Supplier Impersonation Fraud Detection in Business-To-Business Transaction Networks using Self-Organizing Maps

Rémi Canillas[1,2], Omar Hasan[2], Laurent Sarrat[1], and Lionel Brunie[2]

[1] SiS-id, Lyon, France
{name}@{surname}.sisnet.fr
[2] Liris, INSA Lyon, Lyon, France
{name}@{surname}.insa-lyon.fr

Abstract. Supplier Impersonation Fraud (SIF) is a rising issue for Business to Business companies, as the use of remote and quick digital transactions has made the task of identifying fraudsters more difficult. In this paper, we propose data-driven fraud detection system whose goal is to provide an accurate estimation of transactions' legitimacy by using the knowledge contained in the network of transactions created by the interaction of a company with its supplier. We consider the real dataset collected by SIS-ID for this work.
We propose to use a graph-based approach to design an Anomaly Detection System (ADS) based on a Self-Organizing Map (SOM) allowing us to label a suspicious transaction as either legitimate or fraudulent based on its similarity with frequently occurring transactions. Experiments demonstrate that our approach identifies fraudulent transactions with high success in a real-life dataset.

Keywords: fraud detection, graph-based feature engineering, financial networks, B2B network

1 Introduction

Lately, Supplier Impersonation Fraud (SIF) is on the rise, resulting in the loss of hundreds of thousands of Euros in 2018, and ranked 1st most frequent fraud affecting French companies in the latest survey about cyber-criminality conducted in 2019 by Euler Hermes and DFCG [4]. Supplier impersonation consists in a fraudster impersonating a member of a company providing goods and services to another in order to trigger a payment on an account controlled by the fraudster [1]. More and more companies are using digital tools to process, authorize, or even conduct transactions due to numerous advantages provided by digitalization such as the ability to conduct transactions all over the globe in a timely fashion. However, digital transactions make frauds against companies more effective, firstly due to the difficulty to formally identify and trust remote interlocutors that are sometimes geographically very distant from the company headquarters, and secondly due to the increased speed of wired transactions,

allowing money to be moved between accounts in a very short amount of time, thus hindering the process of recovering it after a fraud.

In this work, we present GraphSIF, a SIF detection system that uses a B2B transactions dataset to construct a network modeling the relationships between client and supplier companies in a B2B ecosystem, and describe how these relationships can be used to derive useful knowledge in order to assert the legitimacy of transactions.

We use the transaction network to create a time-evolving behavior sequence summing up the evolution of the graph through time. We then compare the new graph created by adding a suspsicious transaction to the behavior sequence and investigate the potential discrepancy it introduces. If this discrepancy is low then the transaction is considered as legitimate, and if the discrepancy is high then the transaction is considered as likely fraudulent. In order to quantify this discrepancy, a Self-Organizing Map (SOM) is trained on the behavior sequence, and a clustering algorithm is used to quantify the similarity of the tested graph with the ones in the behavior sequence.

The contributions of this paper are the following: a graph-based feature engineering process relying on a bipartite graph constructed from transactions in a B2B context, a classification system that uses Self-Organizing Maps and K-means clustering to investigate the legitimacy of a new transaction, and a comprehensive evaluation of the proposed classification system using data from a real-life B2B ecosystem.

The remaining of the paper is structured as follows: we first present related work in the field of fraud detection, then describe in detail the feature engineering process used to compute the graph used by the SIF detection system. We then describe the classification system we use to label unknown transactions. Finally, we evaluate our SIF detection system.

