modularity-based community detection methods [3]. An advanced version, called the Leiden algorithm, has been proposed in [16]. Modularity-based algorithms produce non-overlapping communities. This means that a node can belong only to a single community. In the real world, this is not really the case for many problems. A node (or a business entity) can belong to several communities. Similarly, a money laundering agent can also support multiple criminal networks. That is why our framework also employs a bottom-up approach. Where, for each node, we build a community around it using the *personalized* pagerank algorithm [2, 8]. This results in fuzzy communities that overlap. The study in [14] assesses various algorithms by analyzing the distributions of (resulting) community sizes. Anomalous communities eventually have to be analyzed by AML analysts. As the size of a community becomes too large, it becomes unrealistic to analyze it by *observing* signs of suspicious patterns. One of our *core* contributions focuses on reducing the size of the alerted (or outputted) communities.

A detailed review published in [9] examines anomaly detection techniques for data represented as graphs. Most community detection methods are designed to identify communities with strong internal connections, while having weak external links to the *other* communities. In transaction datasets (with low degrees of separation), this typically results in communities with short paths, often with just 1 or 2 hops. Consequently, simpler methods such as those in [19] or [6] are more effective in detecting anomalous communities with closely connected neighbors. Due to stringent oversight by data and financial regulators, AML models must be free of bias, restricting the number of usable data points. This includes, for example, avoiding gender, age, racial, socioeconomic, or geographic biases. Modelers must also be cautious of the *guilt by association* fallacy. For example, the method proposed in [12] risks falling into this trap. Data sharing among banks remains a risk, as privacy enhancing technologies such as multiparty computation and homomorphic encryption [21] are not mature enough to be used on a large scale; or for more practical applications. This pushes the need for improving the state-of-the-art for single-bank AML modeling.

Lastly, our research focuses on detecting the anomalous *context* in which money laundering occurs. We use IsolationForest [28] to detect anomalous nodes (or accounts). The detection method can be easily swapped with any graph-based neural network [5] such as GraphSAGE [13] or VGAE [10]. Therefore, we do not care much about the specific anomaly detection algorithm that is being used. More importantly, in a highly regulated industry like financial services, model interpretability is of the utmost importance. The output of the *tabular* feature-based model we train in our framework can be easily explained by either 1) estimating a simple (linear) model on top of the predicted binary label (anomalous/normal); or 2) using a model interpretability framework like SHAP [18] or LIME [11]. There are no straightforward methods for using the same interpretability frameworks on (advanced) neural network architectures designed for graphical data.

III. Datasets

Despite having access to real transaction data from some of the largest European banks, for this paper we will rely on open source datasets. This is because of the obvious risk implications that come with publishing results from real data, containing real money laundering cases. We built our framework using the *real* Libra Internet Bank dataset, which we will call $\mathcal{D}_{lib}^{real}$. The content of this dataset is described in [19].

We also used a synthetic dataset [22] published by the IBM Watson research lab in 2023 as a *secondary* source. We are going to call it $\mathcal{D}_{ibm}^{syn}$. The content of this dataset is elaborately explained in [22]. There are 6 different versions of the dataset, with 2 categories. The first category represents the size of the dataset as low (1 month), medium (3 months), and large (6 months). The second category represents the ratio of anomalies (or illicit transactions) present in the data, as low-illicit and high-illicit. The dataset is injected with a rich set of labels on a transaction level; and on a (broader) *pattern* level. The injected patterns can actually be seen as the anomalous context that we want to identify. The concept of *context* is extensively explained in Section V-A1, with the support of Figure 2 and

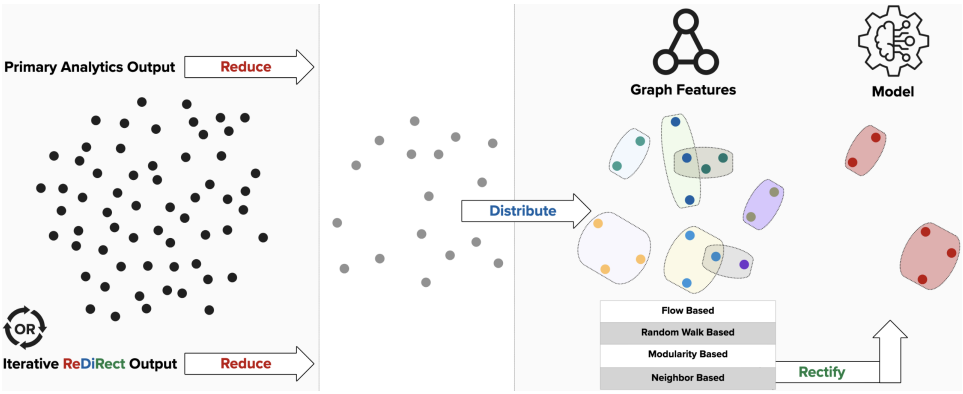

Fig. 1: Conceptualization of the problem formulation for ReDiRect. Left side represents the actual data space; in the next part, nodes are filtered out, either by the *primary* detection system; or by the output of an earlier ReDiRect run. Finally, the fuzzy communities (per node) are labeled as suspicious by an unsupervised machine learning model.

