FlowScope: Spotting Money Laundering Based on Graphs

Xiangfeng Li,[1] Shenghua Liu,[2*#] Zifeng Li,[3] Xiaotian Han,[4]
Chuan Shi,[1 #] Bryan Hooi,[5] He Huang,[6] Xueqi Cheng[2*]
[1]Beijing University of Post and Telecommunication
[2]Institute of Computing Technology, Chinese Academy of Sciences
[3]University of Surrey, [4]Texas A&M University
[5]School of Computer Science, National University of Singapore
[6]China Citic Bank
{aplaceof, shichuan}@bupt.edu.cn, {liushenghua, cxq}@ict.ac.cn, zl00693@surrey.ac.uk, han@tamu.edu,
bryan.hooi.ky@gmail.com, huanghe@citicbank.com

Abstract

Given a graph of the money transfers between accounts of a bank, how can we detect money laundering? Money laundering refers to criminals using the bank's services to move massive amounts of illegal money to untraceable destination accounts, in order to inject their illegal money into the legitimate financial system. Existing graph fraud detection approaches focus on dense subgraph detection, without considering the fact that money laundering involves high-volume *flows* of funds through chains of bank accounts, thereby decreasing their detection accuracy. Instead, we propose to model the transactions using a multipartite graph, and detect the complete flow of money from source to destination using a scalable algorithm, FlowScope. Theoretical analysis shows that FlowScope provides guarantees in terms of the amount of money that fraudsters can transfer without being detected. FlowScope outperforms state-of-the-art baselines in accurately detecting the accounts involved in money laundering, in both injected and real-world data settings.

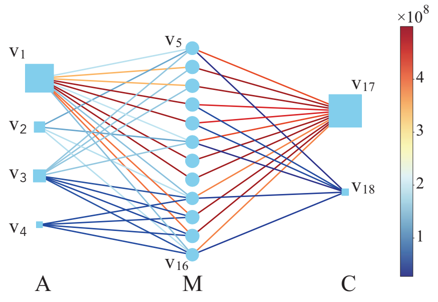


Figure 1: An example of money laundering transfers in a bank, creating a dense tripartite subgraph. The left accounts $\mathcal{A}$ are sources laundering money through the middle accounts $\mathcal{M}$ to the right destination accounts $\mathcal{C}$. Both $\mathcal{A}$ and $\mathcal{C}$ are outer accounts of the bank. Edge color and node size indicates the amount of money transferred.

Introduction

Given a big tripartite or multipartite graph, how can we spot the most suspicious dense flow? Such a flow can be an illegal money laundering (ML) process from sources to untraceable destinations through many adverserial middle accounts and transfers, or transportation flow shows interesting needs of a group of customers. Fig. 1 shows an example of ML within a subgraph, containing two-step flow from source to middle to destination accounts.

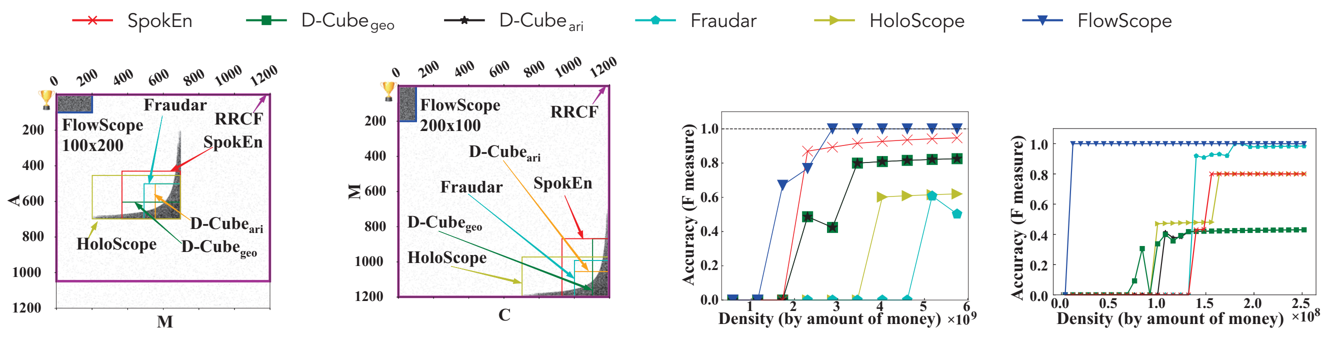
Most existing anti-money laundering algorithms (Wang and Yang 2007; Tang and Yin 2005; Lv, Ji, and Zhang 2008; Awasthi 2012; Paula et al. 2016) ignore the chain transfer scheme of money laundering behavior, and also ignore complex dependencies between transactions, leading to low detection accuracy and easy evasion by adversaries. Dense subgraph or subtensor detection algorithms (Liu, Hooi, and Faloutsos 2017; Shin et al. 2017; Prakash et al. 2010; Hooi et al. 2016) have been used for graph fraud detection, but also only consider single-step transfers. Although (Michalak and Korczak 2011) claims to handle chain transactions, it needs manually labeled data to train this model, whereas such data is rare, and the use of specific labels can make the model less robust for use in different datasets.

In contrast, our FlowScope in this paper considers multistep flows of funds: from senders, through middle accounts, to receivers. In the example based on synthetic data in Fig. 2, fraudulent flows consist of two steps: from $\mathcal{A}$ to $\mathcal{M}$ and $\mathcal{M}$ to $\mathcal{C}$, where the fraudulent accounts are plotted in the upper left of each adjacency matrix. FlowScope correctly detects the accounts involved in this two-step flow, while other algorithms detect parts of the two disconnected hyperbolic blocks, which resemble communities commonly observed in real graphs (Araujo et al. 2014).

