MonLAD: Money Laundering Agents Detection in Transaction Streams*

Xiaobing Sun1,2,†, Wenjie Feng3,†, Shenghua Liu1,2,†, Yuyang Xie3, Siddharth Bhatia4, Bryan Hooi4, Wenhan Wang6, Xueqi Cheng1,2

1 Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
2 University of Chinese Academy of Sciences, Beijing, China
3 Institute of Data Science, National University of Singapore
4 National University of Singapore, Singapore
5 Tsinghua University, Beijing, China
6 Tencent Technology Co. Ltd, Shenzhen, China

sunxiaobing21@mails.ucas.ac.cn, wenchiejfeng.us@gmail.com, liushenghua@ict.ac.cn

ABSTRACT
Given a stream of money transactions between accounts in a bank, how can we accurately detect money laundering agent accounts and suspected behaviors in real-time? Money laundering agents try to hide the origin of illegally obtained money by dispersing multiple small transactions and evade detection by smart strategies. Therefore, it is challenging to accurately catch such fraudsters in an unsupervised manner. Existing approaches do not consider the characteristics of those agent accounts and are not suitable to the streaming settings. Therefore, we propose MonLAD and MonLAD-W to detect money laundering agent accounts in a transaction stream by keeping track of their residuals and other features; we devise AnoScore algorithm to find anomalies based on the robust measure of statistical deviation. Experimental results show that MonLAD outperforms the state-of-the-art baselines on real-world data and finds various suspicious behavior patterns of money laundering. Additionally, several detected suspected accounts have been manually-verified as agents in real money laundering scenario.

CCS CONCEPTS
• Information systems → Electronic commerce; • Theory of computation → Streaming, sublinear and near linear time algorithms; • General and reference → Experimentation.

KEYWORDS
Anomaly detection, Money laundering, Stream algorithm.

ACM Reference Format:
Xiaobing Sun1,2,†, Wenjie Feng3,†, Shenghua Liu1,2,†, Yuyang Xie3, Siddharth Bhatia4, Bryan Hooi4, Wenhan Wang6, Xueqi Cheng1,2. 2022. MonLAD: Money Laundering Agents Detection in Transaction Streams. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM ’22), February 21–25, 2022, Tempe, AZ, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498418

† Both authors contributed equally to the paper. Corresponding author: S. Liu.

1 INTRODUCTION
Money laundering is the process aiming at hiding the origin of illegally obtained money. The estimated amount of money laundered per year is 2–5% of the global GDP, or $800 billion - $2 trillion US dollars and even a lower estimate is underlining the severity of the issue [42]. Source money could come from illegal dealing of commodities, drug trafficking, smuggling, and other criminal activities. Once connected with organized crime and terrorist financing, this will extensively damage the reputation of financial institutions and threaten public security [22].

Thus, it raises the following question: given a stream of transactions for money transfers between accounts, how can we effectively detect money laundering agent accounts in real-time? Money laundering is particularly hard to detect because fraudsters intentionally evade detection via innovative mechanisms, e.g., by funnelling money through multiple accounts, mixing them with legitimate transactions, and making small transfers that fall just below the reporting thresholds [14]. Quick and accurate detection of such suspected accounts and behavior patterns in real-time transactions is the main challenge.

To transfer large amounts of money while keeping each transaction below a “safe” level, the agent accounts must make frequent incoming connections from the source accounts (or agents) and immediate or periodic outgoing connections to the target accounts. These agent accounts exhibit suspiciously fast incoming and outgoing transfer behaviors. Fig. 1 depicts an example of money laundering agents about their transfer structures and behavior series.

Most existing money laundering detection approaches [15, 17, 19, 24, 29, 37] are designed for static records and cannot adapt to streaming scenario. Methods that detect anomalies based on dynamic graphs [3, 7, 8, 34] or outlier detection [4, 9, 28, 40, 41] do not consider the characteristics of money-laundering behavior, resulting in the inferior detection accuracy or not applicable.

In this paper, we propose MonLAD, a scalable sketching algorithm for depicting the behavior of money laundering agent accounts in a transaction stream; it keeps track of the account residual and computes some key statistical features to summarize their behavior; MonLAD-W accurately fits complex patterns in a sliding window manner. We devise AnoScore algorithm to detect anomalies based on the robust statistical deviation, which is theoretically founded and explainable. Experiments on the real-world data show that our method can fast and accurately detect various patterns for money laundering (see Fig. 2(a)-(2(b)) and find many suspicious agent accounts (e.g., Fig. 1), some of which are manually verified.
Figure 1: Money laundering agent behavior pattern example. (a) the connections of incoming and outgoing transfers of a detected suspicious agent (red node, w.r.t. $p_3$ in Fig. 2(a)). (b) the residual series over time of the suspicious agent account, where it shows many suspicious periodical and intensive spikes for receiving money and transferring all out at once. The top subfigure zooms into part of the time.

We make all of our algorithm and experimental code and the majority of the data for the experiments available.

**Summary of Contributions.**

- **Streaming algorithm:** We propose a novel approach MonLAD to detect money laundering agent accounts in a transaction stream, it is able to fast answer the detection query at any time based on the statical features (see Figure 2(a)).
- **Explainable and Flexible:** The proposed AnoSCORE utilizes statistical deviations as the anomaly score instead of a “black box”. Our method can easily adapt to and incorporate different scores from other outlier detection methods.
- **Effectiveness:** Our method outperforms the state-of-the-art baselines on the real-world bank data, it detects various suspicious behavior patterns (e.g., Figure 1), including manually verified fraudulent accounts and periodical patterns.
- **Scalability:** MonLAD is scalable, with linear time complexity in the number of edges of the stream (see Figure 2(c)).

## 2 RELATED WORK

In this section, we review relevant approaches about anomaly and outlier detection, money laundering pattern detection.

**Supervised learning methods:** To involve more attributes and handle high-dimensional data, machine learning models such as SVM [39], decision trees [43], and neural networks [20] are applied to money laundering. [38] uses representation learning techniques to utilize the information contained in graphs. [32] combines network analysis to detect the groups of money laundering activities. Although these algorithms detect money laundering activities in supervised or semi-supervised manners, they suffer from imbalanced class and lacking of adaptability. We focus on detecting money laundering activities in an unsupervised fashion.

**Graph-based methods:** Given as input a stream of edges over time, GOUTLIER [1] scores the likelihood of each edge in the stream based on a structural reservoir sample of edges. SedanSpot [7] and [27] measure edge anomalousness in the stream based on its prior occurrence, preferential attachment and mutual neighbors (homophily). Spotlight [8] detects sudden appearance of many unexpected edges. [23] applies only when multiple graphs with typed nodes and edges evolve simultaneously. Midas [3] identifies micro-cluster based anomalies or suddenly arriving groups of suspiciously similar edges. Nevertheless, those streaming methods do not take into account the characteristics of money laundering. [19] constructed a classifier based on a set of mined rules to detect suspicious transactions in a data stream. However, these traditional rule based algorithms highly rely on domain knowledge and also are easy to be evaded by fraudsters. FlowScope [17] and AutoAudit [16] detect the flow of money laundering in a multipartite graph, however, they cannot handle streaming edges and fail to capture a variety of different behavior patterns well.

Table 1 compares MonLAD and other related methods for the problem of money laundering pattern detection in streams.

### 3 PROBLEM FORMULATION

Here, we summarize three key traits of typical money laundering transfers and then formally define our problem. We use “fan-in” and “fan-out” to refer to the money transfer into and out of an account (especially an agent in intermediate accounts $M$) as a metaphor for fan-like incoming / outgoing connections in social networks.