Algorithm 2. In our work, we focus mainly on large versions of the dataset. By doing so, we also cover the scalability aspects of our framework.

IV. Our Method: ReDiRect

We introduce a novel way of *formulating* the AML problem as represented in Fig. 1. In essence, we want to break down the problem into 3 parts, reduction of the search space (or Reduce); distributed computation of complex graphs-based features (or Distribute); and (finally) identification and optimal *filtering* of the money laundering patterns (or Rectify). We will now go through each part in detail.

A. Reduce

The advantages of reducing the data scope are three-fold, which we will verify in Section V (Experiments), (i) improved model accuracy, (ii) reduction in ILT (Investigation Lead Time)[1], and, (iii) efficient generation of complex graph features.

The initial data space can be effectively reduced by applying simple risk-based AML rules. In practice, transaction monitoring systems in most financial institutions rely on extensive collections of such heuristics. For illustrative purposes, one might consider the following examples: flagging cash deposits exceeding a certain threshold; identifying accounts that perform a high number of rounded-amount deposits; or monitoring trading entities that send multiple transactions to high-risk jurisdictions within a short time window. However, incorporating such rules directly into a research context presents significant challenges. First, disclosing specific rule definitions could inadvertently inform potential money launderers, enabling them to circumvent detection by adapting their behavior. Second, the implementation of these rules in real banking environments typically relies on a broader and more detailed set of data attributes than those available in the synthetic datasets used for research purposes.

[1]We define ILT as the mean time required by an AML analyst to close an alerted case of money laundering

*1) **RM-1** (Reduction Method 1):* For $\mathcal{D}_{ibm}^{syn}$, we instead reverse the problem by: first identifying the *normal* entities, rather than identifying the suspicious entities using risk-based rules. We do so by using some common sense *sequential* filtering, based on AML knowledge, which includes: 1) removing transactions with certain transaction types that can rarely be used in money laundering activities, 2) removing transactions for accounts with a large number of transactions in a given period, and, 3) removing transactions for accounts that interact with a large number of other accounts. The last two filters are indicators for legitimate businesses. We want to reiterate that this is just a work-around. In a real-world setting, risk-based rules provide a much more practical approach with a much greater reduction of data space.

*2) **RM-2**:* For $\mathcal{D}_{lib}^{real}$, we employ more dynamic reduction methods (RM-2 and RM-3). For this method, we take the top X% anomalous nodes produced by the first run of ReDiRect as the input data scope for the next run of ReDiRect. The idea here is that, after removing the most *normal* nodes from the input data, we enable the model to identify the real anomalous nodes more effectively. The value for top X% could be based on the risk appetite of the AML experts.

*3) **RM-3**:* This is kind of an extension of RM-2. For this method, we reduce the input data scope *recursively* using the output of the previous ReDiRect run, as detailed in Algorithm 1.

B. Distribute

From this point on, we *assume* that the transactions produced by the last step are all alerts generated by the *primary* (risk-based) transaction monitoring system of a bank. Our goal is now to reduce the huge volume of false positive alerts.

1) Graph Generation: From the datasets $\mathcal{D}_{lib}^{real}$ and $\mathcal{D}_{ibm}^{syn}$, the respective directed graphs can be extracted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Where, $\mathcal{V}$ represents the bank accounts in the form of nodes (or vertices); and $\mathcal{E}$ represents the transactions in the form of edges. An aggregated (and unique) edge between $s, t \in \mathcal{V}$ is represented as $(s \rightarrow t) \in \mathcal{E}$. Where s is the source account that sends the amount and t is the target account that