Specifically, the FlowScope, proposed in this paper, is a flow-based approach for detecting money laundering behavior that detects chains of transactions. Due to highly skewed portion of ML accounts in real data, we propose an optimiza-

* CAS Key Laboratory of Network Data Science and Technology and University of Chinese Academy of Sciences, Beijing, China

# Corresponding authors



(a) Adjacency matrix $\mathcal{A}$-$\mathcal{M}$ (b) Adjacency matrix $\mathcal{M}$-$\mathcal{C}$ (c) CBank with ground-truth (d) Accuracy for multi-step flows

Figure 2: FlowScope performs the best in money laundering detection. In (a) and (b), FlowScope exactly catches the 2-step flow of money laundering transfers in synthetic data. The two matrices are bipartite adjacency matrices for transfers into ($\mathcal{A}$-$\mathcal{M}$), and out ($\mathcal{M}$-$\mathcal{C}$) of the bank's accounts. The top-left blocks are accounts involved in injected money laundering. The remaining edges are disconnected hyperbolic blocks which resemble normal communities in real graphs (Araujo et al. 2014). Our FlowScope detects exactly the injections. In real-world CBank data, FlowScope achieves the best performance, i.e. accurate and early detection (less money being laundered), with ground-truth label in (c), and injection of multi-step ML flow in (d).

tion problem to find suspicious accounts, instead of using a supervised learning model. Our main ideas are as follows: we model the problem using multipartite graphs, and define a novel anomalousness metric for transfer flows. High values of the metric indicate high-volume flows of funds which also do not leave much money left behind in middle accounts, through specific paths along the graph. In contrast, normal (i.e. honest) accounts do not always transfer money to specific accounts consistently, and also do not clear the middle accounts immediately , thus having lower values of the metric. FlowScope searches for fraudulent accounts by optimizing our metric, jointly optimizing over subsets of source, middle and destination accounts greedily. Moreover, we provide a theoretical guarantee on the near-optimality of our detection, and an upper bound on the amount of money a fraudster can transfer without being detected.

In summary, the main advantages of our work are:

- **Novel anomalousness metric for money-laundering flow:** We propose a novel anomalousness metric for dense and multi-step flow, and show its utility for detecting ML.

- **Theoretical guarantee:** We show the theoretical bound of near-optimality of our algorithm for detecting flow in graphs, as well as an upper bound on the amount of money that fraudsters can transfer, given limited resources of bank accounts.

- **Effectiveness and robustness:** FlowScope outperforms state-of-the-art baselines under various graph topologies, by accurately catching ML in an earlier stage (i.e. when less volume of illegal money is injected, see Fig. 3a). And FlowScope effectively detects adversarial ML when using more fraudulent accounts, and longer transfer chains (see Fig. 3c and 2d).

- **Scalability:** FlowScope is near-linear with the number of edges (i.e. transfers, see Fig. 5b), thus being well-suited to the rapid growth of the banking sector.

Our algorithm is reproducible, and open-sourced [1].

Related Work

The common approaches used for anti-ML are rule-based classification. (Rajput et al. 2014) presented an ontology-based expert system to detect suspicious transactions. (Khanuja and Adane 2014) monitored ongoing transactions and qualified the degree of anomaly by Dempster Shafer Theory. However, rule-based algorithms are easy to be evaded by fraudsters.

Machine learning algorithms are also applied for detecting ML activities. SVM was applied in (Tang and Yin 2005) to process large size of data and achieved higher accuracy. (Michalak and Korczak 2011) used fuzzy matching to catch subgraphs that may contain suspicious accounts. (Lv, Ji, and Zhang 2008) judged whether the capital flow is involved in ML activities using RBF neural networks calculating from time to time. And (Paula et al. 2016) also showed some success for using deep neural networks. However, these algorithms detect the ML activities in a supervised manner, suffering from highly skewed labels and limited adaptability. Additionally, ML detection is usually an adversarial task, and deep models may lack robustness under adversarial attacks. We propose to detect ML activities in an unsupervised manner based on graphs.

In many areas, data objects are inherently interrelated. Graphs provide a powerful mechanism to capture this association (Akoglu, Tong, and Koutra 2015). Many graph-based anomaly detection techniques have been developed for discovering structural anomalies. SpokEn (Prakash et al. 2010) studied patterns in eigenvectors, and was applied for anomaly detection in (Jiang et al. 2014) later. Fraudar (Hooi et al. 2016) proposed a suspiciousness metric which considers on the density of subgraph, and EigenPulse (Zhang et al. 2019) detects the dense subgraph in the streaming graphs.

[1] See code in https://github.com/aplaceof/FlowScope

D-Cube (Shin et al. 2017) considers the dense subtensor problem regarding all dimensions, and CatchCore (Feng, Liu, and Cheng) solves this problem in a Hierarchical way, while D-Spot (Yikun et al. 2019) catches high density on a subset of all dimensions in tensors. HoloScope (Liu, Hooi, and Faloutsos 2017) (Liu, Hooi, and Faloutsos 2018) proposed contrast suspiciousness, focused on the strange behavior which was far normal patterns. RRCF (Guha et al. 2016) tries to preserve pairwise distance, which is very useful for reducing false alarm. However, as these are general-purpose, they do not focus on flow across multiple nodes, which is important for accuracy and robustness against camouflage in the ML activities.

For ML detection, (Dreżewski, Sepielak, and Filipkowski 2015) and (Colladon and Remondi 2017) utilized social network analysis to reveal the underlying roles and organization structure. Clustering-based method (Le Khac and Kechadi 2010) detected ML activities by grouping transactions into clusters. However, those methods do not perform flow tracking, or provide theoretical guarantees.

Problem Formulation

To conceal funds, money launderers make fraudulent transfers from source accounts to destination accounts, through one or many layers of middle accounts. In general, the ML process involves high money flow passing through a bank or a series of banks, we use "bank" in the following to refer a bank or a collection of banks that ML detection is performed on. We then assume that ML has the following traits:

Trait 1 (dense transfers)**.** *Fraudsters create a high-volume and dense subgraph of transfers both into the bank and out of the bank.*

This is because the number of fraudulent accounts is limited, and a high volume of funds need to be transferred into the bank and out of the bank in a short span of time, resulting in a dense subgraph of transfers.

