Universiteit Leiden
Opleiding Informatica

Anomaly detection in real-world networks

Name: Thomas Helling

Date: 23/11/2018

1st supervisor: F.W. Takes
2nd supervisor: J.C. Scholtes

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

LEIDEN UNIVERSITY

Abstract

The overwhelming amount of data that is nowadays available, leads to an increased demand for techniques that automatically identify abnormal or anomalous behavior. In this paper, we consider this problem in the context of networks and investigate how network-based anomaly detection techniques can be used for the purpose of law enforcement and criminal investigations in the field of electronic discovery.

This problem is firstly investigated on static networks. Examples of anomalies in static networks are network intruders in physical networks or spammers spreading unwanted advertisements in online social networks. While existing methods typically identify anomalies from a local perspective, we propose a novel method that identifies nodes from a global perspective. The proposed community-aware CADA algorithm outperforms previous methods on both synthetic and real-world data and scales remarkably well to larger networks.

Second, we approach the problem in networks that change over time. We compare two existing methods that indicate to what extent two graphs are similar to each other. Results on real-world data show that the techniques successfully indicate points in time when something significant happened in the network. In general, these network-driven techniques can uncover critical information about events in the data that was otherwise not discovered.

Contents

List of Figures

List of Tables

Abbreviations

E-Discovery	**E**lectronic **D**iscovery
LFR	**L**ancichinetti**F**ortunato**R**adicchi
ABC	**A**verage **B**etweenness **C**entrality
POI	**P**oints **O**f **I**nterest
POR	**P**oints **O**f **R**elevance
SCAN	**S**tructural **C**lustering **A**lgorithm for **N**etworks
COI	**C**ommunities of **I**nterest
NrMF	**N**on-negative residual **M**atrix **F**actorization
CADA	**C**ommunity-**A**ware **D**etection of **A**nomalies
EDPL	**E**gonet **D**ensity **P**ower **L**aw
EWPL	**E**gonet **W**eight **P**ower **L**aw
EPPL	**E**gonet $\lambda_{w,i}$ **P**ower **L**aw

Chapter 1

Introduction

An important problem in a large variety of fields is the identification of illegal behavior. Spammers, terrorist groups, drug cartels, and network intruders may all be considered as groups that should be acted upon. Due to the rapidly growing amount of data, detecting such behavior by hand becomes increasingly complex and time-consuming. Therefore, machine learning techniques can be adopted to enhance and accelerate the detection of anomalous behaviour in large data sets. The research field of identifying phenomena that diverge from what is considered to be normal is referred to as anomaly detection [3], further discussed in Section 1.1. In Section 1.2, we introduce the problem statement that guides this research. To answer the problem statement, we formulate two research questions in Section 1.3. In Section 1.4, we describe the structure of the rest of the thesis.

1.1 Anomaly detection in networks

The main goal in the field of anomaly detection is to automatically identify anomalies in multi-dimensional data points. However, this approach does not account for the interdependence between data objects. Network analysis can be utilized to represent and analyse complex relationships between objects and therefore provide a powerful approach to anomaly detection. The power of network science has already been shown in a variety of fields, such as pharmacology [4], epidemiology [5], and physics [6].

A network is described as a set of objects (or nodes) and a set of edges (or links), where each interaction represents a connection between a pair of objects. The formal definition of a network is dependent on the context of the application, as the definition of an object or interaction may vary. Note that, while some authors claim there may be a formal

difference between networks and graphs, we use both terms interchangeably throughout this paper.

Using a network-based approach to anomaly detection can be useful due to the fact that fraud is expected to occur in at least two different ways [7]: (1) by word-of-mouth, where it is likely that acquaintances of fraudulent nodes also commit fraud, and (2) by collaboration, where fraudulent nodes attempt to commit fraud as a group. In various domains the potential of network-based anomaly detection was already demonstrated, including spam detection [8] and network intrusion [9].

One of the first network-based anomaly detection algorithms observed the evolution of a telecommunication network to monitor so-called communities of interest (COI) [10]. The interactions between each phone are based upon the call quantity and duration between those calls. By observing the temporal evolution of the graphs on a daily basis, they (1) confirmed the word-of-mouth fraud theory by finding that fraudulent nodes are connected to each other with close proximity, and (2) that fraudulent objects can be detected by measuring the similarity between fraudulent objects and new communities of interest.

