AUTOAUDIT: Mining Accounting and Time-Evolving Graphs

Meng-Chieh Lee
National Chiao Tung University
jeremy08300830.cs06g@nctu.edu.tw

Yue Zhao
Carnegie Mellon University
zhaoy@cmu.edu

Aluna Wang
Carnegie Mellon University
aluna@cmu.edu

Pierre Jinghong Liang
Carnegie Mellon University
liangj@andrew.cmu.edu

Leman Akoglu
Carnegie Mellon University
lakoglu@andrew.cmu.edu

Vincent S. Tseng
National Chiao Tung University
vtseng@cs.nctu.edu.tw

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

***Abstract*—**
How can we spot money laundering in large-scale graph-like accounting datasets? How to identify the most suspicious period in a time-evolving accounting graph? What kind of accounts and events should practitioners prioritize under time constraints? To tackle these crucial challenges in accounting and auditing tasks, we propose a flexible system called AUTOAUDIT, which can be valuable for auditors and risk management professionals. To sum up, there are four major advantages of the proposed system: (a) *"Smurfing" Detection*, spots nearly 100% of injected money laundering transactions automatically in real-world datasets. (b) *Attention Routing*, attends to the most suspicious part of time-evolving graphs and provides an intuitive interpretation. (c) *Insight Discovery*, identifies similar month-pair patterns proved by "success stories" and patterns following Power Laws in log-logistic scales. (d) *Scalability and Generality*, ensures AUTOAUDIT scales linearly and can be easily extended to other real-world graph datasets. Experiments on various real-world datasets illustrate the effectiveness of our method. To facilitate reproducibility and accessibility, we make the code, figure, and results public at https://github.com/mengchillee/AutoAudit.

***Index Terms*—Time-Evolving Graph, Graph Mining, Anomaly Detection**

I. INTRODUCTION

Given a complicated accounting dataset, how can we spot the most suspicious structures and provide practical advice to accountants? Given the graphs rendered from accounting datasets are always directed, weighted, and time-evolving, how can we identify the time frames that exhibit potential correlations and provide the cause? How can we filter out useless information and focus on important signals? We propose AUTOAUDIT (AA) to address three major challenges on mining such time-evolving accounting graphs.

The first challenge relates to facilitating the investigation of money laundering crimes (MLCs)–the importance of leveraging machine learning techniques has been recognized in numerous literature [1], [2], [3], [4]. Among all money laundering crimes, "Smurfing" [5] is one of the most frequent cases [6], [7], [8]. As shown in Figure 1, it involves the use of multiple intermediaries for making small cash deposits, buying monetary instruments, or bank drafts in amounts under the reporting threshold. Establishing reasonable suspicion usually involves reading and analyzing a massive volume of transactions and thousands of documents, which is exceptionally challenging and time-consuming. Consequently, there is necessity to automate the process of reviewing financial records so that "Smurfing" type frauds can be spotted. We propose AA-SMURF, a parameter-free and unsupervised method, to facilitate the detection of "Smurfing" in the accounting data by minimizing the description length of adjacency matrix.

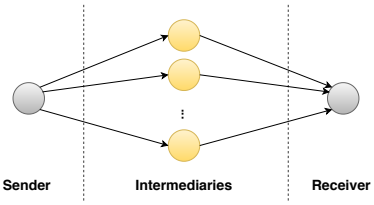


Fig. 1: **"Smurfing"**: One of the most frequent types of money laundering

The second challenge concerns finding the starting point for the "test of transactions." Due to the sheer volume of financial transactions, limited resources for audits, and ever-changing business environment and policies, auditors often find it challenging to determine a starting point for review when examining transactions and documentation. Although there have been studies in detecting anomalies on time-evolving graphs [9], [10], [11], [12], [13], most of them do not offer this information with intuitive explanations. As a result, effective tools that can process account behavior across time to flag critical time-periods for further investigation are needed. We propose AA-FOCUS, by encoding the anomaly score of accounts in each focus-plot over time into sketches, to direct users to the most critical changing points in a long period and explain why they are suspicious.

The third challenge is discovering the potential properties of time-evolving graphs. Obtaining and understanding these temporal-correlation patterns can be extremely useful since periodic financial bookkeeping data, visualized in graphs, are often used for internal control purposes. We propose AA-DISCOVER, to not only bring these temporal patterns to light,

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

but also to ferret out the most effective contributing factors. Moreover, by using AA-DISCOVER, we discover a whole new phenomenon that the log-logistic can fit well for many accounting datasets. For space limitation, AA-DISCOVER along with experimental results are presented in Appendix.

To sum up, we propose AUTOAUDIT in this paper—a systematic method for detecting anomalies not only in accounting datasets, but also in other real-world datasets that shares similar characteristics. Figure 3a shows AA-SMURF automatically reorder the adjacency matrix and put the most likely "Smurfing" patterns in the front (highlighted in red). Moreover, as shown in Figure 3b, AA-FOCUS not only detects the exact period when a significant change occurs, it also finds out why Enron's CEO has been busy solving emergent events.

The advantages of our method are:

- **"Smurfing" Detection**: AA-SMURF, an unsupervised and parameter-free algorithm, can detect injected "Smurfing" pattern in real-world datasets.
- **Attention Routing**: AA-FOCUS identifies most suspicious periods in time-evolving graphs with explanations.
- **Insight Discovery**: AA-DISCOVER unearths three month-pairs with high correlation, proved by "success stories", and additional patterns of accounting datasets. It is put into Appendix due to space limitation.
- **Scalability and Generality**: AUTOAUDIT scales linearly and can be generalized on other real-world graph datasets, such as Enron Email and Czech Financial datasets.

Reproducibility: Our implemented source code and "Smurfing" generator are made public[1].

The rest of this paper is organized as follows. First, We introduce the problem statement and detail our proposed method in Section II. Then, experimental results are then presented in Section III. Section IV concludes. In Appendix[2] (available online in the extended version), we briefly review the background and related work in Section A, and provide additional experiment results.

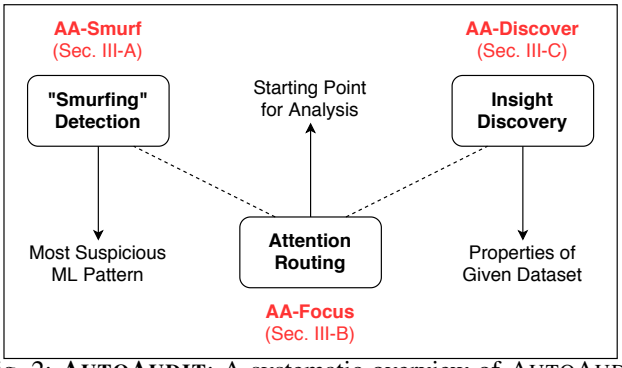


Fig. 2: **AUTOAUDIT**: A systematic overview of AUTOAUDIT

II. PROPOSED METHOD

As shown in Figure 2, AUTOAUDIT solves these three major challenges. We present the details of AA-SMURF on detecting "Smurfing" in Section II-A, and then the design of attention routing in Section II-B. Finally, discovering the properties of given dataset is discussed in Appendix B.