#### 3.1 Key Traits

**Trait 1 (Fast Fan-in and Fan-out).** “Dirty money” will be divided into multiple parts and transferred from sources to destinations; these small-amount but multiple transfers flowing through $M$ are usually completed within a short period.

Thus, by controlling the amount per transfer and the number of transactions, fraudsters evade detection by manipulating a large amount of money. The sooner these transactions complete, the more the fraudsters gain and lower the risk.

**Trait 2 (Frequently Balancing).** Agent accounts crave to frequently reach a balanced state by fan-outs (transfer out to other agents or targets), upon receiving all the money or reaching a volume threshold of multiple fan-ins from the sources.

The remaining money in the agent accounts will be subject to the risk of being detected or frozen, particularly for large amounts. Therefore, fraudsters tend to remain for as little time as possible unless they can empty the account. The remaining amount is then also be used as camouflage by some smart fraudsters.

**Trait 3 (High Throughput).** To process a large amount of money in a short period with a limited number of agent accounts in

| MonLAD | FlowScope [17] | Midas-R [3] | DenseAlert [34] | Spotlight [8] | SedanSpot [7] | AutoAudit [16] |
|--------|---------------|-------------|----------------|--------------|--------------|---------------|
| Money laundering | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| Streaming pattern | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| Statistical deviation | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| Real-valued feature | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |

Table 1: Comparison of MonLAD and relevant approaches.

[1]https://github.com/BGT-M/MonLAD.
where criminal-icon labels manually-veriﬁed accounts involved in money laundering and other outliers also exhibit obvious suspicious behavior pattern. MonLAD outperforms the baselines in two different settings with empty and ﬁlled symbols for each method. (c) MonLAD runs in time linear to the number of edges in a stream.

Figure 2: Performance of proposed MonLAD on real data (CBank). (a) Heatmap of the statistical features used in MonLAD, (b) Comparison of speed and accuracy (F1 score). (c) MonLAD scales linearly with # of edges.

4 PROPOSED METHOD

In this section, we ﬁrst deﬁne some key statistics to recognize the patterns of agent accounts, then propose MonLAD to sketch the

\[ \frac{(F - B)^T}{B^T} \]

\[ B^* + (Ϝ^* - B^*) + \]

\[ \text{Part I} \]
\[ \text{Part II} \]
\[ \text{Part III} \]

\[ p_2 \]
\[ p_3 \]
\[ p_1 \]

\[ \text{Table 2: Symbols and Deﬁnitions.} \]

| Symbol | Deﬁnition |
|--------|-----------|
| \( s_u^t \) | Balanced state of the account \( u \) at time \( t \) |
| \( R_u(t) \) | Residual of the account \( u \) until time \( t \) |
| \( B_u(t) \) | the number of times of balance that the account \( u \) achieves until time \( t \) |
| \( F_u(t) \) | the total number of effective fan-ins of \( u \) within balance until time \( t \) |
| \( \delta_{up}, \delta_{down} \) | minimum thresholds for an effective fan-in and fan-out |
| \( \min_u^t, \max_u^t \) | minimum and maximum residual of \( u \) at time \( t \) during reaching balance |

\[ d_u^t(t) = \sum_{u \in V} \sum_{t_i \in (u,u,u,t_i)} w_{u,u}, d_u^0(t) = \sum_{u \in V} \sum_{t_i \in (u,u,u,t_i)} w_{u,u}. \]

**Residual:** We deﬁne the residual as the difference between the weighted in-degree and out-degree of the account \( u \) until time \( t \),

\[ R_u(t) = d_u^t(t) - d_u^0(t) \]

\[ (1) \]

Note that the residual might also be negative, since the account state (initial residual) is unknown at \( t_0 \). \( R_u(t) \) only indicates the residual after the ﬁrst transaction of the account \( u \) occurred during the observation period. Thus, we deﬁne two key concepts as follows:

**Deﬁnition 1 (Reaching Balanced State).** An account \( u \) reaches a balanced state at \( t \) (denoted as \( s_u^t = 0 \)) after a fan-out transfer, 2They are equal to the total transfer-in and transfer-out amount until \( t \) respectively.
leading its residual to being no more than the minimum of the residual between the most recent time \( t \) and \( t_{11} \), where \( t = 0 \) or \( s_u^t = 0 \), while not balanced at \( t - 1 \) (i.e., \( s_u^{t-1} = 1 \)).

An account reaches a balanced state when the money transferred into it is finally transferred out. So we keep track of the minimum of the residual to find those balanced states. Let \( B_u(t) \) be the number of times of balanced state that \( u \in V \) reaches until time \( t \).

Definition 2 (Effective Fan-in). A money transfer into an account \( u \) is called an effective fan-in if and only if

- the accumulated money transferred into \( u \) (the total fan-in) reaches a threshold \(^3\) since the last balanced state;
- all accumulated money is finally transferred out to reach the next balanced state.

In money laundering scenario, we consider the number of incoming transfers after an effective fan-in (inclusive) until reaching a balanced state. We use \( f_u(t) \) to denote the number of effective fan-ins of the account \( u \) achieves at time \( t \) since the last balanced state (\( u \) may (re-) reach a balanced state depending on the residual afterward), and \( F_u(t) \) as the accumulative \( f_u(t) \) from the initial to the last balanced state before \( t \).

Let \( \delta_{up} \) be the minimum threshold for an effective fan-in and \( \delta_{down} \) be the minimum threshold for an fan-out to be a balanced state. Consider a certain balanced state of the account \( u \) from \( t_0 \) (the initial time or the end of the previous balance) to \( t_k \), for any \( t \in \{ t_0, \ldots, t_k \} \), \( \max_u^t = \max_u^{t_1} \) denotes the maximum (minimum) residual during the process, which is traced by \( s_u^t \), i.e. \( s_u^t = 0 \) if \( t \in \{ t_0, t_k \} \) otherwise \( s_u^t = 1 \).

Example 1 illustrates an explanation based on Figure 3, which shows the change the residual \( R_u(t) \) of \( u \) in \([t_1,t_{11}]\). Note that the residual becomes negative at \( t \in \{ t_5, t_7, t_8, t_{11} \} \) due to the unknown \( R_u(t_0) \), while it must hold that \( R_u(t_0) \geq \max(|R_u(t_5)|,|R_u(t_{11})|) \).

**Example 1 (Concept Example). The changes of residual \( R_u(t) \) in Fig. 3 come from 10 transfers, i.e., 4 fan-outs (at \( t_5, t_7, t_8, \) and \( t_{11} \)) and 6 fan-ins (at other times); it reaches balanced state at \( t_5 \) and \( t_{11} \) according to the definition, i.e., \( s_u^{t_5} = s_u^{t_{11}} = 0 \). The residual achieves a new minimum at \( t_5 \) (i.e. \( \min_u^{t_5} \)) since \( t_1 \). Assuming \( \delta_{up} = \delta_{down} = 20 \), \( f_u(t_4) = 1 \) due to the effective fan-in at \( t_4 \); \( F_u(t_5) = 1 \) and \( B_u(t_5) = 1 \) since \( u \) reaches balanced state only once from \( t_1 \) to \( t_5 \); \( f_u(t_0) = 4 \) since the effective fan-in at \( t_0 \) and there are 3 fan-ins after \( t_0 \); \( F_u(t_1) = F_u(t_2) = 1 \) and \( B_u(t_1) = 5 \) and \( B_u(t_1) = 2 \) due to the second balanced state at \( t_{11} \). In addition, \( \max_u^{t_1} = \max_u^{t_5} = R_u(t_5); \min_u^{t_1} = \min_u^{t_5} = R_u(t_1) \). We can get other relevant values (states) after \( t_5 \) similarly.