Many techniques do not label a node as anomalous, but provide an indication to what extent a certain node is anomalous by assigning an anomaly score to each node [2, 11]. In many cases, the boundary of what should be considered anomalous is not so straightforward; non-fraudulent objects can simply behave anomalously and anomalous objects can behave ordinarily. Therefore, most anomaly detection techniques require qualitative evaluation by a domain expert to conclude whether an object should truly be considered anomalous. It is therefore of utmost importance to include the domain expert in the process of anomaly detection [12].

One application area for network-based anomaly detection is the field of electronic discovery (e-discovery). E-discovery is defined as the discovery of identifying, collecting and producing electronically stored information in response to a request for production in a law suit or investigation [13]. The large volumes of data require techniques to rapidly detect underlying correlations and to identify what kind of network interactions are relevant for the investigations. E-mail is a commonly used medium that actors utilize to organize fraud. The tremendous amount of incoming and outgoing e-mails include a lot of irrelevant information, and automatically identifying the interesting nodes for an investigator may significantly enhance the process of investigation.

1.2 Problem statement

The challenging task in this study is to identify and evaluate network-based anomaly detection techniques that accelerate and enhance the anomaly detection process for a domain expert. Although there are a large number of anomaly detection techniques available, there is no one-size fits all to detect anomalies efficiently and effectively to support the domain expert in real-world data sets. In this research, we investigate how network-based anomaly detection can be beneficial for a domain expert in the field of law enforcement and investigations. Therefore, the problem statement of this research reads as follows.

Problem statement: *How can network-based anomaly detection algorithms support domain experts in detecting anomalous behavior in real-world networks?*

One could wonder when a domain expert is considered to be supported. We believe a domain expert is supported when at least one of the three conditions is met: (1) the domain expert understands how to evaluate and compare several results from anomaly detection techniques, (2) the domain expert is provided with a list of objects that demonstrate divergent behavior as a starting point of the investigation, and (3) the domain expert knows why a certain event in time has occurred. Note that there are more methods to support the detection of anomalous behavior, but these occur in a semi-supervised or supervised setting. This research focuses on anomaly detection in an unsupervised setting.

1.3 Two research questions

In order to answer the problem statement, two research questions are formulated. There are various approaches to detect anomalous behaviour in real-world networks. The first distinction is made between anomaly detection in static or dynamic networks. Anomaly detection in static networks attempts to identify anomalies in a network that does not change over time. Therefore, the first research question reads as follows.

Research question 1: *To what extent can network-based anomaly detection techniques be utilized to identify anomalous behaviour in static real-world networks?*

Contrary to static networks, anomaly detection in dynamic networks attempt to identify anomalous behaviour in networks that change over time. Due to the different nature of the analysis and the underlying data, they both require different approaches and detect different kinds of data points that deviate from what is considered to be normal. Real-world networks operate in a constantly changing environment, where edges and

nodes can be modified, appear, and disappear. The addition of the time component in network-based anomaly detection could be of critical importance to effectively detect unusual behaviour in a dynamic real-world context. Hence, the second research question reads as follows.

Research question 2: *To what extent can the addition of the dynamic component in network-based anomaly detection affect the performance of anomaly detection?*

Answering these two research questions allows us to formulate an answer to the problem statement.

1.4 Structure of the thesis

This rest of this paper is structured as follows. In Chapter 2, we introduce network science and anomaly detection separately. Subsequently, we bring both fields together in Chapter 3. In Chapter 4, we introduce anomaly detection methods that are further investigated in this paper. In Chapter 5, the experimental setup is described to evaluate on anomaly detection techniques. In Chapter 6, the results of the performed experiments are described. We follow up with a discussion about the results in Chapter 7. Conclusively, we answer the problem statement and research questions in Chapter 8.

Chapter 2

Background

Before we are able to understand how anomaly detection techniques can support the domain expert in decision making, we should understand the relationship between fraud and anomaly detection. Therefore, we discuss the fraud triangle in Section 2.1. Then, we discuss two fields separately: network science and anomaly detection. In Section 2.2 we describe the properties of real-world networks. A well-studied and relevant problem in the field of network science is community detection, further discussed in Section 2.3. As there exist no publicly available labeled data set for anomaly detection, we discuss synthetic graph generation in Section 2.4. Lastly, we provide relevant information about anomaly detection in Section 2.5.