Algorithm 1 *Recursive* (data scope) Reduction

initial_size ← SIZE(accounts)
reduce_by ← 50% (should be adjusted accordingly)
break_threshold ← 12% (break if data scope < this %)
left_percentage ← 100%
while true **do**
  if $left\_percentage$ < $break\_threshold$ **then**
    break
  end if
  anomalous ← `ReDiRect`(accounts)
  accounts ← TOP($reduce\_by$ % $anomalous$)
  left_percentage ← SIZE($accounts$) / $initial\_size$
end while

receives the amount. The total amount transferred from s to t is represented as an edge attribute, $\mathcal{A}_{s,t}$. The two node attributes, $\mathcal{S}$ and $\mathcal{R}$, represent the total amount sent and the total amount received by a node, respectively. The weight $\mathcal{W}$ for an edge is calculated as follows:

$$\mathcal{W}(s \rightarrow t) = \frac{\mathcal{A}_{s \rightarrow t}}{\mathcal{S}_s} + \frac{\mathcal{A}_{s \rightarrow t}}{\mathcal{R}_t} \tag{1}$$

This weight serves as the main driver for constructing the communities described in the next section. The addition of the two ratios, one from the point of view of the sender and the other from the point of view of the receiver, ensures that a money launderer can not easily manipulate the system. For example, agents in a network can avoid appearing together (in a community) by using a high-volume intermediary account. In such a situation, only one of the two terms in Equation 1 can be minimized, not both.

2) Community Detection: We employ the following methods to construct communities of nodes with close relationships:

- (**CD-M**) Modularity-based community detection using the Leiden algorithm [16]
- (**CD-RW**) Random walk-based subgraph detection[2] using the Personalized Page Rank algorithm [2]

CD-RW also serves as the *context* for each alerted output. Using this context, an AML expert will assert the validity of an alert. The communities (or contexts) produced by this method have fuzzy boundaries. In other words, a node can belong to multiple communities. This aligns with the real-world scenario, where a money launderer can support multiple money laundering rings. For our main competing benchmark, [19], the same context is provided by the *reduced ego-net* of each alerted account. The parameter θ (minimum allowed page rank) serves as the *fuzziness* factor. It drives what constitutes a close relationship with respect to the *central* node for which we are constructing a community. Consequently, it also limits the size of the resulting communities. The higher its value; the higher the overlap in communities. The value of θ can be adjusted according to risk appetite. This is also the parameter that drives the reduction of ILT.

3) Feature Engineering: Next, we generate features per **CD-M**; and per **CD-RW** community, in a distributed manner. The following are the main categories of features:

- **Simple aggregations**: number of accounts and source-/target-only accounts; number of transactions; number of currencies; timestamp range and standard deviation.
- **Network metrics**: degree assortativity; diameter; maximum degree (in/out/all).
- **Turnover related**: total funds turnover (USD); currency-wise percentage contributions

In addition to these, we also generate the following two sets of node-level features:

- **Flow-based**: funds transferred through the central node (up to *n*-hops), depending on its role as a dispense, pass-through, or sink account
- **Neighbor-based**: aggregations and turnover-related features computed for incoming and outgoing neighbors

C. Rectify

In this part, we rectify the false positives in the output of the last `ReDiRect` run; or in the output of the *primary* detection system, by correctly identifying the communities as anomalous. We train an IsolationForest model [28] using the exhaustive set of features constructed in the *Distribute* step. The anomaly score produced by IsolationForest can be used in combination with several risk-based rules to prioritize anomalies. Although we do not employ those rules here, again because of the risk implications.

If there are good quality labels available from the past, the training can also be done in a supervised learning setting. Or in a semi-supervised setting, where a combination of labeled and unlabeled data can be used to improve the quality of the predicted anomalies.

Optimal prioritization and filtering of anomalies is as crucial as identifying the correct anomalies. For the $\mathcal{D}_{ibm}^{syn}$ dataset, we employ a filtering method in which we iterate over the communities in descending order of their anomalousness. We keep a list of community members for each traversed community and remove those members from subsequent communities to ensure that an AML expert does not have to analyze the same *context* over and over again.

V. Experimental Evaluation

In this section, we will demonstrate the superiority of `ReDiRect` in terms of usefulness and scalability. The fully reproducible code for `ReDiRect`, as well as the experimental evaluation, is available here [3]. All experiments were performed using a machine with the following specifications, chip: Apple M3 Pro; number of cores: 12; memory: 36 GB.

The (lack of) availability and the subjectivity that goes into money laundering labels make the problem space almost infeasible for supervised learning. We therefore build our

[2]The distributed version of the algorithm can be found in the submitted code.