\(^3\)Here ‘1’ means waiting for balanced state and ‘0’ refers to the start (end) of balance.

Note that \( F_u(t) \) always holds since being in the balanced state contains at least a fan-in (including effective fan-in) at any time. Regarding the money laundering scenario, some suspicious behavior patterns corresponding to different types of anomalies \([30]\), including transfers in illegal exchange, illegal gambling, drug trafficking, etc. It can be derived as follows:

- **P1.** \( F_u(t) = B_u(t) \): one-time fan-in and immediate fan-out;
- **P2.** \( F_u(t) \geq B_u(t) \) & a small \( B_u(t) \): dispersive fan-ins and few fan-outs.
- **P3.** \( F_u(t) > B_u(t) \) & a large \( B_u(t) \): dispersive fan-ins and fan-outs.

### 4.2 Our MonLAD Algorithm

\( B_u(t) \) and \( F_u(t) \) are core features to describe the behavior of the account \( u \) in a transfer stream; the other statistics, including \( f_u(t), s_u^{t_0}, \min_u^t, \) and \( \max_u^t \), are auxiliaries; they will constantly be updated as the transaction arrives. So, we design the following rules to update and count. Alg. 1 gives the high-level pseudo-code of our method MonLAD.

When an edge \( e = (u,v,w_{u,v},t) \) arrives, we need to determine whether the source vertex \( u \) reaches a balanced state due to the fan-out and whether the target vertex \( v \) is waiting to reach a balanced state or will start a new process just after the previous balance. The states \( s_u^t \) and \( s_v^t \) will be updated according to Eq. (2) and Eq. (3) respectively, where the contribution of \( w_{u,v} \) to the residuals \( R_u(t) \) and \( R_v(t) \) has been counted by updating \( d_u^{t_0}(t), d_u^{t_1}(t) \).

\[
\begin{align*}
s_u^t = \begin{cases} 0 & \text{if } \max_u^{t-1} - R_u(t) > \delta_{down} & R_u(t) \leq \min_u^{t-1} + \epsilon, \\
\epsilon & \text{otherwise.}
\end{cases}
\end{align*}
\]

This means that \( s_u^t \) will be in a balanced state if a large enough fan-out \( \epsilon \) almost clears the residual \( R_u(t) \) (close to \( \min_u^{t-1} \)), otherwise the previous state holds. Here, we introduce a small residual \( \epsilon > 0 \) to nullify the fraudsters’ attempts to evade detection by keeping a low balance. To be more precise, \( \epsilon \) is the minimum cost of fraudsters and also the maximum tolerance of the detector.

\[
\begin{align*}
s_v^t = \begin{cases} 1 & \text{if } R_v(t) - \min_v^{t-1} > \delta_{up}, \\
\epsilon & \text{otherwise.}
\end{cases}
\end{align*}
\]

that is, \( s_v^t \) is waiting to reach a balanced state if the residual \( R_v(t) \) is still at least \( \delta_{up} \) greater than the previous minimum \( \min_v^{t-1} \), otherwise the previous state holds.

For the target vertex \( v \), \( f_v(t) \) will be updated if Eq. (4) satisfies \( s_v^t = 1 \) (as it is or due to the fan-in e). For the source vertex \( u \), if it reaches a balanced state due to this fan-out, i.e. \( s_u^{t_0} = 0 \) & \( s_u^{t-1} = 1 \), then \( f_u(t) \) will be added onto \( F_u(t) \) as Eq. (5) shows and then be reset to \( 0 \), \( B_u(t) \) will also be updated as Eq. (6) shows. Otherwise, they will keep the same as the previous state.

\[
\begin{align*}
f_u(t) &= \begin{cases} f_u(t-1) + 1 & \text{if } s_u^t = 1, \\
f_u(t-1) & \text{otherwise.}
\end{cases}
\end{align*}
\]

\[
\begin{align*}
F_u(t) &= \begin{cases} F_u(t-1) + f_u(t) & \text{if } s_u^t = 0 \& s_u^{t-1} = 1, \\
F_u(t-1) & \text{otherwise.}
\end{cases}
\end{align*}
\]

\[
\begin{align*}
B_u(t) &= \begin{cases} B_u(t-1) + 1 & \text{if } s_u^t = 0 \& s_u^{t-1} = 1, \\
B_u(t-1) & \text{otherwise.}
\end{cases}
\end{align*}
\]

Figure 3: The residual series of the account \( u \) over time, as a running example of MonLAD with \( \delta_{up} = \delta_{down} = 20, \epsilon = 3 \).
Algorithm 1 MonLAD: Statistical features in stream
Input: Stream of edges over time; thresholds $\delta_{up}$, $\delta_{down}$, and $\epsilon$.
Output: Statistical features per vertex.  
1: while new edge $e = (u, v, w_{uv}, t)$ is received do  
2: if $(u, v)$ never appeared then $\triangleright$ Initialization for newcomers  
3: Initialize $(u, t - 1)$  
4: $R_u(t) = R_u(t) - w_{uv}, R_v(t) = R_v(t) + w_{uv} \triangleright$ For vertex $u$ or $v$  
5: Update 'state' for source $u$ and destination $v$ resp.  
6: sequentially update $f_u(t), F_u(t), B_u(t)$ with Eqsns. (4)-(6)  
7: if $s_u^t = 0 \& s_u^{t-1} = 1$ then  
8: $f_u(t) = 0 \triangleright$ Reset $f_u(t)$  
9: update $\min_{u}^t$ and $\max_{u}^t$ based on Eqsns. (7)-(8)  
10: output $B_u(t), F_u(t), B_v(t), F_v(t)$.  
11: procedure Initialize $(u, t)$  
12: $R_u(t)$ ← 0, $B_u(t)$ ← 0, $s_u^t$ ← 0, $f_u(t)$ ← 0;  
13: $\min_{u}^t$ ← 0, $\max_{u}^t$ ← 0  

Afterward, for the vertex $u$ with the fan-out $e$, $\min_{u}^t$ will be updated by the current residual $R_u(t)$ if it reaches balanced state or $R_u(t)$ is less than $\min_{u}^{t-1}$, for the vertex $v$ with the fan-in $e$, $\max_{u}^t$ will be updated by the residual $R_u(t)$ if it starts a new process and waits for reaching a balanced state ($s_u^t = 1 \& s_u^{t-1} = 0$) or $R_v(t)$ is greater than $\max_{u}^{t-1}$. Eqn. (7) and (8) depict the above update rules.  

$$\begin{align*} 
\min_{u}^t &= \begin{cases} 
R_u(t) & \text{if } (s_u^t = 0 \& s_u^{t-1} = 1) \text{ or } R_u(t) < \min_{u}^{t-1}, \\
\min_{u}^{t-1} & \text{otherwise}, 
\end{cases} \\
\max_{u}^t &= \begin{cases} 
R_v(t) & \text{if } (s_v^t = 1 \& s_v^{t-1} = 0) \text{ or } R_v(t) > \max_{u}^{t-1}, \\
\max_{u}^{t-1} & \text{otherwise}. 
\end{cases} 
\end{align*}$$  

In Algorithm 1, for each edge $e$ in the stream, MonLAD will create and initialize some vertex-specific variables via Initialize for the newcomers if $u$ or $v$ never appear, then it performs update with the above rules, and finally outputs the statistical features of each node at the current time, i.e., $B_u(t)$ and $F_u(t)$.

Figure 3 illustrates a running example of MonLAD. The state $s_u^t$ changes to 0 at $t_5$ and $t_{11}$ due to reaching a balanced state, $\max_{u}^t$ changes at $t \in \{t_2, t_4, t_6, t_8, t_10\}$, and $\min_{u}^t$ changes at $t_5$ and $t_{11}$.

