Graph Analytics for Real-time Scoring of Cross-channel Transactional Fraud

Ian Molloy[1], Suresh Chari[1], Ulrich Finkler[1], Mark Wiggerman[2], Coen Jonker[2], Ted Habeck[1], Youngja Park[1], Frank Jordens[2], and Ron van Schaik[2]

[1] IBM Thomas J. Watson Research Center
[2] ABN AMRO Bank N.V.

Abstract. We present a new approach to cross channel fraud detection: build graphs representing transactions from all channels and use analytics on features extracted from these graphs. Our underlying hypothesis is *community based fraud detection*: an account (holder) performs normal or trusted transactions within a community that is "local" to the account. We explore several notions of community based on graph properties. Our results show that properties such as *shortest distance* between transaction endpoints, whether they are in the same *strongly connected component*, whether the destination has high *page rank*, etc., provide excellent discriminators of fraudulent and normal transactions whereas traditional social network analysis yields poor results. Evaluation on a large dataset from a European bank shows that such methods can substantially reduce *false positives* in traditional fraud scoring. We show that classifiers built purely out of graph properties are very promising, with high AUC, and can complement existing fraud detection approaches.

1 Introduction

Fraud in payment transactions is a large problem for consumers: card fraud alone is estimated to result in losses of €1.33B in the EU in 2013 [7]. The reasons for the continued prevalence of transactional fraud are varied: reliance on outdated technologies (e.g., magnetic stripes), new banking models (e.g., online banking, P2P payments), man-in-the-browser malware, phishing, etc. Banks have turned to preventive technologies, such as EMV (chip-and-pin/signature) for card-present transactions. Mitigations for card-not-present transactions (e.g., online) include: CVV2 numbers, multifactor transactional authentication numbers (TANs), etc. Despite their partial success, fraud is still a big problem: Criminals devise malware specifically crafted to a particular bank's web pages and mobile apps that modify transactions prior to any additional user authorizations. Preventive measures introduce friction by requiring additional equipment or steps that customers need to perform. Thus, fraud detection algorithms and analytics based on transactional information is a fundamental technique all banks rely on. Most fraud detection solutions work on specific channels and use two key methods: rule based systems and statistical analytics. Rule based systems flag known

fraudulent patterns, e.g., multiple transactions at petrol stations in quick succession. Channel specific rules e.g., blacklisting malicious IPs or Tor exit nodes or incorrect billing/shipping addresses are common. Statistical methods attempt to build profiles of customers based on frequency and amount of normal transactions, changes in geography, or merchant types. Real-world constraints require fraud detection to finish in a few milliseconds, limiting the scope and sophistication of analytics, resulting in higher error rates, and even small false positive rates can be extremely costly e.g., irate customers, investigation costs.

This paper describes a completely new approach to cross-channel fraud detection based on graph analytics. Our primary objective is reducing false positives of current methods. Our hypothesis is that payment patterns define communities, and fraud manifests as deviations from these communities. We discover communities by building transaction graphs irrespective of the channel, and analyzing the structure of such graphs. These notions of community become stronger as financial institutions open up new means for person-to-person payments, such as popmoney and QuickPay. Our work is closely related to social network analysis (SNA) that attempts to identify communities and relationships, like friendship and followers. Our analysis shows that semantic differences between social and financial anomalies limit applicability of existing SNA work to this domain.

This work represents the first step in evaluating graph-based anomaly detection for fraud detection: we seek to determine which graph features, if any, provide measurable benefit in identifying fraudulent transactions. Our experiments with cross-channel transaction data from a European bank (ABN AMRO) show that several graph features provide discrimination between fraudulent and benign transactions. In a graph model where a node represents an account and an edge is a completed transaction we show that the following features can be used to substantially reduce false positives (upwards of 30%):

- the shortest distance between the endpoints of the target transaction
- if the endpoints are in the same strongly connected component
- the page rank of the destination and the reverse page rank of the source
- which clusters (defined by different methods) do the endpoints belong to and the likelihood of a transaction between these clusters.

These features, their variants and other path-based features provide excellent discrimination. We find that SNA features, such as node properties and features derived from the egonet of an account, do not perform well. Our solution is orthogonal and complementary to existing techniques, and is not intended to replace existing customer and channel profiling, but rather to improve accuracy by reducing false positives. Scalability of graph algorithms are a major hurdle in their adoption for real-time scoring (a cache miss takes around 100ns [24], limiting processing to around 1000 vertices in 100ms): our features can be precomputed, accelerated using time-storage tradeoffs, approximations, and heuristics to meet performance targets. Finally, we illustrate how to build classifiers, based solely on graph features and independent of traditional statistical features for fraud detection, and evaluate their performance on a small subset of the transactional data showing excellent promise.

There are clear benefits to graph based methods: they are channel independent and applicable to new channels, are more adaptive than rule based systems and more expressive than statistical methods. They define fraud based *directly* on the accounts involved in a transaction, not on *indirect* indicators *e.g.* statistical measures (how much the account has spent in a day) or channel-specific measures (indications of malware on the endpoint). Note that a transaction where the destination account has been modified, just prior to user authorization, cannot be detected by statistical source-account profiling. Our work is, to our knowledge, the first to show a completely new scientific approach to fraud detection in transaction networks that is channel independent. Our results are highly promising with great false positive reduction, and through clever preprocessing and algorithmic selection we can perform scoring in real-time on a commodity x86 machine without specialized hardware.