[3]https://github.com/mhaseebtariq/redirect

framework in an unsupervised learning setting. In addition to that, we are designing our model in an entirely novel way. No existing work focuses on detecting the anomalous *context* of a money laundering network; therefore, we cannot accurately compare the *contextual* evaluation metric that we introduce in our experiments. The main focus of our evaluation will be on the accuracy (of the model); computational efficiency; and (reduction in) investigation lead times, compared to the benchmark methods. When evaluating our method against the $\mathcal{D}_{lib}^{real}$ dataset, we use the exact same metrics (TPR and TPR-AUC) and criteria as used in [19]. In fact, Table V in Section V-B is an extension of Table 2 in [19]. The thresholds used in [19] to select the top anomalies are listed in Table I. We also used the exact thresholds for reporting the TPR's (True Positive Rates) for the $\mathcal{D}_{lib}^{real}$ dataset; and extrapolated them to define the thresholds for the $\mathcal{D}_{ibm}^{syn}$ dataset in Table II.

TABLE I: Thresholds used in [19] to select the top anomalies. The top-x% (predicted anomalies) is calculated from 385,100 accounts.

Anomalies	Count	Threshold	Count / Actual (Threshold Factor)
Actual	**600**	–	**1.00**
Top 0.1%	385	T1	0.64
Top 0.2%	770	T2	1.28
Top 0.5%	1926	T3	3.21
Top 1%	3851	T4	6.42

TABLE II: Thresholds used in [19] for $\mathcal{D}_{ibm}^{syn}$ *large* datasets. LI = Low-Illicit; HI = High-Illicit.

Dataset	Total	T1 (0.64)	T2 (1.28)	T3 (3.21)	T4 (6.42)
$\mathcal{D}_{lib}^{real}$	600	385	770	1926	3851
$\mathcal{D}_{ibm}^{syn}$ LI	1360	873	1745	4366	8729
$\mathcal{D}_{ibm}^{syn}$ HI	9664	6201	12402	31021	62027

A. Experimental Setting

The very definition of an anomaly could be subjective at best, incomplete, or outright wrong at worst. Whether the problem is modeled in a supervised, semi-supervised, self-supervised, or completely unsupervised setting, the starting point is always a set of assumptions. Assumptions are not facts and, therefore, are subjective. How can (then) those *subjective* assumptions be *objectively* evaluated? Identifying suspicious transactions, by isolating a suspiciously unique set of customer behaviors, is a rational design decision. Although not all suspicious transactions would be the result of those behaviors; neither all such behaviors would be indications of money laundering. Therefore, compared to other modeling problems, we argue that *properly* validating anomalies is relatively more crucial in the context of AML modeling.

1) Evaluation Metric: Let $\mathcal{F}^r$ represent a real money laundering flow and $\mathcal{F}^d$, a flow detected as anomalous by `ReDiRect`. In the context of $\mathcal{D}_{ibm}^{syn}$, $\mathcal{F}^r$ are the injected patterns. Fig. 2 shows the simplified version of our proposed metric, applied to an example alerted flow.

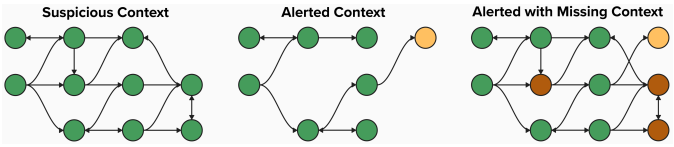

Fig. 2: Example alerted flow: The yellow are extra (or false-positive); and the red are missing (or false-negative) nodes. The contextual completeness would be → 7 (correctly alerted context) / 11 (alerted with missing context) = 0.64.

For each $\mathcal{F}^r$ and $\mathcal{F}^d$, we have to identify how important it is in terms of compliance with the risk. This (again) is an area where we can employ risk-based rules to develop a comprehensive **risk scoring equation**. For our work, we are going to use the turnover magnitude as a proxy for that scoring. The rationale here is: the higher the turnover, the bigger the crime (must be) that produced those funds. Taking terms from Eq. 1, we define a flow turnover as:

$$\mathcal{T} = \sum_{i=1}^{n} \max((\mathcal{S}_i - \mathcal{R}_i), 0) \tag{2}$$

Where, n is the number of accounts in a flow. We further define the normalized version of $\mathcal{T}$ as:

$$\bar{\mathcal{T}} = \min(\frac{\mathcal{T}}{\mathcal{C}}, 1) \tag{3}$$

Where $\mathcal{C}$ is the capping factor, that we define based on the risk appetite. We can now also define the importance of each account in a flow. If in a money laundering network an account only has a minimal contribution in terms of its relative turnover, its exclusion from the network should not be penalized equally. We define the risk score for an account i in $\mathcal{F}^r$ as follows:

$$\mathcal{T}_i^a = \frac{(\mathcal{S}_i + \mathcal{R}_i)}{\mathcal{T}} \tag{4}$$

The normalized version of Eq. 4 can be defined as:

$$\bar{\mathcal{T}}_i^a = \frac{\mathcal{T}_i^a}{\sum_{j=1}^{n} \mathcal{T}_j^a} \tag{5}$$

Using Eq. 2 and 5, we can now construct our *context-weighted* confusion matrix. Algorithm 2 describes the detailed implementation of the calculations. Once we have the context-weighted confusion matrix, the *context-weighted* versions of the F-1 score; recall; precision, etc. can be inferred from the same numbers.