Trait 2 (zero out middle accounts)**.** *The middle accounts serve as a bridge: most of the received money will be transferred out, resulting in a balance between their weighted indegree and outdegree.*

This is because the money left in the middle accounts is subject to risk for fraudsters of being detected and frozen, particularly if the volume is high. Thus fraudsters leave as little money as possible in these accounts.

Algorithms which focus on individual transfers, e.g. feature-based approaches, can be easily evaded by adversaries by keeping each individual transfer realistic. Instead, for the sake of robustness against adversarial camouflage, we focus on *combinations* of high-volume transfers into the bank, internal transfers through several middle accounts, and out of the bank, which cannot be as easily hidden by the fraudsters, as follows:

Problem 1 (ML detection)**.** ***Given*** *a money transfer graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with accounts as nodes $\mathcal{V}$, and transfers as edges $\mathcal{E}$,*

Find: *a dense flow of money transfers (i.e. a subgraph of $\mathcal{G}$),*

Such that:
- *1) the flow involves high-volume money transfers into the bank, and out of the bank to the destinations;*
- *2) it maximizes density as defined in our ML metric.*

The design of the ML metric is very important, because, as we will show, it allows us to offer theoretical guarantees on the near-optimal detection of the dense flow, and an upper bound on the amount of money that fraudsters can transfer.

Table 1: Notations and symbols

Symbol	Interpretation
$\mathcal{W}$	Inner accounts of the bank
$\mathcal{X}, \mathcal{Y}$	Outer accounts mainly transferring money into or receiving money transfers out of the bank
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Tripartite graph of transfers in the bank
$\mathcal{V}$	Nodes of graph $\mathcal{G}$, i.e. $\mathcal{X} \cup \mathcal{W} \cup \mathcal{Y}$
$\mathcal{S}$	Node subset of graph $\mathcal{G}$, i.e. $\mathcal{A} \cup \mathcal{M}_1 \cup \cdots \cup \mathcal{M}_{k-2} \cup \mathcal{C}$
e_{ij}	Total amount of money on edge (i, j)
d_i, d_i^+, d_i^-	Degree (weight), out-degree, in-degree of node
$f_i(\mathcal{S})$, $q_i(\mathcal{S})$	Minimum and maximum of node's out-degree and in-degree, $\forall v_i \in \mathcal{M}_l$
w_i	Weight assigned to a node in priority tree
$\mathcal{S}^*$	Optimal subset of nodes maximizing metric
$\hat{\mathcal{S}}$	Subset returned by FlowScope
$\mathcal{S}'$	Subset just before removing node $v* \in \mathcal{S}^*$ in the next step by FlowScope
$g(\mathcal{S})$	Metric of ML anomalousness

In general, let graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph money transfers. Define $\mathcal{V} = \mathcal{X} \cup \mathcal{W} \cup \mathcal{Y}$, where $\mathcal{W}$ is the inner accounts of the bank, and $\mathcal{X}$ and $\mathcal{Y}$ are sets of outer accounts. $\mathcal{X}$ is the set of accounts that have net transfer of money into the bank, and $\mathcal{Y}$ is the set of accounts that have net transfer out of the bank. An edge $(i, j) \in \mathcal{E}$ indicates that account v_i transfers money into v_j, for $v_i, v_j \in \mathcal{V}$, and e_{ij} is the total amount of money transferred on the edge. Table 1 summarizes the main notations and symbols used in our paper.

Proposed metric

Intuitively, high-volume flows of funds among a small number of accounts become *dense subgraphs* of this multipartite graph. Fraudsters may transfer through multiple layers of middle accounts, making the internal transfers more realistic to evade detection.

We next define how to evaluate the anomalousness of a subgraph induced by subset of nodes. For example, money transfers may follow a k-partite subgraph induced by subset of nodes $\mathcal{S} = \mathcal{A} \cup \mathcal{M}_1 \cup \cdots \cup \mathcal{M}_{k-2} \cup \mathcal{C}$, where $\mathcal{A} \subseteq \mathcal{X}$, $\mathcal{M}_i \subseteq \mathcal{W}$, $\mathcal{C} \subseteq \mathcal{Y}$. Note that $\mathcal{M}_1, \cdots, \mathcal{M}_{k-2}$ can have overlapped middle nodes to consider cyclic transfers during ML as camouflage. In terms of detecting suspicious 3-partite subgraph induced by subset $\mathcal{S}$, we neglect the subscript of subset $\mathcal{M}_i$, thus $\mathcal{S} = \mathcal{A} \cup \mathcal{M} \cup \mathcal{C}$, where $\mathcal{M} \subseteq \mathcal{W}$.

Nevertheless, using a high number of middle layers and disguised cyclic transfers will definitely increase fraudsters' costs and risks in being caught by an audit. Therefore, for limited k, we then describe how we represent the original transfer graph $\mathcal{G}$ for detecting k-partite dense subgraphs.

We simply duplicate $\mathcal{W}$ by $k-2$ times between source nodes $\mathcal{X}$ and destination nodes $\mathcal{Y}$:

$$\mathcal{V}^k = \mathcal{X} \cup \underbrace{\mathcal{W} \cup \cdots \cup \mathcal{W}}_{k-2} \cup \mathcal{Y}$$

for convenient we define $\mathcal{W}_0 = \mathcal{X}$, $\mathcal{W}_{k-1} = \mathcal{Y}$. Next, our ML metric is defined as follows for spotting k-partite dense flow. First define e_{ij} as the total amount of money transfers from v_i to v_j, and define the total (weighted) out-degree and in-degree of node $v_i \in \mathcal{M}_l$, $l \in \{1, 2, \cdot, k-2\}$, w.r.t. node subset $\mathcal{S}$:

$$d_i^+(S) = \sum_{v_j \in \mathcal{M}_{l+1} \wedge (i,j) \in \mathcal{E}} e_{ij}$$

$$d_i^-(S) = \sum_{v_k \in \mathcal{M}_{l-1} \wedge (k,i) \in \mathcal{E}} e_{ki}$$