4.2.1 MonLAD with sliding window: MonLAD-W. In real applications, transaction history involves a variety of accounts with different behaviors, which can be personal or corporate; fraudsters will avoid trading too frequently to evade detection, which differs from the behavior of company accounts that are used for frequent settlement or lopende rekening, so the global statistical features in MonLAD may be inappropriate to distinguish them. Furthermore, entirely accumulative counting also can result in some false positive detection, since normal accounts can also achieve balanced states periodically due to the credit card payback or fund transfers after payroll, thus we should rather capture the local behavior of a short period of time. So, we propose MonLAD-W, as the variant of MonLAD with a sliding time window, to compute the local features of $B$ and $F$. We can use various statistics of the features w.r.t windows,i.e., max, mean. The detail of MonLAD-W algorithm is given in the supplement.

Algorithm 2 ANOScore: Anomaly scoring with EVT
Input: Accounts features $B_u^t$ and $F_v^t = (F_v - B_v)$; threshold $p$ and percentile $\alpha$ for fitting the Pareto distribution.
Output: Suspicious vertex set $M$  
1: $M \leftarrow \emptyset \triangleright$ Use the IQR and $Q_3$ to find the truncation thresholds  
2: Compute IQRs and $Q_3$s for $B_u^t$ and $F_v^t$, to get the $b_1$ and $f_1$  
3: $b_2$ = Pareto($p, \alpha, \{B_u^t | F_v^t = f_1, u \in V\}$)  
4: $f_2$ = Pareto($p, \alpha, \{F_u^t | B_u^t = B_v | u \in V\}$)  
5: for $b \in \{1, b_1\}$ do  
6: $\triangleright$ Anomalies in Part I.  
7: if $(B_u^t > b_2 \& F_v^t > f_1) \text{ or } (B_u^t > b_1 \& F_v^t > f_2)$ then  
8: $M = M \cup \{u | B_u^t = b \& F_v^t = f\}$  
5: for $f \in \{0, f_1\}$ do  
9: $\triangleright$ Anomalies in Part II.  
10: if $(B_u^t > b_2 \& F_v^t > f_1)$ or $(B_u^t > b_1 \& F_v^t > f_2)$ then  
11: $M = M \cup \{u | B_u^t > b \& F_v^t = f\}$  
12: return $M$  

Figure 4: The heatmap and marginal distributions of the statistical features from CBank and the example for ANOscore algorithm. $B_u^t$ and $F_v^t = (F_v - B_v)$ are used, and the result at the end of the observation time are shown. ANOscore detects outliers in Part I, II, III, while the vertices in the high-density region (Part IV) are regarded as normal.

4.3 ANOscore: Anomaly Scoring
Based on the above features of each account, we design the scoring schema, ANOscore, to measure their suspiciousness in the money laundering scenario, respond to the customer queries at any time $t$, and report a group of most suspicious agent accounts as $M$.

The Generalized Pareto Distribution (GPD) is a 3-parameter distribution and has been used to find the law of extreme events (tails) within extreme value theory (EVT) [5] (Ref. the supplement).

The second theorem of EVT is the Pickands-Balkema-de Haan theorem [2, 25]. It states that given a random variable $X$, let $m \in \mathbb{R}$ and define a new random variable $X_m$, that intuitively represents the tail of $X$ past threshold $m$; define $F_m$ as the distribution of $X - m$ conditioned on $X > m$. Then, the conditional excess at threshold $m$ has CDF $F_m(x) = P(X - m \leq x | X > m)$. 

\[ P(X - m \leq x | X > m) \]
As the traits summarized in Sec. 3 and description in Sec. 4.1 show, the behavior of the agent accounts usually has great difference with the normal, this justifies using GPD to model the upper tail of the distribution of their features. The empirical result from the real data as shown in Figure 4 also verifies our motivation, i.e., the distributions of $B_u^*$ and $F_u'$ are $F_u = B_u$ are highly skewed\(^5\).

With the arrival of edges of a stream, MonLAD outputs $B_u(t)$ and $F_u(t)$ for all seen vertices $u \in V$ at time $t$ (MonLAD-W is also suitable). So, we design AnoSCORE in Alg. 2 to answer the customer’s query for anomalies in the stream at any time. AnoSCORE takes the positive features $B_u^*$ and $F_u'$ are $F_u = B_u$ and the probability threshold $\theta$ and percentile $\alpha$ for Pareto distribution as input, finally returns the subset $\mathcal{M}$ containing the most suspicious vertices.

The function Pareto uses the upper-tails part exceeding at threshold $\theta$, corresponding to samples at the bottom $1 - \alpha$ of the ordered population, to fit its parameters, and then outputs the minimum value of fitted data $x$ such that it satisfies $P(X - m > x | X > m) = 1 - F_m(x) < \theta$ for some parameter $p$ and $\alpha$, thus $p$ plays a similar role as the $P$-value to test the null hypothesis (i.e., the points are normal), we set $\alpha = 98\%$ in experiments as [36].

Based on the correlation between the features as Figure 4 shows, AnoSCORE detects those anomalies in three parts over the space of the joint distribution. Specifically, the decision boundaries $b_1$ and $f_1$ are determined by $Q_3 + K \cdot IQR$ with $IQR = Q_3 - Q_1$, $b_2$ and $f_2$ are determined by the Pareto which takes in the samples $\{u\}$ with $F_u' = f_1$ and $B_u^* = b_2$ respectively; we set $K = 1.5$ which is typically used for normally-distributed data \cite{13}. Then, the outliers in Part I, II, III are detected one by one (Lines 5 - 12), while Part IV is regarded as normal. The red dashed line in Figure 4 highlights the final decision boundary for detecting anomalies. In general, $b_2 > b_1$ and $f_2 > f_1$ are always true for sufficient data from large-scale stream; we will set $b_2 = b_1$ and $f_2 = f_1$ for some possible situation where $b_2 < b_1$ and/or $f_2 < f_1$ due to the ill-posed data distribution.

**Theorem 3 (Time Complexity).** The time complexity of the MonLAD algorithm is linear with the number of edges in the stream, $O(|E|)$; MonLAD-W uses $O\left(\frac{|V|}{2} + |V_t|\right)$ time to compute the window feature in addition. AnoSCORE scales linearly with the number of the seen vertices, $O(|V_t|)$, at current time $t$.

### 5 EXPERIMENTS

We design experiments to answer the following questions:

1. **Q1. Effectiveness for pattern detection:** How accurately does MonLAD\(^8\) detect expected behavior patterns (P1-P3) of agent accounts for money laundering? Does AnoSCORE have advantage over other baselines?
2. **Q2. MonLAD spots real-world agent accounts:** What patterns does MonLAD-W detect in real-world datasets? How about the behavior of the suspicious agent accounts?
3. **Q3. Scalability:** Does our method scale linearly with the number of edges?

**Datasets:** We use two real-world datasets, the ‘CBank’ dataset from an anonymous bank under an NDA agreement; the Czech Financial dataset (“CFD”) is an anonymous transferring transaction data of Czech bank released for Discovery Challenge in [21]. Table 3 lists the statistical information of the two datasets.

| Dataset | # of Nodes | # of Edges | Time Span (T) |
|---------|------------|------------|---------------|
| CBank   | 8.77 M     | 47.44 M    | Aug.7 - Aug.13, 2017 |
| CFD     | 11.37 K    | 273.51 K   | Jan.1,1993 - Dec.1,1998 |

**Baselines:** We select following methods as baselines, Spotlight [8], Midas-R [3], and SedanSpot [7] that detect anomalies in edge streams, and D-Cube [33], a batch yet fast algorithm that outputs close or better accuracy than its streaming version, DenseAlert [34].