2 Related Work

Fraud detection is a mature area with almost every bank and card issuer deploying some solution using rules, analytics, and predictive models. They use existing products (e.g., [10, 11, 22]), proprietary risk engines, or combinations of these components. Both SAS and FICO Falcon rely on neural networks for their scoring engine for speed and efficiency. Existing solutions focus on statistical anomalies (min, max, average, etc.) on hand-picked features of accounts to build customer profiles, such as recency, frequency, and monetary value of transactions (RMF) [9] etc., and must be tuned for specific companies. Typically, "cross channel" means combining such properties from multiple channels into one.

There are many works describing applications of machine learning to fraud detection. Most work focuses on feature selection and building supervised classifiers. Many products [10] and published works [3,5,18,26,27] use neural networks to score transactions from statistical features. Other techniques include Bayesian learning [18], decision trees [26], association rules [25], and genetic algorithms [8].

There is a large body of work on social network analysis [2, 13, 19], often focused on identifying anomalous nodes [2], fake accounts [12], or spam [14,17]. As discussed in Section. 5.5 while these methods identify anomalous nodes they are somewhat unconnected with fraud. One can view SNA as focused on identifying anomalous/special nodes while fraud detection finds anomalous edges.

APATE [29] combines traditional features (RMF) with graph-based features connecting credit cards and merchants through transactions. They use influence propagation [17] to measure the propagation of fraud labels through the network. Such analysis identifies "hot spots" where merchants have been compromised and cards used in subsequent fraudulent transactions, such as CNP transactions without a second factor for authorization, which ABN AMRO currently uses.

3 Problem Definition: Data sets and real-time constraints

This section briefly introduces the types of transactions used in the evaluation, the types of fraud addressed, and domain specific constraints that need to be considered. We provide some details on the private dataset used to evaluate our methodology and the possible public datasources. Our test data comes from a European bank that fully deployed EMV in both card-present and card-not-present transactions using chipTANs (Chip Authentication Program). This is very different from the fraud problem in the USA where cards mostly rely on magnetic stripe for card-present transactions and CVV2 for card-not-present transactions. While EMV has some security issues [4, 20] both EMV and TANs substantially reduce vectors for committing fraud and necessitates increasingly sophisticated attacks. Known since 2005 is man-in-the-browser malware which replaces destination account numbers with account numbers the adversary controls, called a mule account. When the customer verifies the transaction, using chipTAN or mTAN, they authorize illegitimate transactions. Mobile malware can intercept mTANs, and customers may still fall victim to phishing attacks, *e.g.* the Boleto scams in Brazil [23].

This work complements existing fraud monitoring solutions by analyzing the utility in graph-based features that are orthogonal to traditional approaches. There are several key requirements: First, it is desirable to be able to score *all* transactions, and not simply subsample. Second, it is preferable to score transactions in real-time and block fraudulent transactions. Some transactions are non-reversible, such as wire transfers, saving both the customer and the bank time and money if the transaction can be blocked rather than discovered through forensics. Thus we expect our analytics to run in the order of milliseconds to support the real-time blocking of suspected transactions.

3.1 Data Sets

The data set that we use is proprietary to ABN AMRO and combines incoming and outgoing transactions from different channels: online banking, point of sale, ATM , mobile banking, and person-to-person payments during the period June 1, 2012 through Aug. 30, 2012. For this period we also had labels: false positive (transactions flagged as fraudulent by the current system, later determined to be benign), transactions correctly flagged as fraudulent, and fraudulent transactions not flagged. There were hundreds of millions of transactions in this period. Other details of the data set including specific properties are described with the results.

The only large public data set is the Bitcoin [21] blockchain, currently having 420 million transactions and about 85 million Bitcoin addresses which can be seen as accounts. Unlike traditional banks, Bitcoin allows (and encourages) a distinct account number for each transaction. We expect that account profiling mechanisms to be unsuccessful on this data and use it only to test scalability of our feature evaluation.

4 Graph Analytic Approach

This section outlines the graph analytic approach, the underlying motivation and how we use transaction graphs to score fraud. We view financial transactions transferring money from one account to another as (temporal) edges connecting accounts. Using this we can build powerful analytics leveraging recent advances in scalable analytics and anomaly detection in graphs. A successful payment can be seen as establishing a trust relationship relying only on the entities in the transaction and not on the channel or any other parameters. An account trusts the accounts/entities that it pays directly the most. The "web of trust" model can be used to transitively infer trust relationship—we then trust accounts that are paid by the accounts we pay and so on—and these relationships naturally define *communities* of normal transactions. If we can find graph features which are indicative of fraud then this approach achieves two big goals:

- Establishes a channel independent mechanism of detecting fraud.
- Graph feature based fraud detection uses a completely new set of features and would complement the accuracy of the traditional fraud detection.

4.1 Formal Definition

While banking systems record numerous attributes about transactions, we restrict ourselves to four attributes: source account, destination account, timestamp, and amount (in a common currency). From the many timestamps associated with a transaction (time initiated, processed, cleared, etc.), we assume there is one authoritative timestamp which we use. Using this data, there are several graph models one can define. First, we can model transactions as a quiver.