A. AA-SMURF *for "Smurfing" Detection*

1) Problem Definition: We focus on the first placement stage of "Smurfing", and define the problem as follows:

Definition 1 ("Smurfing"): Given a time-evolving accounting dataset without ground truth, the goal is to detect "Smurfing", where there are one sender, one receiver, and k intermediaries as shown in Figure 1. The task should render results understandable to auditors and risk management professionals. There are three basic properties of "Smurfing" derived from the information in [14], which can be used for detection:

- **Intermediary Size:** The more the intermediaries, the more suspicious.
- **Inside Purity:** The less the interactions between intermediaries, the more suspicious.
- **Outside Purity:** The less the intermediaries connected or connecting to normal accounts, the more suspicious.

Although intermediaries usually camouflage themselves by setting up connections with normal accounts, they try their best to avoid interacting with other intermediaries. If there exists interactions between intermediaries, it is hard to conceal information and transactions. Moreover, once one of the intermediaries is exposed, it is much easier to track others.

2) Overall Algorithm: Algorithm 1 gives the steps of AA-SMURF. In line 1, we first collect all possible index sets of sender, intermediaries and receiver $\mathcal{P}$, if the number of intermediaries is higher than 2. In line 3-17, we iteratively identify the best "Smurfing" patterns. The score function is given in line 9, consisting of purity and encoding cost. In each iteration, all the sets $I^{'} \in \mathcal{P}$ are examined to find the highest score s in line 4-10. If no pattern generating lower cost is to be found, in line 18, we identify the final result by picking the subset $\mathcal{I}^*$ of $\mathcal{I}$, Index sets containing $\{I_1, I_2, ...\}$, with encoding cost 10% greater than the minimum (early stopping) as described in [15]. We then detail each of the subroutines in the following paragraphs.

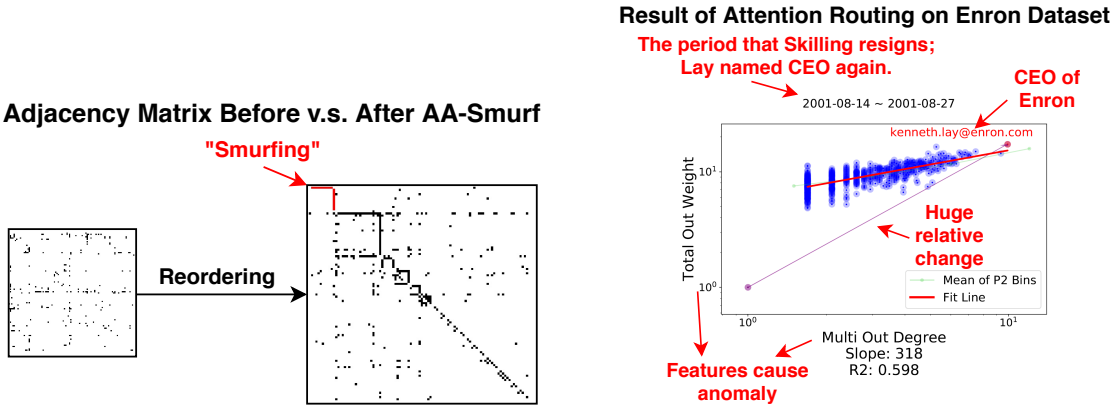
3) Matrix Reordering: Our method is based on the intuition that if an adjacency matrix of graph contains "Smurfing" patterns, after reordering, it is much easier to be compressed. Based on the three aforementioned properties, the optimal matrix containing one "Smurfing" pattern should be as follows:

$$V_{i,j} = \begin{cases} 1 & \text{if } i = 0 \text{ and } j \in [1, k-1] \\ 1 & \text{if } j = k \text{ and } i \in [1, k-1] \; i, j \in [0, n] \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where n is the number of accounts and k is the number of intermediaries. An illustration is shown by the sub-matrix A in Figure 4, where it only contains totally $2k$ elements. The matrix orders of row and column are the same in our method. As shown in Algorithm 2, given a matrix V at iteration J, where $j = 1, ..., J$, we add the indexes of sender, intermediaries and receiver into order $\mathcal{O}$ respectively in line

[1]https://github.com/mengchillee/AutoAudit

[2]Extended version with Appendix: https://arxiv.org/abs/2011.00447



(a) AA-SMURF detects and spots "Smurfing" successfully

(b) AA-FOCUS detects the time of severe change and give the explanation

Fig. 3: AUTOAUDIT contains AA-SMURF (left) to detect "Smurfing" and spots by matrix reordering, and AA-FOCUS (right) to detect the severe event in Enron dataset and explains by the 2-D plot with huge relative change.

Data: Adjacency matrix V from graphs G
Result: Reordered matrix with highest score
1 $\mathcal{P} = GetSets(V), \mathcal{I} = []$;
2 $c = Cost_T(V, [])$;
3 **while** *True* **do**
4 **for** $I^{'} = (i^s, i^m, i^r) \in \mathcal{P} \backslash \mathcal{I}$ **do**
5 $\mathcal{I}^{'} = \mathcal{I} \cup I^{'}$;
6 /* Matrix Reordering § II-A3 */
7 $V^{'} = Reorder(V, \mathcal{I}^{'})$;
8 /* Encoding Cost § II-A4 */
9 $c^{'} = Cost_T(V^{'}, \mathcal{I}^{'})$;
10 /* Purity Calculation § II-A5 */
11 $p^{'} = Purity(V^{'}, \mathcal{I}^{'})$;
12 $s = p^{'} * (c - c^{'})/c$;
13 **end**
14 **if** *No set with cost lower than* c **then**
15 Break;
16 **end**
17 Pick the best V^* and I^* from $\mathcal{P}$ with the largest s;
18 Add I^* into $\mathcal{I}$;
19 $c = Cost_T(V^*, \mathcal{I})$;
20 **end**
21 Pick $\mathcal{I}^* \in \mathcal{I}$ with cost 10% greater than c;
22 Return $Reorder(V, \mathcal{I}^*)$;

Algorithm 1: AA-SMURF: Reorder the adjacency matrix and display the most suspicious patterns

2-6 iteratively. We then reorder the row and column in V by $\mathcal{O}$ in line 8.