2 Related Work

Due to the sensitivity of the data linked to supplier fraud detection for victim companies, we have not been able to find any publicly available research work directly related to SIF. However, we can find several systems designed for fraud detection that also use a network-based approach:

In [9] several network-based fraud detection use-cases are introduced, showing examples of successful use of a database graph to detect bank fraud, insurance fraud and e-commerce fraud. However, the authors focus on specific industrial examples without proposing a formalized evaluation of their solution. Akoglu, Tong and Koutra [3] propose a survey on anomaly detection using graphs, notably in the domain of telecommunication fraud detection. An approach close to our own is found in [2] where an egonet (1-step neighborhood graph) is used to derive features describing a node. However, this approach is applied to a static graph that do not evolve through time, contrary to our approach. The analysis of dynamic graphs through the use of windows, as it is the case in our work, is akin to the ideas developed in [8] and [7] where the graphs are analyzed using

a moving window and detecting anomalous connectivity variation [8] or edges p-value variation [7]. To the best of our knowledge there is no previous attempt at combining a window-based analysis as seen in [8] with the feature approach used in [3]. Van Vlasselaer et al. [10] proposes a approach somewhat similar to ours, using graphs to represent interaction between companies in order to detect social security frauds. However, their work focuses on the application of social network algorithms on a large graph of interconnected entities, whereas our work partition the large graph into smaller neighborhood graphs focused on the specific behavior of a single company. In addition, the system proposed in [10] bases itself on the propagation algorithm and Random Logistic Forests to perform their analysis, and do not make use of Self-Organizing Maps. Furthermore, most of these works uses supervised algorithmsn as they rely on the existence of known legitimate and fraudulent neighboring nodes to conduct their analysis. Our work proposes an unsupervised approach that does not take into account the legitimacy of neighboring nodes to propose a label, but instead uses the topology of a specific ego network.
To the best of our knowledge, our work is the first to use a graph-based approach paired with Self-Organizing Maps in order to address the issue of SIF detection.

3 Problem Description

Let's define two companies C and S that have previously exchanged N historical transactions $\{t_1, t_2, ..., t_N\}$, all performed on the same account a_l in a sequential manner. A fraudster F wants to attempt a fraud by impersonating S and trigger a payment from C to the account a_f. Let's assume that F simply impersonates an executive in S by hacking into his mail account and advising C that all future payments on outstanding invoice should be wired to a_f. With no means to verify the executive's identity, C complies to the request. This triggers all future transactions $t_{N+1}, t_{N+2}, ...$ to be performed on a_f instead of a_l. We will use this example to illustrate how the proposed detection system works.

4 Graph-based Feature Engineering

In order to construct a model of the behavior of a client company using graphs, we first use a sequencing algorithm that aggregates the ordered transactions in bounded windows defining the company's payment behavior in the time frame defined by the window. We then aggregate these transactions into a graph, and we vectorize them by computing the number of patterns occurring in the graph. These vectors create a sequence that we use to train a Self-Organizing Map, resulting in a topography in which similar graphs are regrouped based on the patterns they share. We finally create a test graph containing the transaction to be labeled combined with the most recent transactions, and compute a legitimacy score by quantifying the similarity of the graph created in regards to the ones used to train the SOM. This anomaly score is then discretized to be turned into a label indicating if the suspicious transaction appears fraudulent or

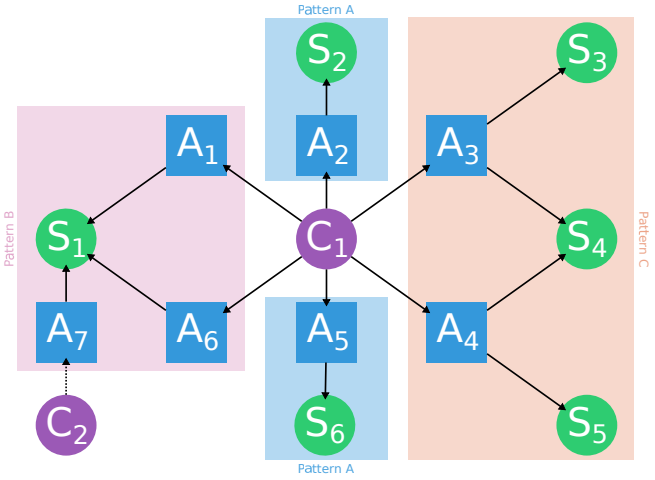


Fig. 1: Transaction graph of client C1.

legitimate.