Definition 1. *A quiver (or multidigraph) $\Gamma = (V, E, s, t)$ where V is a set of vertices of Γ, E is a set of edges, and s and t are mappings $s : E \to V$ and $t : E \to V$ that returns the source and destination vertices for an edge, respectively.*

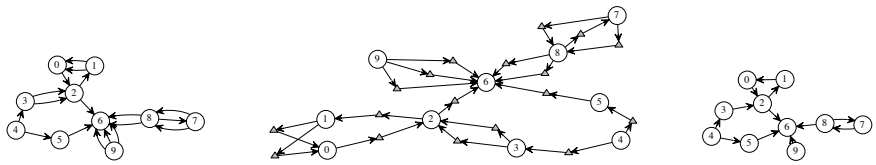
Each edge has two properties: the timestamp and the value of the transaction. Alternatively, we can define a vertex type to represent transactions as follows:

Definition 2. *A heterogeneous information network (HIN) is a digraph $G = (V, E, \gamma, \tau)$ where γ and τ are functions mapping vertices and edges to types: $\gamma : V \to \mathcal{A}$, and $\tau : E \to \mathcal{R}$ for vertex types $\mathcal{A}$ and edge types $\mathcal{R}$.*

We assume two vertex types: accounts and transactions. Only one edge type, pays is necessary, however more expressive HINs are possible. Transaction vertices can be annotated with the timestamp and amount. Collapsing multiple transactions between the same endpoints into one *representative* transaction yields a digraph.

Definition 3. *A digraph is a graph $G = (V, E)$ where V is a set of vertices (accounts), and $E \subseteq V^2$ is a set of edges (transactions). The edges may be weighted (by the total value or the number of transactions etc.).*

These representations can be enhanced by additional information such as account owners, types (savings, chequing, credit card) etc. leading to enhanced insights. We use the quiver representation but process digraphs for scalability.



(a) Quiver (b) HIN (c) Digraph

Fig. 1: Transaction data with three different graph representations. In the HIN, shaded triangles represent transaction vertices.

4.2 Graph Construction and Scoring

Figure 2(a) depicts the overall process of graph construction for fraud scoring. Only transactions that occur strictly prior to the one being evaluated are used to construct the graph (1). Given a transaction $T = X \xrightarrow{v} Y$ (from account X to Y for amount v) to be scored, the accounts are identified in the graph (2). Features from the graph are then extracted (3), and scored using a trained classifier (4).

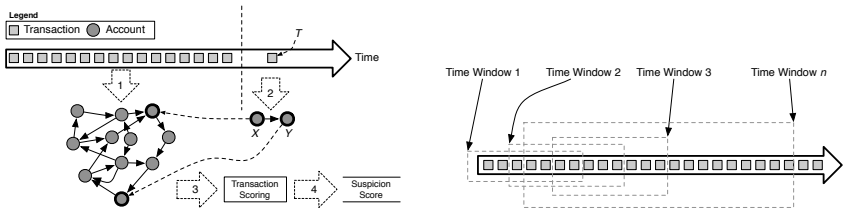
The strategy to select past transactions to construct graphs has tradeoffs in accuracy, speed of construction and maintenance, and scalability of graph algorithms. We can score a transaction against many different graphs taken from (possibly overlapping) time windows as shown in Figure 2(b). Some choices for defining such a prior window for a transaction occurring at time t are: a fixed number of recent transactions (e.g., one million); *real-time* scoring with a time window of size δ, i.e., $[t - \delta, t)$; or rounding time to a unit of granularity n, such as one day, i.e, $\left[n * \left(\left\lfloor \frac{t}{n} \right\rfloor - i\right), n * \left(\left\lfloor \frac{t}{n} \right\rfloor - j\right)\right]$, where $i > j \geq 1$, which we call *discrete time*. This work uses discrete time with a granularity of one week, and score transactions against one, two, four, and ten week windows.

Given a graph G, we extract two basic types of features: features about the vertices, $F_G(X)$, such as the page rank or size of the egonet; and features about the pair of vertices, $F_G(X, Y)$, such as the shortest distance from X to Y. We assume any feature extractor F returns a special value $\perp$ when the input account, or either account for the pair, is not in the graph G, and we can assign a prior fraudulent probability derived from all transactions not seen in a graph.

We can define ensemble scoring variants with this framework using multiple features from the same graph or features from multiple graphs. The classifier we construct for evaluation uses three different features from the same graph for scoring. Standard machine learning techniques on feature selection and training can be used given features from multiple graphs and labelled transactions.

4.3 Community based fraud detection

Our notion of fraud detection is based on notions of community: any transactions done outside a *normal* community are considered suspicious. We explore various



(a) Graphs of past transactions. (b) Discrete time windows.

Fig. 2: How individual transactions relate to graphs to be scored.

notions of community in Section 5 such as: trusted accounts within a small degree of separation (shortest-path metric in Section 5.1), accounts which have a bidirectional flow of money (strongly connected components in Section 5.2), accounts which are "reputed" (PageRank in Section 5.3), and accounts belonging to the same cluster (Section 5.4). Some natural notions of community such as accounts to which there is a sufficient flow of money (maximum-flow) are not feasible to compute in real-time. A longer version of the paper will discuss related notions *e.g.* connectivity, centrality, clustering, link prediction, etc.