4) Encoding Cost of Matrix Reordering: To encode the reordered matrix, we use Minimum Description Length (MDL) principle [16]. It finds the best way to compress matrices based on the aforementioned properties of "Smurfing" patterns. Based on this basic idea, we design a method to compress the binary adjacency matrix. As shown in Figure 4, given a

Data: Matrix V, and index set of sender, intermediaries and receiver $\mathcal{I}$
Result: Reordered matrix $V^{'}$
1 Initialize order $\mathcal{O} = []$;
2 **for** $I_j = (i^s_j, i^m_j, i^r_j) \in \mathcal{I}$ **do**
3 Add sender index i^s_j to $\mathcal{O}$;
4 Add intermediaries indexes i^m_j to $\mathcal{O}$;
5 Add receiver index i^r_j to $\mathcal{O}$;
6 **end**
7 Add the rest indexes into $\mathcal{O}$;
8 $V^{'}$ = reorder the rows and columns of V by $\mathcal{O}$;
9 Return $V^{'}$;

Algorithm 2: Reorder: Reorder the matrix at iteration J by the indexes of sender, intermediaries and receiver

reordered adjacency matrix V_J and index information $\mathcal{I}$ at iteration J, we denote intermediaries number as $k_j = |i^m_j|$ and start indexes as $r_j = \sum_{j'=1}^{j-1} |I_{j'}|$ of "Smurfing" patterns, and separate the reordered adjacency matrix into sub-matrices as follows:

$$\begin{aligned} A_j(V_J, I_j) &= V_J[r_j : r_j + k_j][r_j : r_j + k_j] \\ B_j(V_J, I_j) &= V_J[r_j : r_j + k_j][r_j + k_j + 1 : n] \\ C_j(V_J, I_j) &= V_J[r_j + k_j + 1 : n][r_j : r_j + k_j] \end{aligned} \quad (2)$$

where $j = 1, ..., J$ and $I_j \in \mathcal{I}$. Sub-matrix A_j denotes the potential "Smurfing" pattern separated out by reordering. Sub-matrix B_j denotes intermediaries sending money, and sub-matrix C_j denotes intermediaries receiving money. Sub-matrix D is used to denote errors which cannot be compressed by found patterns, and is defined as follows:

$$D(V_J, I_J) = V_J[r_J + k_J + 1 : n][r_J + k_J + 1 : n] \quad (3)$$

Based on the inside purity and outside purity properties, in order to have fewer interactions, the encoding cost of sub-

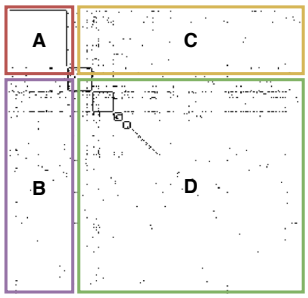


Fig. 4: Reordered binary adjacency matrix with "Smurfing", where sub-matrix A denotes "Smurfing" and the inside purity, sub-matrix B and C denote the outside purity, and sub-matrix D denotes the errors

matrices A, B and C can be computed as follows:

$$\begin{aligned} Cost_A(V_J, \mathcal{I}_J) &= (\textstyle\sum_{j=1}^{J} |A_j(V_J, I_j)| - 2k_j) * (2 \lg (k_j - 1)) \\ Cost_B(V_J, \mathcal{I}_J) &= (\textstyle\sum_{j=1}^{J} |B_j(V_J, I_j)|) * (\lg n + \lg k_j) \\ Cost_C(V_J, \mathcal{I}_J) &= (\textstyle\sum_{j=1}^{J} |C_j(V_J, I_j)|) * (\lg n + \lg k_j) \end{aligned} \tag{4}$$

This function encodes the position of non-zero elements in the given matrix. We subtract $2k_j$ for each A_j because there is no need to record the occurrence of intermediaries. To ensure that most zeros can be compressed in A_j, B_j, and C_j, sub-matrix D should contain as few zeros as possible, which can be calculated as follows:

$$Cost_D(V_J, \mathcal{I}_J) = (|\mathbb{J} - D(V_J, I_J)|) * (2 \lg n) \tag{5}$$

where $\mathbb{J}$ is a matrix of ones. This means that instead of encoding the positions of non-zeros, we encode the positions of zeros in the sub-matrix D_j. To encode real numbers, we use universal code length $lg^*(x) \approx 2lg(x) + 1$. The number of found patterns is encoded by $lg^*(J)$. The indexes for each sender, receiver and intermediaries need $lg(n)$ for each to be encoded. The total cost can then be formed as follows:

$$\begin{aligned} Cost_T(V_J, \mathcal{I}_J) &= Cost_A(V_J, \mathcal{I}_J) + Cost_B(V_J, \mathcal{I}_J) \\ &+ Cost_C(V_J, \mathcal{I}_J) + Cost_D(V_J, \mathcal{I}_J) \\ &+ lg^*(J) + |\mathcal{I}_J| * \lg n \end{aligned} \tag{6}$$

To compute the final score, we use compression rate instead of the actual cost. The compression rate is calculated by normalizing the difference of original and compressed MDL by the original MDL, i.e, $(c - c^{'})/c$, where c denotes the encoding cost of the last iteration and $c^{'}$ denotes the current encoding cost.

5) Purity Calculation: It is obvious that if the numbers of intermediaries increase, the matrix tends to have a higher compression rate. When only using compression rate to select an anomaly, the one with more intermediaries will have a higher chance to be selected, even when its purity is low. To solve this issue, we develop a new score called purity. At iteration J, the purity score p can be computed as follows:

$$Purity(V_J, \mathcal{I}_J) = \textstyle\sum_{j=1}^{J} \frac{2k_j}{|A_j(V_J, I_j)| + |B_j(V_J, I_j)| + |C_j(V_J, I_j)|} \tag{7}$$

where $0 < p \leq 1$. In the best case, where $p = 1$, the edges will only connect from the sender to intermediaries, and from intermediaries to the receiver. We then multiply the compression rate by purity to get the final score s of a given index set.

6) Complexity Analysis:

Lemma 1: AA-SMURF takes time

$$O(nz + |\mathcal{P}|^2) \tag{8}$$

where z denotes the number of non-zero elements and $|\mathcal{P}|$ denotes the number of possible index sets.

Proof. By using the sparse matrix computation, the time complexity of obtaining possible index sets can be reduced to $O(nz)$, where z denotes the number of non-zero elements. The worst case of the while loop in the algorithm is that every iteration we only remove one set from $\mathcal{P}$, taking $O(\frac{(|\mathcal{P}|+1)\mathcal{P}}{2})$ to run (sum of an arithmetic progression). In practice, the number of iterations is linear to the input size, since each account cannot be reordered twice. If an account is identified as an intermediary, the possible sets involving it are removed. The complexity of reordering is $O(|\mathcal{I}|)$, where $|\mathcal{I}|$ is usually a small constant that is negligible. Thus, the complexity is $O(nz + |\mathcal{P}|^2)$. ■

B. AA-FOCUS *for Attention Routing*

1) Problem Definition: In a huge graph dataset, finding the starting point for analysis can be extremely difficult. It becomes even more challenging when the dataset is dynamic, since neither analyzing journal entries entered at a specific time point nor the whole time span would be effective and efficient for detecting changes occurring only in a short period. Furthermore, it is equally important to offer viable explanations for these changes after detecting them. One thing worth noting is that these changes may not always be anomalies, they may also be related to other types of important events. Here we define the Attention Routing problem as follows:

Definition 2 (Attention Routing): Given time-evolving graph datasets, the goal is to detect sudden changes over time, and use the least resources to explain the most patterns.