2.1 Investigations as a domain expert

Criminologist David Cressey introduced the fraud triangle that is used to describe the reasons behind committing fraud [14]. The first of the three factors, motivation, refers to the phenomenon where an individual copes with serious problems (e.g., financial problems) that motivates him/her to commit the crime. The second factor, opportunity, describes to what extent a certain possibility arises to solve its financial problem with a low perceived risk of getting caught. The last factor, rationalization, refers to the fact that most persons that commit fraud do not perceive themselves as criminals, and the individual should be able to justify himself in a way that it makes it an acceptable act.

Understanding the fraud triangle can enhance the prevention and detection of fraudulent behavior, as organizations can act upon individuals that show characteristics of each of the three factors. Although the fraud triangle is able to explain occupational frauds, it has been criticized since it is ineffective in explaining group fraud [15]. The

SEC Accounting and Auditing Enforcement Releases reported that 89% of fraudulent financial reporting included involvement of the CEO, CFO, or both [16]. As the board of directors is not directly involved in managing the account records, it implies collaboration between multiple actors to organize fraud. Utilizing techniques from the field of network science may therefore provide useful information about the individuals that committed unethical behavior collaboratively.

The rapidly growing amount of data of today raises the issue that is known as 'drowning in data', where the overwhelming chunks of data that need to be analyzed for an investigation becomes increasingly complex and time-consuming. Automated techniques are necessary to structure, summarize and evaluate the data quickly, so that individuals that demonstrate fraudulent behavior can be detected and further investigated. A common approach to investigate cases is by organizing the data to answer the golden W's: who, where, what, when, and why. These are commonly combined with two extra questions, how and how much. Answering these questions can shed light upon a multitude of aspects about the reasons behind the fraud.

The investigator iteratively attempts to answer each of these questions with so-called hypothesis testing, where one attempts to confirm a hypothesis by searching for evidence in the large volumes of data. The entire process can already be largely automated to enhance the process of the investigator, so that the investigator can solely focus on reviewing and testing the hypothesis. The process and components of the digital forensic process are shown in Figure 2.1, and are based on [17] and an interview with a domain expert in the field of law enforcement and investigations.

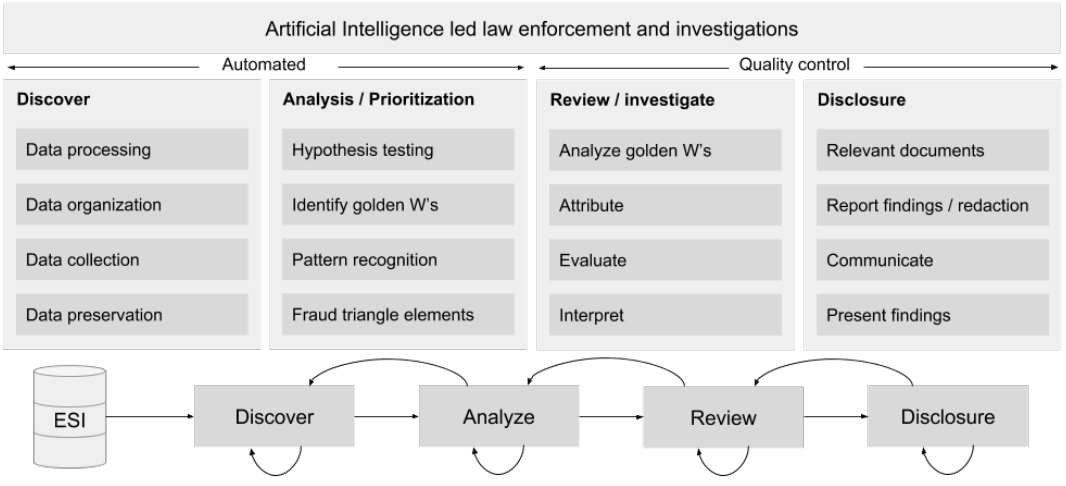


FIGURE 2.1: The components and aspects in each phase of the digital forensic process for investigators.

2.2 Real-world networks

Before we are able to describe what could be considered an anomaly in a network, we should understand what is considered normal behavior in networks and what a network is comprised of.

