Anomaly Detection in Large Social Graphs

Neil Shah

December 9, 2016

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Christos Faloutsos, Chair
Roni Rosenfeld
Jaime Carbonell
Jiawei Han, University of Illinois at Urbana-Champaign

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Keywords: anomaly detection, unsupervised, social graphs, dynamic graphs, attributed graphs, graph understanding, scalability, interpretability

Abstract

Given the ever-growing prevalence of online social services, usefully leveraging massive datasets has become an increasingly important challenge for businesses and end-users alike. Online services capture a wealth of information about user behavior and platform interactions, such as *who-follows-whom* relationships in social networks and *who-rates-what-and-when* relationships in e-commerce networks. Since many of these services rely on data-driven algorithms to recommend relevant content to their users, authenticity of user behavior is paramount to success. But given anonymity on the internet, how do we know which users and actions are real and which are fake? My thesis focuses on this problem and introduces new techniques to effectively and efficiently discern anomalous and fraudulent behavior in online social graphs. Specifically, I propose to work on three thrusts: plain graphs, dynamic graphs and rich graphs.

Firstly, we focus on *plain graphs*, in which only static connectivity information is known. We detail several proposed algorithms spanning the topics of spectral-based fraud detection in a single graph, blame attribution between graph snapshots, and evaluation of the descriptive power of various graph clustering and partitioning methods in identifying anomalous graph structures. Our FBOX algorithm in [59] identifies link fraudsters in social networks with over 93% precision and identifies hundreds of thousands of fake accounts, many of which were yet unsuspended.

Next, we broaden our scope to *dynamic graphs*, in which we are able to leverage connectivity information over a span of time. Many online interactions are timestamped, and thus time and interarrival time between user actions are powerful features which can be used to discern the normal from the abnormal. We describe multiple relevant works which describe how to identify and summarize anomalous temporal graph structures, model interarrival time patterns in user queries to find anomalous search behavior, and identify "group" anomalies comprising of users acting in lockstep. Our FLOCK approach in [58] is the first to tackle the viewbot problem on livestreaming platforms, and finds viewbotted broadcasts and constituent viewbots with over 90% precision and near-perfect recall.

Lastly, we expand our reach to *rich graphs*, in which connectivity information is supplemented by other attributes, such as time, rating, number of messages sent, etc. Rich graphs are common in practice, as online services routinely track many aspects of user behavior to gain multifaceted insights. Multimodal views of data are thus useful in identifying various types of anomalies in different subspaces. To this end, we propose a principled means for ranking users by their abnormality by leveraging statistical patterns in edge attributes. Our EDGECENTRIC approach in [60] uncovers rating patterns in e-commerce datasets and pinpoints fake reviewers with 87% precision at Flipkart.

The techniques described in this thesis span various disciplines including machine learning, graph theory, information theory and social science and are practically applicable to a number of real-world fraud and general anomaly detection scenarios. They are carefully designed and tuned to attain high precision and recall in practice, and be efficient and scale to massive datasets, including social networks, telecommunication networks, e-commerce and collaboration networks with up to *millions* of nodes and *billions* of edges.

Contents

Chapter 1

Introduction

1.1 Introduction

Mining large datasets has become an especially notable focal point in computer science research in recent years due to the ever-increasing scale and complexity of online systems – an estimated 2.5 exabytes of new data is generated every day[1] from commercial transactions, social networks, system log data, electronic sensors and more. This heavily motivates the development of effective and scalable approaches for extracting patterns from large data sources. In reality, many data sources can be construed as graphs, which represent interactions between objects such as humans or computers. Graphs enable modelling of complex phenomena including interactions between users (who-follows-whom on Twitter), product impressions (who-rates-what on Amazon) and email traffic flow (who-emails-whom in a corporate network). While graph analysis can give us insights on common interaction patterns, it is also a powerful tool for identifying anomalous and often fraudulent behavioral patterns including fake user interactions and product ratings to artificially boost popularity, email spam as well as network attacks. However, there are many associated challenges with this task: How can we identify anomalous behavior when only structural graph connectivity is known? How can we additionally leverage temporal information for the same? Furthermore, how can we also integrate more complex and multifaceted features to enable a holistic approach to anomaly detection in graphs?

In some cases, practitioners and data scientists are equipped with only a plain graph, which describes structural connectivity information between objects in a static snapshot, and must use this limited amount of information to find anomalous behavior. This task is common in industrial datasets made available to academics (in which rich features and personally identifiable information are often stripped for privacy or security reasons), as well as in certain types of graphs which inherently represent simple phenomena. In this thesis, we detail a number of approaches which can intelligently utilize this relatively sparse information to identify abnormal individual as well as group-based connectivity patterns.

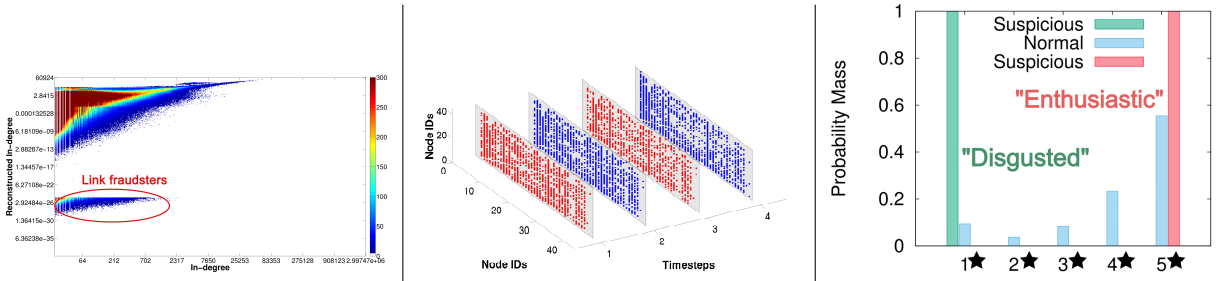
Of course, in domains in which graph objects are users such as in e-commerce and social networks, interactions are often represented as a dynamic graph which changes over time. This stems from the reality that humans themselves behave differently over time. For example, a who-rates-what e-commerce graph typically grows over time, as users rate more products. Similarly, a social network graph also changes over time as users follow and unfollow each other depending on changing interests. Temporal information provides a powerful signal for discerning between authentic and inauthentic user behavior

[1] https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

(for example, a user who rates products consistently 5 seconds apart is likely not behaving normally, and is likely scripted). Our work provides insights into modelling and incorporating dynamism in graphs to identify abnormal behavior.

Finally, most online services track very rich information about their users to improve user experience and the quality of user recommendations for products and other users. For example, e-commerce networks often maintain detailed information about the types of products their users like, transaction cost, time spent viewing product pages, and product ratings and review text. Similarly, social networks have information on how often users view each others pages, exchange messages, endorse their profile statuses and so on. In the presence of rich graphs which capture several types of details about interactions, we tackle the commensurate challenge of identifying different types of anomalies as well as integrating a multitude of signals to inform our detection algorithms.

In this thesis, we work towards new means for anomaly detection in graphs in three major thrusts: mining plain graphs, dynamic graphs and rich graphs. Each of these tasks involves leveraging different degrees of information in effectively and efficiently identifying anomalous behavior. Below, we describe our already completed work, as well as our ongoing and proposed work in the context of these tasks.



(a) Link fraud detection (b) Dynamic graph summarization (c) E-commerce fraud detection

Figure 1.1: An overview of results from our work on anomaly detection in plain, dynamic and rich graphs. (a) shows how our FBOX algorithm catches stealthy fraudsters who are poorly modeled by low-rank spectral methods. (b) depicts a result from our TIMECRUNCH approach for dynamic graph summarization, identifying a dense near-clique in an instant messaging network. (c) shows several types of suspicious versus normal user rating behaviors found by our EDGECENTRIC method.

1.2 Completed Work

We summarize completed work for each of the three tasks below.

1.2.1 Mining Plain Graphs

While our algorithms are designed with generality in mind, our works on plain graphs have varied applications from social network fraud detection, brain graph analysis and general graph summarization. In [59][PDF], we propose a fast algorithm to identify stealthy link fraud attacks in which users aim to artificially inflate their popularity in social networks using fake links. [36][PDF] introduces a blame attribution algorithm to pinpoint nodes and edges which are responsible for extensive change between two graph snapshots, with promising applications in identifying abnormal email behavior and brain graph mining. In [44][PDF] and [43][PDF], we present a comparative analysis of clustering and partitioning algorithms and their relative performance in the graph summarization task, which involves reducing large graphs into a few, interpretable structures.

1.2.2 Mining Dynamic Graphs

The ubiquity of human-generated events results in many scenarios in which dynamic graph analysis is appropriate. In [61][PDF], we present a principled, information theoretic approach for dynamic graph summarization which involves identifying temporally recurrent graph structures like "flickering stars," "periodic cliques" and more. In [29][PDF], we show the temporal bimodality of user web search habits, and use group level analysis to identify abnormal users with "hyperactive" behavior and short rest times. [58][PDF] uses group-level analysis of views followed by a pruning algorithm on livestreaming broadcasts to discern real viewers from fake viewbots.

1.2.3 Mining Rich Graphs

Our work on rich graphs is primarily concerned with understanding and integrating a multitude of signals in the form of features (or attributes) for anomaly detection purposes. In [60][PDF], we detail a method for information theoretic, unsupervised ranking of anomalous nodes by leveraging numerical and categorical edge attributes, such as number-of-stars and inter-arrival times of product ratings. This work provides a principled way to rank comparative suspiciousness between objects with varying edge behavior.

1.3 Ongoing & Proposed Work

We summarize ongoing and proposed work below. This work focuses on the mining rich graphs task.

1.3.1 Mining Rich Graphs

Anomaly Detection with Textual Edge Attributes: (§4.2) How can we extract and leverage edge attributes from text such as product reviews, messages, and posts which we observe in social graphs? Text is an unstructured, but rich data source which can provide insights into anomalous user behaviors (message spam, off-topic discussion, etc.)

Unifying Structural and Edge-attributed Anomaly Detection: (§4.3) How can we integrate the signals from abnormal graph connectivity and abnormal edge attributes? Previous works have primarily focused on graph connectivity without considering richer attributes – unifying the two is a promising direction towards holistic graph-based anomaly detection.

Characterizing Link Fraud: (§4.4) What does link fraud look like in practice? Is there one, or are there several different types? We plan to use honeypot accounts for which we buy fake followers and observe their structural and attribute properties.

Temporal Analysis for Link Fraud: (§4.5) How can we detect when an account purchases fake followers and identify who they are? We again plan to use honeypot accounts upon which we solicit both fake and real followers (colleagues, etc.), and observe patterns and abnormalities in attribute entropy over time.

1.4 Overview

Table 1.1 gives a high-level overview of both the completed, as well as ongoing and proposed work described in this proposal, with subtasks categorized into the associated tasks. Associated section numbers, references, and PDF links are given in the table for the reader's convenience. An asterisk besides a subtask indicates that the associated work is ongoing or proposed, whereas a lack thereof indicates completion.

(PG) Mining Plain Graphs (§2)	**(DG) Mining Dynamic Graphs (§3)**
(PG1) FBOX: Identifying Suspicious Link Behavior (§2.1) [59][PDF] (PG2) DELTACON-ATTR: Cross-graph Blame Attribution (§2.2) [36][PDF] (PG3) VOG-OVERLAP: Leveraging Clustering for Plain Graph Summarization (§2.3) [44][PDF] [43][PDF]	(DG1) TIMECRUNCH: Interpretable Dynamic Graph Summarization (§3.1) [61][PDF] (DG2) M3A: Anomaly Detection in Web Searches (§3.2) [29][PDF] (DG3) FLOCK: Astroturfing in Livestreaming Platforms (§3.3) [58][PDF]
(RG) Mining Rich Graphs (§4)	
(RG1) EDGECENTRIC: Ranking Anomalies in Edge-attributed Graphs (§4.1) [60][PDF] (RG2*) Anomaly Detection with Textual Edge Attributes (§4.2) (RG3*) Unifying Structural and Edge-attributed Anomaly Detection (§4.3) (RG4*) Characterizing Link Fraud (§4.4) (RG5*) Temporal Analysis for Link Fraud (§4.5)	

Table 1.1: Overview of completed, ongoing and proposed work.

For the following three chapters (§2, §3 and §4), we motivate each task with an overarching question. For each completed subtask, we give a high-level summary, and introduce the main ideas and results. For each proposed or ongoing subtask, we motivate the problem's high level research questions as well as discuss preliminary results or potential directions for work.

Chapter 2

Mining Plain Graphs

How can we identify nodes and node groups with abnormal connectivity in plain graphs?

Plain graphs are the building block of both dynamic and rich graphs. They contain limited information which pertains to the structural connectivity between nodes. Plain graph analysis is common in many scenarios in which rich attributes nor timestamps have been collected or are available. In this chapter, we detail several works on identifying anomalous node and group behavior in such graphs. The first two works aim to find individually anomalous nodes in intra-graph and inter-graph settings. The third work complements the preceding ones by instead focusing on identifying abnormal behavior in groups of nodes via graph summarization.

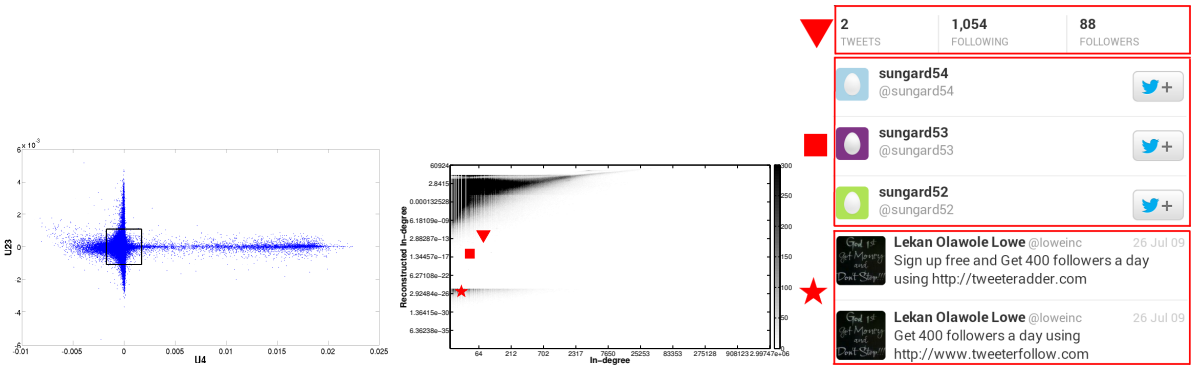
2.1 (PG1) FBOX: Identifying Suspicious Link Behavior

How can we detect suspicious accounts in large online networks? "Fake" links in social graphs due to sockpuppet/bot accounts can enable arbitrary (and frequently spammy or malicious) users and objects to seem credible and popular, thus degrading end-user experience. As a result, numerous sites such as `buy1000followers.co` and `boostlikes.com` exist to provide services such as fake Twitter reviews and Facebook page-likes, typically for just a few dollars for thousands of fake links. In [59], we introduce FBOX, an approach for identifying users and objects involved in link fraud which were undetectable by previous approaches. FBOX is (a) theoretically motivated, (b) shown to be highly effective on real data and (c) is scalable linearly on the number of nonzeros in the input graph. We applied FBOX on a large, public *41.7 million* node and *1.5 billion* edge social graph from Twitter in 2010 and found many suspicious accounts which were still unsuspended at the time of publication.