2) Graph Construction: We first split the time-evolving graph G into I graphs G_i over time by sliding windows with overlap. In Algorithm 3 line 1-4, we construct bipartite graphs W_i for each graph G_i, where the source nodes on the left side denote each node in G_i and the destination nodes on the right side denote explainable focus-plots. We focus on drawing visualizable and understandable 2-d focus-plots, which scatters each data point by only two features. There are total $\binom{q}{2}$ focus-plots generated from the features of nodes, where q denotes the number of extracted features from nodes. In each focus-plot, the anomaly score of each node is calculated by an off-the-shelf unsupervised anomaly detection model. We use Isolation Forest [17] because of its efficiency and robustness. The edge weight w^i of the bipartite graph W_i is then set as the anomaly score of nodes for each focus-plot.

Data: Graphs G split from sliding window
Result: Time t_a with the highest change score and v 2-d focus-plots
1 /* Graph Construction § II-B2 */
2 **for** $G_i \in G$ **do**
3 Extract node feature pairs f_i from G_i ;
4 Construct bipartite graph W_i;
5 **end**
6 /* Sketch Generation § II-B3 */
7 **for** $j = 1, ..., l$ **do**
8 Randomly select $\mathcal{S}_j$ and $\mathcal{D}_j$;
9 Select v focus-plots $\mathcal{V}_j$ from $\mathcal{D}_j$ by Eq. 11;
10 **end**
11 Generate l-d sketch K for all $\mathcal{S}$ and $\mathcal{V}$;
12 /* Change Point Detection and Attention § II-B4 */
13 **for** $i = t, ..., len(G)$ **do**
14 Compute principal eigenvector e_i by SVD from $K_{i-t:i-1}$;
15 Compute cosine distance between e_i and K_i as change score over time;
16 **end**
17 Return t_a with maximum change score and v focus-plots;

Algorithm 3: AA-FOCUS: Attend to the most suspicious time period and explain by the focus-plots

3) Sketch Generation: Once we have bipartite graphs for each timestamp, to detect change in the anomaly scores of each node, each bipartite graph is encoded into sketches to apply further analysis. We borrow the idea from SpotLight [10]. l subsets of source $\mathcal{S}$ and l subsets of destination $\mathcal{D}$ of bipartite graphs have been randomly selected with a given probability, where l denotes the dimensions of sketches. The sketch value in dimension j can then be calculated as follows:

$$f(\mathcal{S}_j, \mathcal{D}_j) = \sum_{s_x \in \mathcal{S}} \max_{d_y \in \mathcal{D}} \sum_{i=1}^{I} w^i_{s_x, d_y} \tag{9}$$

where f denotes the sum of the edge with maximum weight linking from source subset to corresponding destination subset in all W. The total weight of edge (s_x, d_y) in in all W gives a global view of how well this focus-plot d_y explaining s_x.

The difference between our method and SpotLight is that we improve the selection of destination subsets. After randomly selecting destinations (focus-plots) of sketch K_j, we aim to find the subset of $\mathcal{D}_j$, denoted as $\mathcal{V}_j$, which maximally explains the subset of source (nodes) $\mathcal{S}_j$. The gains of adding each focus-plot is then computed by marginal gains [18] as follows:

$$\Delta_f(d|\mathcal{V}^i_j) = f(\mathcal{V}^{i-1}_j \cup \{d\}) - f(\mathcal{V}^{i-1}_j) \tag{10}$$

As shown in line 5-8 of Algorithm 3, we iteratively add the destinations that maximize the marginal gain for v iterations:

$$\mathcal{V}^i_j = \mathcal{V}^{i-1}_j \cup \{\max_{d \in \mathcal{D}_j \setminus \mathcal{V}^{i-1}_j} \Delta_f(d|\mathcal{V}^i_j)\} \tag{11}$$

where $i = 1, ..., v$ and $j = 1, ..., l$. This ensures that the v selected focus-plots have the best performance on explaining the nodes.

4) Change Point Detection and Attention: Having the sketch for each timestamp, we use [12] to measure distance between current and past behavior as the changing behavior of graphs. In Algorithm 3 line 10-13, at timetick i, the behavior of past t timeticks can be summarized by computing the principal eigenvector e of the matrix $K_{i-t:i-1}$ sketches by Singular Value Decomposition (SVD). We use cosine distance to measure the difference as the change score, and identify the dimension with the largest relative change. After finding that dimension, we have a subset of nodes and v focus-plots, where v is the number of plots to output, which give the most informative explanation to this changed behavior.

5) Complexity Analysis:

Lemma 2: AA-FOCUS takes time

$$O(|G|(q^2 f\varphi \log \varphi + tl)) \tag{12}$$

where f and φ denote the number of trees and the max sample size in Isolation Forest, and $|G|$ is the number of graphs over time.

Proof. Isolation Forest takes $O(f\varphi \log \varphi)$ to run, where f denotes the number of trees and φ denotes the max sample size. Since each focus-plot needs to run Isolation Forest once, the complexity to construct a bipartite graph takes $O(q^2 f\varphi \log \varphi)$. For total $|G|$ graphs over time, the complexity of graph construction is $O(|G|q^2 f\varphi \log \varphi)$. Moreover, $O(l)$ is needed to generate the sketches, and $O(|G|tl)$ is needed to extract the principled eigenvectors by SVD. Therefore, the complexity of AA-FOCUS is $O(|G|(q^2 f\varphi \log \varphi + tl))$. ■

III. EXPERIMENTS

In this section, we present our experimental results to answer the following questions:

Q1. **Effectiveness**: How well does AUTOAUDIT work on real-world datasets?
Q2. **Scalability**: How does AUTOAUDIT's running time grow with input size?
Q3. **Discovery**: What is the most interesting thing found by AUTOAUDIT? (See Appendix D and E)

We implement AUTOAUDIT in Python; experiments are run on a 3.2 GHz CPU with 256 GB RAM on a Linux server.

In our experiments, we use three real-world graph datasets. The Accounting dataset from an anonymous institution contains transactions and accounts from one company, which precisely reflects the money flow inside the company. In addition to the Accounting dataset, we include two more real-world time-evolving graph datasets. The Czech Financial dataset [19] contains transactions between customers (inside the bank) and clients (outside the bank), and there is no interaction between customers and clients. The Enron Email dataset [20] collects emails at Enron from about 1998 to 2002, and we used the data from 2001 for concision in our experiment. None of these datasets come with ground truth.

The graphs we used in our experiments are described in the Table I.