4.1 Transaction Graphs

In order to use the contextual environment of a transaction in order to assert its legitimacy, we use the relational aspect of the transactions to create a **Transaction Graph** aggregating all the transactions involving a specific agent (any entity involved in a transaction) in a specific period of time. Figure 1 presents an example of a such a graph. This graph is created by first getting all the transactions involving the elements of a specific transaction. Then, a node is created for each element found in this list of transaction, and two edges are created between the elements involved in the transaction: one directed edge from the client to the account, and one directed edge from the account to the supplier. This transaction graph is a directed bipartite graph with two types of nodes: companies (either client or supplier), or accounts. The edges of the transaction graph are binary weighted: either an edge exists between a company and an account, and its weight is 1, or no edge exists. This model is very simple and aims to focus the analysis on the relationship shared between entity rather than the semantics of such relationships.

4.2 Featurization

The transaction graph, while providing a useful tool for visualization, is unstructured, meaning that is does not represented as a feature vector, and thus is not usable directly to perform a classification. In order to do so, the need to

Table 1: Examples of featurized transactions

Transaction ID	**Pattern A**	**Pattern B**	**Pattern C**	**Pattern D**
T1	2	1	1	0
T2	3	3	0	1
T3	1	0	0	0

transform the transaction graph into structured data arises. We propose the following process in order to do so: First, we isolate **Transaction Patterns** from the transaction graph: we define a transaction pattern as the list of connected sub-graphs obtained when the targeted transaction's client company is removed from the graph. This allows us to create a characterization of the context in which the transaction occurs. We then compute this list of sub-graphs as an histogram where the occurrence of each unique pattern is recorded. We use this histogram as our feature vector in our analysis. Table 1 shows a simple example with three transactions: T1, presented in Fig 1, T2 and T3. For the rest of the paper, we will use a hash of the canonical code [6] of the graph as labels for transaction patterns: this allows us to provide a unique identifier for each pattern.

4.3 Behavior Sequence

Using the transaction graph comes with an important drawback: the temporal order in which the transactions are processed is lost, hence hindering the dynamic analysis of our target. Moreover, with an increasing number of transactions, it becomes harder to determine the impact of a single transaction on the graph. We overcome these issues by creating **Behavior Sequences**: first, the transactions associated with the agent are sorted based on the date in which they occur, and then a window of specified size is created to group transactions in specific sets. Then, instead of a single, all-containing transaction graph, several smaller graphs are created. These graphs create a sequence that illustrate the temporal behavior of the entity.

4.4 Test Graph

In order to quantify the legitimacy of a specific transaction, a **Test Graph** is created: the transaction is added to the latest window of the behavior sequence, and a transaction graph is derived from this window, thus containing the test transaction. It is important to note that by creating the test graph we shift the focus of our Fraud Detection System from a single transaction to a single graph constructed by aggregating all the transactions from a window.

5 Self-Organizing Map Analysis

One of the issue with comparing a test graph with a behavior sequence is that the number of patterns found in the behavior sequence is highly variable, as it

depends on the window size, the number of transactions, and the complexity of the relationships between the client and its suppliers. In order to provide a consistent analysis of the behavior sequence, we first compute a Self-Organizing map that will regroup similar transaction graphs in the same area of the map. We then use K-Means to regroup together these similar graphs. This concludes the training phase. Then in the testing phase we assign a cluster to the test graph, and calculate the distance on the SOM from the node activated by the test graph to the centroid of the cluster assigned to the test graph.

5.1 Training Phase

In order to train the SOMs used in our system, we use the feature vectors found in the behavior graph of the selected client. More specifically, we feed them to the SOM where all the weights of the nodes have been randomly generated. Each feature vector will activate a single node, that will then update its weights to match the value of the histogram, while its topological neighbors will also be updated (though not as much as the first node). Once all the histograms in the behavior sequence have been fed to the SOM, we then proceed to score the test graph.