2.1.1 Main Ideas

(Informal) Problem Definition 1. Stealth Attack Detection:
Given an adjacency matrix $\mathbf{A}$ with rows/columns corresponding to users/objects in graph G
Find fraudulent users $\mathcal{U}$ and objects $\mathcal{O}$ which are undetectable by traditional low-rank spectral methods.

As mentioned above, our main focus is to identify nodes with anomalous structural graph connectivity, which is often indicative of link fraud. [56] showed that spectral methods such as singular value decomposition (SVD) were useful in identifying dense cliques and bipartite cores in graphs. As a refresher, the



(a) Spectral subspace plot (b) Proposed ISRM (c) Suspicious accounts

Figure 2.1: FBOX catches stealth attacks which are missed by spectral methods. (a) shows a low-rank spectral plot on the Twitter social graph which identifies blatant attacks but ignores stealth attackers (at the origin). (b) portrays how the proposed FBOX ISRM (In-link Spectral Reconstruction Map) describes these users by their *reconstruction degree* and identifies several with exceptionally poor reconstruction in the latent space. (c) shows their suspicious profiles with matching glyphs.

k-rank singular value decomposition factors the $n \times m$ matrix $\mathbf{A}$ as the product $\mathbf{U\Sigma V^T}$, where $\mathbf{U}$ is $n \times k$, $\mathbf{\Sigma}$ is $k \times k$ diagonal, and $\mathbf{V}$ is $m \times k$, and is optimal with respect to Frobenius loss.

These methods involved computing a low-rank decomposition, and using the projection strength of various nodes on the latent subspace to guide a dense block expansion algorithm. The main idea behind such methods is that large, dense blocks will be captured by the low-rank decomposition, since such decompositions seek to group similar users and objects together in a latent subspace.

In [59], we show that such methods which focus on nodes that project highly in the latent subspace are inherently unable to detect *stealthy attacks* of below a certain data-dependent and rank-dependent singular-value threshold. Specifically, we present and prove closed-form singular-value properties for common attack types (see [59] for more details). With this knowledge, we show that an informed adversary can engineer attacks that are undetectable under certain rank conditions. These findings additionally have implications for low-rank decomposition for recommendation purposes.

Given that these results indicate that stealthy attacks will explicitly have minimal or no projection in a low-rank subspace (see Figure 2.1a), we next aim to capitalize upon this finding. Specifically, we aim to find those users and objects which have these projection characteristics. In order to identify those users, we propose to summarize each user and each object as a tuple of their *true degree* in the original high-dimensional space, and a *reconstructed degree* in the latent subspace. We define the degree of user i in the rank k subspace as

$$degree(i) \triangleq \|(\mathbf{U\Sigma})_\mathbf{i}\|_2^2$$

where $\mathbf{U}$ and $\mathbf{\Sigma}$ denote the k-rank SVD matrices. A similar definition exists for object j:

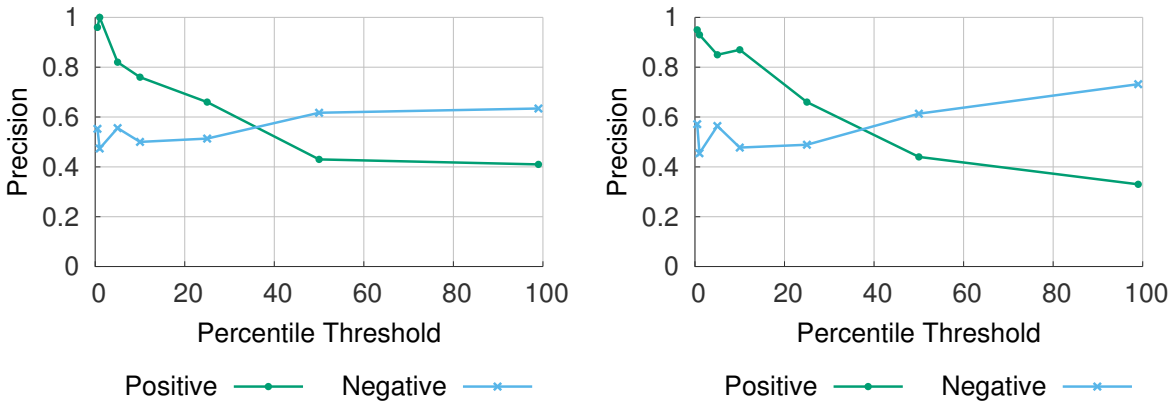
$$degree(j) \triangleq \|(\mathbf{V\Sigma})_\mathbf{j}\|_2^2$$

Note that by these definitions, the true degree corresponds to the node degree in graph-theoretic terms for binary matrices. It can additionally be shown that the true degrees upper bound the reconstructed degrees. In other words, the reconstructed degrees give us an estimate of how much the node projects into the latent

subspace. By plotting the true degrees versus the reconstructed degrees for both users (out-links) and objects (in-links), we can generate in-link and out-link spectral reconstruction maps (ISRM and OSRM) which show how users and objects of various true degree tend to project in the latent subspace (see Figure 2.1b. Our intuition is that uncharacteristically poor reconstruction indicates abnormal behavior, since these nodes exhibit much worse connectivity to the rest of the graph than their peers. We design the a lgorithm which identifies these poorly reconstructing nodes given a user-specified percentile threshold.

2.1.2 Results

As shown in the ISRM in Figure 2.1b, our intuition is correct – objects with higher true degree tend to project strongly with decreasing variance, whereas many fewer and sparser objects tend to project several orders of magnitude less (examples denoted with red glyphs). Figure 2.1c shows that the associated Twitter accounts have highly imbalanced follower-followee ratios, suspicious account names and advertisements for link fraud.



(a) Users (b) Objects

Figure 2.2: (a) and (b) show FBOX's strong (positive) predictive value and low (negative) false-discovery rate in identifying suspicious accounts.

For evaluation, we conduct a classification experiment on the Twitter graph. Specifically, we run FBOX with 7 detection threshold corresponding to reconstruction degree percentiles conditioned on the true-degrees, and randomly sample 50 accounts from the abnormal and normal user sets produced over each run. We do this for both users and objects, resulting in a total of 1400 accounts. We manually labeled these accounts as fraudsters based on a number of criteria, including whether they (a) were already suspended, (b) posted spammy tweets, (c) had suspicious usernames or (d) had sparse profiles but many followers. Figure 2.2 shows that FBOX achieves high ($\geq$ 0.93) positive precision for low detection thresholds, and negative precision improves with higher thresholds. Further results in [59] suggest that FBOX identifies several hundred thousand such fake accounts, and that as many as 70% of them were not yet suspended by Twitter at time of publication.

2.2 (PG2) DELTACON-ATTR: Cross-graph Blame Attribution

How can we pinpoint why two graphs are different, and which are the nodes and edges which most result in the change between graphs? These problems have many applications in anomaly detection and attention routing. For example, if we suddenly notice that a company's network connectivity is different

between today and tomorrow, quickly identifying and addressing the responsible machines and connections is important. In [36], we introduce DELTACON-ATTR, a blame attribution approach for such tasks. DELTACON-ATTR is (a) principled and (b) able to quickly detect structural anomalies in real graphs. We apply DELTACON-ATTR to both synthetic and real (e-mail connectivity and brain connectivity) graphs, and demonstrate both intuitive results and practical findings.

2.2.1 Main Ideas

(Informal) Problem Definition 2. Blame Attribution:
Given two adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, oriented with fixed node correspondences between graphs G_1 and G_2
Find the culprit nodes $\mathcal{V}$ and edges $\mathcal{E}$ most responsible for the change in graph structure.

Traditionally, metrics such as vertex-edge overlap (VEO [52]) and graph edit distance (GED [11]) have been used for quantifying similarity and identifying differences between graphs. However, as [37] shows, these metrics fail to satisfy intuitive properties which graph (dis)similarity metrics should obey – specifically, they both treat all nodes and edges equally. [37] also introduces DELTACON for graph similarity, which is based on an approximate variant of belief propagation, called Fast Belief Propagation (FaBP [33]), which is scalable, and offers convergence guarantees and an easily computable solution based on matrix inversion. DELTACON works by computing pairwise affinity matrices $\mathbf{S}_1$ and $\mathbf{S}_2$ for graphs G_1 and G_2 respectively, by using FaBP. These affinity matrices are derived as

$$\mathbf{S} = \left[\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}\right]^{-1}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix, ϵ is a small constant capturing influence between neighboring nodes, $\mathbf{A}$ is an adjacency matrix, and $\mathbf{D}$ is the diagonal matrix with the degree of node i as the d_{ii} entry. Entry s_{ij} essentially captures the influence that node i exerts on node j. An algebraic manipulation of the Matusita distance [46] between $\mathbf{S}_1$ and $\mathbf{S}_2$, defined as

$$Matusita(\mathbf{S}_1, \mathbf{S}_2) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(\sqrt{s_{1,ij}} - \sqrt{s_{2,ij}})^2}$$

gives the $[0, 1]$-bounded DELTACON similarity score.

DELTACON obeys a number of properties which prior methods do not:

(P1) *Edge Importance* For unweighted graphs, changes that create disconnected components should be penalized more than changes that maintain connectivity properties of the graphs.
(P2) *Edge "Submodularity"* For unweighted graphs, a specific change is more important in a graph with few edges than in a much denser, but equally sized graph.
(P3) *Weight Awareness* In weighted graphs, the bigger the weight of the removed edge, the greater the impact on the similarity measure should be.
(IP) *Focus Awareness* "Random" changes in graphs are less important than "targeted" changes of the same extent.

For this reason, we build our blame attribution algorithm DELTACON-ATTR using DELTACON as a preliminary. Specifically, DELTACON-ATTR relies upon the affinity matrices $\mathbf{S}_1$ and $\mathbf{S}_2$. Our insight is that to pinpoint the nodes most responsible for the change between G_1 and G_2, we can define their *impact* as the Matusita distance between their corresponding row vectors:

$$Matusita(\mathbf{S}_{1,i}, \mathbf{S}_{2,i}) = \sqrt{\sum_{j=1}^{n} (\sqrt{s_{1,vn}} - \sqrt{s_{2,vn}})^2}$$

Thus, these *culprit* nodes with high impact are exactly the ones whose influence has changed the most between graphs. For interpretability, we return only the culprit nodes that had an adjacent edge change between G_1 and G_2, and blame them according to the impact.

Furthermore, we can pinpoint the culprit edges most responsible for change by blaming them in accordance with the sum of the adjacent nodes' impact scores – this way, DELTACON-ATTR will tend to blame edges which touch two high-impact nodes, rather than edges between high and low-impact nodes. Edge culprits can further be categorized into edge deletions and additions, but this is purely a stylistic choice, as their impact computations do not change.

2.2.2 Results

We evaluate DELTACON-ATTR on a number of synthetically created and modified graphs, and compare it to the recently introduced state-of-the-art method CAD [62], which identifies anomalous nodes and edges based on commute time according to a random walk. We perform two experiments to evaluate (a) whether DELTACON-ATTR's ranking of the culprit nodes agrees with intuition, and (b) how DELTACON-ATTR performs in classification accuracy in finding culprits. Figure 2.3 shows that DELTACON-ATTR obeys intuitive properties on synthetic graphs – we show blame attribution results for K5, B10, L10 and S5 with respect to their associated counterparts.

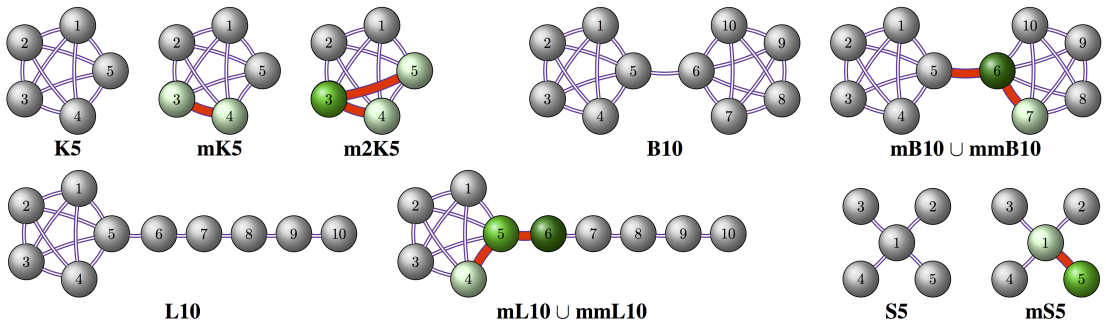


Figure 2.3: DELTACON-ATTR respects properties P1-P3 and IP. Nodes marked green are identified as culprits for change between graphs. Darker shade corresponds to higher rank in the list of culprits. Removed edges are marked in red.

We encourage readers to refer to [36] for further details, but we note here that CAD violates P2 and does not give the same intuitive results for several synthetic graphs. Furthermore, in classification accuracy experiments, we find that DELTACON-ATTR and CAD achieve roughly the same classification accuracy and almost indistinguishable ROC curves. These experiments were conducted on graphs where edge weights were sampled from 4 bivariate Gaussians, and jittered with noise. This resulted in a graph with four clusters with strong intra-connectivity, but sparse and weak inter-connectivity. We defined anomalous nodes as those adjacent to inter-cluster edges.