Next we define the minimum and maximum value between total weighted out-degree and in-degree of a middle account, w.r.t. node subset $\mathcal{S}$:

$$f_i(\mathcal{S}) = \min\{d_i^+(\mathcal{S}), d_i^-(\mathcal{S})\},\ \forall\, v_i \in \mathcal{M}_l \tag{1}$$

$$q_i(\mathcal{S}) = \max\{d_i^+(\mathcal{S}), d_i^-(\mathcal{S})\},\ \forall\, v_i \in \mathcal{M}_l \tag{2}$$

Next, our ML metric is defined as follows for spotting k-partite dense flow:

Definition 1 (Anomalousness of ML for k-partite subgraph)**.** *The anomalousness of a flow from a subset of nodes $\mathcal{A}$, through one layer or many layers of inner accounts $\mathcal{M}$, to another subset $\mathcal{C}$ is:*

$$g^k(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{l=1}^{k-2} \sum_{v_i \in \mathcal{M}_l} f_i(\mathcal{S}) - \lambda(q_i(\mathcal{S}) - f_i(\mathcal{S})) \tag{3}$$

$$= \frac{1}{|\mathcal{S}|} \sum_{l=1}^{k-2} \sum_{v_i \in \mathcal{M}_l} (1+\lambda) f_i(\mathcal{S}) - \lambda q_i(\mathcal{S}),\ k \geq 3 \tag{4}$$

Intuitively, $f_i(S)$ in Eq. (3) is the maximum possible flow that could go through middle account $v_i \in \mathcal{M}$, from accounts $\mathcal{A}$ to accounts $\mathcal{C}$. $q_i(S) - f_i(S)$ is the "remaining money" in v_i's balance after transfers, which can be penalty of ML, since fraudsters prefer to *zero out middle accounts* as Trait 2. The "remaining money" could be retention or deficit (i.e. received from others than $\mathcal{A}$ and $\mathcal{C}$) of middle accounts, occurring as a form of camouflage to evade detection.

The numerator in the objective (3) only uses the degree of middle accounts $\mathcal{M}_1, \cdots, \mathcal{M}_{k-2}$, since out-degree and in-degree of middle accounts are related to themselves and those of source nodes $\mathcal{A}$ and target nodes $\mathcal{C}$.

Interpretation of λ and our metric: We define λ as the imbalance cost rate, a constant coefficient which quantifies to what extent the fraudsters would suffer due to a unit of retention or deficit (cost of camouflage). Now our anomalousness metric $g(\mathcal{S})$ can be interpreted as the *potential profit*, i.e. earnings subtracted by cost, per account that the fraudulent accounts in subset $\mathcal{S}$ could make in a ML process.

Example 1 (Anomalousness of ML in a subgraph)**.** *Fig. 1 illustrates an example of ML from real-world bank. The sizes of nodes sets, $\mathcal{A}$, $\mathcal{M}$ and $\mathcal{C}$, are 4, 12, 2 separately. v_5 both receives and transfer out money of about 452.1 million Yuan, leaving $q_5(\mathcal{S}) - f_5(\mathcal{S}) \approx 0$ in balance. So there is basically no money is left over in v_5, or borrowed from outside or previous deposits. As we can see, our anomalousness metric captures the characteristics of real-world ML.*

As we can see, this example also agrees with Trait 1 and Trait 2 as mentioned before. The carefully designed metric also has the following advantages: a) scalable, and b) offers theoretical guarantees, which means a scalable algorithm can be designed for the metric, and the approximation algorithm provides theoretical bounds to optimality, as we will show later.

Moreover, the metric (4) measures the "density" not only by considering the topology density, but also based on the volume of money transferred i.e. weighted edges. Thus, even if high volumes of money are transferred along a sparse topology, those transfers will be very suspicious in our proposed metric.

Proposed algorithm: FlowScope

We propose a near-greedy algorithm, *FlowScope*, to find subset $\mathcal{S}$ that maximizes the objective $g(\mathcal{S})$ in (3). We first build a priority tree for nodes in $\mathcal{S}$. The weight (i.e. priority) assigned to node v_i is defined as:

$$w_i(\mathcal{S}) = \begin{cases} f_i(\mathcal{S}) - \dfrac{\lambda}{1+\lambda} q_i(\mathcal{S}), & \text{if } v_i \in \mathcal{M}_l \\ d_i(\mathcal{S}), & \text{if } v_i \in \mathcal{A} \cup \mathcal{C} \end{cases} \tag{5}$$

where $d_i(\mathcal{S})$ is the degree of node itself. Note that we can also add prior anomalousness as a weight to $w_i(\mathcal{S})$ for applications.

The algorithm is described in Alg 1. After building the priority tree, we perform the near greedy optimization: subset $\mathcal{S}$ starts at the whole node set; in every iteration we remove the node v in $\mathcal{S}$ with minimum weight in the tree, approximately maximizing objective (3); and then we update the weight of all its connected nodes. The iteration is repeated until one of node sets $\mathcal{A}, \mathcal{M}_1, \cdots, \mathcal{M}_{k-2}, \mathcal{C}$ is empty. Finally, the subset $\hat{\mathcal{S}}$ that we have seen with the largest value $g(\hat{\mathcal{S}})$ is returned. The complexity of FlowScope for k-partite flow is $O(k|\mathcal{E}|\log|\mathcal{V}|)$ [2].