5.2 Testing Algorithm

While SOMs provide a way to project a high dimensional feature vector in a topological plane, and regroup similar feature vectors close to each other, they do not provide by themselves a way to formalize the dissimilarity between two given feature vectors. In order to do so, we created an algorithm that aims to compute an anomaly score based on the trained SOM: the K-Nearest Neighbors Distance Algorithm. This algorithm first uses the K-Means clustering algorithm to create clusters from the graphs contained in the behavior sequence. The distance between each node activated by an histogram, and the centroid of the cluster assigned to it, is computed. A cluster is then attributed to the test graph, and the distance between the node activated by the centroid of the cluster assigned to the test graph, and the node activated by the test graph is calculated. This distance is then used to compute the z-score of the test transaction. The intuition behind this algorithm is that the further away from the center of the cluster, the higher the z-score, and thus the more dissimilar to the members of the cluster the test graph is. In order to give a legitimacy label, the z-score is compared with two user-defined thresholds δ_1 and δ_2 representing the severity of the classification system.

6 Datasets

In this section we provide details about the datasets used to train and evaluate our SIF detection system: the History dataset used to train the model, and the Audit dataset used to test the performance of our system.

Table 2: Features describing a transaction between two companies.

Feature	Type	Description
Client	Nominal (ID)	Identification number of the client issuing the transaction.
Supplier	Nominal (ID)	Identification number of the supplier receiving the transaction.
Account	Nominal (ID)	Identification number of the bank account to which the money is transferred.
Date	Continous (Timestamp)	Timestamp indicating the date when the transaction took place.

6.1 History Dataset

In order to build our data-driven fraud detection system, we use a set of B2B transactions provided by the SiS-id platform[3], aggregating the transactions carried out between July 2016 and July 2019 between 5,921 companies. We dubbed this dataset "History". Table 2 sums up the features available from this dataset. Depending on the transactions' sources, more data can be available, such as the amount of the transaction or details about the goods or services paid by the transaction, but these pieces of information are not available for every record. This dataset contains more than 2 million transactions.

6.2 Audit Dataset

A second set of transactions is provided by SiS-id. It consists of the list of transactions that were analyzed using the expert system of the company in the past 2 years (July 2017 - July 2019). The dataset, called the "Audit" dataset, is composed of 108,102 suspicious transactions submitted by 171 unique client companies. These transactions are attributed a legitimacy label by SiS-id's fraud detection platform that we use as ground truth for our analysis.
SiS-Id's expert system is composed of a rule-based engine that compare the transaction with fraudulent or legitimate known cases defined by investigation experts. If the transaction matches one of these cases, then it is assigned the corresponding label ("low" if fraudulent, "high" if legitimate), while if the transaction does not match any cases a "medium" label is assigned to it.

7 Classification Process

In order to evaluate the legitimacy of a transaction, we run GraphSIF with a set of pre-defined window sizes (ranging from 5 transactions to 200 transactions) and $\delta_1 = 0.50$ and $\delta_2 = 0.9$, using 251 tests transactions previously labeled by SiS-id's expert system, and trained on 16168 unlabeled historical transactions. Results show that the variation in the distance was not high enough to accurately

[3] https://my.sis-id.com

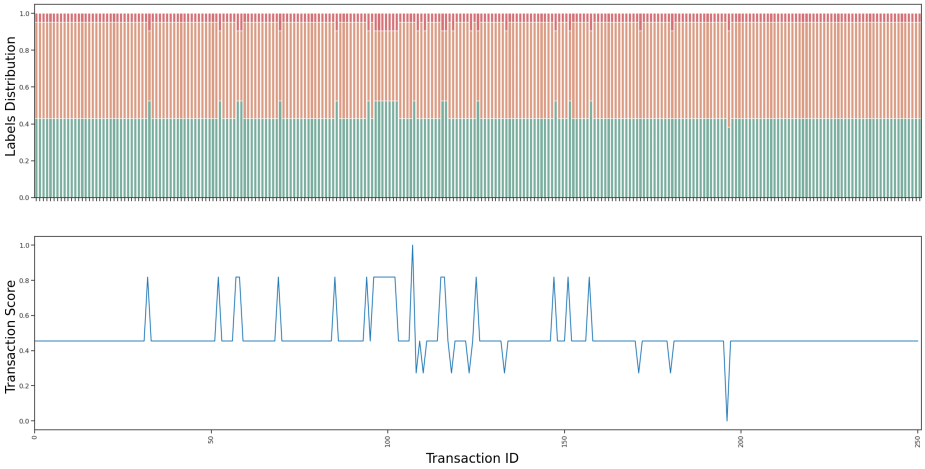


Fig. 2: Score Aggregation - Each of the 251 test transactions (X axis) is classified 40 times with window size ranging from 5 to 200 with thresholds $\delta_1 = 0.50$ and $\delta_2 = 0.9$. The uppermost figure shows the distribution of labels for each transaction. The lowermost figure shows the score computed from this distribution with weights $w_h = 20$, $w_m = 5$ and $w_l = -5$.