We additionally conducted experiments on the Enron e-mail graph of 151 employees discretized into monthly snapshots over a span of 3 years, as well as an array of brain connectome graphs for 114 human

subjects. Thanks to the de-anonymized Enron data, we were able to find some interesting qualitative results which we associate with Enron's public timeline:

- In May 2001, John Lavorato (head of Enron trading operations and CEO of Enron America) sent email to *50 previously un-contacted employees*. May 2001 is an important month for Enron, as it marks the beginning of the downfall as operations for paperwork with LJM (firm used by executives to siphon funds away from Enron) began. Lavorato was given the largest bonus ($5 million) shortly before Enron declared bankruptcy.
- In February 2002, Louise Kitchen (president of Enron Online), Liz Taylor (assistant to president), Mike Grigsby (former VP of Enron Energy Services) and Fletcher Sturm (VP) lost many *(20-50) contacts each*, and made no new ones. These employees likely left the company, as February 2002 was the month Kenneth Lay (CEO) resigned from the board, and a number of guilty executives faced legal action at Congress.

Our experiments on brain connectome graphs involved 114 subjects, each of which was associated with a 70×70 graph representing cortical regions and their inter-connectivity. We additionally had Composite Creativity Index (CCI) scores for the subjects, indicating their performance on a series of creativity tasks. Upon computing pairwise similarity over the graphs with DELTACON and applying hierarchical clustering, we found two clearly separable clusters of brain graphs, for which we found that CCI scores differered statistically significantly. Motivated by the question "what makes the low-CCI and high-CCI brains so different?" we used DELTACON-ATTR to identify the most frequent culprit nodes (cortical regions) for creativity, over pairwise comparisons between the two clusters of subjects. Our findings indicated that connectivity of cortical regions in the left hemisphere is predominantly different between low-CCI and high-CCI groups, whereas connectivity of the right hemisphere is much less to blame. Further details are discussed in both [35] and [36].

2.3 (PG3) VOG-OVERLAP: Leveraging Clustering for Plain Graph Summarization

How can we succinctly summarize large graphs? Summarization is a crucial task given the scale of modern data, as it can abstract away noise and help discover patterns which inform human processes. Graphs are an excellent candidate for the summarization task, as they are able to represent complex phenomena involving communication and other interactions between humans and objects. In [43] and [44], we propose VOG-OVERLAP, a new graph summarization method which outperforms the previous state-of-the-art method VOG [32]. Unlike previous work, VOG-OVERLAP obtains a richer set of candidate structures for summarization across multiple graph decomposition methods, and explicitly penalizes structural overlap between structures to increase the overall graph coverage. We conduct extensive evaluations on seven graph decomposition methods on real-world data, and show that VOG-OVERLAP can utilize these methods in conjunction with improved formulation and summarization heuristics to achieve huge improvements in summarization performance over existing work.

2.3.1 Main Ideas

(Informal) Problem Definition 3. Plain Graph Summarization:
Given an adjacency matrix $\mathbf{A}$ for a graph G
Find a set of possibly overlapping subgraphs to succinctly describe as much of G as possible.

Our work in [43] and [44] leverages the VoG summarization framework. VoG formulates the graph summarization problem as an information theoretic optimization problem in which the goal is to describe the graph using a vocabulary (cliques, bipartite cores, stars, and chains) that minimizes the global description length of the graph. The vocabulary is chosen to contain structures which are ubiquitous in real-world graphs and are often interesting to practitioners. The optimization task uses the minimum description length (MDL) model selection principle, which trades off between model complexity and goodness of fit. The proposed VoG optimization task is

$$minL(G, M) = min\{L(M) + L(\mathbf{E})\}$$

where $\mathbf{M}$ is an approximation of $\mathbf{A}$ induced by the model M, and $\mathbf{E} = \mathbf{M} \oplus \mathbf{A}$ is the error matrix. In order to find candidate subgraphs, VoG adapts a graph re-ordering method called SlashBurn [41] to decompose the input graph into subgraphs, and label the subgraphs using the vocabulary terms based on best fit (again, using MDL). Finally, VoG proposes several heuristics to choose the best summary from the candidate set (a combinatorial problem), including a naïve *vanilla* encoding of all structures, encoding the *top-k* structures sorted by a benefit heuristic and using *GnF* to greedily encodes structures in a single pass down the sorted list.

We notice that VoG has several disadvantages: Firstly, it does not explicitly penalize overlapping structures, meaning that the resulting summary might contain many redundant structures. Secondly, it only leverages a single method (SlashBurn) for graph decomposition, whereas other methods could also be incorporated to perhaps improve summarization quality. Thirdly, the summarization heuristics are very sensitive to the ordering of the candidate structures. VoG-Overlap aims to solve these problems.

Firstly, we notice that the VoG optimization objective shown above does not explicitly penalize overlapping structures. Thus, we propose the modification of this objective to the following:

$$minL(G, M) = min\{L(M) + L(\mathbf{E}) + L(\mathbf{O})\}$$

where $\mathbf{O}$ is a weighted matrix with the o_{ij} entry denoting the number of times that edge (i, j) has been explained by M. This objective discourages inclusion of highly overlapping structures in the model, and steers the structure selection process away from redundancy and towards higher graph coverage. Note that this still aligns with our goal of finding possibly overlapping subgraphs, just not redundant ones.

Secondly, we compare a variety of graph decomposition methods based on their summarization performance. Since different decomposition methods are designed to identify communities in different ways, we expect that analyzing the structural makeup and performance of these methods will help in improving summarization quality. We aim to compare SlashBurn [41], Louvain community detection [8], spectral clustering [48], METIS partitioning [30], BigClam [67], HyCom-Fit [3], and a newly proposed KCBC algorithm which uses recursive k-cores for graph decomposition.

Thirdly, we propose a new summary heuristic called *Step*, and two parallelized variants *Step-P* and *Step-PA* which are robust to the candidate structure ordering unlike VoG's *GnF* heuristic at the cost of increased runtime. The basic idea behind the *Step* heuristic is to choose the best structure from the remaining candidate set iteratively, instead of in a single pass as in *GnF*. This leads to high complexity for large graphs with many candidate structures, motivating the need for speedups via the parallelized variants.

2.3.2 Results

We conducted extensive experiments for (a) comparing the seven graph decomposition methods in terms of their summarization power (compression rate), graph coverage in terms of nodes and edges, structural

makeup and runtime, and (b) comparing the runtime and summarization power of the newly proposed heuristics.

Table 2.1 shows the comparative summarization power in terms of compression across the various decomposition methods. Specifically, we compare the performance of our VoG-Overlap approach using the *GnF* summary heuristic compared with the previous state-of-the-art VoG, which uses only SlashBurn. We notice that VoG-Overlap using SlashBurn performs as well, if not better than VoG across the four datasets. However, when using our proposed KCBC algorithm, summarization performance shows staggering compression improvements (bolded numbers). Comparatively, the remaining methods tend to perform worse in terms of compression performance. While the graph coverage and structural makeup results are excluded here for space constraints, we find that KCBC tends to produce only a few structures with high savings, which tend to be cliques. Our other experiments suggest that in order to build summaries with both expansive graph coverage and high savings, a combination of the algorithms can be used to compensate for each of their biases.

Dataset	VoG	VoG-Overlap + *GnF*						
		SlashBurn	KCBC	Louvain	Spectral	METIS	BigClam	HyCom-Fit
Flickr	95%	80%	**33%**	88%	98%	98%	91%	95%
Enron	75%	75%	**41%**	100%	99%	100%	77%	95%
AS-Oregon	71%	71%	**65%**	95%	94%	96%	85%	98%
Wikipedia-Chocolate	88%	88%	**78%**	99%	99%	100%	89%	100%

Table 2.1: VoG vs. VoG-Overlap + GnF: Compression rate of the clustering techniques with respect to the empty model.

Table 2.2 shows the summarization power of the *Step* heuristic and its variants compared to the *GnF* heuristic using a *unified* model containing the union of structures produced from all seven decomposition methods. Note that the proposed *Step*, *Step-P* and *Step-PA* heuristics achieve substantial 35-70% improvements in compression rate (in bits) compared to the previously proposed *GnF* heuristic on several datasets. Furthermore, *Step-P* and *Step-PA* are much faster computationally than *Step* (with *Step-PA* being the fastest), but achieve the same performance.

Dataset	*Step*	*Step-P*	*Step-PA*	*GnF*
Enron	N/A	N/A	25%	75%
AS-Oregon	35%	35%	35%	71%
Wikipedia-Chocolate	56%	56%	56%	88%

Table 2.2: Compression rate (in bits) of *Step*, *Step-P*, *Step-PA*, and *GnF*. Lower is better; N/A for runs that were terminated early.

2.4 Conclusion

In this section, we propose several algorithms for identifying anomalous nodes within and between graphs, as well as anomalous structures within a graph. fBox pinpoints many thousands of suspected fake accounts in social networks with over 93% precision. DeltaCon-Attr obeys intuitive principles with respect to graph (dis)similarity on synthetic graphs, and shows strong practical results in finding anomalous behaviors in the Enron e-mail network and attributing differences in creativity between human brain graphs. Lastly, VoG-Overlap improves upon the previous state-of-the-art for graph summarization by reducing storage cost of graphs by 35-70% via more robust summarization heuristics and our new KCBC graph decomposiion algorithm.

Chapter 3

Mining Dynamic Graphs

How can we leverage temporal information to find abnormal behaviors in dynamic graphs?

Dynamic graphs are ubiquitous, as all interactions between humans and objects occur at particular timestamps. Since humans and their interactions constantly change over time, it is natural to aim to incorporate temporal information as a feature in anomaly detection algorithms. Below, we describe several works which aim to identify abnormal node, edge and group behavior in dynamic graphs. The first work focuses on identifying anomalous and recurrent structural connectivity patterns in dynamic graphs via graph summarization. However, given that temporal signal is itself a very rich feature, the next two works focus heavily on modeling actions over time with limited use of structural connectivity information.

3.1 (DG1) TIMECRUNCH: Interpretable Dynamic Graph Summarization

How can we describe a large, dynamic graph with just a few phrases? Is it random? If not, what are the most apparent deviations from randomness – a dense block over time, or perhaps a star appearing with some periodicity? These deviations indicate patterns. In [61], we introduce TIMECRUNCH, which aims to find and rank the patterns which exist in dynamic graphs using a set of *temporal phrases* which jointly express temporal recurrence and graph connectivity. To the best of our knowledge, our work is the first to focus on the summarization task for dynamic graphs, beyond dense-block detection. TIMECRUNCH (a) has an information theoretic grounding, (b) is effective and scalable, and (c) is able to find interesting structural patterns in several large, real-world dynamic graphs.

3.1.1 Main Ideas

(Informal) Problem Definition 4. Dynamic Graph Summarization:
Given an adjacency tensor $\mathbf{A}$ for a dynamic graph G
Find a set of possibly overlapping temporal subgraphs to succinctly describe G.

[61] formulates the dynamic graph summarization task as an information theoretic optimization problem, using the minimum description length (MDL) principle for model selection. Specifically, the optimization task is

$$minL(G, M) = min\{L(M) + L(\mathbf{E})\}$$

where $\mathbf{M}$ is an approximation of the adjacency tensor $\mathbf{A}$ induced by the model M, and $\mathbf{E} = \mathbf{M} \oplus \mathbf{A}$ is the error tensor. This objective was previously proposed by VoG [32]. In fact, we leverage the same static graph decomposition and subgraph labeling framework as VoG at the start of the TimeCrunch algorithm. Both VoG and MDL are briefly discussed in §2.3.1, and are thus not expanded upon here.

The main idea of TimeCrunch is to use a set of *temporal phrases* which represent a cross product between static structures (full clique, bipartite core, star, chain) and temporal recurrence patterns (oneshot, constant, ranged, periodic, flickering) to concisely summarize the dynamic graph G. We assume that G is comprised of a series of plain graphs $G_1 \dots G_n$, discretized in a suitable granularity to the application. Unlike in VoG, where only structural connectivity needs to be modelled, the dynamic graph summarization problem setting necessitates the modelling of temporal patterns as well. Each temporal phrase jointly represents a recurrence pattern and structural connectivity. Temporal phrases with nodes and timesteps effectively describe temporal structures, which explicitly induce edges in $\mathbf{M}$. In order to arrive at a summary of temporal structures, TimeCrunch has four steps: (a) generating candidate static structures, (b) labeling candidate static structures, (c) stitching candidate temporal structures and (d) composing the summary.

Steps (a) and (b) utilize VoG's graph decomposition and static subgraph labeling framework to generate labeled static structures from each of the static graphs comprising the dynamic graph series $G_1 \dots G_n$. Essentially, this step gives us a set of static structures and the associated timesteps at which they occur.

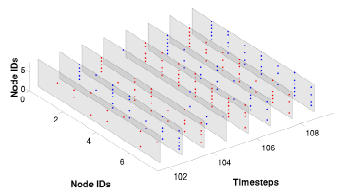
Next, in order to identify recurrent temporal subgraphs, we aim to stitch together the static subgraphs with the same structural connectivity and high node overlap across timesteps. As naïve pairwise comparison between structures to check for similarity is quadratic and intractable for large graphs, we propose a fast, approximate clustering scheme based on rank-one SVD (§2.1.1 briefly introduces SVD) and matrix deflation to cluster similar structures together. Then, once we have clustered recurrent and highly-overlapping structures together, we use MDL again to assign the recurrence pattern which fits each cluster's time series the best. This produces the set of candidate temporal structure.

Identifying the best (most concise) summary from the set of fixed temporal structures is a combinatorial problem, and thus we turn to heuristics to give us approximate solutions. We experiment with the following heuristics: *Vanilla*, *Top-10*, *Top-100* and *Stepwise*. The *Vanilla* heuristic encodes all temporal structures naively, whereas the *Top-10* and *Top-100* structures respectively encode the 10 and 100 "best" structures by a local encoding benefit heuristic. *Stepwise* takes a single pass over the sorted candidate temporal structures and only includes structures which improve the global encoding cost at the time of consideration.