A network $G = (V, E)$ consists of a set of vertices V and a set of edges E. There are many different ways to describe vertices and edges. Throughout this paper, We use the terms nodes, objects, and vertices interchangeably. The same holds true for edges, links, and connections. The total number of nodes and edges are referred to as n and m respectively. The degree of a node k_v is the number of adjacent lines for node v. One can also make a distinction between the number of incoming (indegree) or outgoing (outdegree) links of a node. Initially it was assumed that the node degree distribution of a real-world network follows a Poisson distribution. However, the opposite seemed true [18]. As an example, consider a large tulip exporter with five departments. It is irrational to assume that each employee has an equal probability to interact with random people from all departments, as most employees will interact with many people from their own department. This is one of the many network properties that correctly depict a *real-world* network. We mention the most significant of them below.

(1) The scale-free property [18]. Instead of following a Poisson degree distribution, a real-world network follows a asymptotic power law degree distribution in the form of $P(k) \sim k^{-\lambda}$, where $P(k)$ describes the fraction of nodes that have k connections in the network. λ is the power law exponent that typically falls in the range of $2 < \lambda < 3$. This means that there are many nodes that link to few other people and a few nodes that connect to many other nodes. The latter type of nodes are often referred to as hubs, as they connect to many people that are not connected to each other. In the example of the tulip exporter, one could expect that managers are the ones communicating interdepartmental, and therefore function as hubs.

(2) The small world phenomenon [19]. The distance between a pair of nodes is described as the shortest path between two nodes. The average distance between each pair of nodes appears to be relatively small, a phenomenon also known as the six degrees of separation [20]. It describes that there is an average of six edges between any pair of nodes. For example, you might know someone that knows the prime minister of your country, that in turn shook hands with the president of the United States. Hence, there are only three handshakes between you and the American president. Hubs play an important role in the six degrees of separation since they significantly decrease the number of steps between any pair of nodes.

(3) Low density. Networks are sparse, meaning that a node is mostly connected to a small part of the network and not to the entire network. Density is measured as the number of edges divided by the maximum number of edges in the network. For a directed network, the maximum number of edges is $n(n-1)$ and for an undirected network the maximum number of edges is $n(n-1)/2$. We compute the network density by dividing the number of edges in the network by the maximum number of edges.

(4) High clustering coefficient [20]. The clustering coefficient of a node describes to what extent direct neighbors of a node connect to each other. The clustering coefficient of a node can be computed by dividing the number of edges between neighbors of a node by the maximum number of such edges. The direct neighborhood of a node and links between those nodes is often described as the egonet of a node. The average clustering coefficient of a network is the average clustering coefficient over all nodes. A real network exhibits a high average clustering coefficient.

These four properties collaboratively describe small-world networks. Many networks follow these network properties, such as protein interaction networks, science collaboration networks, and actor networks.

2.3 Community detection

A well-studied task in networks is to identify groups of persons that tend to cluster together, also known as community detection [21]. A large variety of community detection algorithms exist in literature, and we will discuss two of them more in more detail: Louvain [22] and Infomap [23, 24].

2.3.1 Louvain Modularity

The Louvain Modularity algorithm attemps to optimize the so-called Modularity measure. Modularity is a measure of quality for the division in a network that was introduced by Girvan and Newman [25]. It measures the fraction of edges m that connect vertices in the same community in relation to the expected value of the the network if the m edges are randomly distributed while maintaining the same vertex degree distance. Mathematically, the modularity function Q is denoted as follows.

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \tau \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{2.1}$$

Here, c_i is the community to which vertex i is assigned. Moreover, A_{ij} is a binary number that states 1 if there exists an edge between vertices i and j. $\delta(c_i, c_j)$ is 1 if vertices i and j fall in the same community and 0 otherwise. $k_i k_j / 2m$ denotes the probability that there is an edge between i and j if the graph is randomly generated. τ is the so-called resolution parameter that denotes the size of the communities to be found: a smaller τ provides more communities. k_i is the degree of node i. For a weighted network, A_{ij} can take any non-negative value.

As each node should be assigned to a cluster to measure the Modularity Q, and identifying the optimal community structure is proven to be a NP-hard problem [26], we use the greedy Louvain method described in [22]. In short, the Louvain algorithm is executed as follows. First, all nodes are assigned to their own community. The algorithm will then iteratively walk through the nodes, and replace each node to the community that maximizes Modularity. Once we passed all nodes, we aggregate the nodes and edges of each community into one node and in turn iterate through the aggregated nodes to find the highest increment in modularity. The algorithm terminates once there is no improvement in Modularity found.