Real-world Graph Datasets

Name	Nodes	Edges	Time Span
Accounting	254	285,298	01/01/2016 to 02/06/2017
Czech Financial	11,374	273,508	01/05/1993 to 12/14/1998
Enron Email	16,771	1,487,863	01/01/2001 to 12/31/2001

TABLE I: The summary of real-world graph datasets used

A. *"Smurfing" Detection*

1) Setup: To evaluate AA-SMURF, we inject anomalies in our Accounting and Czech datasets. In the Accounting dataset, we randomly pick one sender and one receiver, and then pick n intermediaries. The interactions between intermediaries have been deleted to mimic "Smurfing". In the Czech dataset, sender and receiver are randomly picked from client accounts, and intermediaries are picked from customer accounts. This imitates "Smurfing" where money is imported from outside of the bank and transferred out. The interaction between customer accounts are randomly generated with the probability of 0.05%. We injected the datasets with intermediaries from 10 to 50 respectively. Noise patterns with either more intermediaries or more interaction between intermediaries are randomly added to test robustness. For each number of intermediaries we run experiments for 10 times to ensure low variance. Because of no existing baseline, so we run an ablation study like evaluation instead. Baseline AA-SMURF w/o MDL and AA-SMURF w/o Purity only contain one of the two ingredients in score function (purity or encoding cost). Baseline number of intermediaries directly detects "Smurfing" by outputting the pattern with the highest number of intermediaries.

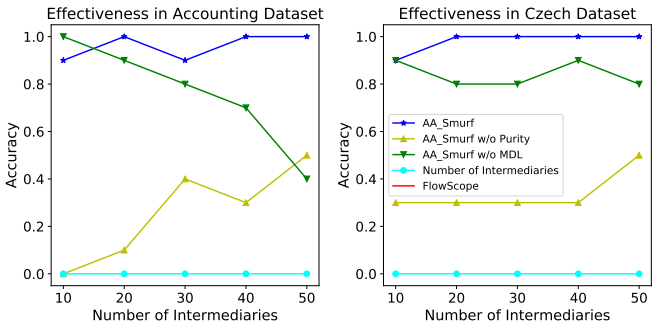


Fig. 5: **AA-SMURF**: AA-SMURF achieves nearly 100% accuracy and outperforms other baselines in Accounting Dataset and Czech Dataset

2) Effectiveness: As shown in Figure 5a and Figure 5b, AA-SMURF successfully detects "Smurfing" patterns in both Accounting and Czech datasets and outperforms other baselines by achieving nearly 100% accuracy. Baseline number of intermediaries performs the worst, indicating the necessity of designing a "Smurfing" detection method. Moverover, FlowScope [4] is not usable in our problem because they enabled intermediaries can have many interactions. In Figure 3a, we also can see AA-SMURF reorders the matrix by putting the most suspicious patterns in the front, so that they can be more easily noticed by auditors and risk management professionals.

3) Scalability: To evaluate the scalability of AA-SMURF, we vary the number of possible index sets, which highly correlate with running time. As shown in Figure 6a with black triangles, AA-SMURF scales linearly with the number of possible index sets $\mathcal{P}$ with only 1 machine. We also parallelized it by 4 machines to get a much higher efficiency, shown in Figure 6a with red squares. To demonstrate that AA-SMURF is highly parallelizable, we vary the number of machines and report $\frac{T_1}{T_M}$, where T_1 denotes the running time by 1 machine, and M denotes the number of machines. In Figure 6b, we find that it achieves a 4.5 times speedup by only 12 machines.

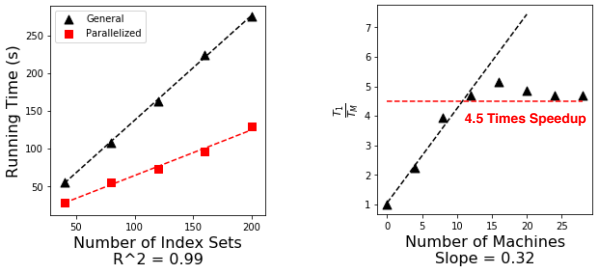
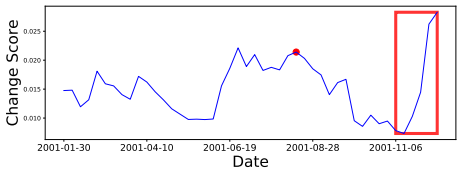


Fig. 6: **AA-SMURF**: It scales linearly with number of index sets by one machine (left), and speeds up if running with more than 1 machine (right).

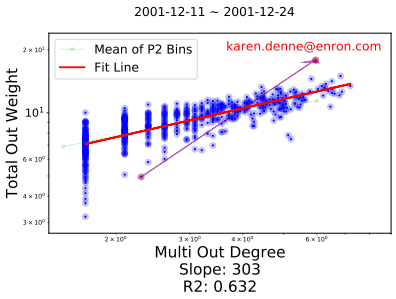
B. *Attention Routing*

1) Setup: In this experiment, graphs are separated along time by 14-day time window with 50% overlap, and then we extract all the following features from graphs: (i) Unique In Degree (ii) Multi In Degree (iii) Unique Out Degree (iv) Multi Out Degree (v) Total In Weight (vi) Mean In Weight (vii) Median In Weight (viii) Variance In Weight (ix) Total Out Weight (x) Mean Out Weight (xi) Median Out Weight (xii) Variance Out Weight, where multiplicity denotes repetition. These features have been transformed into logarithm scale before being input into Isolation Forest. The dimension of each sketch is set to be 256.

2) Effectiveness: To test our effectiveness, we test AA-FOCUS on the Enron Email dataset, with which we could access accurate events on the news timeline. In Figure 7a, we find that the change score fluctuates over time. The red dot in Figure 7a corresponds to the time when Jeff Skilling resigned after releasing the August 2001 quarterly financial report and Kenneth Lay returned to being the chief executive. In Figure 3b, the unique out degree and median out weight of Kenneth Lay increased dramatically in a short time, reflecting this to be an important event. Figure 7a (red box) shows the period that Enron declared bankruptcy. As shown in Figure 7b, Karen Denne, the spokeswoman of Enron, have been busy handling media interviews, explaining the remarkable increase of the total out weight and multi out degree.



(a) The change score over time, where the significant changes are shown in red point and red box



(b) Karen Danne had became more active after the company declared bankruptcy, reflecting on her email length and number

Fig. 7: **Attention Routing**: In Enron dataset, we capture the change when the CEO returned with company's bankruptcy

3) Scalability: To evaluate the scalability of AA-FOCUS, we empirically vary the number of focus-plots and the dimension of sketch. As shown in Figure 8a and Figure 8b with black triangles, AA-FOCUS scales linearly with both features while only using 1 machine; the efficiency boosts up after parallelizing it by 4 machines, shown with red squares. Furthermore, we vary the number of machines to show that it is easy to parallelize and for 4 times speedup by 8 machines.

