A New Algorithm for Money Laundering Detection Based on Structural Similarity

Reza Soltani, Uyen Trang Nguyen, Yang Yang, Mohammad Faghani, Alaa Yagoub, Aijun An
Department of Electrical Engineering and Computer Science
York University
Toronto, Canada
(rts, utn, yangcs, faghani, yagoub, aan)@cse.yorku.ca

Abstract— **Money Laundering (ML) is the process of cleaning 'dirty' money, thereby making the source of funds no longer identifiable. Detecting money laundering activities is a challenging task due to huge volumes of financial transactions being made in a global market on a daily basis. This paper proposes a novel approach for detecting money laundering transactions among large volumes of financial data in an efficient and accurate manner. We propose a framework that applies case reduction methods to progressively reduce the input data set to a significantly smaller size. The framework then scans the reduced data to find pairs of transactions with common attributes and behaviours that are potentially involved in ML activities. It then applies a clustering method to detect potential ML groups. We present preliminary experimental results that demonstrate the effectiveness of the proposed framework.**

Keywords: money laundering, money laundering detection, graph theory, structural similarity

I. Introduction

Money Laundering (ML) is the third largest business in the world and accounts for about 2.7% of the global gross domectic product (GDP) [1]. Money laundering is the process of cleaning 'dirty' money, thereby making the source of funds no longer identifiable. Dirty monies are the funds obtained from criminal activities. Criminal activities include drug trafficking, illegal gambling and tax evasion. Consequently, preventing and eliminating money laundering activities restrict the proceeds obtained from criminal behaviour and reduce the growth of crime.

There are many methods of money laundering. Criminals can hide the source of money by using the funds in casinos or real estate purchases, or by overvaluing legitimate invoices. In general, a money laundering procedure is composed of three major steps: *placement*, *layering* and *integration* [2]. Placement is the process of introducing the dirty money into the financial system by some mean. Layering is the processing of carrying out complex transactions to hide the source of the funds. Finally, integration is to withdraw the proceeds from a destination bank account. The purpose of performing complex layering is to confuse anti-money laundering instruments.

For many years, governments and businesses all around the world have established regulations and recommendations to fight money laundering. In Canada FINTRAC (Financial Transactions and Reports Analysis Centre of Canada) [2] is the government agency responsible for providing regulations and policies to all Canadian entities that interact with money and securities. These entities include accountants, banks, and real estate agencies. The regulations demand entities to report large transactions (online and offline) and suspicious transactions. FINTRAC also provides guidelines for the general public. Similar agencies exist in other countries. FinCen is the United States counterpart that provides regulations and guidance to financial institutes. There are many international agencies such as the Financial Action Task Force (FATF) [3] that set standards and promote effective implementations of regulatory measures for combating money laundering.

However with the growth of global economy, e-businesses and online transactions, it is predicted that ML crimes will grow steadily [4]. Given the great volume of transactions being made in a global market on a daily basis, the amount of data to be processed by anti-money-laundering (AML) systems is substantial.

In existing money laundering detection (MLD) works [4, 7, 11, 12, 14], complex operations are performed on large data sets to detect potential ML transactions. In the era of economic globalization and big data, the enormous amount of financial data to be processed daily will render most of existing MLD algorithms unscalable. In this paper, we present preliminary results of a *scalable* MLD framework designed to detect money-laundering transactions among large volumes of financial data in an efficient and accurate manner. The objective of this research is lay the ground work for a framework that can process transactions in near real time and have the ability to flag suspicion accounts within 72 hours of committing the fraudulent transactions.

The proposed framework adopts a number of case reduction methods to progressively reduce the dataset to a significantly smaller size. The framework then scans the data to find pairs of transactions with common attributes and behaviours that are potentially involved in ML activities. It then applies a clustering method to detect potential ML groups. (Suspicious transactions and accounts output by the framework will have to be investigated by humans to confirm their guilt. The purpose of MLD algorithms is to reduce the originally large data set to a much smaller number of transactions to be inspected by humans.) In this paper, we first describe the proposed MLD framework and then present preliminary experimental results that demonstrate the effectiveness of the proposed framework.