2.3.2 Infomap

To understand Infomap, one first needs to understand the concept of random walkers. A random walker is somebody or something that is randomly traversing through the edges of the graph. By analyzing the path of a random walker for a longer period of time, one expects to find a pattern, where a random walker traverses through a community for a longer period of time. Infomap attempts to describe the list of nodes that are visited by a random walker in a minimal way, i.e., it tries to find the minimum description length of the path of a random walker.

Compressing the description length is often done with Huffman coding, where terms that occur often get a short code word, while terms that occur rarely get a long code word. Finding the theoretical compression limit can be done using Shannon's source coding theorem, that describes that the average length of a code word can not be less than the entropy of a random variable R, if l code words are used to describe the l states of R that occur with frequencies p_i. The entropy of R can be described as $H(X) = -\sum_1^n p_i log(p_i)$. The average number of bits to describe a step from a random walker can then be computed in terms of the frequency distribution P to visit nodes in the network with a lower bound of H(P).

Infomap uses a two-level description to compress information in the graph. Firstly, each cluster is provided with a unique name following Huffman coding, where nodes in the

same cluster are provided with names from a different Huffman coding. An extra 'exit' code word is provided to capture whether the random walker is leaving a cluster, followed by the code word of the new cluster. By using the two-level description approach, one can significantly reduce the description length of the random walker in the network. Given the set of communities C, the average description length of a single step by the random walker is computed with the so-called map equation L as follows.

$$L(C) = qH(O) + \sum_{i=1}^{C} p_i H(I^i) \tag{2.2}$$

Here, q describes the probability to leave a cluster, and $H(O)$ the entropy of the module names. Furthermore, $H(I^i)$ is the entropy of within cluster movements, including the exit code word for cluster i. Finally, p_i is formally described as the fraction of within-cluster movements for cluster i, and the probability of leaving cluster i so that $\sum_{i=1}^{C} = 1 + q$. Now, one can search for an optimal community structure by minimizing the map equation L.

For the minimization, the authors follow the Louvain method by replacing nodes to the clusters that minimize the map equation. Once no more increase in L can be found, the same aggregation procedure is also followed. Furthermore, two extra steps are performed to ensure that the algorithm does not get stuck on local minima: (1) submodule movements, where each cluster is treated as its own network and the algorithm is applied on that network. If any submodules were found in the submodule, the algorithm is reapplied on the entire network where the submodules can move freely between other modules. and (2) single-node movements, where each node is assigned to be part of its own module, and then the algorithm is ran subsequently to see whether the node may belong to a different cluster.

2.4 Graph generation

As real-world networks often consist of noise that obscure the evaluation of performance measures, and the availability of labeled data sets in the field of anomaly detection is scarce, a common approach to measure the performance of network-based anomaly detection algorithms is to generate synthetic graphs that obey the properties of real-world networks. Synthetic graph generation is a field of large interest, and the number of graph generators has increased significantly over the past decades [18, 27]. The identification of a suitable graph generator is therefore a difficult problem. For reasons

we show later, communities in graphs are closely related to anomaly detection in a static context.

Therefore, the LancichinettiFortunatoRadicchi (LFR)-benchmark, a graph generator that generates graphs with the unique property that it obeys the network properties of heterogeneity for the community sizes and node degrees, appears to be most satisfactory for static anomaly detection [27]. Moreover, the range of parameters to tune makes the benchmark extremely suitable for the generation of unweighted, weighted, undirected, and directed networks. To elaborate on the parameters, it is best to describe them in the context of how the graphs are constructed. The graphs are constructed as follows.

1. A network of n nodes is generated where each node is given a degree extracted from the power law minus exponent λ_1 with $k_{\min}$ and $k_{\max}$ so that the average degree is d.

2. Community sizes are generated from the power law minus exponent λ_2 with community sizes $s_{\min}$ and $s_{\max}$ so that $s_{\min} > k_{\min}$ and $s_{\max} > k_{\max}$, and the sum of community sizes is n. If these are not chosen, the community sizes will be chosen close to the degree extremes.

3. Each node is assigned to a community, as long as the community does not exceed the set community size. If the number of edges within the community exceeds the community size, the node becomes homeless. Homeless nodes are randomly assigned to a community. If the community size is exceeded then a randomly selected node of that community becomes homeless. The process terminates if all communities are completed.