To evaluate the proposed *context-weighted* recall in the context of AML modeling, we compare it against the normal recall metric. In Table III, the first column shows an example money laundering flow that we aim to detect using our model. For that flow, the third column shows the maximum money laundering contributions possible per account. Finally, the fourth and fifth columns show two scenarios for the detected context. In Scenario 1, a model identifies the anomalous context (as) involving accounts A, C, D, E, and F. In Scenario

TABLE III: Summary of an example ML (money laundering) flow: transactions, maximum ML share (or contribution), and detection scenarios. Scenario 2 has better context-weighted recall than Scenario 1.

Flow: Source→Target (Amount)	**Node**	**ML Share**	**Caught in Scenario 1**	**Caught in Scenario 2**
F→A (5 K)	A	11K	✓	✓
A→D (1 K)	B	10K	—	✓
C→A (2 K)	C	2K	✓	✓
E→A (2 K)	D	1K	✓	—
A→B (10 K)	E	2K	✓	—
—	F	5K	✓	—
		Recall	**83%**	50%
		Context-weighted Recall	68%	**84%**

2, a model identifies the context as A, B, and F. If we used the normal recall measure, we would have preferred the output (or detected context) of the first model. Although after a closer inspection, we can see that the second model captures the actors (most) *central* to the money laundering activity; i.e., the source (A), as well as the destination (B) of the laundered funds.

Algorithm 2 *Context-Weighted* Confusion Matrix

$\mathcal{X} \leftarrow \max(\|\mathcal{F}_1^d\|, \|\mathcal{F}_2^d\|, ..., \|\mathcal{F}_n^d\|)$
SET weighted_[TP,FP,TN,FN] $\leftarrow 0$
for i in $\boldsymbol{\mathcal{F}}^r$ **do**
  $j \leftarrow$ Get the index for the best matched $\mathcal{F}_i^r$ in $\boldsymbol{\mathcal{F}}^d$
  matched $\leftarrow \mathcal{F}_i^r \cap \mathcal{F}_j^d$
  TP $\leftarrow 0$
  for k in $matched$ **do**
    $TP \mathrel{+}= \bar{\mathcal{T}}_k^a$
  end for
  $FP \leftarrow$ (($\|\mathcal{F}_j^d\|$ - $\|matched\|$) / $\|\mathcal{F}_j^d\|$) $\times \bar{\mathcal{T}}_i$
  $TN \leftarrow$ (($\mathcal{X}$ - $\|\mathcal{F}_j^d\|$) / $\mathcal{X}$) $\times \bar{\mathcal{T}}_i$
  $FN \leftarrow 1 - TP$
  if $matched$ is empty **then**
    $TN \leftarrow 0$
    $FN \leftarrow \bar{\mathcal{T}}_i$
  end if
  $weighted\_[*]$ += $[TP, FP, TN, FN]$
end for
redundant_flows $\leftarrow$ Flows in $\boldsymbol{\mathcal{F}}^d$ with no match in $\boldsymbol{\mathcal{F}}^r$
for l in $redundant\_flows$ **do**
  TP, $FN \leftarrow 0$
  $FP \leftarrow \bar{\mathcal{T}}_l$
  $TN \leftarrow$ (($\mathcal{X}$ - $\|\mathcal{F}_l^d\|$) / $\mathcal{X}$) $\times \bar{\mathcal{T}}_l$
  $weighted\_[*]$ += $[TP, FP, TN, FN]$
end for

B. Results on the $\mathcal{D}_{lib}^{real}$ dataset

Table V shows the clear superiority of `ReDiRect` in terms of the reported TPR's. In some cases, we see improvements of up to ∼**12%**, even when the data scope is reduced to a very small proportion. In some cases, especially with higher selection thresholds, we see bigger improvements with greater reductions in the data scope. This claim can be further validated by analyzing Figure 3.