The remainder of the paper is organized as follows. Section II describes related research on anti-money laundering solutions. In Section III we present the proposed framework. Section IV describes the results of the experiments performed on the framework. Finally, section V provides concluding remarks and outlines our future work.

II. Related Work

This section summarizes existing works in the field of anti money laundering research. Money laundering detection started in 1970 when financial institutes began reporting large transactions to their governments. Statistical methods such as Bayesian models and temporal sequence matching were used in late 1990s to detect money laundering [5]**.** Later on in 2004, machine-learning approaches were applied to find ML patterns. Donoho [6] as well as Wang *et al.* [7] proposed C4.5 decision tree algorithms to detect money laundering risks. Tang *et al.* [8] proposed a support vector machine to detect money laundering activities. Lin-Tao Lv *et al.* proposed radial-based function neural network model to detect ML activities [9].

In recent years graph-based approaches have gained popularity. Zhang *et al.* [4] use financial transaction networks and community detection algorithms to find ML groups. Awasthi *et al.* combines genetic algorithm with clustering methods to find money-laundering communities [10]. Chen *et al.* use transaction attributes such as amount, timestamp to perform matching transactions. It then uses fuzzy logic to find money-laundering activities [11]. Michalak and Korczak [12] propose a graph-mining algorithm to detect ML activities. Their research takes a different direction from analyzing individual transaction and instead focuses on the relationships among accounts and transactions.

Our research presented in this paper shows that it is possible to find money-laundering accounts by studying the communities in financial networks. However, processing large financial network structures is costly. Our proposed framework applies case reduction methods to progressively reduce the original data set to a significantly smaller size. It then applies a clustering method to detect potential ML transactions. Efficient methods to reduce the size of the original data set is crucial to money laundering detection since huge amounts of data is produced daily from e-businesses, e-commerce and other online financial transactions. In this paper we present a new approach in reducing the size of the data, different from existing works.

To protect their clients' privacy and to maintain the confidentiality of clients' activities, financial institutions safeguard transaction information with utmost diligence. One of the challenges in AML research is to obtain real financial data in order to evaluate new algorithms. Therefore synthetic data have been used for research in money laundering detection [13]. In the topic of synthetic data generation, E. Lopez-Rojas and S. Axelsson [14] propose a method constructed on the idea of multi-agent based simulation to generate synthetic transaction data. Multi-agent based simulation [15] relies on the interactions among autonomous and interactive agents to model complex systems. Their data generation system takes into account numerous financial attributes including customer ID, date of transaction, type of transaction and amount transferred. Some of the transactions are labeled as *suspicious* in order to train the supervised AML algorithm. In [12], Michalak and Korczak design a model that represent a micro-economy. This model consists of three types of entities: businesses, individuals and government offices. The behaviour and quantity of each entity is regulated by its parameters. In this model, salary payments, purchases of goods, and tax payments form new transactions. A collection of transactions obtained for a duration of one year is used as the test data in their work.

Our proposed framework has been evaluated using synthetic test data to obtain preliminary results reported in this paper. The data consist of financial accounts and transactions with basic attributes common to all accounts and transactions in real world. Section III.A describes the dataset in detail.

III. The Proposed Framework and money laudering detection Algorithm

This section describes the proposed MLD framework in detail. The framework is designed such that, given a set of bank accounts and transactions, the framework will return a list containing potential groups of money laundering accounts. The framework starts by reviewing the input and searching for *matching transactions*. Matching transactions are financial transactions that have common attributes such as similarity in deposit and withdrawal amounts. Next the framework generates a network representation of all matching transactions. It then uses network-based algorithms to filter out unnecessary accounts and transactions. The framework then applies a clustering method to find suspicious ML communities within the network. Finally, the extracted communities are rearranged, sorted and returned as the output of the framework.