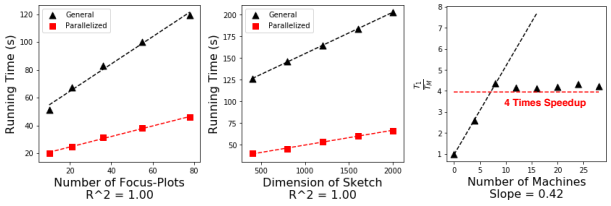


Fig. 8: **AA-FOCUS**: It scales linearly with number of focus-plots (left) and dimension of sketch (middle) by one machine, and speeds up if running with more than 1 machine (right).

IV. CONCLUSIONS

We present AUTOAUDIT (AA), which addresses the anomaly detection problem on time-evolving accounting datasets. This kind of data is usually complicated and hard to organize. Our main purpose is to automatically spot anomalies, such as money laundering, providing huge convenience for auditors and risk management professionals. Our approach is also general enough to be easily modified to solve problems in different domains. The following are our major contributions:

1) AA-SMURF, a parameter-free and unsupervised method to detect "Smurfing" type money laundering.
2) AA-FOCUS, attend to the most suspicious plots and accounts in time-evolving graphs.
3) AA-DISCOVER, discover month-pairs with high correlation, proved by "success stories", and patterns in accounting dataset follow Power-Laws in log-log scales.
4) Scalability with running time linear in input size, and generality on other real-world graph datasets.

ACKNOWLEDGMENT

This research was partially supported by Ministry of Science and Technology Taiwan under grant no. 109-2218-E-009-014.

REFERENCES

[1] A. Chadha and P. Kaur, "Handling smurfing through big data," in *Big Data Analytics*. Springer, 2018, pp. 459–470.
[2] E. L. Paula, M. Ladeira, R. N. Carvalho, and T. Marzagão, "Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering," in *ICMLA*. IEEE, 2016.
[3] D. Savage, Q. Wang, P. Chou, X. Zhang, and X. Yu, "Detection of money laundering groups using supervised learning in networks," *arXiv preprint arXiv:1608.00708*, 2016.
[4] X. Li, S. Liu, Z. Li, X. Han, C. Shi, B. Hooi, H. Huang, and X. Cheng, "Flowscope: Spotting money laundering based on graphs."
[5] P. Reuter and E. M. Truman, "Money laundering: Methods and markets," *Chasing Dirty Money: The Fight Against Money Laundering. Peterson Institute*, 2003.
[6] J. Madinger, *Money laundering: A guide for criminal investigators*. CRC Press, 2011.
[7] L. Corselli, "Italy: money transfer, money laundering and intermediary liability," *Journal of Financial Crime*, 2020.
[8] S. Schneider, *Money laundering in Canada: an analysis of RCMP cases*. Nathanson Centre for the Study of Organized Crime and Corruption Toronto, 2004.
[9] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova, "Anomaly detection in dynamic networks: A survey," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 7, no. 3, 2015.
[10] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, "Spotlight: Detecting anomalies in streaming graphs," in *KDD*, 2018, pp. 1378–1386.
[11] E. A. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," in *KDD*. ACM, 2016, pp. 1035–1044.
[12] L. Akoglu and C. Faloutsos, "Event detection in time series of mobile communication graphs," in *Army science conference*, vol. 1, 2010.
[13] D. Eswaran and C. Faloutsos, "Sedanspot: Detecting anomalies in edge streams," in *ICDM*. IEEE, 2018, pp. 953–958.
[14] M. Levi and P. Reuter, "Money laundering," *Crime and Justice*, vol. 34, no. 1, 2006.
[15] Y. Matsubara, Y. Sakurai, and C. Faloutsos, "Autoplait: automatic mining of co-evolving time sequences," in *SIGMOD Conference*. ACM, 2014.
[16] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, 1978.
[17] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *ICDM*. IEEE, 2008.
[18] N. Gupta, D. Eswaran, N. Shah, L. Akoglu, and C. Faloutsos, "Lookout on time-evolving graphs: Succinctly explaining anomalies from any detector," *CoRR*, vol. abs/1710.05333, 2017.
[19] P. Berka, "Workshop notes on discovery challenge pkdd'99," 1999.
[20] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in *ECML*. Springer, 2004, pp. 217–226.
[21] D. H. Chau, "Data mining meets hci: Making sense of large graphs," CMU Pittsburgh PA Machine Learning Dept, Tech. Rep., 2012.
[22] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, "Netprobe: a fast and scalable system for fraud detection in online auction networks," in *WWW*. ACM, 2007, pp. 201–210.
[23] D. H. . Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, "Polonium: Tera-scale graph mining and inference for malware detection," in *SDM*. SIAM, 2011, pp. 131–142.
[24] P. Devineni, D. Koutra, M. Faloutsos, and C. Faloutsos, "If walls could talk: Patterns and anomalies in facebook wallposts," in *ASONAM*, 2015.

APPENDIX

A. Background and Related Work

1) Anti-Money Laundering (AML): Money laundering has been a long-term trouble to the modern financial system. Many scholars and researchers have tried to combat "Smurfing". As shown in Figure 1, it contains three major stages: placement, layering and integration. Placement denotes injecting a huge amount of money from illegal activities to financial systems; layering separates the money into several parts to avoid Suspicious Activity Report. Finally, these money will be integrated into the target account.

On one hand, some studies used transaction-based AML to detect "Smurfing". [1] used Hadoop MapReduce to efficiently examine the transactions, and transactions that exceeds the fine-tuned thresholds will be detected. [2] used AutoEncoder to calculate the reconstruction error for anomaly detection. The higher the error, the more suspicious the transaction. However, these methods ignore the dependence between accounts, so that they are more likely to trigger false alarms.

On the other hand, considering transactions as a graph is a perfect solution for clarifying interactions between accounts. [3] combined network analysis with supervised learning to detect the groups of money laundering activities. Nevertheless, these methods highly rely on carefully labelled data, where the availability is low and the cost is significant. Thus, [4] proposed to approach AML as an optimization problem by using multipartite graphs. Nevertheless, it requires a lot of senders and receivers. To ameliorate the aforementioned problems, we propose a more practical definition, and carry out the result with unsupervised method.

2) Anomaly Detection in Time-Evolving Graphs: Anomaly detection is one of the closest real-world applications in graph mining, which has been well-studied in the past few years. Time-evolving graphs are dynamic and sudden changes are usually inklings of potential anomalies. Careful scrutiny of this field can be found in [9]. In addition, [10] proposed a randomized sketching-based approach to detect sudden changes in streaming graphs, and theoretically proved that anomalies lie far apart from normal instances. StreamSpot [11] is a clustering-based method that detects anomalies in streaming heterogeneous graphs with dynamic maintenance. An algorithm is proposed to detect changing points in time-evolving graphs by the time series feature extracted from all nodes [12]. [13] proposed their method based on two suggested signs of anomalous edges; sudden occurrence and connection with sparsely-connected parts.