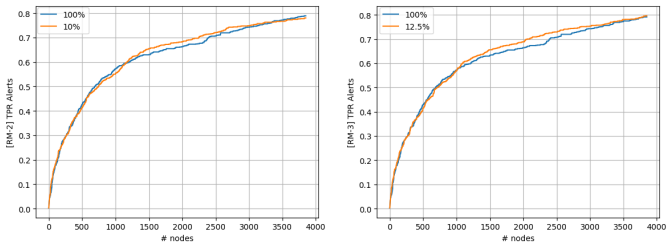

Fig. 3: TPR AUC plots comparing the initial run of `ReDiRect` (no reduction in data), with the outputs of RM-2 (10% reduction) and RM-3 (12.5% reduction) methods.

TABLE IV: Mean investigation lead times. Bold font represents the best times. Cells with shaded background represent variations of `ReDiRect`.

Method	**Mean ILT (Investigation Lead Time)**				
		Threshold Factor			
	Actual	**T1**	**T2**	**T3**	**T4**
EgoNetRed [19]	12.4	102.84	67.24	37.62	23.59
`ReDiRect`100%	8.35	23.24	23.49	21.11	17.54
[RM-2] 75%	8.33	23.36	23.4	20.34	16.59
[RM-2] 40%	7.99	21.72	21.4	17.48	13.63
[RM-2] 10%	6.78	**16.67**	**15.44**	13.63	10.98
[RM-3] 50%	8.10	22.40	22.28	18.57	14.38
[RM-3] 25%	7.86	20.50	20.06	16.24	12.93
[RM-3] 12.5%	**6.47**	17.84	16.03	**13.08**	**10.67**

1) Reduction in ILT (Investigation Lead Time): This is the main cost driver for any risk or AML department in a bank. Reducing the mean time it takes for an analyst to close an alert is of the utmost importance. To achieve that, we need to ensure that analysts have the most precise *context* for each alert they are investigating. In the case of `ReDiRect`, that would be the n-hop community we construct for each node (or account). For [19] that would be the reduced ego-net of each account.

For simplicity, we assume that an analyst takes a minute to investigate one account in an alerted *context*. This is an ideal proxy for complexity, as a higher number of accounts translates to a higher number of interactions and network intricacies. We can therefore define ILT for a given alert as the size of the *context* provided with that alert. Table IV shows the potential reduction in ILT, with decreasing size of the input data scope. In some cases, we have a reduction of up to a factor of ∼**6**×. To put this into perspective, an analyst can now perform the duties of 6 analysts in the same amount of time.

C. Results on the $\mathcal{D}_{ibm}^{syn}$ dataset

Here, we are going to focus on the *context-weighted* recall metric (/confusion matrix) defined earlier. We want to stress here that the same metric cannot be applied to the D_{lib}^{real} dataset, or any other openly available dataset out there. Only

TABLE V: Detailed evaluation results on the $\mathcal{D}_{lib}^{real}$ dataset. **Bold** font represents the best version. Red font is when ReDiRect is better than the competitors.

Method	Alerts					Reports				
	AUC-**T4**	TPR-**T1**	TPR-**T2**	TPR -**T3**	TPR -**T4**	AUC-**T4**	TPR-**T1**	TPR-**T2**	TPR-**T3**	TPR-**T4**
EgoNet [6]	0.599	0.287	0.437	0.660	0.815	0.574	0.146	0.336	0.672	0.850
EgoNetRed [19]	0.604	**0.405**	0.514	0.660	0.744	0.603	**0.355**	0.437	0.696	0.772
ReDiRect 100%	0.617	0.364	0.524	0.660	0.791	**0.676**	0.318	**0.546**	**0.727**	**0.955**
—"—[RM-2] 75%	0.623	0.364	0.527	0.670	0.804	0.674	0.318	**0.546**	**0.727**	**0.955**
—"—[RM-2] 40%	0.634	0.374	**0.529**	0.696	**0.819**	0.653	0.318	0.409	**0.727**	**0.955**
—"—[RM-2] 10%	0.623	0.372	0.510	0.680	0.781	0.586	0.318	0.364	0.636	0.773
—"—[RM-3] 50%	0.631	0.372	0.525	0.690	0.814	0.658	0.318	0.409	0.727	0.955
—"—[RM-3] 25%	0.634	0.369	0.516	0.700	0.803	0.623	0.318	0.318	0.727	0.909
—"—[RM-3] 12.5%	0.629	0.368	0.513	0.685	0.796	0.593	0.318	0.318	0.636	0.773

TABLE VI: *Context-weighted* recall for the $\mathcal{D}_{ibm}^{syn}$ datasets. LI = Low-Illicit; HI = High-Illicit.