5 Binary features for fraud detection

This section provides results of the experiments on the transaction data. We discuss the analytics evaluated, details of the evaluation, observations and finally, if the feature can distinguish between classes of transactions: random benign, false positives (generated by the current system), and fraud. The transaction data and labels used were extracted from 2 June 2012 to September 2012. From this we construct graph models over different windows of time and use these to score subsequent transactions. We experimented with different window sizes to determine the appropriate size: choosing a small window can cause instability in the graph features while choosing a large windows results in graphs computationally expensive graphs without recency. We constructed graphs from the transaction data of lengths 1–4 weeks. Before we discuss the specific features and how we will evaluate their utility, we must consider how the features may be used in practice. There are several ways in which graph features can be integrated into existing fraud detection and management system depending on the particular use case or type of fraud. For example, if false positive reduction is the primary goal then only transactions the existing fraud detection system flags need to be scored. Specifically, when the existing fraud detection system flags a transaction as potentially fraudulent, graph features are extracted and scored. If the graph-based analytics indicate the transaction is not fraud, then it may be dropped without further processing. The reduction in false positives can either yield significant savings for the bank (e.g., fewer incidents to investigate). Alternatively, the savings can be spent in other ways. For example, investigating the same number

of transactions, decreasing the false negatives and catching more fraud, or by decreasing customer friction by increasing the allowed risks. In other use cases, transactions can also be scored using graph-based features (see Section 7). This can be performed independently (as we demonstrate), or in combination with existing analytics and channel-specific features, such as geolocation, transaction amount, and other channel specific features.

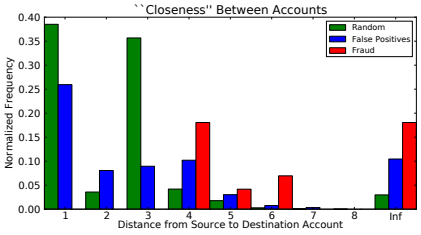
Our experiments are designed to evaluate two of these use cases. First, we evaluate how well they reduce false positives produced by existing analytics. We also indicate how well the individual features perform at identifying fraudulent transactions if they were to be used alone (note that typical fraud detection systems score using tens-to-hundreds of features). Finally, we will build a classifier using several features and score held-out transactions.

5.1 Shortest Path

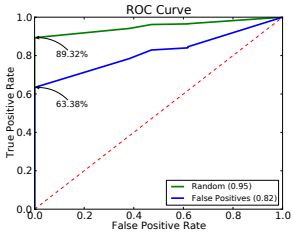
This analytic is easiest to understand: Viewing a transaction as a trust relationship, a path between two accounts is a series of trust relationships. The central hypothesis we investigate is: trust in financial networks is transitive *i.e. if account x pays y and y pays z then can we view this a trust relationship between x and z*. This model is especially relevant for person-to-person payments. We test whether this transitive notion of trust can be used to identify fraudulent transactions. Specifically, we measure the length of the shortest chain of transactions between the source and destination accounts and evaluate its ability to discriminate between normal and fraudulent transaction. In effect the analytic is *the existence of a short path is indicative of a non-fraudulent transaction.*

For directed transaction graphs, where each edge has a weight (value), there are two possible notions of closeness: shortest path and shortest distance. Shortest path treats all edges equally, and counts the length of the path that connect two accounts. Shortest distance could weight this path with the transaction value or any other metric. For any pair of transactions, we compute *three* shortest paths: the direct shortest path from the source account to the destination; the reverse directed shortest path, i.e., the path from the destination account to the source and the undirected shortest path. An undirected shortest path may exist even if there are no paths from the source to the destination or vice-versa.

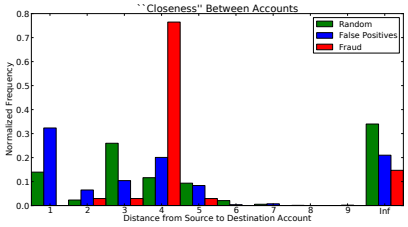
Experimental Results We evaluate shortest path metrics using a graph built from four weeks of transactions and scoring transactions in the subsequent week with it. The *null hypothesis* is: *the shortest path feature does not statistically distinguish normal from fraudulent transactions.* Over all periods, the null hypothesis is rejected: the distribution of shortest paths for fraudulent transactions compared to that of a sample of normal transactions differs significantly (χ^2 for $p << 0.05$). For each shortest path variant, we evaluate the intrinsic value of a one-dimensional binary classifier at distinguishing between normal and fraudulent transactions, producing a receiver operating characteristic (ROC) curve illustrating the false positive and false negative tradeoffs. Figure 3 shows the



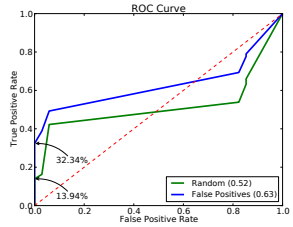
(a) Shortest Path Distribution



(b) Shortest Path ROC Curve



(c) Reverse Shortest Path Distribution



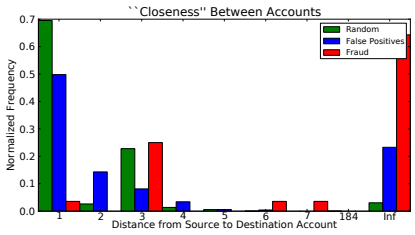
(d) Reverse Path ROC Curve

Fig. 3: Distribution of shortest path for transactions during 2012-06-29 through 2012-07-05 in graph constructed from the previous four weeks of transactions.

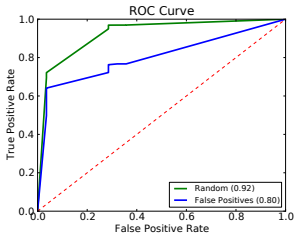
distribution of the shortest path (in the graph constructed from the prior 4 weeks) for transactions in the period of 2012-06-29 through 2012-07-05. The histograms are scaled for the different populations. Normal and false positive transactions show a clearly distinct statistical distribution compared to fraudulent transactions. In this case, the shortest path binary feature is able to reduce false positive rates by 63%!

The results for the reverse shortest path, a closeness measure for "repayment", is shown in Figure 3. Reverse shortest path is the poorest performing of the three illustrative measures, but still yields an almost 14% reduction in false positives. Other time periods yielded equally impressive, or better, reductions for shortest path.