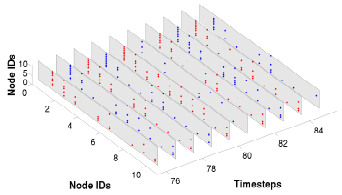
3.1.2 Results

We conduct both quantitative evaluations comparing summarization power of the various heuristics and the temporal structural makeup of several real-world dynamic graphs, as well as qualitative evaluations showing examples of interesting activities that TimeCrunch helps us find in these graphs. Of the heuristics, we find that *Stepwise* is able to concisely and consistently summarize dynamic graphs with the best compression rate and the fewest number of structures. We refer the reader to [61] to investigate quantitative experiments.

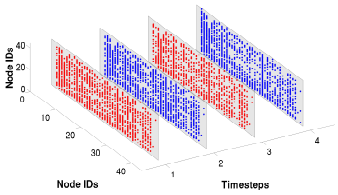
Figure 3.1 contains nine examples of interesting, recurrent temporal subgraphs which we find across five dynamic graphs: *Enron* (who-emails-whom), *Yahoo-IM* (who-messages-whom), *Honeynet* ("honeypot" network attacks, who-attacks-whom), *DBLP* (coauthorship, who-publishes-with-whom), and *Phonecall*



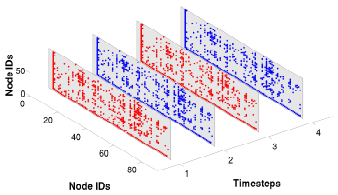
(a) 8 employees of the *Enron* legal team forming a *flickering near clique*



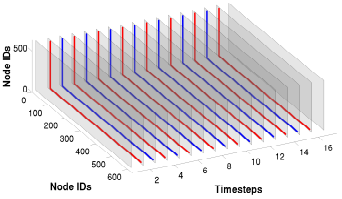
(b) 10 employees of the *Enron* legal team forming a *flickering star* with the boss as the hub



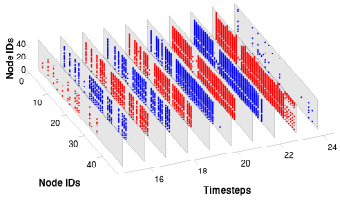
(c) 40 users in *Yahoo-IM* forming a *constant near clique* with 55% density over the observed 4 weeks



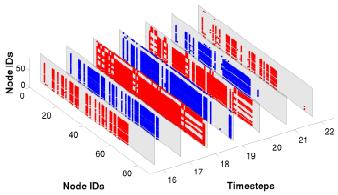
(d) 82 users in *Yahoo-IM* forming a *constant star* over the observed 4 weeks



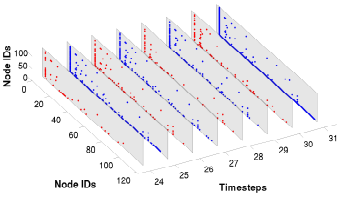
(e) 589 honeypot machines were attacked on *Honeynet* over 2 weeks, forming a *ranged star*



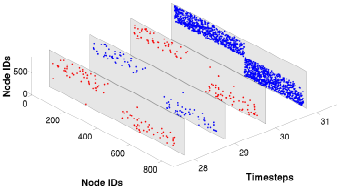
(f) 43 authors that publish together in biotechnology journals forming a *ranged near clique* on *DBLP*



(g) 82 authors forming a *ranged near clique* on *DBLP*, with burgeoning collaboration from timesteps 18-20 (2007-2009)



(h) 111 callers in *Phonecall* forming a *periodic star* appearing strongly on odd numbered days, especially Dec. 25 and 31



(i) 792 callers in *Phonecall* forming a *oneshot near bipartite core* appearing strongly on Dec. 31

Figure 3.1: TIMECRUNCH finds meaningful temporal structures in real graphs. We show the reordered subgraph adjacency matrices over multiple timesteps. Individual timesteps are outlined in gray, and edges are plotted with alternating red and blue color for discernibility.

(who-calls-whom in a large, anonymous Asian city). Some of the associated findings are highlighted below:

- Figure 3.1c shows a constant near-clique of 40 users messaging each other with 55% density in a Yahoo instant messaging network. We conjecture this is a large group-chat with users dropping in and out over time, or a number of bots messaging each other.
- Figure 3.1e shows a ranged star of 589 "honeypot" machines intentionally left vulnerable to attract attackers, being attacked by a single attacker machine over the span of two weeks. Interestingly, we also noticed that over 70% of oneshot star attacks on *Honeynet* occurred on December 31st or January 1st, indicating a high frequency of "new-year" attacks.
- Figure 3.1i shows an extremely abnormal oneshot near-bipartite core in the phonecall dataset consisting of two groups of almost 400 callers who almost never call within their own groups, but call between groups occasionally. We notice a strong densification of this pattern on December 31st,

perhaps corresponding to calls from well-wishers.

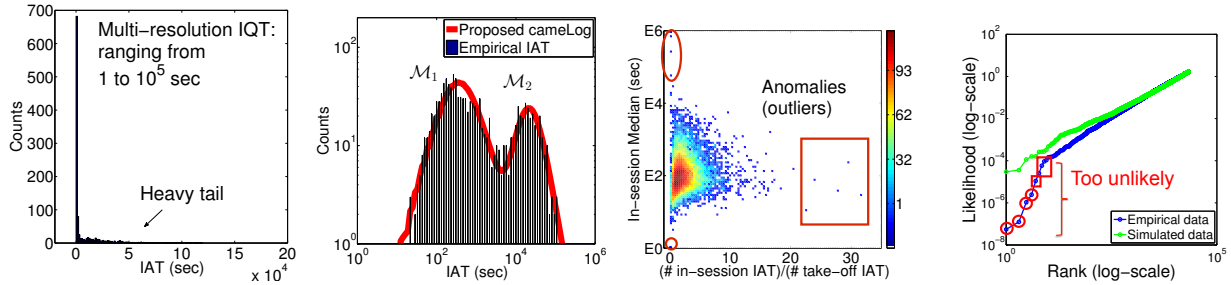
We note that ground truth is generally not available in these graphs, and therefore our plausible explanations of the unveiled structures are mere conjectures. However, the output of TIMECRUNCH offers great value to domain experts and practitioners who have access to ground truth and some investigative capacity.

3.2 (DG2) M3A: Modeling Interarrival Time in Web Searches

Consider that a user "Alice" submits one web search every five minutes, for three hours in a row – is this normal? How can we detect abnormal search behaviors amongst Alice and others? Are there any temporal patterns for user search behavior? These questions motivate our work in this section. We studied one of the largest publicly available query logs containing over *30 million* queries from 600 thousand users. We propose the M3A model, meta-model and anomaly detection framework, which operates on both user and group level to identify patterns in user web search. M3A proposes a new model for interarrival time (IAT) between search queries (also known as inter-query time, IQT) which (a) has improved modelling accuracy over other well-known distributions, (b) offers appealing quantitative interpretations, and (c) is useful for anomaly detection.

3.2.1 Main Ideas

(Informal) Problem Definition 5. Interarrival Time Behavior Modeling:
Given web query timestamps $t_1 \ldots t_n$ for each user in a set $\mathcal{U}$
Find a model describing the interarrival time between queries, and deviations from the model.



(a) Empirical IAT in lin. scale. (b) Empirical IAT in log scale and Camel-Log fit. (c) Group-level analysis. (d) Rank-weirdness plot.

Figure 3.2: M3A detects patterns and anomalies in IAT. (a) shows a histogram of IATs for a single user in linear scale. (b) shows how our Camel-Log model uncovers and models the bimodal IAT in logarithmic scale. (c) illustrates group-level analysis with scatter plot of two highly correlated Camel-Log parameters, indicating several types of anomalies (circled and boxed). (d) shows an automated way of spotting anomalies through our group-level Meta-Click metamodel: the blue deviants (in circles/boxes) correspond to the outliers (in circles/boxes) in (c).

Traditionally, IAT between events has usually been modeled as a Poisson process (independent, exponentially distributed IAT). But, is this an accurate model for user search behavior? To find this answer, we explore a large, industrial query log from Google, covering 36 million queries submitted from 657 thousand users, with timestamps at the resolution of one second. As Figure 3.2a shows, the empirical

IAT distribution actually has a rather heavy tail as opposed to an exponentially decaying one. This suggests that the assumptions of the Poisson process do not hold in practice, given that the arrival rate of queries may change, and certain queries may be submitted depending on previous queries. This poses the question: how can we actually model the IAT distribution accurately?

Figure 3.2b shows the same distribution as in Figure 3.2a, but in logarithmic scale. Interestingly, we notice a sharp bimodality in the distribution. The first contribution of M3A is the Camel-Log model. The main idea of Camel-Log is to model IAT between a user's queries using a mixture of two log-logistic (LL) distributions. As a reminder, a log-logistic distribution is a skewed, heavy-tail distribution for which the logarithm of observed values is distributed logistically. We find that LL has been previously used to model the IAT of internet communications and comments on Youtube [66], making it an attractive choice. Additionally, we find that a mixture of LLs outperforms a mixture of both exponential and Pareto distributions in terms of Kolmogorov-Smirnov tests, log-likelihood and Bayesian Information Criterion (BIC) on both the large query log dataset, as well as an auxiliary Reddit comments dataset which we manually scraped. The intuition behind the aforementioned bimodality is that users tend to have both "in-session" as well as "take-off" IATs, corresponding to a search session involving a few searches, followed by a wait/sleep time before using the search service again (see [29] for detailed experiments). The PDF of the Camel-Log distribution is written as

$$f_{Camel-Log}(t) = \theta \cdot f_{\mathcal{LL}}(t; \alpha_{IN}, \beta_{IN}) + (1-\theta) \cdot f_{\mathcal{LL}}(t; \alpha_{OFF}, \beta_{OFF})$$

where α_{IN}, β_{IN} and $\alpha_{OFF}, \beta_{OFF}$ correspond to the in-session and take-off LL parameters respectively, and θ is a mixture parameter. We can infer the parameters using the maximum-likelihood estimate.

Given Camel-Log as a means of modelling the IATs of individual users, we next ask: are there regularities in the parameters of all users? In fact, we find that the ratio R ($\triangleq \frac{\theta}{1-\theta}$) and log-median M ($\triangleq \log(\alpha_{IN})$ are strongly correlated (see Figure 3.2c). In order to model the joint distribution, we propose the use of copulas, which provide a means of modeling a multivariate, joint distribution by considering the dependency structure between univariate marginals (in this case, R and M). We notice that R and M are both well-described as following LL themselves, and thus we use Gumbel's copula (popular for modeling joint distributions of variables with heavy tails) to model their joint distribution. Using Gumbel's copula, we find that the CDF of the joint distribution between R and M, which we call Meta-Click, is written as

$$F_{Meta-Click}(r, m; \eta\alpha_R, \beta_R, \alpha_M, \beta_M) = e^{-([\log(1+(r/\alpha_R)^{-\beta_R})]^{\eta} + [\log(1+(m/\alpha_M)^{-\beta_M})]^{\eta})^{1/\eta}}$$

where α_R, β_R and α_M, β_M are the R and M LL parameters (inferred with MLE) respectively, and η is a copula hyperparameter (estimated using Kendall tau correlation). We find that using an appropriate value for η, the contour plot for Meta-Click matches the empirically observe distribution very well.

We propose using the Camel-Log and Meta-Click models to estimate the likelihood of a user's search behavior, and identify abnormal users in this way (see Figure 3.2d).

3.2.2 Results

We first evaluate the goodness-of-fit for Camel-Log. Figure 3.3 shows three examples of web search users' empirical IAT distributions in logarithmic scale with the associated MLE-fitted Camel-Log distribution, juxtaposed with the associated QQ plots showing a near-ideal fit in each case. This indicates that Camel-Log is able to model real users' varying habits quite well. See [29] for additional experiments showing similarly good fitting results on users from the Reddit comment dataset.

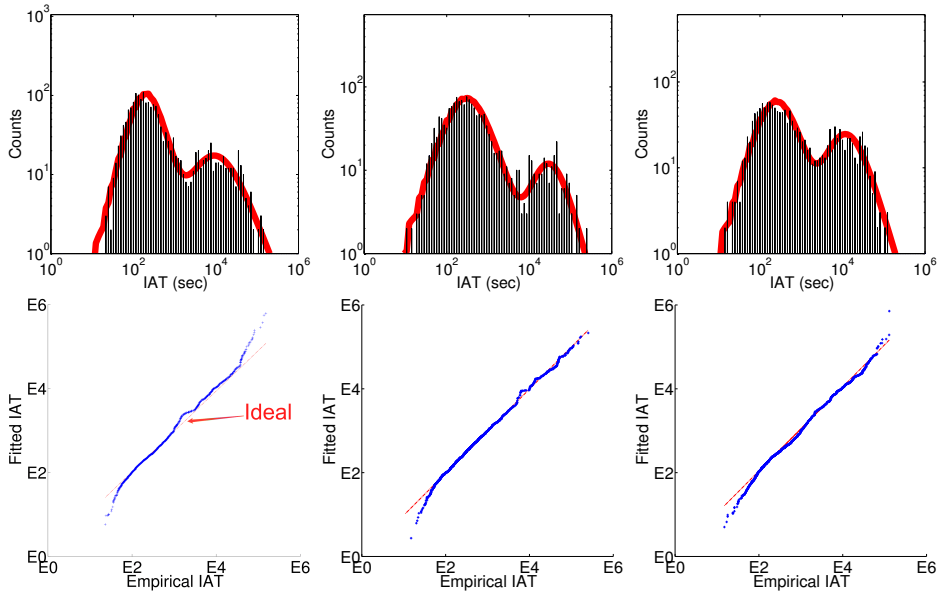


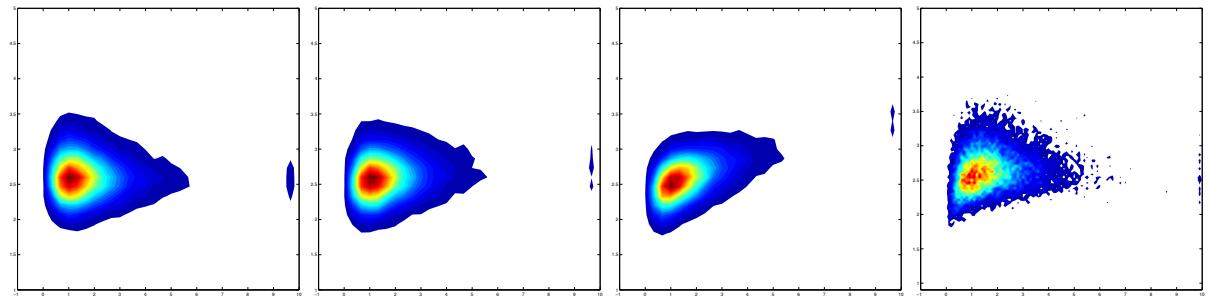
Figure 3.3: Camel-Log consistently and accurately captures the bimodal search behavior. (a)-(c) show the Camel-Log fit (red line) on different users' marginal IAT distributions. (d)-(f) show the Quantile-Quantile (QQ) plots which further substantiate that Camel-Log fits empirical data very well (45°line is ideal).