4. The algorithm rewires the nodes so that each node has a fraction of approximately ρ internal neighbors and $1 - \rho$ external neighbors, whilst the nodes retain their degree.

2.5 Anomaly detection

As described earlier, anomaly detection refers to the problem of identifying data points that diverge from what is considered to be normal [3]. It can result in insights into how to prevent and act upon types of malicious behavior, such as a breach into a system or a hacked credit card. Figure 2.2 provides a two-dimensional visual representation of a data set that includes two types of anomalies; (1) a white crow, a data point that clearly does not belong to any of the two clusters, and (2) an in-disguise anomaly, a data point that attempts to behave like a normal data point but actually attempts to breach into the cluster.

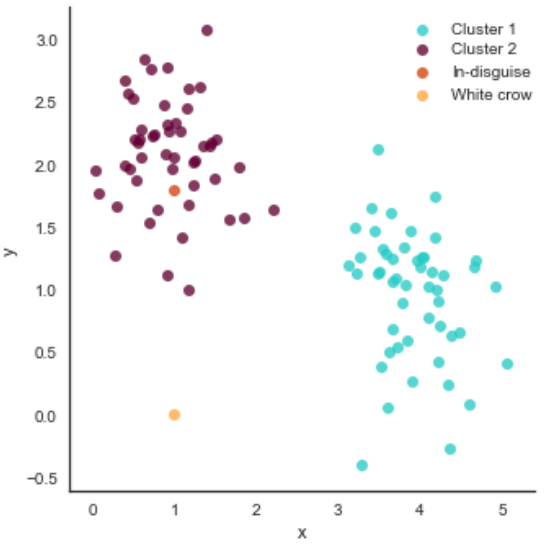


FIGURE 2.2: A visual representation of a data set where an anomaly occurs.

2.5.1 Types of anomaly detection

Research in anomaly detection dates back to the 19th century [28]. Since then, many techniques have been developed in the field of anomaly detection. The majority of these techniques can be divided into six categories [3]: (1) classification based, (2) clustering based, (3) nearest neighbor based, (4) statistical, (5) information theoretic, and (6) spectral. The techniques in each category are similar to the data mining categories, only with another purpose. As an example, a clustering-based technique attempts to detect anomalies by clustering the data set and identifying the nodes that do not belong to any of the clusters, as is clearly the identification of a white crow anomaly illustrated in Figure 2.2.

Furthermore, there are three different approaches to detect anomalous behavior: (1) unsupervised anomaly detection, that identifies anomalies in unlabeled data, (2) supervised anomaly detection, where the data is already labeled anomalous or not, and (3) semi-supervised anomaly detection, where a model is constructed on a labeled training data set, and one measures the likelihood that other instances follow the same anomalous pattern. In this research, we identify anomalous nodes in an unsupervised fashion because there are no publicly available labeled data sets.

2.5.2 Challenges in anomaly detection

One of the main challenges in anomaly detection is to determine whether a node should truly be considered anomalous or not. As an example, consider a student that attempts to conduct a survey on a social network channel on the one hand. The student may send a lot of friend requests to gain more exposure to complete the survey. On the other hand, we consider a spammer, that also attempts to send malicious messages to as many people as possible. The spammer also sends a lot of friend requests to maximize its reach. The behavior of the student and spammer can be extremely similar, while the spammer should be considered fraudulent and the student should not.

Therefore, the exact definition of each anomaly differs for each domain. Anomalies are usually the result of malicious actions, and it is of critical importance to correctly define the anomaly one is looking for. Still, the persons that commit illegal actions may be aware of the methods to uncover illegal activities and attempt to hide by acting as normal persons would (illustrated as the in-disguise anomaly in Figure 2.2). Therefore, domain experts should be involved throughout the process to determine whether the flagged node should be acted upon.

2.5.3 Anomaly detection examples

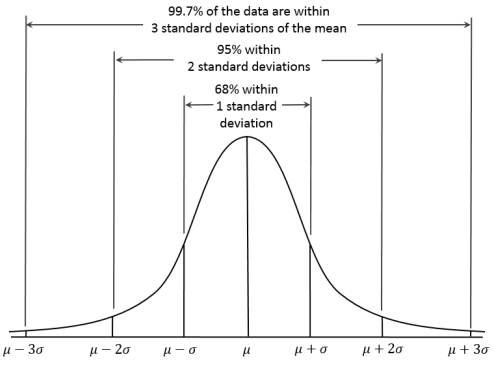


FIGURE 2.3: A normal distribution (obtained from [1]).