**Evaluation Metric:** We evaluate the performance with $F1$ score. For the baselines detecting suspicious edges rather than vertices, we treat an edge as a hit if any of its ends are labeled anomalous. For a fair comparison, we report the best $F1$ score for those baselines returning a rank of suspicious edges.

**Experimental Setup:** All experiments are carried out on a 2.7GHz Intel Xeon E7-8837 CPUs processor and 512GB RAM running Linux. MonLAD is implemented in Python. In all the experiments, we set $\alpha = 98\%$, $p = 0.05$ and $\delta_{up} = \delta_{down} = \epsilon = 10k$ for our methods (unless specified otherwise). We average the results over 5 trials for all synthetic experiments for anomalies injection.

### 5.1 Q1. Effectiveness in pattern detection.

#### 5.1.1 Effectiveness of MonLAD.

Considering the various behavior patterns in money laundering scenario as summarized in Sec. 4.1, we adopt numerous injection schema upon the CBank dataset to verify MonLAD’s performance for spotting different kinds of anomalies. Based on the characteristics of patterns P1-P3, we control the behavior of agent accounts via different $f_1$ and $b_1$.

We use the clean CBank data, i.e., removing all suspicious accounts detected by MonLAD, as background for injection. For patterns P1 and P2, we inject $|M| = 200$ agent accounts as anomalies and randomly select the source or target accounts for money transfer-in and out; the amount of money for each transaction is sampled from a Dirichlet distribution\(^9\) to ensure that the total amount (no more than 1E8) is uniformly assigned to each edge; all the money transferred into an account is clear out and reaching a balanced state finally; the time of each transaction is randomly sampled from the time range.

Figure 5(a)-5(b) show the performance of MonLAD and baselines for detecting different patterns. We report the results of MonLAD with different $p$ (0.01 to 0.05 with step 0.01) for clear comparisons. As we can see, MonLAD consistently outperforms all baselines for P2 pattern by achieving at least 68% improvement in Fig. 5(a) (with $p = 0.01$). Fig. 5(b) demonstrates the performance for detecting P1 behavior pattern, we can see FlowScope also achieves high accuracy (triangle marked) since these found accounts have fewer transfer connections but higher average amounts. Compared with Fig. 5(a), however, FlowScope gives the worst accuracy for low-density cases. We compare the accuracy and speed with baselines

---

\(^5\)We use the difference feature since $F_u \geq B_u$ always hold for any vertex ($\cdot$).

\(^6\)Q1 and Q2 represent 1st and 3rd quartile of samples respectively.

\(^7\)The detailed proof of the Theorem 3 is given in the supplement.

\(^8\)We use MonLAD(W) to refer to MonLAD(W) followed with AnoSCORE here.

\(^9\)It generates one sample (edge) each time with the parameter $\alpha = [\alpha_1, \ldots, \alpha_{|E|}]$ with $\alpha_i = 100$ for $1 \leq i \leq |E|$ to guarantee a small variance among the sampling.
under two specific settings, $f_\ast = 500$ & $B_\ast = 4$ in Fig. 5(a) and $f_\ast = 1$ & $B_\ast = 1k$ in Fig. 5(b). Fig. 2(b) shows the result, where the empty and filled markers correspond to the above two cases; we can see that MonLAD achieves nearly $4 \times$ speedup over FlowScope with very similar accuracy and outperforms other competitors. We evaluate the performance of MonLAD against the number of injected agent accounts ($|M|$) and give the result in Fig. 5(c). MonLAD achieves the best F1 score and has obvious advantages over baselines; its accuracy decreases as $|M|$ increases for $p \leq 0.02$, since the injection gradually dominates the tail part of data, resulting in lower recall when the precision has reached the maximum.

### 5.2 Q2. Spotting real-world agent accounts

#### 5.2.1 Analysis of MonLAD

We apply MonLAD to CBank data for detecting suspected real-world agent accounts and analyze their behavior patterns. As Figure 2(a) shows, MonLAD spots various types of anomalies for money laundering where the decision boundary is denoted as the red dashed line. In the absence of labels, we randomly select some detected accounts from different parts for case study about their transfer structures, residual series, and statistical features ($B_\ast$ & $f_\ast$). Moreover, 12 accounts are manually verified to be fraudulent agents, they can be easily and accurately caught by MonLAD and are labeled with ‘criminal-icron’ near $p_2$ in Fig. 2(a).

Figure 1 and Figure 6 show the behavior patterns of some selected accounts ($p_3$–$p_4$). For the account $p_3$ in Fig. 1, it receives 63 transfers from two different accounts ($\approx 34.72M$ in total) and transferred them to 6 other accounts multiple times, there are two accounts accounted for the majority and other small amount transfers are very likely to be camouflage transactions; from the residual series, we can see that a large amount of money flowed through $p_3$ with $P1$ behavior leading to reaching a number of balanced states, which is a typical suspicious pattern.

Additionally, from Figure 6(a)–6(b), we find that the account $p_1$ has similar transfer structure with $p_3$, but it has multiple fan-ins with the behavior similar as $P2$ pattern as Fig. 6(c) shows; so $p_1$ transfers money in a mixed manner of $P1$ and $P2$, i.e. $P3$. As we observe from Fig. 6(d)–6(f), the fan-ins of the verified account $p_2$ is more than 1,000× its fan-outs, which is consistent with the typical $P2$ pattern. For the account $p_4$, it has the same behaviors as $P2$ based on the zoom-in sub-figure in Fig. 9(b) and Fig. 6(i), while it is less suspicious than $p_2$ due to close to the decision boundary.

In contrast, the residual of the normal accounts in Part IV rarely reaches balanced states, we provide detailed analysis for the behavior of some randomly selected normal examples in the supplementary. Therefore, MonLAD indeed finds some suspicious agent accounts for money laundering.

#### 5.2.2 Analysis of MonLAD-W

We verify the effectiveness of MonLAD-W for detecting suspicious behavior with periodical balanced pattern and compare with MonLAD. We use CBank dataset and set the parameter of MonLAD same as the Sec. 5.2. We set window size $K = 1h$, sliding stride $s = 1h$ for MonLAD-W which use the maximum value of the statistical feature series as score, $p = 1e-3$ for AnoScore.

Figure 7(a) illustrates residual series of two detected accounts as examples for the case study. The account at the bottom is detected by MonLAD-W but not MonLAD. It achieves high-frequent balanced states within one hour (each takes less than $5min$), which is rather suspicious; it was active only on Aug. 4th, which results in a smaller value of $B$ than the $B$s of those normal accounts that have been active since the early days from the point of cumulative counting.

Therefore, the anomalous accounts like above mentioned one will not be detected by MonLAD. Furthermore, we randomly select another account that has the same features ($B = 8$ & $F' = 1$) from MonLAD-W as the above account. As shown in the top of Fig. 7(a), we can see that it had a similar periodical balanced pattern, yet with a much longer period (i.e. $= 0.5 d$). Although the top one is more likely to be a normal account, MonLAD cannot distinguish it from the more suspicious one at the bottom. In contrast, MonLAD-W is able to avoid such false positive (FP) detection since it only counts the number of features within a specified time window. For MonLAD-W, the feature values of these accounts are $B = 1F' = 0$ (top) and $B = 7F' = 1$ (bottom).

Therefore, MonLAD-W is able to capture the local behavior pattern while avoiding some FN and FP cases caused by the accumulative effect in MonLAD.

#### 5.2.3 Development at a real-world platform

We also apply MonLAD to a real money transfer dataset collected from Tencent’s WeChat App, which consists of millions of accounts and transaction records for one week$^{10}$ with the format (source, destination, time, amount).