3) Attention Routing: This term was first coined in [21], which is used to increase the quality of analyzing massive networks. The major purpose is to find the most critical nodes and subgraphs. [22] designed a system to detect fraud users in a large online auction dataset, modeling users and transactions as a Markov Random Field. [23] detected the malware in a tera-scale bipartite graph based on scalable Belief Propagation algorithm. LookOut [18] enhanced the interpretability and succinctness, only showing a limited number of plots which maximize the anomaly score.

B. AA-DISCOVER *for Insight Discovery*

Effective audits require auditors to identify risks through understanding an entity—its environment, operation, and control activities. The real-world challenge has been the lack of fast tools to process the massive amount of detailed transaction data and establish robust patterns regarding an entity's business operations and accounting practices. Quickly discovering periodic temporal correlations in large accounting data is profoundly meaningful because it aids auditors in better accessing risk and detecting aberrant transactions.

1) Problem Definition: Temporal correlations usually exist in a time-evolving graph dataset, which are worth studying because they are useful in discovering the behavior rules of nodes. The nodes that exhibit unusual behavior can be treated as outliers. The distributions of different features in the dataset also give us abundant information. Here we define the problem in the following:

Definition 3 (Insight Discovery): Given a time-evolving graph dataset, there are two major goals to discover the insights of dataset:

- unearth the correlation between time frames and the most significant factor incurring this phenomenon, and
- examine the distributions to verify whether they follow Power-Law patterns.

2) Information Extraction and Visualization: SVD is a method of matrix decomposition which is extremely useful in extracting essential information. It decomposes the matrix into U, the columns containing left singular vectors, Σ with singular values on its diagonal, and V^T containing right singular vectors. To further investigate the correlation between months, we form a 2-dimensional matrix as shown in Figure 9, where the first dimension is the pairs of the source and destination accounts, and the second dimension is months. We use SVD to decompose this matrix to derive three matrices U, Σ, and V^T. The rows of V^T illustrate correlations between each month, and the corresponding columns of U explain the correlations according to the behaviors of each pair. Therefore we can project rows and columns on a 2-dimensional plane, and the lower the distance between the elements of rows and columns, the higher the relevance. This visualization can be used to discover active months of each account pair.

3) Power-Law Pattern: Log-logistic distribution has been well studied and used in economics for many years, which reflects how the "rich get richer." A continuous random variable follows the log-logistic distribution, if and only if its logarithm follows the logistic distribution [24]. One of the best properties of the log-logistic is that its odds ratio follows the Power-Law, which is the ratio of Cumulative Distribution Function (CDF) over the complementary CDF (CCDF). We use Linear Regression to fit odds ratio and calculate the slope ρ. We then derive the equation as follows:

$$\beta = -\rho \tag{13}$$

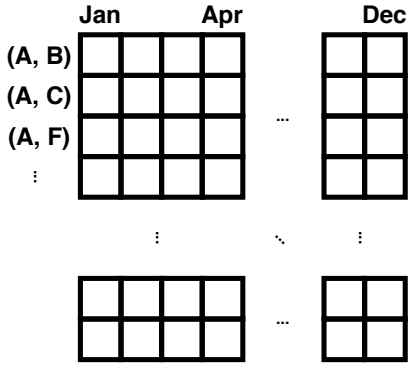


Fig. 9: A 2-dimension matrix where the first dimension is the pair of accounts, and the second dimension is months

where β denotes the slope of CCDF and ρ denotes the slope of odds ratio. This equation is used to examine whether a distribution follows the Power-Law in the experiments.

C. Additional Experiment Results for "Smurfing" Detection

To implement AA-SMURF on streaming graphs, we further calculate the compression rate by encoding cost over time. As shown in Figure 10, AA-SMURF handle streaming graphs well, where it successfully detects "Smurfing" pattern by giving the highest compression rate over time.

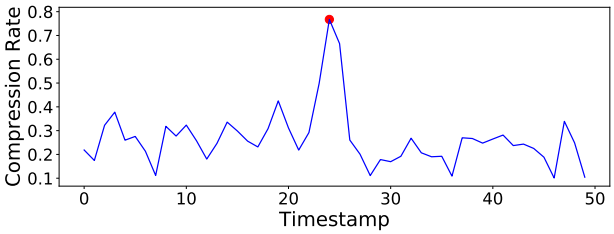
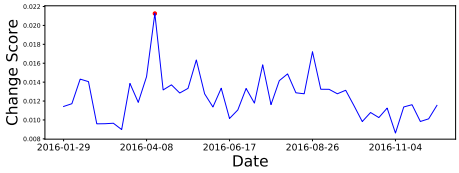


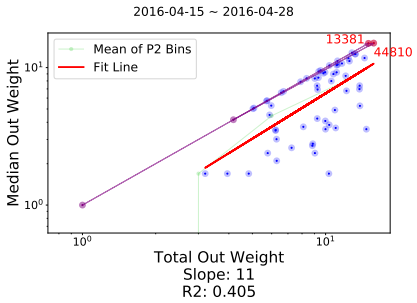
Fig. 10: **AA-SMURF**: AA-SMURF detects "Smurfing" in the time-evolving dataset on the red point with high compression rate

D. Additional Experiment Results for Attention Routing

We use AA-FOCUS to investigate activities in the Accounting dataset. As shown in Figure 11a, the change score reaches the highest peak at the detected red point in the last week of April. Figure 11b further shows during this focal week, the change of the Cost of Good Sold (COSG) account 44810 is significant in terms of median and total out weight. The COSG account moves from the purple dot to the red dot, where the purple dot summarizes the account behavior during the four weeks before the focal week, and the red dot describes the account behavior in the focal week. During previous four weeks, the median and total out weight do not equal zero (purple dot), which indicates a small movement of the COGS account in terms of the account balance. The median and total out weight then increase dramatically in the focal week (red dot), suggesting the occurrence of exceptionally significant



(a) The change score over time, where the significant change is shown in red point



(b) The explanation where account 44810 and 13381 had significant changes in terms of median and total out weight

Fig. 11: **Attention Routing**: In Accounting Dataset, we discovery manufacturer's alternation of information system

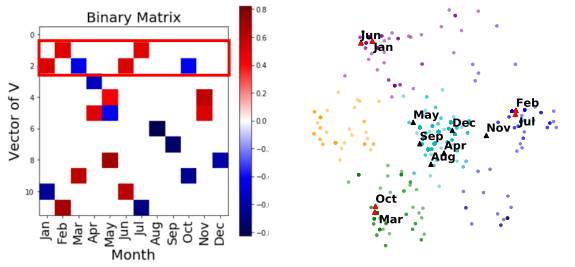
movement concerning the COGS account balance. According to Figure 11b, similar implications can be drawn for the Finished Goods Inventory account 13381.