determine if a transaction graph was significantly different from the ones in the behavior sequence. In order to circumvent this issue, we aggregated the occurrences of each class label for each of the test transaction and reported it in Figure 2 (upper part, number of occurrences normalized for scale). In this figure we notice more clearly the different labels given by the analysis of the different windows. We define three weights w_h, w_m and w_l for each of the legitimacy labels, in order to "reward" legitimate transaction and "penalize" abnormal ones. Then the weighted average $w = \frac{w_h o(h)+w_m o(m)+w_l o(l)}{w_l+w_m+w_l}$ is computed, where $o(.)$ is the number of occurrences of a label for a specific test transaction. We plot this score (normalized) in the lower part of Figure 2, computed with $w_h = 20$, $w_m = 5$ and $w_l = -5$.

Finally, the score for each transaction is then compared with δ_1 and δ_2, as shown in Figure 3, where the green, orange, and red zones represents the area where transactions are labeled with the "high", "medium" and "low" legitimacy label respectively. This allows us to label 6 transactions with the "high" legitimacy label, 15 transactions with the "medium" legitimacy label, and 230 transactions with the "low" legitimacy label.

8 Problem Resolution using GraphSIF

We will now describe how GraphSIF applies to the example defined in section 3, assuming only a single window size w is used. First, a behavior sequence of size $\frac{N}{w}$ is created from the N transactions between C and S. As the N historical transactions are identical, identical histograms will be created. Then, each