---

$^{10}$Concrete information is not public due to privacy protection.
Figure 6: Case study of the detected anomalies in real-world data (CBank). Here illustrate the transfer structure, residual series $R_s(t)$, and $F_s(t)$ of those randomly selected suspicious accounts ($p_1, p_2, p_4$) labeled in Figure 2(a).

Figure 7: Performance for real data and case study. (a) Two example detection cases with periodical pattern as the FP (false positive) from MonLAD and the TP (true positive) from MonLAD-W. (b) The heatmap of the features from MonLAD based on the real data from Tencent Wechat App and some manually verified suspected fraudster accounts.

5.3 Q3. Scalability

We measure how rapidly MonLAD’s update time increases as the stream grows. We used the accumulative edge streams of CBank per 12 hour and gives the running time of the algorithm until the specific time. Figure 2(c) shows the linear scaling of MonLAD’s running time with the number of edges of the stream.

6 CONCLUSIONS

In this paper, we propose MonLAD and MonLAD-W for detecting the money laundering anomaly agent accounts in a transaction stream; we introduce a statistical score schema, ANoSCORE, to find anomalies that have obvious deviation behavior. The experiments on real-world datasets show that our method MonLAD achieves state-of-the-art performance and interpretable results, and our methods are also linearly scalable.

ACKNOWLEDGMENTS

This paper is partially supported by the National Science Foundation of China under Grant No.91746301, 61772498, U1911401, 61872206. This paper is also supported by the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No. XDA19020400 and 2020 Tencent Wechat Rhino-Bird Focused Research Program.
REFERENCES

[1] Charu C Aggarwal, Yuchen Zhao, and S Yu Philip. 2011. Outlier detection in graph streams. In ICDE.
[2] August A Balkema and Laurens De Haan. 1974. Residual life time at great age. The Annals of probability (1974).
[3] Siddharth Bhatia, Bryan Hooi, Minji Yoon, Kijung Shin, and C. Faloutsos. 2020. MIDAS: Microcluster-Based Detector of Anomalies in Edge Streams. AAAI (2020).
[4] Zhangyu Cheng, Chengmeng Zou, and Jianwei Dong. 2019. Outlier detection using isolation forest and local outlier factor. In Proceedings of the conference on research in adaptive and convergent systems. 161–168.
[5] Stuart Coles, Joanna Bawa, Lesley Trenner, and Pat Dorazio. 2001. An introduction to statistical modeling of extreme values. Vol. 208. Springer.
[6] Paul Embrechts, Claudia Kluppelberg, and Thomas Mikosch. 1999. Modelling extremal events. British actuarial journal (1999).
[7] Dhivya Eswaran and Christos Faloutsos. 2018. Sedanopt: Detecting anomalies in edge streams. In ICDM.
[8] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra. 2018. SpotLight: Detecting Anomalies in Streaming Graphs. SIGKDD (2018).
[9] Wenjie Feng, Shenghua Liu, Christos Faloutsos, Bryan Hooi, Huawei Shen, and Xueqi Cheng. 2019. Beyond Outliers and on to Micro-clusters: Vision-Guided Anomaly Detection. In The 23rd Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 541–554.
[10] Markus Goldstein and Andreas Dengel. 2012. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. KI-2012: Poster and Demo Track (2012).
[11] Johanna Hardin and David M Rocke. 2004. Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator. Computational Statistics & Data Analysis (2004).
[12] Zengyou He, Xiaofei Xu, and Shengchun Deng. 2003. Discovering cluster-based local outliers. Pattern Recognition Letters (2003).
[13] David C Hoaglin, Boris Iglewicz, and John W Tukey. 1986. Performance of some resistant rules for outlier labeling. JASA (1986).
[14] John Hunt. 2011. The new frontier of money laundering: how terrorist organizations use cyberlaundering to fund their activities, and how governments are trying to stop them. Information & Communications Technology Law (2011).
[15] Nihen An Le Khac and M-Tahar Kechachi. 2010. Application of data mining for money laundering detection: A case study. In ICDMW.
[16] Meng-Chieh Lee, Yue Zhao, Aluna Wang, Pierre Jngkhong Liang, Leman Akoglu, Vincent S Tseng, and Christos Faloutsos. 2020. AutoAudit: Mining Accounting and Time-Evolving Graphs. arXiv preprint arXiv:2011.00447 (2020).
[17] Xiangfeng Li, Shenghua Liu, Zifeng Li, Xiaotian Han, Chuan Shi, Bryan Hooi, He Huang, and Xueqi Cheng. 2020. FlowScope: Spotting Money Laundering Based in Edge Streams. In WSDM.
[18] Jiabao Zhang, Shenghua Liu, Wenting Hou, Siddharth Bhatia, Hua-Wei Shen, and Wei Lu. 2016. A scalable approach for outlier detection in edge streams using sketch-based approximations. In SDM.
[19] Reza Soltani, Uyen Trang Nguyen, Yang Yang, Mohammad Faghani, Ala Yagoub, and Aijun An. 2016. A new algorithm for money laundering detection based on structural similarity. In Ubiquitous Computing, Electronics & Mobile Communications Conference (UEMCON), IEEE Annual.
[20] Lucia Larise Stavarache, Donatas Narbutis, Toyotaro Suzumura, Ray Harishanker, and Augustas Žaltkaskas. 2019. Exploring Multi-Banking Customer-to-Customer Relations in AML Context with Poincaré Embeddings. arXiv:1912.07701 (2019).
[21] Jun Tang and Jian Yin. 2005. Developing an intelligent data discriminating system of anti-money laundering based on SVM. In ICMCL.
[22] Seikanth Thudumu, Philip Branch, Juejung Jin, and Jygdutt Jack Singh. 2020. A comprehensive survey of anomaly detection techniques for high dimensional big data. Journal of Big Data, 7 (1) (2020), 1–30.
[23] M. Togbe, M. Barry, A. Boly, Yousra Chabchoub, Raja Chiky, Jacob Montiel, and Reza Soltani. 2017. Anomaly detection in streams with extreme value theory. In SIGKDD.
[24] Riad Alsabti and Ali Shamsuddin. 2001. A novel anomaly detection scheme based on principal component classifier. Technical Report.
[25] Emaad Manzoor, Sadegh M Milajerdi, and Leman Akoglu. 2016. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In SIGKDD.
[26] Eibe Frank, Frank Hage, Helmut Landwehr, and Christos Faloutsos. 2016. Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering. In ICMLA.
[27] James Pickands III et al. 1975. Statistical inference using extreme order statistics. Annals of statistics (1975).
[28] Siddharth Ramaswany, Rajeev Rastogi, and Krysook Shim. 2000. Efficient algorithms for mining outliers from large data sets. In SIGMOD.
[29] Stephen Randalou, Steve Harenberg, Kohutj Sharma, and Nagiza F Samatova. 2016. A scalable approach for outlier detection in edge streams using sketch-based approximations. In SDM.
[30] L. Retting, M. Khayati, P. Cudré-Mauroux, and Michal Piorkowski. 2015. Online anomaly detection over Big Data streams. IEEE Big Data (2015).
[31] Peter Reuter. 2004. Chasing dirty money: The fight against money laundering. Peterson Institute.
[32] Peter Reuter and Edwin M Truman. 2003. Money laundering: Methods and markets. Chas. Dirty Money Fight against money Launder (2003).
[33] Peter J Rousseeuw and Katrien Van Driesen. 1999. A fast algorithm for the minimum covariance determinant estimator. Technometrics (1999).
[34] David Savage, Qingmai Wang, Pauline Chou, Xiu Zheng Zhang, and Xinghuo Yu. 2016. Detection of money laundering groups using supervised learning in networks. arXiv:1608.00708 (2016).
[35] Kijung Shin, Bryan Hooi, Jisoo Kim, and Christos Faloutsos. 2017. D-cube: Dense-block detection in terabyte-scale tensors. In WSDM.
[36] Kijung Shin, Bryan Hooi, J. Kim, and C. Faloutsos. 2017. DenseAlert: Incremental Dense-Subbtensor Detection in Tensor Streams. SIGKDD (2017).
[37] Mei-Ling Shyu, Shu-Ching Chen, Kanokri Sarinnapakorn, and Li-Wu Chang. 2003. A novel anomaly detection scheme based on principal component classifier. Technical Report.
[38] Alwan Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. 2017. Anomaly detection in streams with extreme value theory. In SIGKDD.
[39] Peter Reuter and Edwin M Truman. 2003. Money laundering: Methods and markets. Chas. Dirty Money Fight against money Launder (2003).
[40] Reza Soltani, Uyen Trang Nguyen, Yang Yang, Mohammad Faghani, Ala Yagoub, and Aijun An. 2016. A new algorithm for money laundering detection based on structural similarity. In Ubiquitous Computing, Electronics & Mobile Communications Conference (UEMCON), IEEE Annual.
[41] Lucia Larisse Stavarache, Donatas Narbutis, Toyotaro Suzumura, Ray Harishanker, and Augustas Zaltakas. 2019. Exploring Multi-Banking Customer-to-Customer Relations in AML Context with Poincaré Embeddings. arXiv:1912.07701 (2019).
[42] Jun Tang and Jian Yin. 2005. Developing an intelligent data discriminating system of anti-money laundering based on SVM. In ICMCL.
[43] Seikanth Thudumu, Philip Branch, Juejung Jin, and Jygdutt Jack Singh. 2020. A comprehensive survey of anomaly detection techniques for high dimensional big data. Journal of Big Data, 7 (1) (2020), 1–30.
[44] M. Togbe, M. Barry, A. Boly, Yousra Chabchoub, Raja Chiky, Jacob Montiel, and Reza Soltani. 2017. Anomaly detection in streams with extreme value theory. In SIGKDD.
[45] Riad Alsabti and Ali Shamsuddin. 2001. A novel anomaly detection scheme based on principal component classifier. Technical Report.
[46] Alwan Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. 2017. Anomaly detection in streams with extreme value theory. In SIGKDD.
[47] Reza Soltani, Uyen Trang Nguyen, Yang Yang, Mohammad Faghani, Ala Yagoub, and Aijun An. 2016. A new algorithm for money laundering detection based on structural similarity. In Ubiquitous Computing, Electronics & Mobile Communications Conference (UEMCON), IEEE Annual.
[48] Lucia Larisse Stavarache, Donatas Narbutis, Toyotaro Suzumura, Ray Harishanker, and Augustas Žaltkaskas. 2019. Exploring Multi-Banking Customer-to-Customer Relations in AML Context with Poincaré Embeddings. arXiv:1912.07701 (2019).
[49] Jun Tang and Jian Yin. 2005. Developing an intelligent data discriminating system of anti-money laundering based on SVM. In ICMCL.
[50] Seikanth Thudumu, Philip Branch, Juejung Jin, and Jygdutt Jack Singh. 2020. A comprehensive survey of anomaly detection techniques for high dimensional big data. Journal of Big Data, 7 (1) (2020), 1–30.
Algorithm 3 describes the framework of \( A \) (variance, etc. The maximum value is used in our experiments. A.3 Generalized Pareto Distribution