A. Input Dataset

The input to the algorithm is a set of financial transactions and a set of financial accounts. Each account has two attributes:

- *Account-ID:* the unique numeric ID of a financial account
- *Account-Trust-Flag*: this flag can be either *True* or *False*. The value *True* indicates the account is whitelisted. White lists contain accounts that are trustworthy. To reduce false positive results, financial accounts of trustworthy owners can be (optionally) added to the whitelist. Accounts in a white list are not selected as potential ML accounts even when they are exhibiting ML behaviour.

Each transaction has the following five attributes:

- *TransactionID:* the unique numeric ID of a financial transaction
- *SenderAccountID:* the unique ID of the account sending the funds
- *ReceiverAccountID:* the unique ID of the account receiving the funds

- *Amount:* The amount of funds being transfered
- *Timestamp*: The date and time of when the transfer took place

The dataset is comprised of two files containing the financial account and transactions in CSV (comma seperated values) format. The following is a sample CSV file given to the framework as input:

```
TransactionID, SenderAccountID, ReceiverAccountID, Amount, TimeStamp
11,845,785,3033, 1467422592
21, 950,404,10000,6, 1467423332
22, 404,626,9800, 1467452512
35, 209, 118, 10000, 1467472512
37, 118, 710, 9800, 1467572512
```

B. Discovering Matching Transactions

Money laundering crimes are often executed by certain professional groups which we term ML groups. ML groups receive illegal money from clients and transfer it to the final payee through their member ML accounts. Figure 1 shows the basic topologies of ML groups described by Michalak and Korczak[12]. Each of the two topologies consist of a sender account, a reciever account and one or many intermediary accounts. These three types of accounts correspond to the placement, layering and integration phases of ML activities explained in Section I. The sender account is used to introduce the dirty funds into the financial system. The purpose of retransfering the funds through intermediate accounts is to obsecure the source of funds. The funds are ultimately obtained from the receiver account as clean money. In Figure 1.a funds are divided into smaller batches and are advanced to a receiver account. Conversly in Figure 1.b the entire fund is passed through multiple intermediary accounts and eventually deposited into the receiver account. Actual ML activities are generally based on the basic topologies shown in Figure 1. Therefore detection of such sub-structures assist in detecting larger scopes of ML networks [12].

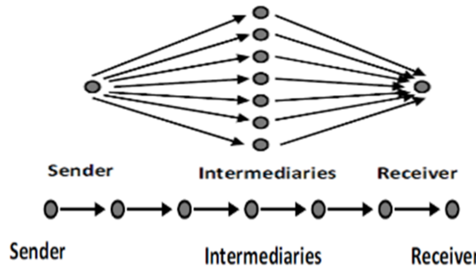

Fig. 1. Two common topologies of money laundering networks obtained from [12]. a) Top: A common ML topology with one sender account and one reciever account, and many intermediary accounts forming a column. The funds are divided by the sender among intermediary accounts, which then resend the funds to the receiver account. b) Bottom: A common ML topology consiting of one sender and one receiver and a row of intermediary accounts. The intermediary accounts pass the entire amount (minus the optional commision) of funds to the next account in the row until ultimately they are received at the receiver account.

The role of ML groups is to retransfer the money through multiple accounts efficiently and with little or no change in the amount. These characteristics allow ML accounts and transactions to exhibit certain unique properties. These properties can be used to pair certain transactions together. Chen and Mathe [11] introduced a fuzzy score between pairs of transactions. These scores are used to evaluate the ML likelihood of financial accounts.