Next, we evaluate Camel-Log's goodness-of-fit against other candidate distributions: specifically, we consider the mixture of two exponential, and two Pareto distributions as contenders. We evaluate the comparative fit using three metrics: KS test, log-likelihood and BIC. Camel-Log's performance with the KS test is the best compared to the contenders, indicating that the next-best contender is the exponential mixture, whereas the Pareto mixture performs very poorly. In addition, Camel-Log explains 78% of users better compared to the exponential mixture, and over 99% of users better compared to the Pareto mixture. Similarly, Camel-Log has smaller BIC than 66% of users compared to the exponential mixture, and over 99% again due to the Pareto mixture.

Figure 3.4 further shows that the Meta-Click copula-based joint model of R and M well-approximates the empirical distribution. The rank-weirdness plot in Figure 3.2d shows a number of anomalous users which were identified as very unlikely using Meta-Click. We found that these users were often abnormally active and have disproportionate in-session and take-off ratios. We confirmed that many of these anomalous users detected by M3A did not have any sleep time and issued many "orphan" queries which resulted in no follow-through, suggesting that these are likely web-bots used to promote traffic counters, certain URLs or advertisements.

3.3 (DG3) FLOCK: Astroturfing in Livestreaming Platforms

How can we identify viewbots in livestreaming platforms? Livestreaming platforms have become increasingly popular in recent years as a means for sharing and advertising creative content (video games, music, etc.) Popular content streamers who get high viewership on their streams can earn livings by means of advertisement revenue, donations and channel subscirptions. Unfortunately, this incentivized popularity has resulted in an incentive for fraudsters to *astroturf*, or artificially inflate viewership by providing fake "live" views to customers via viewbots. In [58], we propose FLOCK, a method for combating astroturfing on livestreaming platforms. FLOCK is (a) to our knowledge, the first work to characterize and



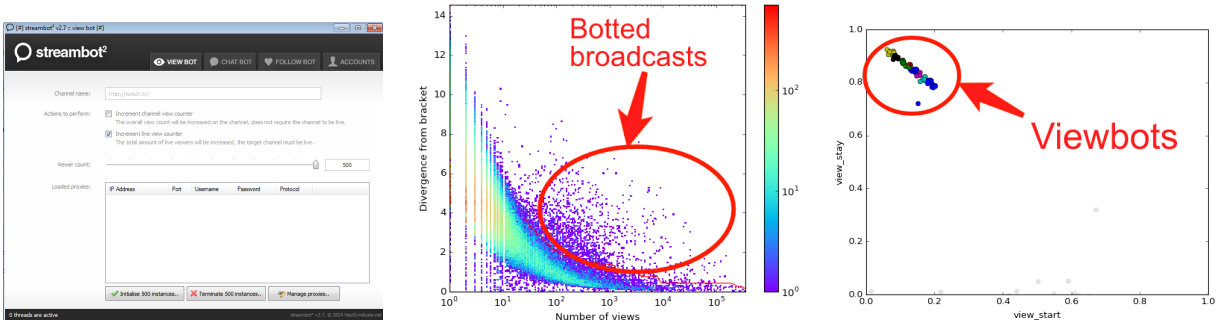
(a) $\eta = 1$ (b) $\eta = 1.12$ (c) $\eta = 1.3$ (d) Real data

Figure 3.4: Meta-Click is able to capture the joint distribution for R vs. M well. (a)-(c) show contour plots for Meta-Click with various η values. (d) shows the real distribution. In (b), $\eta = 1.12$, which is the value estimated from the real data.

tackle this problem, (b) principled and unsupervised, and (c) achieves high precision in identifying botted broadcasts and views in a real-world livestreaming workload of over *16 million* views and *92 thousand* broadcasts.

3.3.1 Main Ideas

(Informal) Problem Definition 6. Viewbot Identification:
Given a set of views $\mathcal{V}$ over streamed broadcasts $\mathcal{B}$, with associated start/end times for each broadcast $b \in \mathcal{B}$ and view $v \in \mathcal{V}$
Find the set of viewbotted broadcasts $\mathcal{B}_{botted}$ and constituent botted views $\mathcal{V}_{botted}$.



(a) Viewbotting software (b) Viewcount-deviance plot (c) Broadcast-behavior plot

Figure 3.5: FLOCK finds botted broadcasts and their botted views. (a) shows an example broadcaster-controlled botting tool used to astroturf views. (b) shows our FLOCK's viewcount-deviance plot which plots each broadcast's viewcount and model deviance – the red boundary separates botted from normal broadcasts. (c) shows a behavior plot for a botted broadcast, where each point is a view's start and stay fraction through the broadcast – notice the highly synchronized viewbot (colored) behavior where most views persist throughout the broadcast duration.

The viewbot identification problem is a challenging one, as traditional dynamic graph based methods for identifying fraud, such as PARAFAC tensor decomposition [45] and dense block search [6] are inherently unsuitable. The livestreaming context involves IPs sending HTTP requests for video content every few

seconds, indicating that they are watching a live broadcast. Consecutive requests to a given broadcast form a "view." As a result, almost every broadcast inherently forms dense blocks in the IP-time space. Furthermore, viewbots are started and stopped upon broadcaster's command (see Figure 3.5a) when they are currently streaming, making block detection on the IP-broadcast space ineffective.

We claim that labeling individual views as real or fake from information contained in HTTP requests (browser type, environment, etc.) is both difficult given the lack of ground truth and error-prone. Instead, we leverage their more robust temporal features for anomaly detection. The intuition behind FLOCK is to take an unsupervised approach which enables us to focus on temporal viewing behavior in aggregate, and identify behaviors which stand out from our model of normal viewer behavior.

We first build a model of normal broadcast behavior, where each broadcast is considered a "bag-of-views," and each view is represented with its start (v_{start}) and stay (v_{stay}) fraction over the broadcast duration. Thus, if a view starts halfway through a broadcast and stays until the end, it is represented as a tuple $(v_{start} = 0.5, v_{stay} = 0.5)$. This means that $v_{start} + v_{stay} \leq 1$. Then, we discretize the start-stay space over fixed-width intervals and model the broadcast as a multinomial distribution over the resulting space according to maximum likelihood estimate (MLE) parameters. Given that we expect viewing behavior to differ given broadcast length (viewers are likely to watch proportionally more of a 10 minute vs. 10 hour broadcast), we partition the set of broadcasts into brackets which discretize the space of broadcast durations, and model each bracket distribution separately as

$$\hat{t}(v_{start} = X, v_{stay} = Y) = \sum_{b \in \zeta(t)} \left(\sum_{v \in \rho(b)} \frac{1_{XY}(v)}{\sum_{b \in \zeta(t)} |\rho(b)|} \right)$$

where 1_{XY} is the indicator function for bin $(v_{start} = X, v_{stay} = Y)$ for view $v \in V$, $\hat{b}$ and $\hat{t}$ are the distributions for broadcast b and bracket t, $\rho(b)$ is the set of b's views and $\zeta(t)$ is the set of t's broadcasts. This gives us a joint probability density function of the relative frequencies with which viewers start and stop watching broadcasts of various durations according to MLE. We use these empirically determined parameters to describe the model of normal user behavior on a per-bracket basis. We find that over a large body of views, they are consistent from day to day.

Next, we focus on a broadcast-level analysis, where we aim to identify abnormal broadcasts which have suspicious viewing behavior. Broadcast-level analysis is useful, as it enables us to consider the abnormality of groups of views vs. individual ones. Figure 3.5b shows a viewcount-deviance plot, where each broadcast is plotted by its viewcount and the deviance (KL divergence) from its bracket distribution. Notice that most broadcasts have low deviance, except for the sparse cloud. We identify these as the botted broadcasts using an empirically tuned decision boundary.

Finally, we aim to discern the botted from the real views in the botted broadcasts. Our intuition is that botted views result in high deviance between the broadcast and bracket distributions. Furthermore, given that bots often act in close synchrony, we expect them to form dense microclusters in the start-stay space. To leverage these intuitions, we first cluster the views in each botted broadcast using a method called X-means [54], and next try to prune out clusters of views which result in reduced deviance between broadcast and bracket distributions. Given that the problem is combinatorial even for fixed clusters, we propose several heuristics:

- *Topmost* – remove the single cluster which results in maximal reduction from the original broadcast
- *Iterative* – rank the clusters according to reduction, greedily prune in that order, and repeat

- *Stepwise* – repeat *Topmost* until convergence (locally optimal choices)

The pruned views are considered botted (see Figure 3.5c).

3.3.2 Results

We first evaluate FLOCK on its performance in identifying botted broadcasts. Unfortunately, calculating precision and recall metrics are quite difficult given the lack of ground truth. However, to calculate precision, we resort to manual labeling of a random sample of 100 broadcasts each from the outlier and non-outlier regions according to our decision boundary in Figure 3.5b. We labeled the broadcasts using criteria based on whether we observed any high-density clusters of seemingly out-of-place views in the broadcast behavior plots, as well as if we observed low IP/ASN entropy of the views. On the dataset of labeled broadcasts, we found that FLOCK achieved 98% positive and 99% negative precision.

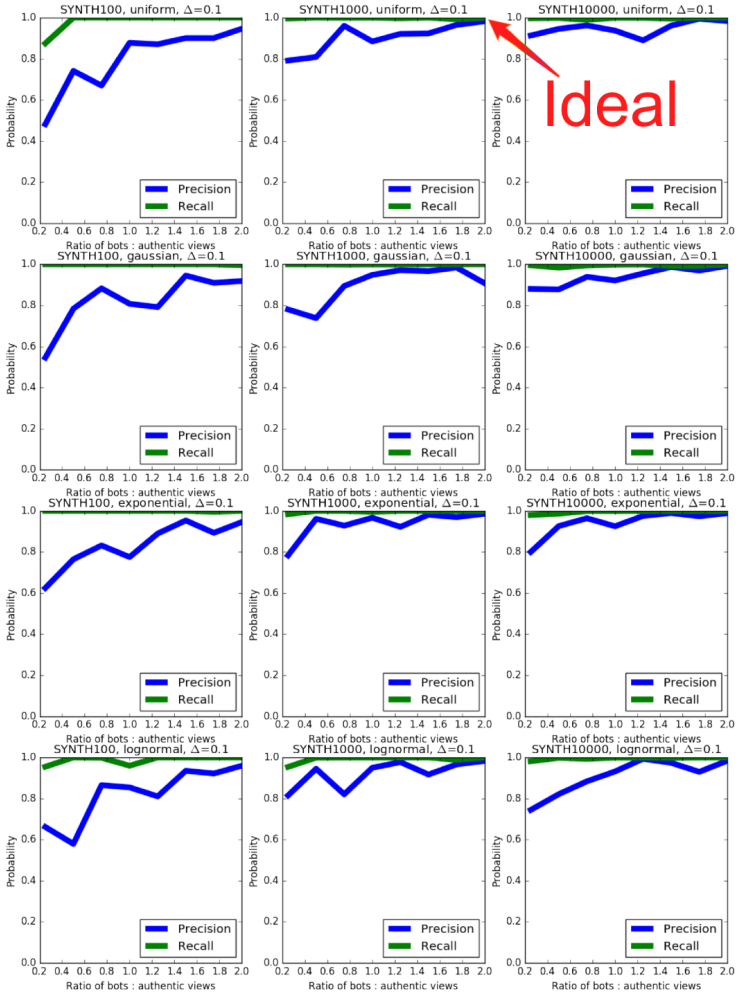


Figure 3.6: FLOCK achieves high precision/recall in discerning botted from authentic views.

We next evaluate FLOCK's pruning heuristics on their performance in improving the KL divergence objective (via mean absolute deviation) as well as total runtime (and time complexity, described in [58]). Higher MAD/MAPD is better given that it indicates a larger improvement in the reduction in deviance. We find that *Iterative* shows over 150% improvement over *Topmost* in terms of MAD, whereas *Stepwise*

only gives a 3% improvement over *Iterative*. Furthermore, *Stepwise* performs almost twice as slow due to its higher complexity for broadcasts with large numbers of view clusters. As a result, we choose to use *Iterative* in practice as it gives the best tradeoff between improving the objective and speed.

We next aim to evaluate FLOCK's performance in identifying botted views. We resort to synthetic experiments, in which we simulate botted broadcasts by generating authentic views from a real bracket distribution and jittering them, and adding botted views by sampling their interarrival and intertermination times (IAT/ITT) from uniform, Gaussian, exponential and lognormal distributions. We chose these attack distributions as viewbot providers might use them to engineer real attacks by spacing out bots. As Figure 3.5c plot indicates that botted views are often synchronous, we inject the botted views in a short duration. Figure 3.6 shows positive precision/recall results (on the botted views) from an array of experiments in which we vary authentic viewcount, botted viewcount, and attack distribution parameters before running FLOCK with the *Iterative* heuristic. We observe consistently high (≥ 0.95) recall in almost all cases, and strong (≥ 0.9) precision for broadcasts with high levels of botted activity.

3.4 Conclusion

This section details three proposed algorithms which aim to summarize structural temporal recurrence patterns in graphs, as well as focus on modeling actions over time with limited structural focus for anomaly detection purposes. TIMECRUNCH compresses dynamic graphs near-linearly on the number of edges and finds numerous abnormal structural patterns such as "constant stars" and "ranged near-cliques" on instant messaging, collaboration, phonecall, computer and e-mail networks which correspond to suspected bots and network attacks. M3A shows the near-perfect fit of modeling interarrival times between web queries as a mixture of log-logistic distributions, which fits empirical user search behavior better for 78% and 99% of users compared to exponential and Pareto mixtures, respectively. Lastly, FLOCK is the first work focusing on characterizing fake viewing behavior on livestreaming platforms, and shows over 98% precision in identifying viewbotted broadcasts, and over 90% precision and recall in identifying synthetic viewbot attacks.