\[ \text{FAUC indicates better performance. We set } \alpha = 0.5 \text{ and } \rho = 0.2 \text{ for } \text{ANO} \text{SCORE due to the limited data size. The parameter setting of baselines and detailed injection scheme are given in A.6.} \]

**Table 4: \text{ANO} \text{SCORE outperforms other outlier-detection methods over CFD dataset}**

| Anomalies type | CBLF | HBOS | IF | KNN | MCD | PCA | \text{ANO} \text{SCORE} |
|----------------|------|------|----|-----|-----|-----|-------------------------|
| Part I \( f_1 \) | 0.706 | 0.202 | 0.494 | 0.131 | 0.188 | 0.573 | 0.744 |
| Part III \( B_3 \) | 0.641 | 0.324 | 0.775 | 0.554 | 0.839 | 0.590 | 0.687 |
| All Parts \( f_P \) | 0.791 | 0.612 | 0.845 | 0.570 | 0.750 | 0.848 | 0.810 |
| All Parts \( B_3 \) | 0.808 | 0.608 | 0.776 | 0.556 | 0.731 | 0.752 | 0.736 |
| Average | 0.742 | 0.437 | 0.722 | 0.453 | 0.627 | 0.691 | 0.744 |

Table 4 shows the detection results of different detection methods for various injection settings. On average, \text{ANO} \text{SCORE achieves the best performance and is in par with CBLF and IF, they have own strength for different parts which corresponds to different behavior patterns. However, \text{ANO} \text{SCORE is more interpretable and intuitive due to unitizing the statistical deviations for the money laundering scenario. Besides, we can see that some method also performs well in some specific parts, e.g., MCD for 'Part III' and PCA for 'All Parts (\( f_P \))'; HBOS and KNN have the worst performance. In summary, our MonLAD is more flexible to in favor of different outlier detection methods and achieves appealing results.**

### A.2 MonLAD-W algorithm

Algorithm 3 describes the framework of MonLAD-W, which can seamlessly connect with MonLAD, it computes and outputs the local statistical features of the current time window with Line 4-14.

As can be seen, MonLAD-W yields \( B_{\text{win}} \) and \( F_{\text{win}} \) for all accounts, forming a group of series, across the time window; thus we can also analyze trends or statistics for them, like maximum, mean, variance, etc. The maximum value is used in our experiments.

### A.3 Generalized Pareto Distribution

Specified by the parameters location \( \mu \), scale \( \sigma > 0 \), and shape \( \xi \), the cumulative distribution function (CDF) of GPD is defined as,

\[
\text{GPD}_{\mu,\sigma}(x) = \begin{cases} 
1 - \left( \frac{1 + \frac{x-\mu}{\sigma}}{\xi} \right)^{-1/\xi} & \text{if } \xi \neq 0 \\
1 - \exp\left(-\frac{x-\mu}{\sigma}\right) & \text{if } \xi = 0 
\end{cases}
\]

The second theorem of EVT is the Pickands-Balkema-de Haan theorem \([2, 25]\). It states that given a random variable \( X \), let \( m \in \mathbb{R} \) and define a new random variable \( X_m \), that intuitively represents the tail of \( X \) past threshold \( m \); define \( \text{CDF}(\text{fauc}) \) as the distribution of \( X > m \) conditioned on \( X > m \). Then, the conditional excess at threshold \( m \) has CDF \( \text{CDF}(x) = P(X - m \leq x | X > m) \).

Being agnostic to the distribution of original data under a weak condition, the key properties of GPD are its flexibility in smoothly interpolating between light- and heavy-tailed regimes (extreme events), and its universality property.

**Property 1 (Universality of GPD). Let \( F \) be any distribution function from a broad class of distributions\(^1\). For the CDF of the GPD, there exists \( \xi \) and \( \sigma(m) \) that approximate the tail of \( F \) arbitrarily closely, that is, \( \lim_{m \to \infty \sigma(m)} \sup x \left| \text{CDF}(x) - \text{GPD}(\xi, \sigma(m)) \right| = 0 \), where \( \sigma(m) \) is the right endpoint of \( F \) and can be infinite.