In the proposed framework, *matching transactions* are transactions that share certain attributes. Two transactions *A* and *B* are a pair of matching transactions if they have the following properties:

- The destination of transaction *A* (*ToAccountID*) is the same as the sender (*FromAccountID)* of transaction *B*. In other words, transaction *A*'s receiver must be transaction *B*'s sender. This property helps to detect intermediary accounts assigned with the task of *retransfering* funds.
- The amount value (*Amount*) of both transactions *A* and *B* should be equal or differ by less than a predefined constant. The reason is that money laundering agents may retransfer the whole amount, or hold back a small amount as commission for their work [11].
- The amount value (*Amount*) of both transactions *A* and *B* should be above a certain threshold (e.g., above $10,000). Based on current banking guidelines, this is the amount that triggers MLD operations [2].
- The difference in time between both transactions must be within a threshold (e.g., within 72 hours). As indicated in [11], ML accounts are in the business of retransfering funds from the initial sender to the final receiver in an efficient manner. This means that a shorter delay between retransfers implies a higher possibility of money laundering activity.

The above properties can be formulated as the following equation. For a given set of transactions $T = \{t_{ij} | i, j = 1 \cdots N\}$ where *i* and *j* are two distinct bank accounts and *N* denotes the total number of transactions to be processed, the set of matching transactions *P* is defined as:

$$P = \left\{ \langle t_{ij}, t_{jk} \rangle \middle| \begin{array}{c} A(t_{ij}) > \alpha, \\ 1 \geq \frac{A(t_{jk})}{A(t_{ij})} > \beta, \\ T(t_{ij}, t_{jk}) < \theta \end{array} \right\} \tag{1}$$

In Eq. (1), the function $A(t_{ij})$ returns the amount of fund being transferred between account *i* and *j*. This amount should be higher than the constant α, the threshold above which a transaction is considered a potential ML activity, e.g., α = \$10,000. The rate of equality between the amounts of incoming transaction t_{ij} and outgoing transaction t_{jk} should be higher than a user-defined constant β for the two transactions to be considered suspicious. β is the minimum rate of commission held back by (paid to) a money laundering agent. The function $T(t_{ij}, t_{jk})$ returns the time difference between two transactions t_{ij} and t_{jk}. The constant θ is the maximum time difference allowed between two transactions (e.g., 72 hours) in order for them to be considered as a potential ML activity. The resulting P contains a set of unique pairs of transactions where each transaction appears only once.

Given the sample file input listed at the end of Section III.A and the following parameters: α = \$10,000, $\beta = 0.9$ and θ = 72 hours, the output set P consists of the following items: P = {<21, 22>, <35, 37>}. Each item in the set contains the IDs of matching transactions.

C. Generate graphs of Matching Transactions

Given the set P the next step is to generate the weighted directed graph $G(V, E)$, where V are nodes representing accounts and E are the directed edges representing transactions. The transactions connect one account to another. An outbound edge from node A represents a transaction from account A to another account. Likewise an inbound edge connected to account A represents a transaction from another account to account A. Every edge has a weight associated with it. Let w_{ij} represent the weight of the edge from node i to node j. The value w_{ij} represent the number of transactions from account i to account j. To generate the graph the following procedure is executed:

```
For each pair of matching-transactions <$t_{ij}$,$t_{jk}$> in P:
    1- Add three nodes $v_i$, $v_j$, $v_k$ into graph G if they do not yet exist. Otherwise, go to step 2.
    2- Add directed edges $e_{ij}$ and $e_{jk}$ in G if they do not yet exist. Otherwise, increment the weights $w_{ij}$ and $w_{jk}$ by one.
```

The use of the edge weights will be discussed in detail in the next sub-section.

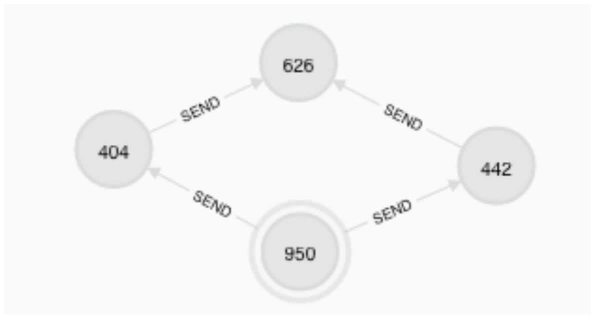

Fig. 2. The graph representation of two pairs of matching transactions. This graph is produced by step C of the framework. The edges show funds are sent from a sender account with ID 950 to two intermediary account with IDs 404 and 442, from which the funds are sent to the receiving account with ID 626.