A simple but effective statistical approach to anomaly detection relies on the assumption that the data follows a normal distribution. Illustrated in Figure 2.3, a normal distribution has the property that about 68% of the data points falls within one standard deviations σ of the mean μ and 95% of the data falls within 2σ of μ. If the data is normally distributed, anomalous data points are those points with the furthest distance towards μ. A threshold can be set to flag all data points that are not covered within 3σ from the μ as anomalous. Often, the median $\tilde{x}$ is used instead of the mean as the mean can become skewed due to outliers. Although this is a very intuitive approach, the assumptions that the data follows a normal distribution and can be reduced to a one-dimensional value does not always hold true.

Another relatively simple approach to anomaly detection is by utilizing the k-means clustering algorithm [29]. k-means clustering focuses on assigning each observation, that

consists of a vector of features, to one of the k clusters. It starts by initializing k clusters and assigning or estimating the centroid of the k clusters. The algorithm iteratively runs two steps: (1) the clustering step, where each observation is assigned to the cluster of the closest centroid, and (2) the centroid update step, where the centroid is repositioned in the vector space by averaging each feature that belongs to that centroid. The distance between each feature vector can be computed with several distance measures, yet the Euclidean distance is commonly used and is described as follows.

$$D_{Euc}(p,q) = \sqrt{\sum_{i=1}^{N}(p_i - q_i)^2} \tag{2.3}$$

Here, p and q are two vectors of size N, where p_i and q_i denote the ith value of the vector space. When searching for anomalies with the k-means algorithm, the challenge is to find those data points in the cluster that are furthest away from the centroids of the clusters. In other words, we select the nodes that have a large intra-cluster distance given a certain threshold. Although k-means offers no accuracy guarantees, it offers simplicity and performance. In case of Figure 2.2, the white-crow anomaly can be detected by initializing the algorithm with $k = 2$ to find the purple and blue clusters. The white crow anomaly will either be assigned to the purple or blue cluster, but the distance to the centroid of both clusters remains high, which makes the k-means algorithm a suitable approach for detecting white-crow anomalies.

Chapter 3

Related work

Contrary to anomaly detection in regular data sets, using a network-based approach to capture the relationship between objects can provide different insights into the network and its anomalies. As mentioned before, in many domains it is expected that fraud spreads by word-of-mouth, and that organized fraud occurs in closely related groups [7]. Therefore, we bring two fields together in this Chapter: anomaly detection and network science. We discuss the network anomaly types in Section 3.1. In Section 3.2, we provide a variety of applications of network-based anomaly detection. In Section 3.3, we address five different methods on how to evaluate on anomaly detection algorithms. The contributions of this research are described in Section 3.4.

3.1 Network anomaly types

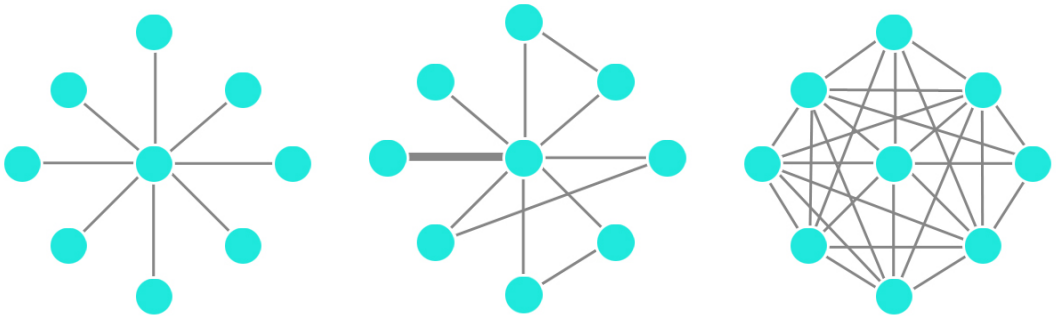
According to [7, 30, 31], There are four types of anomalies that can be detected in networks: (1) anomalous nodes, (2) anomalous edges, (3) anomalous sub-graphs, and (4) events. We discuss each one of them below shortly.