**Algorithm 3 MonLAD-W: MonLAD with sliding time window**

**Input:** Stream of edges over time; thresholds \( \delta_{\text{up}}, \delta_{\text{down}} \), and \( \varepsilon \); window size \( K \), sliding stride \( s \).

**Output:** Statistical features per window.

1. \( t_{\text{begin}} = t_{\text{end}} = 0; \)
2. Initialize feature vectors \( B_{\text{begin}}(\ast) \leftarrow 0, F_{\text{begin}}(\ast) \leftarrow 0 \)
3. while new edge \( e = (u, v, w_{uv}, t) \) is received do
4. if \( t > t_{\text{end}} \) then \( \triangleright \text{Receive an unseen time tick} \)
5. \( B_{t_{\text{end}}}(\ast) \leftarrow B(\ast), F_{t_{\text{end}}}(\ast) \leftarrow F(\ast) \) \( \triangleright \text{Record for } t_{\text{end}} \)
6. \( t = K + s \cdot \left\lfloor \frac{t - K}{s} \right\rfloor \) \( \triangleright \text{Bound of current window} \)
7. if \( t_{\text{end}} \geq K \) and \( t_{\text{end}} \leq t \) then \( \triangleright \text{Window update} \)
8. \( B_{\text{win}} = B_{t_{\text{begin}}}(\ast) - B_{t_{\text{begin}}}(\ast) \)
9. \( F_{\text{win}} = F_{t_{\text{end}}}(\ast) - F_{t_{\text{begin}}}(\ast) \)
10. \( t_{\text{begin}} \leftarrow \max \{ t_{i} \mid t_{i} \leq t_{\text{begin}} + s \} \)
11. for \( i < t_{\text{begin}} \) do \( \triangleright \text{Window slides with } s \)
12. Remove \( B_{t_{i}}(\ast) \) and \( F_{t_{i}}(\ast) \)
13. yield \( B_{\text{win}}, F_{\text{win}} \) \( \triangleright \text{Output the current features} \)
14. \( t_{\text{end}} \leftarrow t \)
15. \( B(u), F(u), B(v), F(v) = \text{MonLAD}(e, \delta_{\text{up}}, \delta_{\text{down}}, \varepsilon) \)

**A.4 Analysis of Time Complexity**

For MonLAD, for each edge, the algorithm will update the corresponding variables according to the rules, which takes \( O(1) \). Thus, the time complexity of the MonLAD algorithm is linear with the number of edges in the stream, \( O(|E|) \).

For MonLAD-W, additionally, the most time consuming steps are Lines 8 – 13 in Algorithm 3, which are executed at the end of each window. Let \( |T| \) be the number of time ticks, \( K \) be the window size and \( s \) be the sliding stride size, then the number of time windows will be \( \frac{|T|}{s} + 1 \); vectors subtraction at line 8 – 9 takes \( O(|V|) \). Therefore, the additional time complexity of MonLAD-W is \( O\left(\frac{|T|}{s} \times |V|\right) \).

\(^1\)This class includes almost all commonly used distributions \([6]\).
we randomly pick 2 normal accounts \( p_5 \) and \( p_6 \) (\( B_{p_5} = F_{p_6} = 0 \)) from the Part IV.

For \( \text{ANOSCORE} \), the thresholds computing at Lines 3 – 4, Line 6, and Line 11 for the upper tails part of distribution will take \( O(1) \), and it takes \( O(|V_I|) \) to determine whether each node is abnormal in Lines 8 – 9. Thus, \( \text{ANOSCORE} \) scales linearly with the number of the seen vertices, \( O(|V_I|) \), at the current time \( t \).

### A.5 Normal Behavior Patterns.
To further verify the behavior patterns detected with \( \text{MONLAD} \), we randomly select 2 normal accounts (fall in the part IV of the features heatmap in Figure 8) from \( \text{CBank} \) dataset as comparison.

The results are illustrated in Figure 9. In Figure 9(c), we find that the account \( p_5 \) has the same behaviors as \( \text{P1} \) pattern (i.e., one-time fan-in and immediate fan-out), but it reaches the balanced state only three times, which are labeled with red circle marker, and the total amount of transfers is far less than the suspicious accounts \( p_3 \) and \( p_4 \) analyzed in the main paper (see Figure 9(a)-9(b)), thus \( p_5 \) is more likely to be a normal account. Meanwhile, as shown in Figure 9(d), there is no obvious pattern in the residual series of the account \( p_6 \) (\( B_{p_6} = F_{p_6} = 0 \)) and it never reaches a balanced state, which is consistent with the normal behavior.

In summary, those results suggest that the normal accounts rarely reach a balanced state compared with the anomaly accounts whose behavior are more suspected. \( \text{MONLAD} \) can spot the anomalies based on those behavior patterns.

### A.6 Detailed Experiment Results.
Here we provide the detailed information about the experimental settings (baseline parameter settings and injection scheme), qualitative analysis and comparison for the additional results of \( \text{ANOSCORE} \) and baselines.

#### Implementations.
In this experiment, we consider 6 outlier detection algorithms implemented by [45] for comparison:

- **Clustering-Based Local Outlier Factor (CBLOF)** [12]: We set the number of clusters \( k = 8 \), the coefficients for deciding small and large clusters \( \alpha = 0.9 \) and \( \beta = 5 \), as recommended in the original work.
- **Histogram-based Outlier Score (HBOS)** [10]: We set the number of bins \( k = 10 \).
- **Isolation Forest (IF)** [18]: We set the number of trees to \( t = 100 \) and the sub-sampling size to \( \psi = 256 \), as recommended in the original work.
- **KNN** [26]: We set the number of nearest neighbors \( k = 5 \).
- **Minimum Covariance Determinant (MCD)** [11, 31]: We set the proportion of points to be included in the support of the raw estimate \( h_j = 0.8 \), leading to a better accuracy than the setting in the original work.
- **PCA** [35]: We use euclidean distance as distance metric and keep all principal components to calculate the outlier scores.

#### Injection scheme.
We use the clean CFD dataset as the base for injection after removing all suspicious accounts detected by \( \text{MONLAD} \) with the corresponding detection thresholds are \( b_1 = 13 \), \( b_2 = 20 \), \( f_1 = 95 \) and \( f_2 = 120 \).

For each part, we inject 20 points as anomalies. Specifically, for the Part I, we vary \( F_i \) from 100 to 130 and randomly select \( B_i \) from 1 to the threshold \( b_1 = (13) \) and ensure injected anomalies fall in this desired part. Similarly, for the Part III, we vary \( B_i \) from 15 to 35 and randomly select \( F_i \) from 1 to the threshold \( f_1 = (95) \). For the mixture cases, we change the above two experimental features (i.e., \( F_i \), and \( B_i \)) separately and randomly inject 40 points into Part II. And when one experimental feature changes, such as \( F_i \) (or \( B_i \)), we will fix \( B_i \) (or \( F_i \)) of Part III (or Part I) to the threshold \( b_2 = 20 \) (or \( f_2 = 120 \)).

#### Analysis and Comparison.
Based on the above performances, we analyze each method as follows: HBOS assumes the feature independence while \( F_i \) and \( B_i \) are related; KNN ranks points by the distance to its \( k \) nearest neighbor, however, we assume that the suspicious accounts usually locate on the upper tail of data so that close to the normal. Additionally, MCD is proposed based on Gaussian-distributed data while the distribution of our features are highly skewed as shown in Fig. 3 in the main paper. In contrast, methods without special assumptions perform better. CBLOF focuses on the physical significance of outliers, PCA utilizes the correlation between data and IF focus on isolated points, yet they are designed for general outliers detection purpose and are not fully suitable for the money laundering detection scenario. However, our proposed \( \text{ANOSCORE} \) can avoid the above problems and achieves higher accuracy.