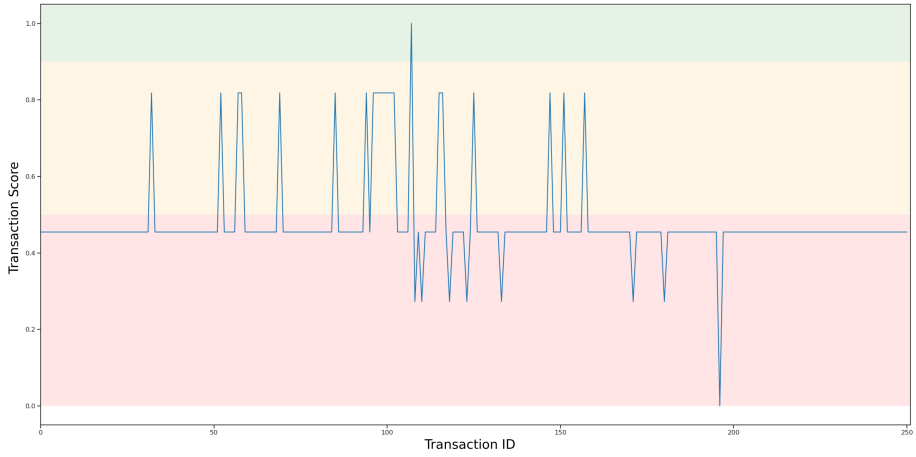


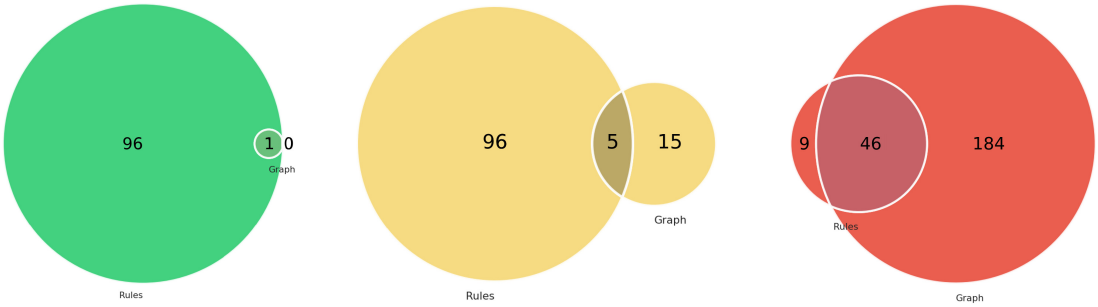
Fig. 3: Score Thresholding with $\delta_1 = 0.50$ and $\delta_1 = 0.90$ - Transactions with a score higher than δ_1 are labeled with a "high" legitimacy label. A score between δ_1 and δ_2 is given a "medium" legitimacy label. A score lower than δ_2 is given a "low" legitimacy label.

transaction from t_{N+1} is added to the $\frac{N}{w}^{th}$ behavior sequence's graph, and as a new account (a_f) is added, a new graph will be created. Thus, the feature vector representing this graph will be significantly different from the previous one, and be labeled as fraudulent by GraphSIF.

9 Experimental Results

In this section, we compare the results obtained in the previous section with the ones obtained with SiS-id's rule-based fraud detection system by first providing an overview of the differences and then focusing on three key metrics for fraud detection: accuracy, efficiency and maintainability. Unfortunately, due to the sensitive nature of the business data, the trained model is not publicly available. However a request might be submitted to the authors in order to obtain an anonymized sample of the dataset.

We first present three Venn diagrams showing the overlap of labels given by the rule engine (Rules) and graph-based detection system (Graph). First, we notice that there are a lot of transactions that has not been given the same label by both of the detection systems. Most of these difference come from the fact that numerous "high" and "medium" legitimacy labels given by the rule-based engine have been labeled with "low" legitimacy labels by the graph-based analysis. Only 0.01% of the transactions labeled with the "high" legitimacy label by the rule engine are shared with the graph analysis, as shown in Figure 4a, while the medium" label is only shared in 0.05% of the transactions (Figure 4b). However, 83.6% of the "low" legitimacy transaction are shared by the two systems, as shown in 4c.



(a) "High" Labels (b) "Medium" Labels (c) "Low" Labels

Fig. 4: Overlap of rule-based ("Rules") and graph-based ("Graph") analysis classification results for each label.

9.1 Precision

In order to evaluate GraphSIF's ability to detect a fraudulent transaction, we use a definition of precision slightly different from the one traditionally found in Machine Learning. More precisely, we define the accuracy $P = \frac{d_f}{N_f}$ of a fraud detection system as the number of fraud detected by the system d_f divided by the total number of fraud N_f found in the evaluation dataset. In this work we use the number of transaction labeled with a "low" legitimacy score by SiS-id's expert system as N_f.

We use Figure 4c to compute the precision. We have $N_f = 9 + 46$ the number of transactions given a "low" legitimacy label by the rule engine, and $d_f = 46$ the number of transactions also given the "low" legitimacy label by the graph-based analysis, thus giving $P = \frac{46}{55} = 0.836$.

While our approach leads to a high rate of fraud detection, the number of false positives (legitimate transactions labeled as fraud) is also high. False positives might delay payment and seed distrust between business partners. The elaboration of a cost-sensitive metric is thus an interesting research direction.
Figure 4 shows that our results differs from the rule engine for the "high" and "medium" labels. This might be explained by the fact that the data investigated is very different: while the expert system relies on expert knowledge about the transaction, GraphSIF only considers the relations between companies and accounts in order to perform its analysis.

9.2 Efficiency

The efficiency of a SIF detection system is the time taken by the system to assign a label to a suspicious transaction. This metric is of utmost importance, because the quicker a fraud attempt is detected, the less a system is vulnerable to the fraud. On average, the training time took roughly 33 seconds using a

laptop with 7.4 GB RAM running Ubuntu 18.04, while testing the 251 suspicious transactions took 18 seconds. Thus the time taken by the graph-based analysis to label one transaction is 71.7 ms for a single window. However, as the final result is derived from the analysis of all the windows, this score must be multiplied by the number of windows size, i.e $100/5 = 20$ in our case, so $E = 20 * 71.7 = 1434$ ms. Thus, there exists a trade-off between efficiency and accuracy, as adding more windows to the analysis will lead to more fine-grained results, but at the cost of increasing linearly the processing time and thus the efficiency. However, it is very likely that GraphSIF computation for each windows size can be performed in a parallel fashion, thus reducing the computation time for testing a transaction.