In a real-world setting, we have no prior knowledge on how many middle layers are used by fraudsters. However, too many steps of transfers also raise the risks and costs. So it is possible that we give an upper bound K. To detect multi-step laundering, we then try every possible k up to at most K, and return the subset maximizing the follows:

$$\mathcal{S}^*, k^* = \arg\max_{S, k \leq K} \{g^k(\mathcal{S})\}$$

[2]See complexity analysis and all proofs in the supplement: https://github.com/aplaceof/FlowScope/blob/master/FlowScope-supplement.pdf

```
Algorithm 1: FlowScope
    Input: Graph $G = (\mathcal{V}, \mathcal{E})$
    Output: Node set of dense tripartite flow: $\hat{\mathcal{S}}$
1   $\mathcal{A} \leftarrow \mathcal{X}, \mathcal{M}_1 \leftarrow \mathcal{W}, \cdots, \mathcal{M}_{k-2} \leftarrow \mathcal{W}, \mathcal{C} \leftarrow \mathcal{Y}$
        // generate $k$-partite node subsets
        from $\mathcal{G}$
2   $\mathcal{S} \leftarrow \mathcal{A} \cup \mathcal{M}_1 \cup \cdots \cup \mathcal{M}_{k-2} \cup \mathcal{C}$
3   $w_i \leftarrow$ calculate node weight as Eq. (5)
4   $\mathcal{T} \leftarrow$ build priority tree for $\mathcal{S}$ with $w_i(\mathcal{S})$
5   while $\mathcal{A}, \mathcal{M}_1, \cdots, \mathcal{M}_{k-2}$ and $\mathcal{C}$ is not empty do
6       $v \leftarrow$ find the minimum weighted node in $\mathcal{T}$
7       $\mathcal{S} \leftarrow \mathcal{S} \setminus \{v\}$
8       update priorities in $\mathcal{T}$ for all neighbors of $v$
9       $g(\mathcal{S}) \leftarrow$ calculate as Eq. (3)
10  end
11  return $\hat{\mathcal{S}}$ that maximizes $g(\mathcal{S})$ seen during the loop.
```

In terms of detecting another dense subgraphs, we can simply remove the previous one, and rerun our algorithm. Moreover, the algorithm is near-greedy, yet provide a theoretical guarantee to approximate the optimal as we will show below.

Theoretical Analysis

For simplicity, we analyze FlowScope theoretically in case of three transfer layers. First of all, we show two lemmas derived from our algorithm before showing the theoretical bound [2] .

Let $\mathcal{S}^* = \mathcal{A}^* \cup \mathcal{M}^* \cup \mathcal{C}^*$ be the optimal subset. Then we have

Lemma 1. *$\forall v_i \in \mathcal{S}^*$, then $(1+\lambda)w_i(\mathcal{S}^*) \geq g(\mathcal{S}^*)$.*

Lemma 2. *$\forall v_i, \mathcal{S}'$ such that $v_i \in \mathcal{A}^* \cup \mathcal{C}^*$ and $\mathcal{S}^* \subseteq \mathcal{S}'$, then $w_i(\mathcal{S}') \geq w_i(\mathcal{S}^*)$;*
$\forall v_i, \mathcal{S}'$ such that $v_i \in \mathcal{M}^*$ and $\mathcal{S}^* \subseteq \mathcal{S}'$, then $w_i(\mathcal{S}') \geq w_i(\mathcal{S}^*) - \frac{\lambda}{\lambda+1}(q_i(\mathcal{S}') - q_i(\mathcal{S}^*))$.

Lemma 1 provides a lower bound of the weight of nodes in the optimal subset. Lemma 2 shows the relationship between the weight of a particular node measured in the graph over optimal set $\mathcal{S}^*$ and its weight in the graph over the larger superset $\mathcal{S}'$.

Now let $\mathcal{S}'$ be the subset just before *the first node $v_i \in \mathcal{S}^*$ was removed*, and $\hat{\mathcal{S}}$ be the subset returned by FlowScope. Thus, the theoretical bound that our FlowScope algorithm can achieve is:

Theorem 1 (Theoretical Bound)**.** *Given graph $\mathcal{G}$ and anomalousness metric of ML $g(\mathcal{S})$, the subset $\hat{\mathcal{S}}$ returned by FlowScope satisfies:*

$$g(\hat{\mathcal{S}}) \geq \frac{|\mathcal{M}'|}{|\mathcal{S}'|}(g(\mathcal{S}^*) - \lambda\varepsilon) \quad (6)$$

In formula (6), $\varepsilon = \max_{v_i \in \mathcal{S}^*}\{q_i(\mathcal{V}) - q_i(S^*)\}$ i.e. the highest volume of money transfers that a laundering account $v_i \in \mathcal{S}^*$ sends to or receives from other non-fraudulent accounts as possible camouflage. Note that the volume of camouflage ε usually cannot be as large as the volume of fraudulent transfers. Camouflage is much smaller, otherwise fraudsters take too much cost and risk for camouflage. Coefficient $\frac{|\mathcal{M}'|}{|\mathcal{S}'|} = (1 + \frac{|\mathcal{A}'|+|\mathcal{C}'|}{|\mathcal{M}'|})^{-1}$ cannot be arbitrarily small, because $|\mathcal{M}'| \geq |\mathcal{M}^*|$ as ML fraudsters use many middle accounts. Since $v_i \in \mathcal{S}^*$ at the same time is the next node removed from $\mathcal{S}'$ by FlowScope, then $\forall v_j \in \mathcal{A}' \cup \mathcal{C}'$ satisfies: $w_j(\mathcal{S}') \geq w_i(\mathcal{S}') \approx w_i(\mathcal{S}^*) \geq \min_{v_j \in \mathcal{S}^*}\{w_i(\mathcal{S}^*)\}$, assuming that a small amount or none of money transfers are used as camouflage. Hence such nodes are limited in a real transfer graph, and $|\mathcal{A}'|+|\mathcal{C}'|$ cannot be very large.