E. Additional Experiment Results for Insight Discovery

In this experiment, we selected the data from a general ledger database of an anonymous manufacturer in 2016.

1) Temporal Correlation: In order to further illustrate the factors causing this phenomenon, we extract the information by SVD described in Section B2. As shown in Figure 12a, V_1^T and V_2^T rows reflect the similarities between each month-pairs. We then project them into a 2-d plane marked by red triangles as shown in Figure 12b. To further illustrate the relation between month-pairs and account-pairs, we project the corresponding U_1 and U_2 columns marked by dots on the same plane, where the colors are generated by K-Means. The clusters around month-pairs are the activated account-pairs in corresponding month-pairs, which we call them "petals". For example, the purple cluster are the account-pairs that were active only in January and June. The yellow cluster on the left of the plane denotes those account-pairs being in two month-pairs, i.e., (Jan, Jun) and (Mar, Oct). Moreover, the cluster in the middle of the plane is called "stamen", which contains months with no correlation. These "petals" and "stamen" give us extremely intuitive information to investigate correlations between months.

The anonymous manufacturer we investigate conducted onboard training of new employees in March and October. Typically, such events are held in the selected months of the year (and perhaps during the downtime of the employees affected). When such events happen, certain account-pairs



(a) The rows of matrix V containing right singular vectors

(b) The account pairs are gathered into clusters on U_1 and U_2 plane

Fig. 12: **Temporal Correlation**: We discover the temporal correlation between months

are activated, such as the cost of issuing an award to the best-performing employees and a specific funding source is usually used for these HR events. As a result, the account-pair from the funding source to the award expense only takes place in the two months of the year. Meanwhile, this manufacturer set its intercompany reconciliations in March and October, possibly because of the relatively more intense waves of intercompany transactions in these months. These waves of intercompany transactions and scheduled reconciliations would trigger the pairings between intercompany accounts. As shown in Figure 12b, March and October are clustered as a similar month-pair.

2) Patterns Following Power-Law: In the accounting dataset, we draw several distributions of CCDF and Odds Ratio and surprisingly find that log-logistic distribution fits well. The features we extracted for computing distribution are as follows: (i) Transaction Amount (ii) Interval Arriving Time (iii) Pair Amount (iv) Pair Multiplicity (v) Out Weight / 1000 (vi) In Weight / 1000 (vii) Multi Out Degree (viii) Multi In Degree (ix) Unique Out Degree (x) Unique In Degree. We use account-pair as the unit of distributions (ii) and (iii). Weight is divided by 1000 to eliminate small numbers. Multi degree means that the degree to one specific node could be more than one, where in a unique degree there will only be one. As shown in Table II, the results demonstrate the accuracy of Equation B3. Moreover, there are three essential exponents 1, 0.6, and 1.6, and all distributions in the Accounting dataset follow these three exponents. For distributions from (i) to (ii), ρ usually equals to 1; for the ones from (iii) to (viii), ρ usually equals to 0.6; for the others from (ix) to (x), ρ usually equals to 1.6.

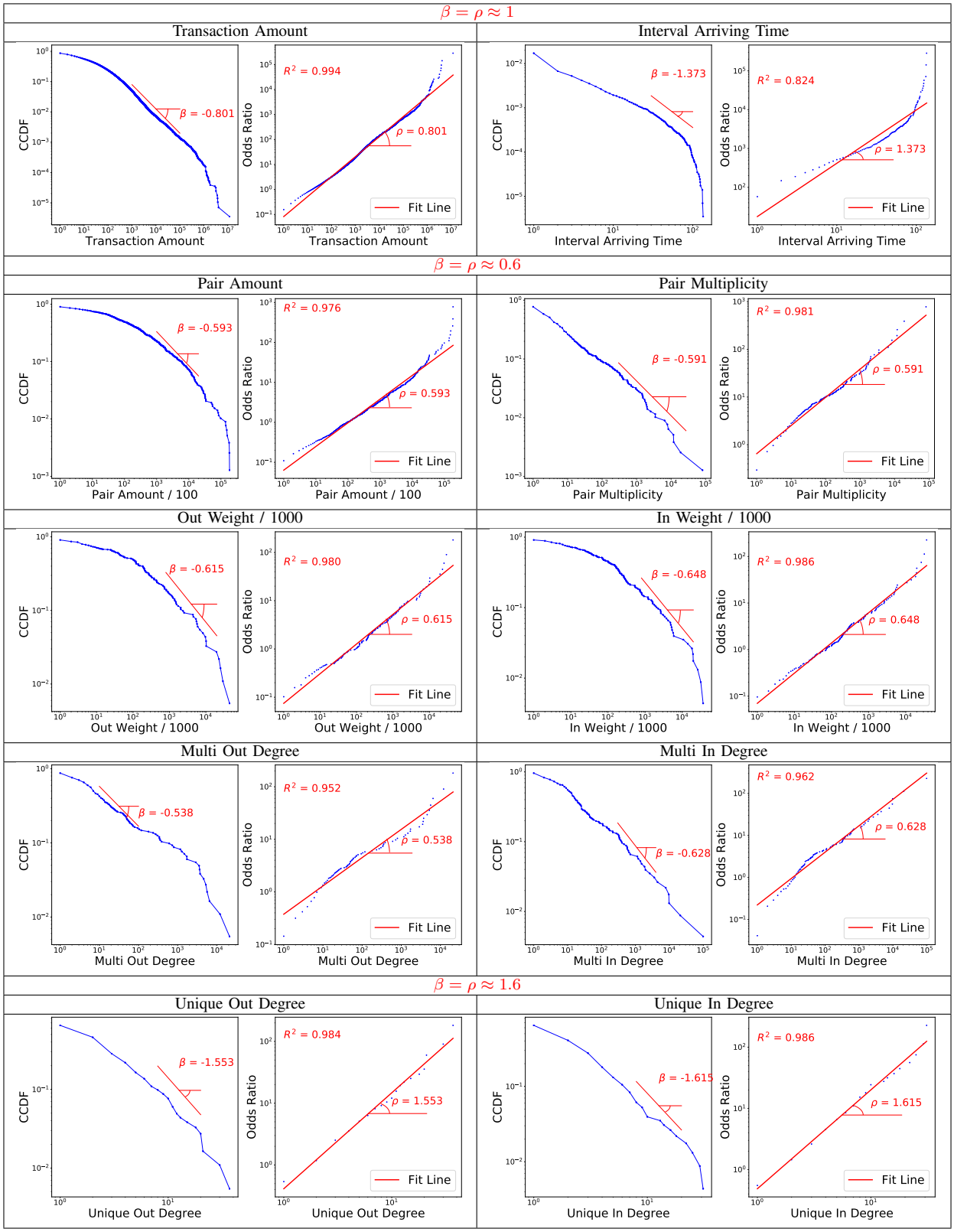


TABLE II: **AA-DISCOVER**: AA-DISCOVER discovers the patterns in accounting dataset follow Power-Law.