9.3 Maintainability

The maintainability of SIF detection system represents the time and effort it needs to adapt to a new situation. In the context of GraphSIF, this time roughly corresponds to the time it takes to update the data model when more historical transactions are added to the dataset. Experiments show that the training time for the graph-based system is 33 seconds on average for each window size. It is slightly higher (35 seconds) for small window size as more graphs are created from the dataset. This training time corresponds to the maintainability time, and it relatively high, as several steps need to be conducted before the model is complete.

10 Conclusion

In this paper, we introduce GraphSIF, a novel feature-engineering process that creates a feature vector based on the relationship between a client company and the accounts it used to pay its supplier company, providing a new tool to describe the underlying transaction mechanism involved in their interaction.

Several recent papers such as [11] [5] and [7] propose an human interpretation of the patterns uncovered by their approach and how they might suggest illegal behavior. The focus of our work is to emphasize on the variation of behavior, instead of the behavior itself. However the relation between the uncovered patterns and fraud attempts is currently under investigation.

In conclusion, we used the temporal information contained in the transactions of the History dataset to create a behavior sequence composed of the transactions emitted by a client aggregated in several bounded time windows. We showed how to use this behavior sequence to create a data model based on Self-Organizing maps representing the behavior of a client company through time. We then used this data model to infer the legitimacy of new transactions using the K-means clustering algorithm, along with an aggregation algorithm allowing us to combine the results obtained for different window size in a comprehensive score.

We presented the result of our classification system first on a single client to investigate its performance locally, and showed that the results differs widely from the rule engine. However, GraphSIF shows a very good accuracy locally,

and quick training and testing time, and thus can be used in a complementary fashion with the expert-based analysis.

References

1. AIG. Impersonation Fraud Claims Scenarios. https://www.aig.com/content/dam/aig/america-canada/us/documents/business/management-liability/impersonation-fraud-claims-scenarios-brochure.pdf, 2019. [Online; accessed October 10, 2019].
2. Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball: spotting anomalies in weighted graphs. In *Proceedings of the 14th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining-Volume Part II*, pages 410–421. Springer-Verlag, 2010.
3. Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.
4. Euler-Hermes DFCG. Barometre Euler Hermes-DFCG 2019. https://www.eulerhermes.fr/actualites/etude-fraude-2019.html, 2019. [Online; accessed October 10, 2019].
5. Luc A. Dubols, Daryl K. Gray, and Edward J. Tweedie. Surgical images: Soft tissue Calcinosis cutis, 2007.
6. Stephen G. Hartke and A. J. Radcliffe. McKay's canonical graph labeling algorithm. 0000:99–111, 2012.
7. Misael Mongiovì, Petko Bogdanov, Razvan Ranca, Evangelos E. Papalexakisy, Christos Faloutsos, and Ambuj K. Singh. NetSpot: Spotting significant anomalous regions on dynamic networks. *Proceedings of the 2013 SIAM International Conference on Data Mining, SDM 2013*, 2:28–36, 2013.
8. Carey E Priebe, John M Conroy, David J Marchette, and Youngser Park. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, 11(3):229–247, 2005.
9. Gorka Sadowksi and Philip Rathle. Fraud Detection: Discovering Connections with Graph Databases The #1 Database for Connected Data Fraud Detection: Discovering Connections Using Graph Databases. (January), 2017.
10. Véronique Van Vlasselaer, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. GOTCHA! Network-Based Fraud Detection for Social Security Fraud. *Management Science*, 63(9):3090–3110, 2016.
11. Johannes Wachs and János Kertész. A network approach to cartel detection in public auction markets. *Scientific Reports*, 9(1):1–18, 2019.