Next we show that the limited bound (6) holds:

Theorem 2 (Bounding Money Laundering)**.** *Suppose $\hat{S}$ is the subset returned by FlowScope. The maximum amount of money that may be laundered per account in our dataset without being detected is given by:*

$$\frac{\sum_{v_i \in \mathcal{S}^*} f_i(\mathcal{S}^*)}{n_0} \leq \frac{1}{1-\lambda\eta}\left(\frac{|S'|}{|M'|}g(\hat{S}) + \lambda\varepsilon\right)$$

where n_0 is the number of accounts used by fraudsters, ε denotes the maximum amount of transfer to non-fraudulent accounts as we expected in a particular account that gets involved in ML, and η denotes the deficit or retention, the criminals, if exist, are suffering as a percentage of dirty money laundered:

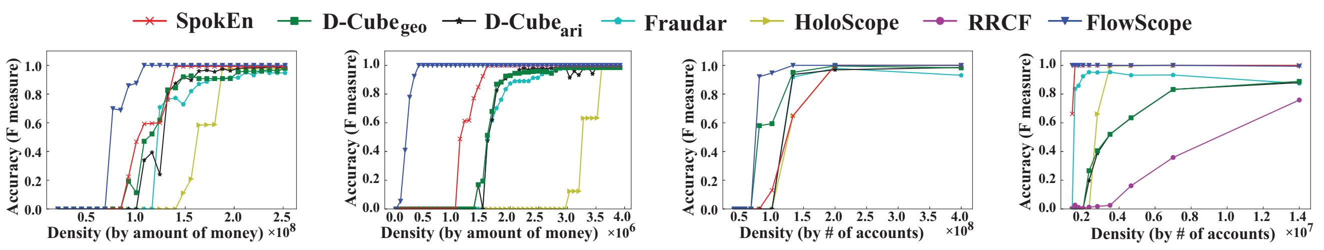
$$\eta = \frac{\sum_{v_i \in \mathcal{S}^*}(q_i(\mathcal{S}^*) - f_i(\mathcal{S}^*))}{\sum_{v_i \in \mathcal{S}^*} f_i(\mathcal{S}^*)}, \eta \in [0, \frac{1}{\lambda}]$$

Theorem 2 provides an estimation of the amount of money that could be laundered per account in our dataset without being detected, and gives the formulation as a function of the retention or deficit fraudsters can tolerate and the amount of transfer to other non-fraudulent accounts they have.

Experiments

The datasets used are summarized in Table 2, **CBank dataset:** Real-world transfer data from an anonymous bank under an NDA agreement, with a group of money laundering accounts being labeled, and the accounts opened in other banks are labeled. **Czech Financial Dataset (CFD):** An anonymous transfers of Czech bank released for Discovery Challenge in PKDD'99 (Lütkebohle).

To evaluate our FlowScope under various money laundering behaviors, we inject ML as follows: fraudulent accounts are randomly picked from $\mathcal{V}$ as the multipartite groups, e.g. $\mathcal{A}, \mathcal{M}$ and $\mathcal{C}$. Denote the total number of injected accounts as N. The edges between each group are randomly generated with probability p. The total amount of laundering money D, starting from $\mathcal{A}$, through $\mathcal{M}$, to $\mathcal{C}$, is assigned to generate edges proportional to Gaussian distribution ($mean = 10$ and $std = 1$). The balance of accounts in $\mathcal{M}$ is kept according to the total in-degree of each node. Camouflage edges are randomly connected to normal accounts, with amount of money for each edge uniformly generated from 100 to 1000. Note that we remove labeled fraudsters in CBank when performing injection experiments.



(a) CBank: descending money (b) CFD: descending money (c) CBank: ascending # of accounts (d) CFD: ascending # of accounts

Figure 3: FlowScope outperforms baselines robustly under different adversarial densities by descending amount of money or ascending # of accounts. In Fig. 3a-3b, we keep # of accounts as (7,5,3), and evaluate on different amount of money. FlowScope can detect earlier ML, i.e. the detection reaches *high accuracy* when fraudsters launder a *smaller* amount of money. In Fig. 3c- 3d, we keep amount of money as 6 billion and 210 million separately, and evaluate on different # of accounts with ratio 7:5:3. Other possible # of accounts and ratios are tested in our result table.

Datasets	#Nodes	#Edges	Time Span
CBank	6.13M	43.98M	Aug.07,2017 - Aug.13,2017
	0.99M,3.15M, 1.99M		
CFD	11.38K	273.51K	Jan.01,1993 - Dec.31,1998

Table 2: The statistics of the real-world datasets.

The evaluation of our FlowScope is organized according to the following four questions: effectiveness, real-world performance, robustness against longer transfer chains and scalability [3].

Effectiveness: one middle layer (Q1)

We first conduct an experiment on a synthetic dataset show in Fig. 2a-2b. The two adjacency matrices of transfer subgraphs are generated by SNAP network library (Leskovec and Faloutsos 2007). Hyperbolic and rectangular blocks have volume density around 0.8 and 0.7 respectively. Thus we can see that FlowScope detects exactly the fraudulent blocks, while other baselines return parts of two disconnected hyperbolic blocks, which is thought to be normal communities (Araujo et al. 2014).

In the real scenario, the crimes can make arbitrary of topology with the respect of $\mathcal{A}$, $\mathcal{M}$ and $\mathcal{C}$ accounts to deceive the anti-ML system. So fraudsters are injected with a lot of variants to check the robustness of the algorithm in this experiment [4].

The influence of the amount of money: In this experiment, given sizes of $\mathcal{A}, \mathcal{M}, \mathcal{C}$ and edge probabilities fixed, we increase the amount of injected money laundered step by step while fixing the other conditions. The results were shown in Fig. 3a-3b, in both cases, FlowScope detects the ML behavior early and accurately, and the methods based on bipartite graph are unable to catch suspicious tripartite dense flow in the graph. RRCF (Guha et al. 2016) gives the lowest performance on the injected experiment, as it generally catches the largest amount of money transferred naturally within the bank. HoloScope is also overly influenced by such large amount of transfers due to its definition of the 'contrast suspiciousness' measure.

The influence of the number of fraudulent accounts: Another possible case is that the money launderers try to employ as many as people possible to disperse the dirty money, making the ML behavior much harder to detect. In this experiment we increase the number of fraudsters while keeping other conditions unchanged. The value of the amount of injected money D over the total number of fraudsters N, i.e. density(by the number of accounts) is used as horizontal axis in Fig. 3c-3d, which illustrate FlowScope's effectiveness and robustness against large ML groups.

Summary comparisons in table: We summarize the comparison results in transfer data of CBank and CFD in Table 3. We use a slash '/' to separate results from two kinds of injections in the table for saving space: 1) Varying amount of money injected: injection of different volumes of money, from 300 million to 3 billion, given fixed number of accounts in each layer. For example, in the row of 7:5:3, we inject money laundering through 7, 5, and 3 accounts in the respective layers; 2) Varying # of accounts injected: injection of different # of accounts given a fixed volume of money, e.g. in the row of 7:5:3, we inject (7, 5, 3) fraudulent accounts in the respective layers, then (14, 10, 6), and so on.