Fig. 2 shows part of the sample graph generated from two pairs of matching transactions. The nodes of the graph represent the financial accounts. The directed edges represent the financial transaction. The first pair of matching transactions consists of the transactions connecting accounts 950 to 404 and account 404 to 626. Similarly, the second pair of matching transaction connects account 950 to 442 and account 442 to 626. On Figure 2 the flow of funds originates from account 950 and ends at 626.

D. Filtering based on Balance Score

Money laundering accounts are typically involved in large volume of transactions, and they retransfer significantly larger amounts of funds in comparison to benign accounts. This means that nodes with higher degree (i.e. higher number of connected edges and weights) are more likely to be part of an ML group. Moreover the number of outgoing transactions from an ML account must correlate with number of its incoming transactions. Let w_{ij} denote the weight of an edge connecting node i to node j, the following Eq. (2) gives the *balance score (B-score)* of a node.

$$B(i) = \frac{2\sum_{j=1}^{N} w_{ij} \sum_{j=1}^{N} w_{ji}}{\left(\sum_{j=1}^{N} w_{ij}\right)^2 + \left(\sum_{j=1}^{N} w_{ji}\right)^2} \times \log\left(\min\left(\sum_{j=1}^{N} w_{ij}, \sum_{i=1}^{N} w_{ji}\right)\right) \quad (2)$$

The balance score of a node is determined by the ratio of the weights of incoming and outgoing edges multiplied by the log of minimum of its incoming and outgoing edges as shown in Eq. (2). The purpose of the logarithm expression is to produce a smaller value for the final balance score. Eq. (2) produces a higher value when the number of edges connected to the node is higher and the difference between the weights of incoming and outgoing edges is smaller (higher balance between incoming and outgoing edges).

A node with a balance score above a pre-defined threshold Q is considered a potential money-laundering node, and is stored in set C to be considered in the next step. The constant Q represents the level of filtering on the nodes. As Q increases, the number of potential ML accounts selected by the framework is reduced. ML accounts are typically involved in larger than usual numbers of transactions. Therefore, the value of Q is typically set to a value above the average balance score of all nodes in the graph.

As an example, consider applying Eq. (2) to obtain the balance score of account 442 shown in Fig. 2. The result should be a value between 0 and 3 inclusively (given a maximum of 1000 inbound/outbound transactions). Assuming that the numbers of incoming and outgoing transactions are 20 each (i.e., each inbound/outbound edge has a weight of 20), the resulting value B is 1.3. If this value is above the predefined threshold Q, account 442 will be added to the list C. Otherwise, it will be removed from further processing.

E. Analysis Based on Structural Similarity

Money laundering accounts usually involve the same clients. Structure similarity [16] is an effective method for detecting groups of nodes involving in the same money laundering case. Structural similarity achieves this goal by clustering together nodes with common neighbours and adjacent edges of similar weights. The structural similarity algorithm used in this paper is called *Shrink* and is introduced in [16]. The modified *Shrink* algorithm to evaluate node similarity is described below.

For each node $u \in V$ let $\Gamma(u)$ represent the set of nodes with inbound edges directed to *u*. Let $\Gamma'(u)$ represent all nodes with outbound edges originating from node *u*. Let $w_{uu} = 1$ to avoid division by 0 in Eq. (3) shown below. Then the structural similarity between two nodes *u* and *v* is given by:

$$\sigma(u,v) = \frac{\sum_{x\in\Gamma(u)\cap\Gamma(v)} w_{xu}w_{xv}}{\sqrt{\sum_{x\in\Gamma(u)} w_{xu}{}^2}\sqrt{\sum_{x\in\Gamma(v)} w_{xv}{}^2}} \times \frac{\sum_{x\in\Gamma'(u)\cap\Gamma(v)} w_{ux}w_{vx}}{\sqrt{\sum_{x\in\Gamma'(u)} w_{xu}{}^2}\sqrt{\sum_{x\in\Gamma'(v)} w_{xv}{}^2}} \tag{3}$$