Chapter 4

Mining Rich Graphs

How can we make use of attributes in rich graphs to identify abnormal nodes and their interactions?

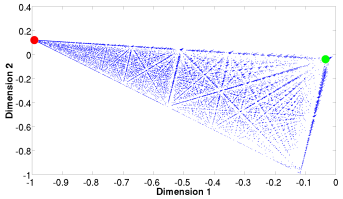
In the many contexts in which graphs represent human behaviors, we can represent complex information about the interactions via attributed, rich graphs. Attributes in rich graphs can describe auxiliary information about nodes and edges, such as rating and timestamp values in an e-commerce graph, number of messages exchanged in an online social network, and phone call duration in a telecommunications network. In this chapter, we aim to analyze and leverage these rich information types for anomaly and fraud detection. Below, we first describe our completed work on ranking anomalous nodes in graphs with categorical and numerical edge attributes. We next discuss proposed work which expands this problem to include more complex textual and structural edge attributes. Finally, we discuss our proposed work on utilizing node attributes for link fraud detection.

4.1 (RG1) EDGECENTRIC: Ranking Anomalies in Edge-attributed Graphs

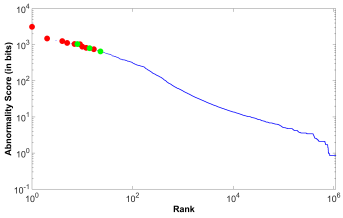
Given a graph with attributed edges, what can we say about the behavior of the nodes? For example, in a user-rates-product graph with 1-5 star ratings on edges, can we discern which users rate (and products are rated) abnormally? Furthermore, between two users with varying edge behavior, can we say which is more suspicious? This situation is common when considering rich real-world graphs which store complex information on the interactions between nodes (ratings, timestamps, etc.) In [60], we propose EDGECENTRIC, an unsupervised approach to utilize exactly this information to rank nodes based on their adjacent edge abnormality. EDGECENTRIC (a) leverages edge attributes instead of structural connectivity and node information, (b) is grounded in information theory and (c) successfully spots numerous anomalies in large, edge-attributed real-world graphs including the Flipkart e-commerce graph with over *3 million* product reviews between *1.1 million* users and *545 thousand* products.

4.1.1 Main Ideas

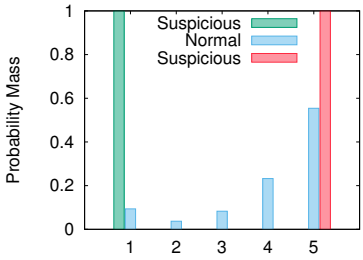
(Informal) Problem Definition 7. Edge-attributed Anomaly Detection:
Given a rich graph G with possibly multiple numerical or categorical edge attributes chosen from the attribute set $\mathcal{A}$
Find a function δ to score the abnormality of each node on its (adjacent) edges.



(a) Two clusters (red and green) of hard-to-discern fraudsters shown in a collapsed 2D subspace, reduced from the original 5D subspace over user rating values (1-5).



(b) Our approach, EDGECENTRIC, identifies the users at the red and green clusters as highly abnormal.



(c) We find that the abnormal users in the red cluster give only 5 star ratings, whereas users in the green cluster give only 1 star ratings.

Figure 4.1: EDGECENTRIC spots abnormal users on real graphs. On Flipkart user-product ratings, EDGECENTRIC finds users who greatly deviate from typical behavior – the red and green clusters contain single-mindedly "enthusiastic" and "disgusted" users who only give 5 or 1 star reviews respectively, compared to the global (J-shape) behavior shown in blue.

EDGECENTRIC leverages the minimum description length (MDL) principle (see §2.3.1 for a brief introduction) to rank abnormality of nodes based on the distribution over the attribute values of adjacent eges. Unlike most previous works which use MDL to find the best model given some data, we instead use it to evaluate how well the data fits a given model. The intuition behind our approach is that data which fits the edge attribute model well enjoys high compression, while data which is poorly represented is more costly to encode.

In order to see how we can leverage MDL to formulate the abnormality function δ, we must first consider our model and data representations. In this description, we use the user-rates-product example with a rating attribute (1-5 stars) to simplify explanations; however, the attribute(s) can actually be any numerical or categorical feature. Since MDL requires lossless encoding, we aim to have a model from which we can losslessly reconstruct the set of ratings (and generally, attribute values) for each user and product. For this reason, we treat our model as the empirical distribution over the rating attribute values across all edges in the dataset. The blue J-shaped curve in 4.1c indicates such a global model distribution across ratings. Given this global distribution C as our model, we next ask: how expensive is it to encode a given user v's rating distribution with respect to the model? In fact, the extra cost of encoding a sample from v's distribution $\hat{v}$ using the optimal code for C's distribution is $KL(\hat{v} \parallel C)$ bits, where KL denotes the Kullback-Leibler divergence. To encode the entire set of v's ratings f_v, the extra cost in bits (overhead) is defined as

$$\delta_{base}(v) = |f_v| \cdot KL(\hat{v} \parallel C)$$

This is the base formulation for our abnormality function δ. Intuitively, the function penalizes both increasing deviance between the user distribution $\hat{v}$ and the global distribution C, as well as increasing scale, or volume of the deviance (in number of ratings). Nodes which are party to many actions (edges) and behave abnormally with respect to the model thus cost more bits to encode.

We additionally posit that since user behavior is often more complex than singular, global trends, it can be useful to have different models of rating behavior. For example, some users will tend to rate more positively, some more negatively, and some perhaps uniformly. δ_{base} can be extended to incorporate

multifaceted behaviors by clustering users into several groups of varying proportions, and calculating the expected overhead in bits with respect to these cluster distributions versus a single global one:

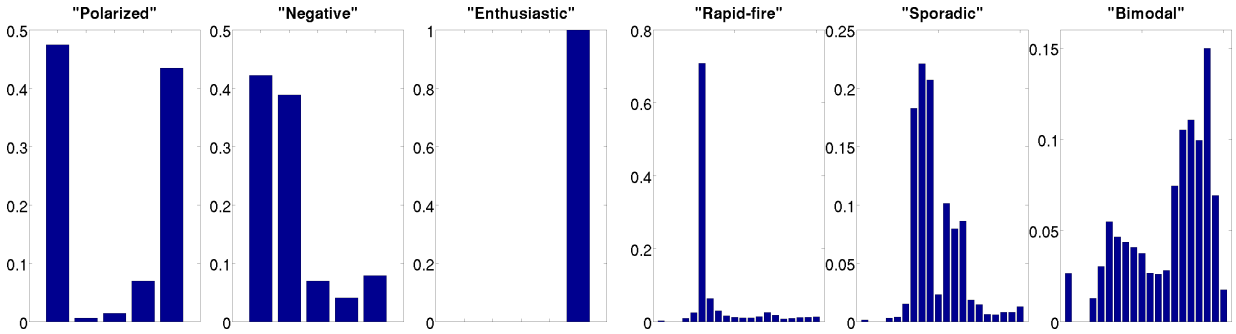
$$\delta_{mf}(v) = |f_v| \cdot \sum_{g=1}^{h} \left(\rho_g \cdot KL(\hat{v} \parallel C_g)\right)$$

where C_g and ρ_g give the gth cluster distribution and its associated proportion (out of 1) across all nodes, respectively. We additionally extend this multifaceted abnormality function to multi-attribute and multi-relational contexts by considering individual attributes and relations additively and independently. By avoiding considering joint distributions of multiple attributes in cases where they exist, we avoid the combinatorial explosion and sparsity issues associated with the curse of dimensionality.

To rank nodes in this fashion, EDGECENTRIC first aggregates the attribute values for each attribute over outgoing edges conditioned on each node type in G. Next, given the attribute types and ranges, we discretize linearly or logarithmically for numerical attributes (discretizing categorical attributes is trivial). Next, we employ a BIC-based clustering method called X-means [54] across each node's attribute distributions, assigning nodes to clusters based on l_2 distance. Given the cluster and node attribute distributions, we compute the final abnormality score δ (see [60] for details) for each node. Finally, we rank the nodes according to this value and return the rankings to the practitioner for further investigation.

4.1.2 Results

To evalute EDGECENTRIC, we used three large, real-world e-commerce datasets with rating and interarrival time (IAT) attributes: Flipkart and Amazon Health user-rates-product-at-time graphs, as well as an online software marketplace (SWM) user-rates-application graph. We ask: what kinds of edge-attribute behavior do we observe in real graphs? And furthermore, is EDGECENTRIC able to spot abnormally behaving nodes in these graphs?



(a) Ratings (b) IATs

Figure 4.2: Sample cluster distributions for user ratings (a) and IATs (b) on Flipkart. Bins in (a) correspond to 1-5. Bins in (b) are log. spaced – the first few span from several seconds to a few minutes.

Figure 4.2 show some Flipkart user behaviors uncovered by clustering the rating distributions. (a) shows three types of raters:

- "Polarized" raters who give mostly 1s or 5s
- "Negative" raters who tend to rate very pessimistically
- "Enthusiastic" raters who tend to rate only 5s

(b) shows three types of IAT patterns:

- "Rapid-fire" users who suspiciously rate mostly a few seconds or minutes apart
- "Sporadic" users who tend to rate days to weeks apart
- "Bimodal" users who go months without rating products, but occasionally have periods of frequent activity

Upon applying EDGECENTRIC to this dataset, we gave a list of the 250 most abnormal accounts to domain experts at Flipkart, who investigated and labeled the users. EDGECENTRIC attained 0.87 precision at 100 users, and over 0.7 precision over the top 250. As Flipkart does not yet seem to have organized fraud, these results are significant. Most fraudsters spammed 4s/5s to multiple products from a single seller, or spammed 1s/2s to products from competitors. The most abnormal user had given 3,692 ratings (all 5s) with an average IAT of just a few seconds.

We found similar results on the SWM dataset, for which we only used the rating attribute. We found the users with highest abnormality scores had very spammy behavior. The most abnormal user had given 186 5s to a single application. The accompanying reviews included quotes like *"Awesome!!!,Get this app now and earn points for a $10 gift card"* and *"Awesome App!!!! FREE money ,The app is great to earn points for FREE money. Get it today!"* While we do not have ground truth, we found that the top 20 users according to EDGECENTRIC often posted repetitive, spammy text in addition to skewed ratings.

Our analysis on the Amazon Health dataset pinpointed abnormal users who were often high-status reviewers who were given products for free in exchange for ratings. As a result, we noticed phrases such as *"Note: sample unit provided for reviewing purposes."* at the end of reviews. We found that these users were extremely biased and likely disingenuous, universally giving 80-90% 5s. The most abnormal products were similarly top fitness products such as *BlenderBottle* and *FitBit* which received many tens of thousands of ratings which were unnaturally skewed, suggesting suspicious activity.

4.2 (RG2*) Anomaly Detection with Textual Edge Attributes

High-level Question: *How can we use textual edge attributes to find abnormal user behavior?*

In §4.1, we discussed our EDGECENTRIC method which uses categorical and numerical edge attributes to find users behaving abnormally in rich graphs. However, such graphs often contain textual features as well, describing emails, chat messages between users, product reviews, etc. We propose to study whether, and how we can incorporate textual attributes into this unsupervised anomaly detection framework. In fact, opinion spam and fraud detection in product reviews has been a hotbed of text mining research in recent years, with most methods focusing on supervised classification using n-gram and psycholinguistic features [50] [39].

As text is a complex and unstructured data type, we must carefully identify means in which we can extract more well-defined information from it. Given that EDGECENTRIC uses the idea of treating edge attributes as samples from the adjacent source node's attribute distribution, we aim to generalize this idea to textual features. One promising approach is to extract categorical and numerical features from the text on each edge itself. Some possible features which can be computed on an edge/message level include

- Length of message
- Character and word entropy of messages
- Proportion of non-alphanumeric characters

This transformation allows us to transform the complex text attribute into simpler attributes which are easier to model. Another possibility is to treat each node as a document, and the aggregate of outgoing edges as a bag-of-words. This representation can allow us to apply techniques such as LDA [7] to more directly extract a distribution from the text. These varying features and approaches model different aspects of the text, including redundancy, likelihood of spam, relevance and topical content, and more.

We anticipate the major contributions of this work will be the identification of, and practical experiments concerning highly telling textual features which can enable efficient unsupervised ranking of anomalies. Furthermore, the work will serve to nicely extend the EDGECENTRIC framework to support yet another desirable attribute class.

4.3 (RG3*) Unifying Structural and Edge-attributed Anomaly Detection

High-level Question: *How can we integrate structural and edge-attributed anomaly detection into a unified framework?*

Structural anomaly detection (primarily discussed in §2) often focuses on using random-walk based methods, matrix factorization and graph decomposition to identify abnormal connectivity behavior in an unsupervised fashion. These methods tend to operate solely on extracting meaning from the graph adjacency matrix by means of identifying dense blocks, abnormally high degree but poorly connected nodes, and other types of interesting structures for practitioners (chains, stars, etc.) As such, they do not consider attribute information in finding outliers.

Conversely, methods which focus on edge attributes like EDGECENTRIC (§4.1) focus on statistical profiling and finding outlier nodes who have abnormally deviant edge-attribute distributions at large scale. However, EDGECENTRIC does not make much use of graph connectivity beyond identifying the adjacent nodes and edges. It is unable to leverage information about how the edges are distributed across different groups of nodes.