Similar results are obtained for other time periods validating shortest path as a good feature for fraud detection. In these periods, with the exception of a fraudulent transaction where a prior relationship existed, a minimum of 39% false positive reduction was obtained; the smallest improvement was 13.9% false positive reduction using the reverse shortest path (and an AUC of only 0.52). We have explored many variants such as building graph models from transactions over a larger time period. In the 4-week graph models in Figure 3 there is a small percentage (about 5%) of normal transactions whose endpoints don't appear in the 4 week graph *i.e.* the accounts were inactive for the previous four weeks. This impacts the accuracy of classifiers we can build with this feature. Building models with larger windows of transactions will reduce the number of inactive



(a) Shortest Path Distribution



(b) Shortest Path ROC Curve

Fig. 4: Distribution of shortest path for transaction in the period 2012-08-17 through 2012-08-23 using a ten-week graph model

accounts: building a graph model from 10 weeks of transactions from June 1, 2012 through Aug 16, 2012 reduces the number of inactive accounts to about 1.8%. As Figure 4 shows, scoring two weeks of subsequent transactions with this graph model displays the same statistical discrimination. Note however that there is a fraudulent transaction where the endpoints conducted a valid transaction before. A longer version of this paper will report on discrimination using shortest *distance* weighting by the value of the edge. These results show that shortest path and distance and their variants provide excellent discrimination between normal and fraudulent transactions.

5.2 Strongly Connected Components

A strongly connected component (SCC) is a subgraph such that all pairs of vertices are connected. We use SCC as a notion of bi-directional transitive trust: when account x pays account y, if x and y are in the same strongly connected component, then, in theory, money already flows from x to y, and money flows from y back to x. When there exists a path from x to y, yet there is no return path, we can view y as a sink account. Paying money into sink accounts could be risky and we suspect such sink account (or entire subgraphs) to be suspicious. However, these may be external accounts for which we do not have visibility.

We evaluate the following aspects of strongly connected components: size and distribution of SCCs, impact of window size on SCCs, whether the two accounts involved in a transaction are members of the same SCC.

The sizes of strongly connected components naturally follow a powerlaw and the size of the largest components increases with the length of the time window. Unlike other clustering methods, adding a new transaction can only merge two SCCs into one, making it easily adapted to real-time applications and SCC numbers can be stored and later looked up. This is important because the smaller the component, the less likely the two randomly selected accounts will be members of the same SCC.

Transactions and Strongly Connected Components We use strongly connected component for identifying fraudulent transactions and reducing false positives based on the hypothesis that transactions *within* an SCC are less likely to be fraudulent than transactions than span two strongly connected components. Such edges (transactions) in graph theory are known as bridges.

To evaluate, we build graphs for one- two- and four-week windows that advance one week at a time. Next, transactions for a target period are scored against the strongly connected components and placed into one of four categories: the source and destination accounts are in the same SCC; source end destination accounts are in different strongly connected components, but a prior transaction exists; source and destination accounts are in different strongly connected components and a prior transactions does not exist; either the source or destination accounts was inactive and is not a member of any strongly connected component

The results from our ten week experiment are shown in Figure 5, clearly illustrating that two accounts are more likely to be in the same SCC, or have a prior transaction, for the benign transactions than the fraudulent transactions (the null hypothesis is rejected). We also measure the increase in conditional probability given which of the four classes the transaction falls into. We summarize these results using the conditional probability gain by grouping all graphs (one, two, four, and ten week windows) based on their duration in Table 1. We can clearly see that the majority of fraudulent transactions occur where one of the endpoints was previously inactive or the accounts are members of different strongly connected components.

Fig. 5: SCC results for transactions 2012-08-17 through 2012-08-30 on graph from prior ten weeks.

Window Size	Gain
1	0.58
2	1.02
3	2.68
4	2.13
10	4.88

Table 1: Conditional probability gain from SCCs.

5.3 Page Rank

PageRank [6] is an algorithm for measuring the importance and trust in accounts, originally developed to model the importance of web pages. In our graphs PageRank measures the reputation of an account in terms of the payments made to it. Our hypothesis is that accounts with a high PageRank are less likely to be fraudulent. In the PageRank algorithm, an account evenly distributes its own PageRank to the accounts it pays, and the algorithm iterates until convergence: $\mathsf{PR}(u) = \frac{1-d}{N} + d \sum_{v \in P(u)} \frac{\mathsf{PR}(v)}{|P(v)|}$, where $P(u)$ is the set of accounts u pays and d is a damping factor. Originally, the damping factor modeled the probability a random web surfer stops on a page. In financial transactions, the damping factor can be used to model an account saving. We use a default constant damping

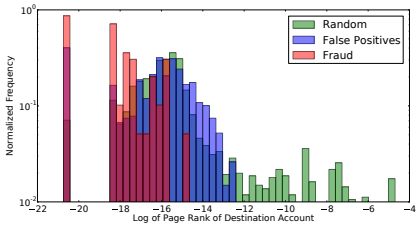


Fig. 6: Distribution of PageRank for transactions during 2012-06-29 through 2012-07-05 in graph constructed from the previous four weeks of transactions.

factor of 0.85; future work will explore using a per-account damping factor based on past spending behavior.