Then two metrics are evaluated for comparison. **FAUC**: the areas under curve of F-measure as in Fig. 3. We normalize the density in horizontal axis to scale FAUC between 0 and 1. Higher FAUC indicates better performance. **'F1 $\geq$ 0.9'**: the lower bound of money volume (in million \$) or upper bound of account size such that the F-measure curve stays above or equal to 0.9, as low volume of money and large size of accounts can reduce detection accuracy.

Thus we can see from the table, FlowScope achieves the best results at most of the settings, indicating earlier and more accurate detection for more fraudulent accounts than

[3]RRCF is only compared in a smaller size of CFD dataset, because of the efficiency problem.

[4]We abbrev ($|\mathcal{A}|$,$|\mathcal{M}|$, $|\mathcal{C}|$) = (a,m,c) as sizes=(a,m,c) and ($|\mathcal{A}|$:$|\mathcal{M}|$: $|\mathcal{C}|$) = (a:m:c) as ratio=(a:m:c).

Dataset	metrics*	A:M:C	D-Cube$_{ari}$	D-Cube$_{geo}$	Fraudar	HoloScope	SpokEn	RRCF	FlowScope
CBank	FAUC	5:9:1	0.417 / 0.600	0.591 / 0.810	0.347 / 0.634	0.276 / 0.466	0.610 / 0.753	- / -	**0.633** / **0.800**
		5:5:5	0.502 / 0.658	0.501 / 0.709	0.467 / 0.683	0.379 / 0.655	0.598 / 0.708	- / -	**0.757** / **0.843**
		7:5:3	0.533 / 0.727	0.522 / 0.779	0.529 / 0.704	0.377 / 0.547	0.633 / 0.708	- / -	**0.761** / **0.843**
	$F1 \geq 0.9$ (million \$7 node size)	5:9:1	190 / 30	- / 45	- / 30	210 / 15	154 / 30	- / -	**132** / **75**
		5:5:5	150 / 45	- / 45	- / 42	- / 30	116 / 45	- / -	**84.0** / **90**
		7:5:3	175 / 30	166 / 54	180 / 33	- / 15	122 / 30	- / -	**76.0** / **90**
CFD	FAUC	5:9:1	0.498 / 0.577	0.528 / 0.577	0.409 / 0.770	0.125 / 0.773	0.716 / **0.894**	0.253 / 0.538	**0.939** / 0.877
		5:5:5	0.565 / 0.633	0.592 / 0.736	0.549 / 0.867	0.143 / 0.810	0.716 / 0.897	0.236 / 0.364	**0.962** / **0.900**
		7:5:3	0.580 / 0.734	0.520 / 0.725	0.593 / 0.826	0.0356 / 0.818	0.728 / 0.898	0.213 / 0.434	**0.970** / **0.900**
	$F1 \geq 0.9$ (million \$7 node size)	5:9:1	2.21 / 15	2.19 / 15	- / 60	3.52 / 60	1.71 / 120	- / 15	**0.400** / **150**
		5:5:5	1.87 / 30	2.05 / 30	- / 150	- / 75	1.23 / **150**	- / 15	**0.240** / **150**
		7:5:3	- / 30	- / 30	- / 120	- / 60	1.46 / 135	- / 15	**0.240** / **150**

Table 3: Experimental results. The results of injected different volumes of money and # of accounts are reported in one cell separated by '/'. And '-' in the table means never returning a possible solution either in limited time or injection density.

baselines, which has better results when the ratio is 7:5:3.

Real-world performance (Q2)

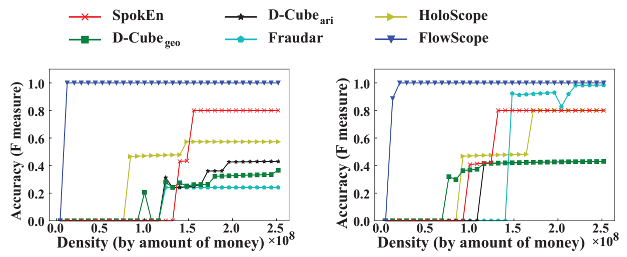
Our CBank dataset contains labeled ML activity: based on the $\mathcal{A}$ to $\mathcal{M}$ to $\mathcal{C}$ ML schema, the number of each type of accounts is 4, 12, 2 as shown in Fig. 1. We test how accurately and early, we can detect the fraudsters in CBank dataset. To do this, we first scale down the percentage of dirty money laundered from source accounts to destination accounts, then gradually increase the volume of money laundering linearly back to the actual value in the dataset. As shown in Fig. 2c, our FlowScope is the earliest to catch the ML behaviors accurately, as the total volume of dirty money is roughly 10% lower than other baselines given the same accuracy, and is the only algorithm catching the exact ML fraudsters.

Robustness against longer transfer chains (Q3)

Further, we investigate the case where the criminals transfer the dirty money using two levels of middle accounts, which is much more difficult to detect. The injection is made in the same way for the two groups of middle accounts. As illustrated in Fig. 2d (setting sizes=(3,7,7,3) and p=0.6 in CBank data) and Fig. 4a-4b, for different sizes of the transfer groups. The results show that even when ML has longer transfer chains, our algorithm by far outperforms the baselines, indicating FlowScope's ability to catch suspicious multi-step flow of money transfers.