The structural similarity between every pair of nodes found on set *C* is calculated. The result is a similarity matrix *S* in which each cell stores a value $\sigma(u,v)$ corresponding to nodes *u* and *v*. A pair of sufficiently similar nodes – also know as a *dense pair* – is a pair with nodes *u* and *v* such that their $\sigma(u,v)$ is above a threshold γ. Let set *D* hold the set of all unique dense pairs.

Table I displays an example similarity matrix *S* from four different accounts. Assuming that the threshold value γ is 0.2. Set *D* will contain the pairs {<442, 404>, <950, 442>, <404, 950>}

TABLE I. SIMILARITY MATRIX S FROM SAMPLE DATA.

$\sigma(u,v)$	626	442	950	404
626	0.0	0.0	0.0	0.0
442	0.0	0.0	0.6	0.97
950	0.0	0.6	0.0	0.7
404	0.0	0.97	0.7	0.0

F. *Mining Money Laundering Groups*

The last step is to derive the money laundering groups from the set of dense pairs *D*. This is accomplished by merging together the pairs that have at least one common node. The merge procedure is as follow:

```
for each pair d in D :
    for each node i in d :
        for each unprocessed pair d' in (D – d):
            if i is in d' :
                merge d and d'
                remove d' from D
```

The final set *D* consists of groups of accounts that are potentially involved in money laundering activities. As an example, assuming the original set *D* is based on the similarity matrix shown in Table I. The final set *D* produced by the above procedure is {<442, 404, 950>}. This means the accounts with IDs 442, 404 and 950 constitute a money laundering group. This step is following by human investigators performing further analysis on the obtained ML groups.

IV. EXPERIMENTS

This section describes prreliminary experiments performed to evaluate the effectiveness of the proposed framework.

A. *Testing environment and implementation*

The proposed framework is implemented and tested on a machine with 2.6 Ghz Intel Core i5, and 8GB of RAM running 64-bit Mac OS El Capitan. The implementation is done using Java (J2SE-1.7) on Eclipse IDE. To handle graph structures, we use the *Neo4j* graph engine [17]. Neo4j is a Java-based database engine offering Java API, a querying language named Cypher, and a web-based console to interact with its database.

B. *Dataset*

A significant challenge with this type of research is obtaining realistic financial data to evaluate the proposed algorithms. Financial institutions are very protective in regards to sharing financial information of their customers. Therefore, to evaluate the performance of the proposed framework synthetic financial data is generated and used as test data. A Java program that takes as input two sets of parameters performs the data generation task. The first set of parameters specifies the properties of clean financial transactions. The second set of parameters specifies the properties of money laundering transactions.

We conducted five tests. Following are their parameters as described in Table II as well:

- Tests 1, 2 and 3: The graph contains 1,000 nodes (client accounts). We generated 300 clean transactions and 10 ML transactions.
- Test 4: The graph has 10,000 nodes. There are 500 clean and 100 ML transactions.
- Test 5: There are 10,000 nodes in the graph, 200 clean and 200 ML transactions.

TABLE II. EXPERIMENT PARAMETERS

	Tests 1,2,3	Test 4	Test 5
Number of accounts	1,000	10,000	10,000
Clean Transactions			
Clean transactions	300	500	200
Transactions between two nodes	1 to 3	1 to 3	1 to 3
Amount in a transaction	\$1 to \$5,000	\$1 to \$5,000	\$1 to \$5,000
Time between deposit and withdrawal (days)	1 to 24	1 to 24	1 to 24
Money Laundering Transactions			
ML Transactions	10	100	200
Transactions between two nodes	6 to 12	6 to 12	6 to 12
Amount in a transaction	\$9,900 to \$10,000	\$9,900 to \$10,000	\$9,900 to \$10,000
Time between deposit and withdrawal (days)	1 to 2	1 to 2	1 to 2