Method	Threshold Factor			
	T1	**T2**	**T3**	**T4**
[LI] RM-1	0.034	0.075	0.190	0.351
[HI] No reduction	0.050	0.087	0.271	0.450
[HI] RM-1	0.287	0.498	0.675	0.713
[HI] RM-2 50%	0.237	0.477	0.658	0.683

because of the *rich* flow-level labels present in $\mathcal{D}_{ibm}^{syn}$, we are able to apply our proposed metric.

Table VI reports the *context-weighted* recall with different data reduction methods. It is again clear from the results that the reduction in the input data scope considerably improves accuracy. However, RM-2 does not work well here. We believe that this is due to the presence of an *unrealistically* high number of accounts involved in money laundering activities. For such a dataset, a heuristic-based approach (like RM-1) might be more suitable. Using Algorithm 2 we can also construct the *context-weighted* confusion matrix as shown in Table VII, for a more holistic view of the model performance. An interesting thing to note is that the numbers reported here are *even* (slightly) better than those produced from [23], where the model is trained in a **supervised** setting instead. Although, to state the obvious, the comparison is not entirely fair, as we have different data scope and a different variation of metrics.

TABLE VII: *Context-weighted* confusion matrix for $\mathcal{D}_{ibm}^{syn}$ (high-illicit) [RM-1], with T4 selection threshold.

		p	**n**	**Total**
Actual	**p'**	369	149	**518**
	n'	3497	63063	**66560**
Total		**3866**	**63211**	

D. Execution times

In terms of scalability, we were able to run all of our experiments on a personal laptop. Even executions for the larger $\mathcal{D}_{ibm}^{syn}$ datasets, with up to 180 million transactions, do not take more than a couple of hours.

From Figure 4, we can validate that the framework scales almost linearly with increasing data size. Furthermore, Figure 5 shows how the execution times for different steps can be reduced (drastically) by having higher levels of parallelization.

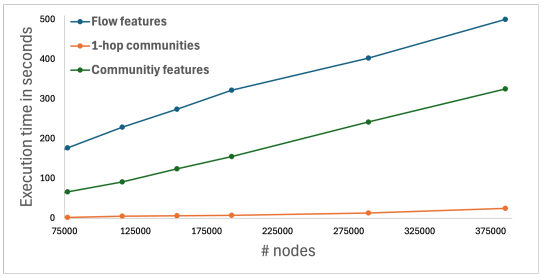

Fig. 4: The execution times for each of the heavy duty tasks in the framework, with increasing number of nodes in the $\mathcal{D}_{lib}^{real}$ dataset.

Fig. 5: Execution times for $\mathcal{D}_{ibm}^{syn}$ large dataset, with increasing level of distribution.

VI. Conclusion and Future Work

We showed that with ReDiRect it is possible to generate high-quality and *comprehensive* money laundering alerts, while keeping false positives to a minimum. Not just the overall false positive signals; but also false positives within a community identified as anomalous. The fine tuning of the last step *Rectify* has much room for improvement. We can employ stratified and stochastic model trainings to minimize the impact of groups of nodes appearing in multiple communities. Advanced deep learning models such as GNN [5], autoencoders [7, 10], etc. could be suitable candidates to identify anomalous communities in a more robust manner.

The selection of the top anomalies is also an area where we can have some improvements. Anomalous communities with overlapping nodes can be joined or merged in a smart way,

reducing (even further) the number of anomalies to analyze. We can also apply AML typology-specific risk scoring to shortlist more relevant alerts. For example, to capture trade-based money laundering [26] networks, we can select communities with (certain types of) businesses that are mostly used in such schemes. We plan to cover all of these areas of improvement in our future research. Finally, we invite fellow-researchers to use Tables VI and VII as the first benchmarks, based on the (proposed) *context-weighted* metrics.

References

[1] S.N. Welling. "Smurfs, money laundering and the federal criminal law: The crime of structuring transactions". In: *Fla. Law* 41 (1989), pp. 287–343.

[2] Lawrence Page et al. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report. Stanford InfoLab, Nov. 1999.

[3] M. E. J. Newman. "Modularity and community structure in networks". In: *Proceedings of the National Academy of Sciences* 103.23 (2006), pp. 8577–8582.

[4] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), P10008.

[5] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80.

[6] Leman et al. Akoglu. "oddball: Spotting Anomalies in Weighted Graphs". In: *Advances in Knowledge Discovery and Data Mining*. 2010. ISBN: 978-3-642-13672-6.

[7] Pierre Baldi. "Autoencoders, Unsupervised Learning, and Deep Architectures". In: *Proceedings of ICML*. Vol. 27. PMLR, July 2012, pp. 37–49.