PageRank can be used either unweighted or weighted. In the weighted form, the distribution of an account's PageRank to its neighbors can be made proportional to the transaction amount. Alternatively, we could weight edges based on the number or frequency of the transactions. This may not be accurate as it is possible to send messages and hence use mobile payments as a chat service with €0.01 transactions. However, since frequent payments can indicate a level of trust, future work will investigate this method of weighting edges. We experiment with four different versions of PageRank: Forward unweighted, Forward weighted, Reverse unweighted, Reverse weighted. In the reverse forms, we reverse the directions of the transactions with the intuition that accounts performing many transactions are less likely to be in fraudulent transactions. We evaluate all four forms of PageRank against the same sample graphs used to evaluate shortest path: compute PageRank on a four-week graph and use that to score transactions from the subsequent week. In each case, we use PageRank of the source and destination accounts and evaluate the following question: *does the PageRank for victims or destination account of fraud follow a different distribution than randomly selected accounts?*

Figure 6 shows an example of how PageRank of the destination discriminates between normal and fraudulent transactions which is also seen in other periods evaluated. Table 2 shows the average performance of using the variants of the PageRank algorithm The unweighted PageRank of the destination produces a respectably average 39.3% reduction in false positives (ranging from 22–48%). Using the weighted version improves the reduction to 44.54% on average. The reverse PageRank of the source produces less impressive but useful reduction of 16.98% on average and 17.70% for the weighted version.

5.4 Clustering

We use clustering techniques to discover accounts which have similar profiles *e.g.* connectedness and transactional patterns, and score transactions based on whether they are consistent with the cluster. In particular, we use prior models

Feature evaluated	Average ZER	Average AUC
PageRank of Destination	39.575%	0.760
Weighted PageRank of Destination	44.54%	0.788
Reverse PageRank of Source	16.98%	0.625
Weighted Reverse PageRank of Source	17.70%	0.652

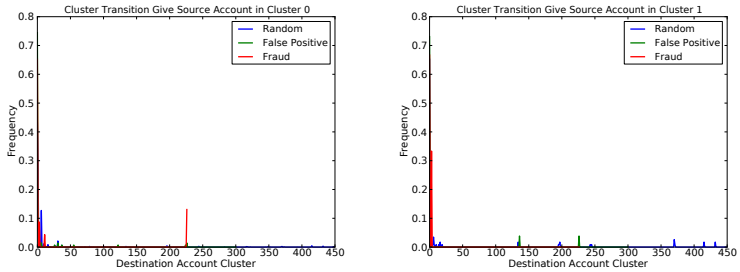
Table 2: PageRank Results

of inter-cluster transaction likelihoods to score the transaction risk. We explore different features for clustering, each providing a different perspective on the data: *basic features* of vertices such as the number of outgoing edges, distinct destinations, incoming edges and accounts, types of edges (external, foreign, etc.); *egonet features* are extracted from the egonet of a vertex including weight and number of all transactions, number of distinct transactions of each type, number of nodes of each type etc.; *connectivity features* represent transactional information. We build an undirected graph where vertices denote accounts, and edges represent transactions. The edges are weighted by a function of the number of transactions and the total transaction amount between the two accounts.

Clustering algorithms: We use the BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [30] algorithm to cluster basic and egonet features since they represent accounts as vectors in a multi-dimensional space. BIRCH is an algorithm which performs hierarchical clustering over a very large data-set, and is known to handle well very large and noisy data sets. We use the Markov Cluster (MCL) algorithm [28] to cluster nodes connected to endpoints similarly based on the connectivity features. MCL is a fast and scalable cluster algorithm for graphs based on simulation of random flow in graphs.

Cluster evaluation: Since known class membership labels are not available, we measure the results by cluster quality: how homogenous the clusters are and how well clusters are separated. Another measure is cluster stability across different time periods. We evaluated these strategies against the 1-week and 4-week transaction graphs. The produced clusters have excellent homogeneity, separation and stability for both egonet and basic features. We defer the detailed cluster evaluation figures to a longer version of the paper.

Using Clusters for Scoring: We hypothesize that accounts in the same cluster will have similar spending habits, i.e., pay similar accounts. We test this by investigating the inter-cluster transitions *i.e.* for all transactions from source accounts in the same cluster, we measure the distribution of the destination account clusters. We evaluate whether cluster-to-cluster distribution for fraudulent transactions deviates from that obtained by the false positives and random transactions. We show a single result leaving the rest to a longer version of the paper. We observed such spikes across multiple time periods for the clusters produced using the basic features. However, almost all fraudulent transactions are inter-cluster with egonet features, limiting its ability to score transactons.



(a) Cluster 0 (b) Cluster 1

Fig. 7: Outgoing cluster transitions illustrating anomalous spikes for fraudulent transactions.

5.5 Egonet Features

Egonets, or subgraphs of radius two centered around a single account (the ego node) yield features similar to those obtained from the basic analysis. The egonet includes any transaction between the neighbors of the ego node, making concepts such as "incoming" and "outgoing" transactions and accounts, as used in the basic features, less meaningful. However transactions may be classified as those including the ego node, and those strictly between the neighbors of the ego node. Egonets have been studied extensively for anomaly detection in social network graphs and corporate graphs. For example, egonets have been used to analyze the Enron email corpus [16], political donations, and blogs. We investigate the use of a suite of egonet based anomaly detection features called OddBall [2]. OddBall explores powerlaws that exist between properties of an egonet: For example, in an undirected graph, the number of edges in an egonet of n vertices, is bound between $n-1$ and $n*(n-1)$, giving a powerlaw between 1 and 2 approximately. OddBall analyzes such relationships for common properties of an egonet, computes powerlaws, and identifies anomalies that deviate significantly (are far from a regression line) or are in areas of low density (are not near other accounts when plotted by egonet properties). Several properties of egonets are of interest, mainly the number of nodes, the number of edges (total and distinct), the total weight of the edges, and the principal eigenvalue.