In all the tests, the amounts of funds involved in clean transactions range from \$1 to \$5,000, and the time difference between a deposit and a withdrawal range from one to 24 days. There are one to three transactions between any two clean nodes. In ML transactions, the amounts of funds range from \$9,900 to \$10,000, and the time difference between a deposit and a withdrawal is 48 hours or less. There are six to 12 transactions between any two ML nodes because ML accounts are involved in higher than usual numbers of transactions. Table II summarizes the parameters of the experiments.

C. Results and Discussion

The proposed AML framework can be configure with a number of parameters. These parameters are introduced in Section III. The following is the list of parameters used in the experiments:

- *degreeConstant*: also known as the Q constant, which ranges from 0 to 3 inclusive (for maximum of 10,000 transactions). This is the balance score threshold discussed in Section III.D. A higher threshold yields fewer potential ML accounts, leading to a higher false negative rate but a smaller false positive rate.

- *densePairConstant*: also known as the γ constant, which ranges from 0 to 1 inclusive. This is the dense pair threshold introduced in Section III.E. A higher dense pair threshold results in more restrictive filtering on dense pairs.

- *amountThreshold*: also known as the α variable in Eq. (1), which is the minimum amount of funds transferred by a transaction to be considered an ML transaction. This threshold is set to \$10,000 for all experiments. This value is used in most papers on MLD research [2].

- *allowedAmountDifference*: also known as the β variable in Eq. (1), which is the difference between a deposit and a withdrawal amount. This is the money held back by (paid to) the account owner for his/her money laundering service. In all experiments the value of this parameter is set to \$100, which means that the difference between the amount of funds received and resent by an account should not be larger than \$100. Reducing the value of this parameter decreases the number of results processed by the framework, which may consequently decrease the number of true positive results.

- *allowedTimeDifference*: also known as the θ variable in Eq. (1), which the maximum time allowed between a deposit and a withdrawal of ML money. ML funds typically change hands very quickly [11]. In all experiments the allowed time difference is set to two days, meaning that ML funds are retransferred to another account within two days or less. Reducing this parameter lowers the number of true positive results.

The results of the five tests are shown in Table III. As it shows in the third column of Table III (Test 3), executing the framework with values 0.1 and 0.2 as the *degreeConstant* and *densePairConstant* values, respectively, leads to the detection of 9 out of 10 ML groups, and 28 of 29 ML accounts. This translates to 90% accuracy in detection of ML groups and 96% accuracy in detection of ML accounts. This leads to the conclusion that using the lowest possible value as *degreeConstant* yields the highest accuracy in the proposed framework at the cost of possible increase in false positive results (which can be inspected later by humans to remove these false positive results from the list of ML accounts).

The result obtained from Tests 4 and 5 demonstrate the high accuracy of the framework on larger datasets. The current framework is able to detect ML groups that follow the structure presented in Fig. 1(a). Future experiments will focus on evaluating the framework on more complex ML structures.

TABLE III. RESULTS OF EXPERIMENTS

Procedure	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Test 5**
Parameters					
degreeConstant	0.8	0.4	0.1	0.1	0.1
densePairConstant	0.2	0.2	0.2	0.2	0.2
amountThreshold	\$10,000	\$10,000	\$10,000	\$10,000	\$10,000
allowedAmountDifference	\$100	\$100	\$100	\$100	\$100
allowedTimeDifference (days)	2	2	2	2	2
Result of Tests					
Number of matched transactions	369	291	280	2881	5671
Accounts with high balance score	23	31	29	318	632
Total number of dense pairs	40	41	34	439	876
Number of ML accounts identified	**20**	**29**	**28**	**303**	**593**
Total number of ML accounts	**28**	**31**	**29**	**318**	**629**
Number of ML groups identified	**7**	**8**	**9**	**85**	**164**
Total number of ML groups	**10**	**10**	**10**	**100**	**200**
Percentage of ML groups detected	**70%**	**80%**	**90%**	**95%**	**94%**
Percentage of ML accounts detected	**71%**	**93%**	**96%**	**85%**	**82%**