[8] Peter A. Lofgren et al. "FAST-PPR: scaling personalized pagerank estimation for large graphs". In: *Proceedings of SIGKDD*. 2014, pp. 1436–1445.

[9] Leman et al. Akoglu. *Graph based anomaly detection and description: a survey*. 2015. DOI: 10.1007/s10618-014-0365-y.

[10] Thomas N. Kipf et al. *Variational Graph Auto-Encoders*. 2016. URL: https://arxiv.org/abs/1611.07308.

[11] Marco Tulio et al. Ribeiro. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD*. 2016. DOI: 10.1145/2939672.2939778.

[12] David Savage et al. "Detection of money laundering groups using supervised learning in networks". In: *ArXiv* (Aug. 2016).

[13] William L. et al. Hamilton. "Inductive representation learning on large graphs". In: *Proceedings of the 31st NIPS*. 2017. ISBN: 9781510860964.

[14] Paul et al. Wagenseller. *Size Matters: A Comparative Analysis of Community Detection Algorithms*. 2018.

[15] Zhongying Zhao et al. "A comparative study on community detection methods in complex networks". In: *Journal of Intelligent & Fuzzy Systems* 35 (June 2018).

[16] V. A. Traag, L. Waltman, and N. J. van Eck. "From Louvain to Leiden: guaranteeing well-connected communities". In: *Scientific Reports* 9.1 (2019), p. 5233.

[17] Xiangfeng Li et al. "FlowScope: Spotting Money Laundering Based on Graphs". In: *Proceedings of AAAI* 34.04 (Apr. 2020), pp. 4731–4738.

[18] Scott M. Lundberg et al. "From local explanations to global understanding with explainable AI for trees". In: *Nature Machine Intelligence* 2.1 (2020), pp. 2522–5839.

[19] Bogdan et al. Dumitrescu. "Anomaly Detection in Graphs of Bank Transactions for Anti Money Laundering Applications". In: *IEEE Access* 10 (2022). DOI: 10.1109/ACCESS.2022.3170467.

[20] Xiaobing Sun et al. "MonLAD: Money Laundering Agents Detection in Transaction Streams". In: *WSDM '22*. 2022, pp. 976–986.

[21] Panagiotis Chatzigiannis et al. *Privacy-Enhancing Technologies for Financial Data Sharing*. 2023.

[22] Erik Altman et al. "Realistic synthetic financial transactions for anti-money laundering models". In: *NIPS '23*.

[23] Jovan Blanuša et al. "Graph Feature Preprocessor: Real-time Subgraph-based Feature Extraction for Financial Crime Detection". In: ICAIF '24.

[24] Doug Bonderud. *AML statistics of 2023*. URL: https://withpersona.com/blog/the-most-mind-blowing-money-laundering-statistics-of-2022. (accessed: 08.03.2025).

[25] Europol. *Money Muling*. URL: https://www.europol.europa.eu/operations-services-and-innovation/public-awareness-and-prevention-guides/money-muling. (accessed: 14.06.2023).

[26] FATF. *Trade-Based Money Laundering*. URL: http://fatf-gafi.org/en/publications/Methodsandtrends/Trade-basedmoneylaundering.html. (accessed: 15.03.2023).

[27] Rezaul et al. Karim. "Scalable Semi-Supervised Graph Learning Techniques for Anti Money Laundering". In: (). DOI: 10.1109/ACCESS.2024.3383784.

[28] Fei et al. Liu. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422.

[29] Reuters. *ABN Amro to settle money laundering probe for $574 mln*. URL: https://www.reuters.com/business/abn-amro-settle-money-laundering-probe-574-million-2021-04-19/. (accessed: 14.03.2023).

[30] Xiaobing Sun et al. "CubeFlow: Money Laundering Detection with Coupled Tensors". In: *PAKDD 2021*.

[31] Haseeb Tariq and Marwan Hassani. "Extracting Money Laundering Transactions from Quasi-Temporal Graph Representation". In: *ACM SIGAPP SAC '26*. DOI: 10.1145/3748522.3779790.

[32] Haseeb Tariq and Marwan Hassani. "Topology-Agnostic Detection of Temporal Money Laundering Flows in Billion-Scale Transactions". In: *PKDD '25*. ISBN: 978-3-031-74643-7. DOI: 10.1007/978-3-031-74643-7_29.

[33] UNODC. *Money Laundering Overview*. URL: https://www.unodc.org/unodc/en/money-laundering/overview.html. (accessed: 25.03.2023).