We evaluated how well OddBall performs at identifying accounts involved in fraudulent transactions, either the victim or the fraudster. The results indicate egonets are a poor indicator of malicious activity in financial networks, however they were successful in identifying several anomalous accounts. Figure 8 presents a scatter plot of two properties of an egonet, egonode was involved in fraud (green squares and red triangles), and the anomalies (yellow triangles). These anomalous accounts are actually *vostro accounts*, accounts for other banks held for accounting purposes, and should not be considered malicious.

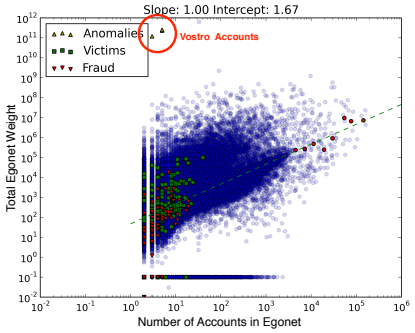


Fig. 8: OddBall anomalies identifying vostro accounts.

6 Scalability

This section briefly discusses the scalability of the graph analytics discussed. To be effective in practice, a fraud detection algorithm must be able to score all transactions, in real-time (under 100ms). Any property that depends on a single account, such as which SCC the account belong to, or the PageRank, can be precomputed. Properties that require the pair of accounts, such as the shortest path, are prohibitive to precompute and store, and must be computed in real-time. For large graphs, however, a single shortest path query can be costly; in some of our experiments exceeding 30 seconds. To handle throughput, we seek solutions which don't require massive parallelism or special hardware.

Architectural differences, such as faster processors and memory, typically yield a 2–8x performance increase, falling short of our goals. By bounding the maximum length of the paths, we can ensure that *most* queries are returned in sufficient time, however graphs and accounts with high branching factors still yield queries that are exhaustive and search the entire graph. Instead, we use approximation algorithms that store minimum spanning trees for a select subset of vertices, in an approach similar to other approximate algorithms [1, 15]. Unlike [1], we store a small number (100) of spanning trees, reducing precomputation time from hours to minutes, and drastically reducing storage costs. To improve approximations that aren't the result of intersections between two spanning trees, we perform a breadth first search that caps the number of edges evaluated to ensure termination in a few ms. Our experiments used Neo4J, a graph database for storing and processing some of the data, but the performance was inadequate. We store computed features in MongoDB, and use graph-tool, a Python Boost graph library package, for some algorithms, such as PageRank. Our shortest path approximation was implemented in C++. All tests were performed on an AMD Opteron 8439 with 256 GB RAM.

7 Classifiers

While our primary objective was to reduce the number of false positives, we experimented with building classifiers, using only graph features evaluated and computable in real-time. We restricted our classifiers to features derived from SCC, PageRank, and shortest path (forward, reverse, and undirected). We divide transactions into one month graphs and train a support vector machine (SVM) classifier with a radial bias function (RBF) kernel on the first month and test on the remaining data. All features are extracted from the graph pertaining to the prior month and zero-mean unit-variance normalization is performed (relative to the first month's features). The results are shown in Figure 9, which also indicates how currently deployed analytics perform. Because the deployed analytics make use of rules and blacklists, they cannot be parameterized and we can only represent them as a point on the ROC curve. These rules may reject transactions due to failed PIN or TAN, known mule accounts, or phishing campaigns. Note that the task here is different than previous sections, where the task was false positive reduction (the positive class is benign) while here it is fraud detection (positive class is fraudulent).

The results are clearly encouraging for the use graph features for fraud scoring. Our highest performing classifier reached an AUC score of 0.93; higher than the optimistic binary evaluations shown in the previous section. Importantly, this implies the features some of the features are independent, and combining them may yield better results. It is only built on three types of features: shortest path, SCC, and PageRank, but identifies many fraudulent transactions missed by the deployed analytics. The classifiers here built on graph feature alone currently yield too many false positives to be deployed in practice. However, graph models contribute a new set of features and we expect that ensemble classifiers built from these and traditional features to be substantially more accurate.

Further work is needed on selecting the optimal time window, or windows, for producing graphs. The same one-month graph is used to score a month's worth of transactions, and not one week as we used previously. Here we observed a drop in AUC scores for the shortest path features, from 0.95 to 0.67 for example, but by combining all shortest path computations we were able to improve to an AUC of 0.888. Of course, best results are expected with ensembles of graph features and current fraud scoring mechanisms.

8 Conclusion

We have described a new approach to cross channel fraud detection based on graph analytics. Our results show that graph features can provide excellent discrimination between fraudulent and normal transactions. This strengthens our belief that graphs of financial transactions form accurate representations of actual social communities and relations. We expect that improvements in the scalability of graph analytics, this new approach can be a feasible solution to enhance the accuracy of fraud detection. Further, they offer a default fraud model for all new transaction channels. We expect that graph models will also be beneficial to applications such as commercial banking and anti-money laundering.

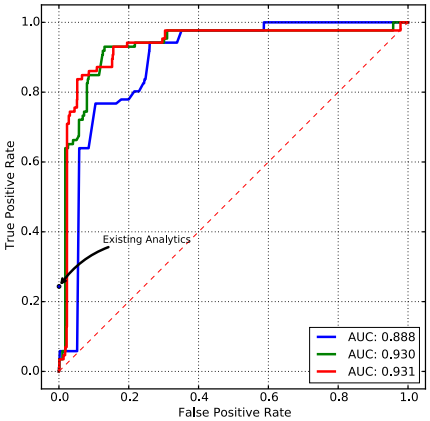


Fig. 9: Classifier performance using only graph features: Shortest path and SCC (blue), adding BFS query properties (green), and adding PageRank (red).