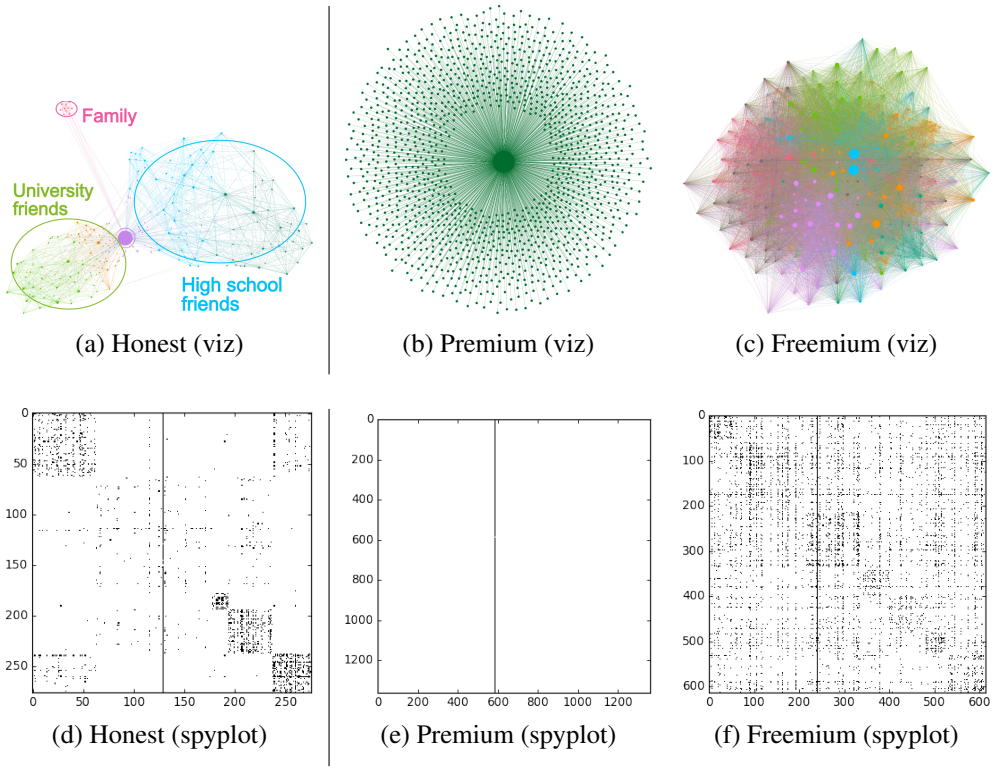
For example, if 500 users rate the same 200 products, it is suspicious due to the structural unlikeliness of a dense 500×200 bipartite core in the data. However, if we now consider that all the ratings were 5 stars, it is suspicious due to the unlikeliness of the rating attributes. In this proposed work, we aim to find ways to bridge the technical gap between these two directions which operate on orthogonal information sources.

One promising approach for doing so is treating the adjacent edges of each node as being drawn from a graph "membership" distribution. [21] takes this approach by applying the traditional LDA topic modeling algorithm to the graph community detection setting, where nodes are treated as documents, and adjacent edges as their words. We can then construe the topic distributions on each node as distribution of communities where the given edges connect. From this point, we can model connectivity much the same way as any other attribute on edges and incorporate this into the EDGECENTRIC framework. Incorporating structural connectivity into an edge-attributed anomaly detection framework is a promising direction for holistic anomaly detection on rich graphs.

4.4 (RG4*) Characterizing Link Fraud

High-level Question: *Can we distinguish the behavior of link fraudsters from normal users using their connectivity information and node attributes?*

Most previous works on social network link fraud detection try to separate honest users from fraudsters while implicitly assuming a single type of fraudulent behavior. But, is this assumption actually true? And if not, what are the distinct characteristics of such fraudulent behaviors? To answer these questions, we are conducting ongoing work in which we use "honeypots," or intentionally vulnerable victim Twitter accounts on which we purchase fake followers from different providers to observe their attribute properties and connectivity behavior. Purchasing fake followers on honeypots and meticulously tracking their activities gives us a labeled fraudster dataset which is especially hard to find in practice, and enables us to study varying aspects of fraudulent accounts, including their delivery rate, follower and followee count, tweet messages and more. For example, fraudsters might all deliver a batch of accounts every 5 minutes, or may have high (> 1000) followee but low (< 100) follower counts. They may all post spammy tweets, obfuscate external links with URL shorteners, and so on.

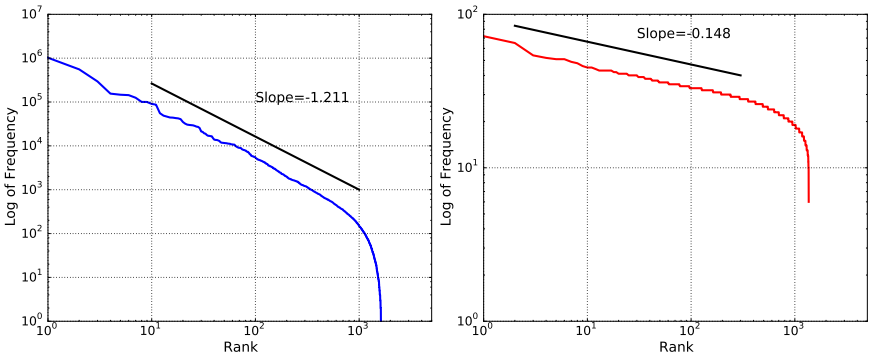


(a) Honest (viz) (b) Premium (viz) (c) Freemium (viz)

(d) Honest (spyplot) (e) Premium (spyplot) (f) Freemium (spyplot)

Figure 4.3: Freemium and premium fraud has noticeably different local network structure compared to honest following behavior. Nodes are colored by modularity class, and sized proportionally to in-degree in (a)-(c). Associated, reordered adjacency matrices are shown in (d)-(f) – the vertical line in each spyplot indicates the followers all following a central node. Notice the block community structure in honest followers compared to the star structure in premium and dense near-clique structure in freemium fraudsters.

In fact, our current work suggests that link fraud is in fact not unimodal like most previous works assume, but rather multimodal. Specifically, we find two different overarching types of fraud which have characteristically different properties. First, we notice there are "premium" fraudsters, which tend to be artificially populated sybil accounts and are only used to deliver fake links to paying customers. Secondly, there are "freemium" fraudsters, who are in fact real users who otherwise give their account credentials to a provider who trades follows between the real accounts for free, and additionally uses these real accounts to satisfy some paying customers. Figure 4.3 shows the differences in local network structure of these



(a) Genuine user #followers (b) Fraudulent user #followers

Figure 4.4: Genuine and fraudulent users' attribute distributions differ. (a) and (b) show varying slopes indicating distinct distributions of follower counts across genuine and fraudulent users, respectively.

types of fraudsters versus honest followers, centered on two honeypots and a real account for which we collected network connectivity information.

Our ongoing efforts focus on additionally studying the account attribute distributions over these followers as well. We ask: do they have any regularities? Can we use these features in distinguishing reliably between honest and fake users? Furthermore, can we build a classifier on both account and network features and show that a multi-class problem formulation which separates the honest, freemium and premium classes is more accurate than binary setting? Answering these questions has important applications in how supervised link fraud detection is conducted in practice.

4.5 (RG5*) Temporal Analysis for Link Fraud

High-level Question: *Can we identify link fraud campaigns by examining account follower attribute distributions over time?*

While our ongoing work in §4.4 focuses on identifying static features about the different modes of link fraud in the hopes of improving classification of individual accounts, in this proposed work, we aim to characterize the dynamic behavior of account followers to find groups, or campaigns of synchronized link fraudsters. As fake links are often delivered within short periods of time and in sizable amounts, it stands to reason that a group-level analysis could more easily catch many fraudsters simultaneously. We plan to track the distributions over accounts' follower attributes over time to examine patterns and abnormal behaviors in the types of followers that a given account gets.

We hypothesize that there should be some regularities in the types of followers that an account attracts from day to day. This regularity across multiple attributes (whether the follower accounts have default profiles, when they were created, their language and country of origin, etc.) can be measured using entropy or some other metrics to quantify/summarize the individual attribute distribution over the incremental follower set. Positive and negative spikes in these statistics over time can be noteworthy, and indicate anomalous occurrences. For example, if the account of an English speaking user tends to get almost exclusively English speaking followers each day, but one day gets followers who predominantly speak Russian, this suggests that the account was abnormally delivered some links that would otherwise not be characteristic of his followership. Figure 4.4 shows the differences in the "follower count" attribute distribution across the set of followers of a genuine, and known fraudulent Twitter user by means of rank-

frequency plots in log-log scale. These figures serve to show the sharp contrast between the shapes of the attribute distributions between these behavior types.

We plan to track the publicly accessible activity of incremental followers of both honest accounts, and honeypots for which we attract premium and freemium followers, and determine which features are useful in discerning fraudulent behaviors. This idea is especially promising, as it would both (a) further complicate the process for fraudsters to collect/generate usable accounts, as well as (b) greatly complicate the way fake links need to be delivered to buyers such that they avoid being caught. Given that the pool of usable accounts for fraudsters is likely limited, and account properties are not very trivial to change over time, we conjecture that a successful temporal method to detect fraud campaigns would greatly decrease profit margins for fraudsters.

4.6 Conclusion

In this section, we discuss proposed approaches for identifying anomalous nodes in attributed graphs. Our EDGECENTRIC algorithm offers an interpretable and principled way of ranking abnormal nodes in graphs with categorical and numerical edge attributes, and shows over 87% precision in finding fraudulent users in the Flipkart e-commerce network, amongst others. We next discuss two planned future works for extending the approach to account for a wider class of anomalies, including textual edge-attributes (reviews, messages, etc.) and structural connectivity. Lastly, we discuss two works which use honeypot accounts to (a) characterize the structural connectivity and attribute behavior of real link fraudsters, and (b) spot link fraud campaigns via temporal analysis of follower behaviors.

Chapter 5

Related Work

5.1 Plain Graphs

Matrix factorization approaches: Most works on anomaly detection for plain graphs aim to identify rare substructures or abnormal node connectivity behavior. One class of works aims to use spectral decomposition. [56] leverages low-rank SVD can to extract dense blocks based on the "eigen-spokes" pattern formed in the singular vector spectral subspace plots of distinct communities. [25] expands upon this pattern, and shows the manifestation of fraudster camouflage and partially overlapping attacks as "tilting rays" and "pearls" in spectral subspace plots on social graphs. [24] identifies abnormal social graph connectivity based on the synchronicity and normality of top singular vector features of nodes' neighbors.

Propagation approaches: A number of algorithms have been proposed for finding fake links in social networks by using random-walk and propagation based methods. [69] proposes a decentralized method for finding fake links which relies on using intersecting random walks to leverage abnormal graph cut between honest and fake regions of the social graph. [68] follows a similar idea, but modifies the intersection conditions from the previous work to offer stronger theoretical bounds. [12] ranks nodes based on their likelihood of being fake links based on propagating "trust" over the graph. [13] employs a similar trust propagation idea, but incorporates user reports and negative feedback to improve the ranking. [51] is another propagation-oriented approach which uses the well-known Belief Propagation algorithm applied on e-commerce graphs to identify fraudsters in a semi-supervised setting. [33] shows the equivalence conditions between random walk and propagation approaches, both of which are means of measuring guilt by association. [42] idnetifies several patterns related to interaction between nodes in a who-influences-whom phone application network.

Compression approaches: A number of works use MDL and compression-based arguments to find outliers. [14] finds dense blocks of nodes by reordering the graph adjacency matrix according to an MDL objective, and identifies anomalous edges based on their induced deltas in overall encoding cost. [30] and [48] propose multilevel partitioning and spectral clustering schemes which can also be used to find dense blocks. [23] uses a greedy algorithm and column weighting to identify dense blocks in social networks. [32] uses MDL for the graph summarization task, which aims to find interesting graph structures beyond dense blocks. [1] finds anomalous nodes in graphs based on egonet features.

5.2 Dynamic Graphs

Time series approaches: Dynamic graph anomaly detection expands this problem definition to identify abnormal temporal behaviors and change points in connectivity over time. One class of such works involve computing features over a sequence of graphs and detecting change points in their time series. [10] uses Graph Edit Distance (GED) to measure the "error" between graphs in terms of topological operations. [53] proposes the λ-distance for measuring changes between graphs in terms of the spectrum of the adjacency matrix or graph Laplacian. [52] identifies several alternative similarity measures including Signature Similarity (SS) based on hashing, and vertex edge overlap (VEO) which measures similarity based on overlap. [5] extract local egonet features over nodes in consecutive timesteps and finds anomalies in the summary statistics over all nodes.

Matrix factorization approaches: Other approaches use matrix or tensor decomposition over graph snapshots to identify abnormalities. [65] uses compact matrix decomposition to summarize each graph snapshot as a low-rank summary, for which the reconstruction error is tracked over time to find anomalous snapshots. [64] proposes streaming tensor analysis in the same vein, which involves tracking the "energy" of incremental tensor decompositions to identify anomalous edges. [34] proposes using the PARAFAC tensor decomposition to find rare events and microclusters. [45] also uses tensor decomposition for intrusion detection in networks.

Compression approaches: Some methods extend the MDL objective from plain graphs to a temporal setting. [4] uses repeated rank-one tensor decompositions and deflation, paired with an MDL objective to extract dense blocks from the rank-one vectors. [63] partitions each graph snapshot individually using MDL, and again uses MDL to separate the set of snapshots into "segments" of similar behavior – the change points between segments represent anomalies.

Scan statistics-based approaches: Scan statistics are also frequently used for dynamic anomaly detection. [57] use sliding window analysis to identify snapshots in the Enron email network which have abnormally dense connectivity. [47] proposes scanning for the count of stars and chains, as motivated by hacker attacks in real networks.

5.3 Rich Graphs

Node-attributed subgraph mining approaches: Anomaly detection in rich graphs aims to leverage any and all information available in the form of connectivity and attributes to find abnormalities. Several works aim to cluster node-attributed graphs into dense groups of similar users, which can be both indicative of communities and fraudulent users. [70] proposes a means of learning an objectively which jointly considers structural density and node-attributed similarity in order to partition a graph into mostly homogeneous node-attributed clusters. [18] aims to solve a similar problem by means of a pruning algorithm, but focuses instead on extracting clusters rather than partitioning the graph. [17] jointly models structural connectivity and node attributes using hidden Markov random fields in order to find community outliers, or nodes which have anomalous attributes with respect to their local community. [55] tackles a similar problem, except it also aims to use user input to identify relevant attribute subspaces for clustering and outlier detection.

Compression approaches: Several methods formulate MDL objectives for the rich graph anomaly detection problem as well. One of the earliest such works was introduced in [49], which uses MDL to identify anomalous categorical attribute-takes-value subgraphs which compress the graph poorly. [2] finds dense blocks of similar connectivity and node attribute values by altering a clustering step with a joint adjacency and attribute matrix shuffling step to greedily minimize an MDL objective.