Sensitivity and Scalability (Q4)

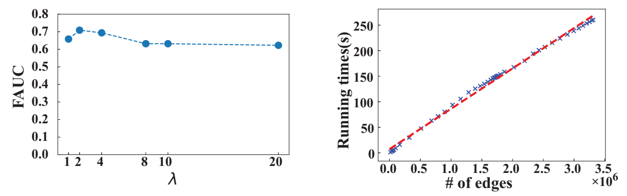
Sensitivity: FlowScope is not sensitive to constant coefficient λ within reasonable range (see Fig. 5a). We hence set λ to 4.0 for all our experiments as we assume $1/\lambda$= 0.25 is the highest rate of retention or deficit as camouflage that fraudsters can afford. We conduct experiments on CBank with ground-truth labels with a series of λ fixing the other conditions. FAUC is measured for each λ. As illustrated in Fig. 5a, FlowScope is not sensitive to λ within a reasonable range.
Scalability: We use the CBank dataset by accumulating time by hours, to get increasing-size data. As Fig. 5b shows, FlowScope is nearly linear with the number of edges.



(a) CBank: Transfer chains of (5,5,5,5)

(b) CBank: Transfer chains of (2,8,8,2)

Figure 4: FlowScope is robust against longer transfer chains.



(a) FlowScope is stable for a reasonable range of λ.

(b) FlowScope scales near linearly

Figure 5: Model analysis of FlowScope.

Conclusion

In this paper, we model the money laundering problem as detecting the densest multi-step flow, and define a novel density metric. An algorithm FlowScope is proposed for searching dense flow in big transaction graphs from banks accurately yet efficiently. We propose a novel anomalousness metric for dense multipartite flow. FlowScope offers theoretical guarantees on the near optimal detection of the dense flow. Experiments demonstrate the effectness of FlowScope's utility in detecting money laundering, as it outperforms state-of-the-art baselines under various experiment settings. Meanwhile, FlowScope is near-linear with the number of transactions. The code is open-sourced for reproducibility.

Acknowledgments

This material is based upon work supported by the Strategic Priority Research Program of CAS (No. XDA19020400), the National Natural Science Foundation of China (No. 61772082), the Beijing Municipal Natural Science Foundation (4182043), the NFS of China (No. 91746301, 61772498, 61872206), and the Beijing NSF (No. 4172059), and the Tencent Open Fund.

References

Akoglu, L.; Tong, H.; and Koutra, D. 2015. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery* 29(3):626–688.

Araujo, M.; Günnemann, S.; Mateos, G.; and Faloutsos, C. 2014. Beyond blocks: Hyperbolic community detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 50–65. Springer.

Awasthi, A. 2012. Clustering algorithms for anti-money laundering using graph theory and social network analysis.

Colladon, A. F., and Remondi, E. 2017. Using social network analysis to prevent money laundering. *Expert Systems with Applications* 67:49–58.

Dreżewski, R.; Sepielak, J.; and Filipkowski, W. 2015. The application of social network analysis algorithms in a system supporting money laundering detection. *Information Sciences* 295:18–32.

Feng, W.; Liu, S.; and Cheng, X. Catchcore: Catching hierarchical dense subtensor.

Guha, S.; Mishra, N.; Roy, G.; and Schrijvers, O. 2016. Robust random conferenceut forest based anomaly detection on streams. In *International conference on machine learning*, 2712–2721.

Hooi, B.; Song, H. A.; Beutel, A.; Shah, N.; Shin, K.; and Faloutsos, C. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 895–904. ACM.

Jiang, M.; Cui, P.; Beutel, A.; Faloutsos, C.; and Yang, S. 2014. Inferring strange behavior from connectivity pattern in social networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 126–138. Springer.

Khanuja, H. K., and Adane, D. S. 2014. Forensic analysis for monitoring database transactions. In *International Symposium on Security in Computing and Communication*, 201–210. Springer.

Le Khac, N. A., and Kechadi, M.-T. 2010. Application of data mining for anti-money laundering detection: A case study. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, 577–584. IEEE.

Leskovec, J., and Faloutsos, C. 2007. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th international conference on Machine learning*, 497–504. ACM.

Liu, S.; Hooi, B.; and Faloutsos, C. 2017. Holoscope: Topology-and-spike aware fraud detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1539–1548. ACM.

Liu, S.; Hooi, B.; and Faloutsos, C. 2018. A contrast metric for fraud detection in rich graphs. *IEEE Transactions on Knowledge and Data Engineering*.

Lütkebohle, I. BWorld Robot Control Software. https://data.world/lpetrocelli/czech-financial-dataset-real-anonymized-transactions/. [Online; accessed 2-November-2018].

Lv, L.-T.; Ji, N.; and Zhang, J.-L. 2008. A rbf neural network model for anti-money laundering. In *Wavelet Analysis and Pattern Recognition, 2008. ICWAPR'08. International Conference on*, volume 1, 209–215. IEEE.

Michalak, K., and Korczak, J. 2011. Graph mining approach to suspicious transaction detection. In *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, 69–75. IEEE.

Paula, E. L.; Ladeira, M.; Carvalho, R. N.; and Marzagao, T. 2016. Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 954–960. IEEE.

Prakash, B. A.; Sridharan, A.; Seshadri, M.; Machiraju, S.; and Faloutsos, C. 2010. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 435–448. Springer.

Rajput, Q.; Khan, N. S.; Larik, A.; and Haider, S. 2014. Ontology based expert-system for suspicious transactions detection. *Computer and Information Science* 7(1):103.

Shin, K.; Hooi, B.; Kim, J.; and Faloutsos, C. 2017. D-cube: Dense-block detection in terabyte-scale tensors. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 681–689. ACM.

Tang, J., and Yin, J. 2005. Developing an intelligent data discriminating system of anti-money laundering based on svm. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 6, 3453–3457. IEEE.

Wang, S.-N., and Yang, J.-G. 2007. A money laundering risk evaluation method based on decision tree. In *Machine Learning and Cybernetics, 2007 International Conference on*, volume 1, 283–286. IEEE.

Yikun, B.; Xin, L.; Ling, H.; Yitao, D.; Xue, L.; and Wei, X. 2019. No place to hide: Catching fraudulent entities in tensors. In *The World Wide Web Conference*, 83–93. ACM.

Zhang, J.; Liu, S.; Yu, W.; Feng, W.; and Cheng, X. 2019. Eigenpulse: Detecting surges in large streaming graphs with row augmentation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 501–513. Springer.