References

1. R. Agarwal, M. Caesar, B. Godfrey, and B. Y. Zhao. Shortest paths in microseconds. *CoRR*, abs/1309.0874, 2013.
2. L. Akoglu, M. McGlohon, and C. Faloutsos. Anomaly Detection in Large Graphs. Technical Report CMU-CS-09-173, Carnegie Mellon University, Nov. 2009.
3. E. Aleskerov, B. Freisleben, and B. Rao. CARDWATCH: a neural network based database mining system for credit card fraud detection. In *IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, 1997.
4. M. Bond, O. Choudary, S. J. Murdoch, S. Skorobogatov, and R. Anderson. Chip and Skim: Cloning EMV Cards with the Pre-play Attack. *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 49–64, 2014.
5. R. Brause, T. Langsdorf, and M. Hepp. Neural data mining for credit card fraud detection. *11th International Conference on Tools with Artificial Intelligence. TAI 99*, pages 103–106, 1999.
6. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and {ISDN} Systems*, 30(17):107 – 117, 1998. Proceedings of the Seventh International World Wide Web Conference.
7. A. Brown, D. Divitt, and A. Rolfe. Card Fraud Report 2015. Technical report, Alaric, Mar. 2015.
8. E. Duman and I. Elikucuk. Solving Credit Card Fraud Detection Problem by the New Metaheuristics Migrating Birds Optimization. In *Advances in Computational Intelligence*, pages 62–71. Springer Berlin Heidelberg, Berlin, Heidelberg, June 2013.
9. P. S. Fader, B. Hardie, and K. L. Lee. RFM and CLV: Using iso-value curves for customer base analysis. *Journal of Marketing Research*, 42(4):415–430, 2005.
10. FICO. FICO Falcon Fraud Manager for Debit and Credit Card. Technical report, FICO, 2012.
11. fiserv. fiserv: Compliance & fraud management. `https://www.fiserv.com/risk-compliance/financial-crime-risk-management.aspx`, 2015.

12. N. Z. Gong, M. Frank, and P. Mittal. SybilBelief: A Semi-Supervised Learning Approach for Structure-Based Sybil Detection. *IEEE Transactions on Information Forensics and Security*, 2014.
13. N. Z. Gong, W. Xu, L. Huang, P. Mittal, E. Stefanov, V. Sekar, and D. Song. Evolution of social-attribute networks: measurements, modeling, and implications using google+. In *the 2012 ACM conference*, pages 131–144, New York, New York, USA, Nov. 2012. ACM.
14. C. Grier, K. Thomas, V. Paxson, and M. Zhang. @ spam: the underground on 140 characters or less. *Proceedings of the 17th ACM conference on Computer and communications security*, pages 27–37, 2010.
15. A. Gubichev, S. Bedathur, S. Seufert, and G. Weikum. Fast and accurate estimation of shortest paths in large graphs. In *CIKM '10: Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 499–508, 2010.
16. B. Klimt and Y. Yang. The enron corpus. In *In ECML*, pages 217–226, 2004.
17. R. C. C. F. Leman Akoglu. Opinion Fraud Detection in Online Reviews by Network Effects. In *International AAAI Conference on Weblogs and Social Media*, pages 1–10, Apr. 2013.
18. S. Maes, K. Tuyls, and B. Vanschoenwinkel. Credit card fraud detection using Bayesian and neural networks. In *Proceedings of the 1st international NAISO congress on neuro fuzzy technologies*, 2002.
19. A. Mislove, M. Marcon, P. K. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. *Internet Measurement Comference*, pages 29–42, 2007.
20. S. J. Murdoch, S. Drimer, R. Anderson, and M. Bond. Chip and PIN is Broken. In *2010 IEEE Symposium on Security and Privacy*, pages 433–446. IEEE, 2010.
21. Nakamoto and Satoshi. Bitcoin: A Peer-to-Peer Electronic Cash System, 2009.
22. NICE. Nice actimize: Fraud detection & prevention. `http://www.niceactimize.com/fraud-detection-and-prevention`, 2015.
23. RSA. RSA Discovers Massive Boleto Fraud Ring in Brazil. Technical report, EMC, July 2014.
24. S. Saini, J. Chang, and H. Jin. Performance Evaluation of the Intel Sandy Bridge Based NASA Pleiades Using Scientific and Engineering Applications. In *High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation*, pages 25–51. Springer International Publishing, Cham, Nov. 2013.
25. D. Sánchez, M. A. Vila, L. Cerda, and J. M. Serrano. Association rules applied to credit card fraud detection. *Expert Systems with Applications: An International Journal*, 36(2):3630–3640, Mar. 2009.
26. A. Shen, R. Tong, and Y. Deng. Application of Classification Models on Credit Card Fraud Detection. In *2007 International Conference on Service Systems and Service Management*, pages 1–4. IEEE, 2007.
27. M. Syeda, Y.-Q. Zhang, and Y. Pan. *Parallel granular neural networks for fast credit card fraud detection*, volume 1. IEEE, 2002.
28. S. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, 2000.
29. V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens. APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 75:38–48, July 2015.
30. T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. *ACM SIGMOD*, 25(2), 1996.