V. Conclusion

The proposed MLD framework aims to find potential money-laundering groups among a large number of financial transactions. In order to improve the efficiency of the framework, case reduction methods such as matching-transaction detection and balance score filter are used to narrow down the list of potential ML accounts. Next by taking advantage of structural similarity, we can identify and group potential money laundering accounts. Our preliminary experimental results show a high degree of accuracy in detection of ML accounts.

In our future work, we plan to improve the detection accuracy further by incorporating neural networks and fuzzy logic approaches, and to support more complex ML network structures. We will also conduct experiments on real financial data obtained from banks to evaluate the framework under real-life scenarios.

VI. References

1. Fatf-gafi.org, 'Money Laundering - Financial Action Task Force (FATF)', 2014. [Online]. Available: http://www.fatf-gafi.org/faq/moneylaundering/. [Accessed: 22- Dec- 2015].
2. Fintrac-canafe.gc.ca, 'What is Money Laundering', 2011. [Online]. Available: http://www.fintrac-canafe.gc.ca/fintrac-canafe/definitions/money-argent-eng.asp. [Accessed: 22- Dec- 2015].
3. "Fatf-gafi.org - Financial Action Task Force (FATF)", Fatf-gafi.org, 2016. [Online]. Available: http://www.Fatf-gafi.org. [Accessed: 22- Dec- 2015].
4. Zhang, Zhongfei Mark, John J. Salerno, and Philip S. Yu. "Applying data mining in investigating money laundering crimes." Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2003.
5. Phua, Clifton, et al. "A comprehensive survey of data mining-based fraud detection research." arXiv preprint arXiv:1009.6119 (2010).
6. Donoho, Steve. "Early detection of insider trading in option markets." Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004.
7. Wang, Su-Nan, and Jian-Gang Yang. "A money laundering risk evaluation method based on decision tree." 2007 International Conference on Machine Learning and Cybernetics. Vol. 1. IEEE, 2007.
8. Tang, Jun, and Jian Yin. "Developing an intelligent data discriminating system of anti-money laundering based on SVM." 2005 International Conference on Machine Learning and Cybernetics. Vol. 6. IEEE, 2005.
9. Lv, Lin-Tao, Na Ji, and Jiu-Long Zhang. "A RBF neural network model for anti-money laundering." 2008 International Conference on Wavelet Analysis and Pattern Recognition. Vol. 1. IEEE, 2008.
10. Awasthi, Abhishek. "Clustering algorithms for anti-money laundering using graph theory and social network analysis." (2012).
11. Chen, Yu-To, and Johan Mathe. "Fuzzy Computing Applications for Anti-Money Laundering and Distributed Storage System Load Monitoring." (2011).
12. Michalak, Krzysztof, and Jerzy Korczak. "Graph mining approach to suspicious transaction detection." Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on. IEEE, 2011.
13. Barse, Emilie Lundin, Hakan Kvarnstrom, and Erland Jonsson. "Synthesizing test data for fraud detection systems." Computer Security Applications Conference, 2003. Proceedings. 19th Annual. IEEE, 2003.
14. Lopez-Rojas, Edgar Alonso, and Stefan Axelsson. "Multi agent based simulation (mabs) of financial transactions for anti money laundering (aml)." Nordic Conference on Secure IT Systems. Blekinge Institute of Technology, 2012.
15. Salamon, Tomas. Design of agent-based models. Eva & Tomas Bruckner Publishing, 2011.
16. Huang, Jianbin, et al. "SHRINK: a structural clustering algorithm for detecting hierarchical communities in networks." Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, 2010.
17. Neo4j Graph Database, 'Neo4j, the World's Leading Graph Database', 2014. [Online]. Available: http://neo4j.com/. [Accessed: 22- Dec- 2015].