Edge-attributed approaches: [22] takes a Bayesian approach to finding e-commerce fraud by modeling ratings and interarrival times via a graphical model. [28] proposes mining association rules from review data based on relationships between the reviewer, product and brand versus the rating, and identifying unexpected rules and rule groups as likely spammers. [40] aims to identify e-commerce fraud via several disparate features including targeted ratings and similarity in review text. [31] focuses on finding colluding agents who pollute responses in crowdsourcing services for monetary gain via pairwise user response comparisons in categorical question/answer tasks.

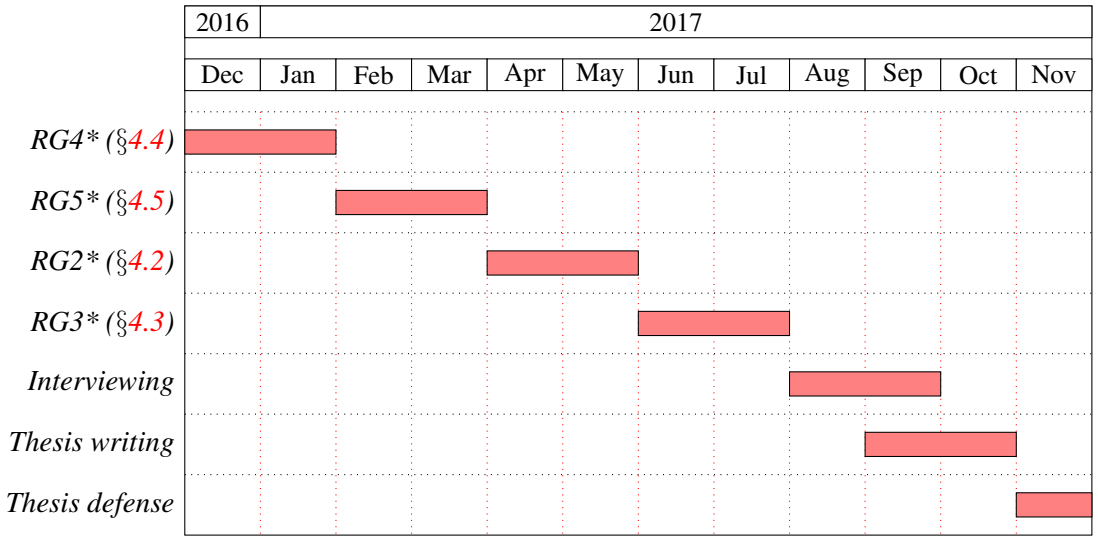
Unsupervised feature-based approaches: [19, 20] propose approaches to better sampling for rare class discovery in active learning scenarios using local topological density estimation. [9, 15] propose the popular local-outlier factor (LOF) and density based spatial clustering of applications with noise (DBSCAN) methods which give points in a feature space outlier scores based on nearest neighbors. [38] gives a broad overview of these and other similarly motivated density-based clustering approaches. [26] proposes a scalable way to find top-n local outliers in large databases using microclusters to compress the data. [27] suggests incorporating "reverse nearest neighbors" of points in addition to nearest neighbors to more robustly find local outliers in spaces where neighborhood density varies greatly. [16] discusses a method for integrating output of various anomaly detection algorithms by measuring probabilistic concensus between the algorithms via propagation.

Chapter 6

Timeline

My proposed timeline is as follows:

- **Dec 2016 - Jan 2017:** Characterizing link fraud (§4.4)
- **Feb 2017 - Mar 2017:** Temporal analysis for link fraud (§4.5)
- **Apr 2017 - May 2017:** Anomaly detection with textual edge attributes (§4.2)
- **Jun 2017 - Jul 2017:** Unifying structural and edge-attributed anomaly detection (§4.3)
- **Aug 2017 - Sep 2017:** Interviewing
- **Sep 2017 - Oct 2017:** Thesis writing
- **Nov 2017:** Thesis defense



Chapter 7

Conclusion

In this thesis, we study the problem of identifying anomalous and fraudulent behavior in large, social graphs. Our work makes numerous contributions over three main thrusts:

(PG) Mining plain graphs: [59] finds many previously unsuspended link fraudsters in social networks which are modeled poorly by low-rank factorization methods. [36] introduces a principled cross-graph node and edge blame attribution algorithm. [43] improves over state-of-the-art graph summarization and finds abnormal structures with high graph coverage. Of these, [59] identifies social networks link fraudsters with over 93% precision and identifies fake accounts of which as many as 70% were yet unsuspended.

(DG) Mining dynamic graphs: [61] introduces a dynamic graph summarization algorithm which abnormal temporal structures in real graphs. [29] models the bimodality of search interarrival times and finds hyperactive users with odd temporal behavior. [58] characterizes the livestreaming problem in astroturfing and aims to solve it using a model of view start/end times. Of these, [58] identifies botted livestreaming broadcasts with over 98% precision, and the constituent botted views with over 90% precision and near-perfect recall.

(RG) Mining rich graphs: [60] leverages categorical and numerical edge attributes in graphs to rank nodes based on the abnormality of their asscoiated interactions, and achieves high precision in finding fraudsters on a real e-commerce network. Of these, [60] uncovers latent user and fraudster behaviors in real datasets, including "enthusiastic" and "disgusted" raters, and achieves 87% precision on top-ranked users at Flipkart.

We additionally discuss a number of proposed and ongoing works on the third task, concerning rich graphs. Two of these works focus on studying node attributes and graph connectivity in static and dynamic contexts to identify properties of link fraud and pinpoint fraudulent campaigns, respectively. The other two works focus more algorithmically on leveraging text for edge-attributed anomaly detection, and unifying structural and edge-attributed anomaly detection. We believe that these additional works will improve the state-of-the-art in graph-based anomaly detection.

Bibliography

[1] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining*, pages 410–421. Springer, 2010.

[2] Leman Akoglu, Hanghang Tong, Brendan Meeder, and Christos Faloutsos. Pics: Parameter-free identification of cohesive subgroups in large attributed graphs. In *SDM*, pages 439–450. SIAM, 2012.

[3] Miguel Araujo, Stephan Günnemann, Gonzalo Mateos, and Christos Faloutsos. Beyond blocks: Hyperbolic community detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer, 2014.

[4] Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos E Papalexakis, and Danai Koutra. Com2: fast automatic discovery of temporal (comet) communities. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 271–283. Springer, 2014.

[5] Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. Netsimile: a scalable approach to size-independent network similarity. *arXiv preprint arXiv:1209.2684*, 2012.

[6] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd international conference on World Wide Web*, pages 119–130. ACM, 2013.

[7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[8] VD Blondel, JL Guillaume, R Lambiotte, and E Lefebvre. The louvain method for community detection in large networks. *J of Statistical Mechanics: Theory and Experiment*, 10:P10008, 2011.

[9] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[10] Horst Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[11] Horst Bunke, Peter J Dickinson, Miro Kraetzl, and Walter D Wallis. *A graph-theoretic approach to enterprise network dynamics*, volume 24. Springer Science & Business Media, 2007.

[12] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 197–210, 2012.

[13] Qiang Cao and Xiaowei Yang. Sybilfence: Improving social-graph-based sybil defenses with user negative feedback. *arXiv preprint arXiv:1304.3819*, 2013.

[14] Deepayan Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 112–124. Springer, 2004.

[15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[16] Jing Gao, Wei Fan, Deepak Turaga, Olivier Verscheure, Xiaoqiao Meng, Lu Su, and Jiawei Han. Consensus extraction from heterogeneous detectors to improve performance over network traffic anomaly detection. In *INFOCOM, 2011 Proceedings IEEE*, pages 181–185. IEEE, 2011.

[17] Jing Gao, Feng Liang, Wei Fan, Chi Wang, Yizhou Sun, and Jiawei Han. On community outliers and their efficient detection in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 813–822. ACM, 2010.

[18] Stephan Gunnemann, Ines Farber, Brigitte Boden, and Thomas Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *2010 IEEE International Conference on Data Mining*, pages 845–850. IEEE, 2010.

[19] Jingrui He and Jaime Carbonell. Prior-free rare category detection. *learning*, 2:5, 2009.

[20] Jingrui He and Jaime G Carbonell. Rare class discovery based on active learning. 2008.

[21] Keith Henderson and Tina Eliassi-Rad. Applying latent dirichlet allocation to group discovery in large graphs. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1456–1461. ACM, 2009.

[22] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Gunneman, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. Birdnest: Bayesian inference for ratings-fraud detection. *arXiv preprint arXiv:1511.06030*, 2015.

[23] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 895–904. ACM, 2016.

[24] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Catchsync: catching synchronized behavior in large directed graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 941–950. ACM, 2014.

[25] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Inferring lockstep behavior from connectivity pattern in large graphs. *Knowledge and Information Systems*, pages 1–30, 2015.

[26] Wen Jin, Anthony KH Tung, and Jiawei Han. Mining top-n local outliers in large databases. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 293–298. ACM, 2001.

[27] Wen Jin, Anthony KH Tung, Jiawei Han, and Wei Wang. Ranking outliers using symmetric neighborhood relationship. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 577–593. Springer, 2006.

[28] Nitin Jindal, Bing Liu, and Ee-Peng Lim. Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1549–1552. ACM, 2010.

[29] Da-Cheng Juan, Neil Shah, Mingyu Tang, Zhiliang Qian, Diana Marculescu, and Christos Faloutsos. M3a: Model, metamodel, and anomaly detection in web searches. *arXiv preprint arXiv:1606.05978*, 2016.

[30] George Karypis and Vipin Kumar. Metis–unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.

[31] Ashiqur R KhudaBukhsh, Jaime G Carbonell, and Peter J Jansen. Detecting non-adversarial collusion in crowdsourcing. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[32] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. Summarizing and understanding large graphs. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(3):183–202, 2015.

[33] Danai Koutra, Tai-You Ke, U Kang, Duen Horng Polo Chau, Hsing-Kuo Kenneth Pao, and Christos Faloutsos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 245–260. Springer, 2011.

[34] Danai Koutra, Evangelos E Papalexakis, and Christos Faloutsos. Tensorsplat: Spotting latent anomalies in time. In *Informatics (PCI), 2012 16th Panhellenic Conference on*, pages 144–149. IEEE, 2012.

[35] Danai Koutra, Neil Shah, Joshua Vogelstein, Brian Gallagher, and Christos Faloutsos. Deltacon for brain graph clustering and attribution.

[36] Danai Koutra, Neil Shah, Joshua T Vogelstein, Brian Gallagher, and Christos Faloutsos. Deltacon: Principled massive-graph similarity function with attribution. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(3):28, 2016.

[37] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. Deltacon: A principled massive-graph similarity function. In *Proceedings of the SIAM International Conference in Data Mining. Society for Industrial and Applied Mathematics*, pages 162–170. SIAM, 2013.

[38] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[39] Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. Learning to identify review spam. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 2488, 2011.

[40] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 939–948. ACM, 2010.

[41] Yongsub Lim, U Kang, and Christos Faloutsos. Slashburn: Graph compression and mining beyond caveman communities. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3077–3089, 2014.

[42] Yibin Lin, Agha Ali Raza, Jay-Yoon Lee, Danai Koutra, Roni Rosenfeld, and Christos Faloutsos. Influence propagation: Patterns, model and a case study. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 386–397. Springer, 2014.

[43] Yike Liu, Tara Safavi, Neil Shah, and Danai Koutra. Reducing million-node graphs to a few structural patterns: A unified approach. 2016.

[44] Yike Liu, Neil Shah, and Danai Koutra. An empirical comparison of the summarization power of graph clustering methods. *arXiv preprint arXiv:1511.06820*, 2015.

[45] Hing-Hao Mao, Chung-Jung Wu, Evangelos E Papalexakis, Christos Faloutsos, Kuo-Chen Lee, and Tien-Cheu Kao. Malspot: Multi2 malicious network behavior patterns analysis. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 1–14. Springer, 2014.

[46] Kameo Matusita. Decision rules, based on the distance, for problems of fit, two samples, and estimation. *The Annals of Mathematical Statistics*, pages 631–640, 1955.

[47] Joshua Neil, Curtis Hash, Alexander Brugh, Mike Fisk, and Curtis B Storlie. Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics*, 55(4):403–414, 2013.

[48] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[49] Caleb C Noble and Diane J Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2003.

[50] Myle Ott, Claire Cardie, and Jeffrey T Hancock. Negative deceptive opinion spam. In *HLT-NAACL*, pages 497–501, 2013.

[51] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210. ACM, 2007.

[52] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications*, 1(1):19–30, 2010.

[53] Mitchell Peabody. *Finding groups of graphs in databases*. PhD thesis, Citeseer, 2002.

[54] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, volume 1, 2000.

[55] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1346–1355. ACM, 2014.

[56] B Aditya Prakash, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *Advances in knowledge discovery and data mining*, pages 435–448. Springer, 2010.

[57] 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.

[58] Neil Shah. Flock: Combating astroturfing on livestreaming platforms. 2016.

[59] Neil Shah, Alex Beutel, Brian Gallagher, and Christos Faloutsos. Spotting suspicious link behavior with fbox: An adversarial perspective. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 959–964. IEEE, 2014.

[60] Neil Shah, Alex Beutel, Bryan Hooi, Leman Akoglu, Stephan Gunnemann, Disha Makhija, Mohit Kumar, and Christos Faloutsos. Edgecentric: Anomaly detection in edge-attributed networks. *arXiv preprint arXiv:1510.05544*, 2015.

[61] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1055–1064. ACM, 2015.

[62] Kumar Sricharan and Kamalika Das. Localizing anomalous changes in time-evolving graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 1347–1358. ACM, 2014.

[63] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and S Yu Philip. Graphscope. In *KDD-2007: 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[64] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383. ACM, 2006.

[65] Jimeng Sun, Yinglian Xie, Hui Zhang, and Christos Faloutsos. Less is more: Compact matrix decomposition for large sparse graphs. In *SDM*, pages 366–377. SIAM, 2007.

[66] Pedro Olmo S Vaz de Melo, Christos Faloutsos, Renato Assunção, and Antonio Loureiro. The self-feeding process: a unifying model for communication dynamics in the web. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1319–1330. ACM, 2013.

[67] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.

[68] Haifeng Yu, Phillip B Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 3–17. IEEE, 2008.

[69] Haifeng Yu, Michael Kaminsky, Phillip B Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. *ACM SIGCOMM Computer Communication Review*, 36(4):267–278, 2006.

[70] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2(1):718–729, 2009.