Node anomaly is an object that behaves considerably different compared to other objects in the network. One of the most prominent node anomalies in the field of static network-based anomaly detection, is the node that connects to random nodes in the network [2, 11, 32–34]. There are two reasons why node anomalies can be considered anomalous: (1) these nodes are considered hubs and therefore play an important role in the network by connecting two or more distinct communities (e.g. opinion leaders), and (2) these nodes are not aware of the global network structure and therefore connect to many random nodes (e.g., spammers that send e-mails to as many e-mail addresses as they could find on the web).

Edge anomaly describes interactions between two nodes that seem irregular. An edge anomaly can occur in a variety of ways. An typical example of an edge anomaly is that there exist an extremely strong tie between two nodes, while both nodes do not have strong ties to any other nodes in the network [2, 33]. Other than strong ties, interactions that connect two groups of objects in the network that were otherwise not connected could be considered anomalous [35].

Sub-graph anomaly can be described as a group of nodes that collaboratively show anomalous or fraudulent behaviour. In large groups, it is unlikely that everybody knows each other and groups that are very strong connected to each other may indicate collusion. For example, in a review network where users are connected if they both reviewed a specific product, groups of fake reviewers can be identified as they constantly review each other or the same products [2, 36].

Event anomaly is an anomaly where a significant change in the network happened between time step t and previous time steps. An event anomaly can only be detected in a dynamic network, where several network features are monitored over consecutive graphs. A graph in the stream of consecutive graphs can be flagged anomalous if the graph significantly differs from the rest of the graphs [37, 38]. Analyzing a network over time may provide new insights into the network, but also requires techniques with low computational complexity to analyze the network in a rapid fashion.



(a) Node anomaly (b) Edge anomaly (c) Sub-graph anomaly

FIGURE 3.1: Examples of anomaly types in static networks as defined in [2]. Note that the Event anomaly is not illustrated, because that only occurs in a dynamic context.

3.2 Anomaly detection in networks

The anomalies described in the previous section can be detected with various techniques. We divided the techniques in five categories: (1) local-structure based, (2)

global-structure based, (3) clustering based, (4) factorization based and (5) event detection.

3.2.1 Local-structure based

The first type of network-based anomaly detection refers to anomaly detection by only measuring local structural properties of each node. As an example, the Oddball algorithm was introduced for weighted graphs that (1) extracts a number of features, such as the number of nodes and edges, from the ego-network of each node. (2) identifies a pattern of normal behaviour by fitting so-called power laws on these features, and (3) detects points that deviate from these power laws. The algorithm is able to detect stars (node anomaly) or near-cliques (sub-graph anomaly), and dominant links (edge anomaly) in a weighted graph [2]. Each of the detected anomalies was also described in the context of a real-world network. For example, a star may accurately represent spammers in networks, because spammers are typically not aware of the local clustering structure and therefore send a message to people that are not connected to each other.

3.2.2 Global-structure based

Many studies in the field of network science focus on measuring the centrality of each node, i.e., the importance of a node in the network. The results can provide insights into the key players in the network and may therefore be of critical importance to identify objects that influence other nodes in the network. Using centrality-based methods, one can uncover the most important nodes in a network (those nodes with a high centrality). While centrality measures do not truly belong to the field of network-based anomaly detection, these techniques can be beneficial for the domain expert to get insight into the network. Therefore, we mention the five well-known ones below.

The degree centrality is described as the number of nodes that a node is connected to in the network. It therefore indicates the popularity of a node, being in contact with a lot of other nodes in the network. One could also make a distinction between indegree centrality (the popularity of an object), and outdegree centrality (objects that want to connect to many other objects).

The betweenness centrality is the fraction of shortest paths in the network that pass a specific node. Therefore, it indicates which nodes influence the flow throughout the network. Hence, these are nodes that act as a 'bridge' in the network, and therefore play an important role in connecting communities. Another approach is to compute the betweenness centrality on a local part of the graph, so one can identify local hubs

in the network. For example, Hassanzadeh et al. extended the Oddball algorithm by extracting the average betweenness centrality of each ego-network and comparing that against the number of edges, detecting stars or near-stars and cliques or near-cliques more accurately than the Oddball algorithm [32].

An alternative centrality measure is closeness centrality, one that indicates how close the node is to all other nodes in the network. It is computed as the fraction of shortest paths between a node and all other nodes. Nodes with a high closeness centrality are nodes that should be reached out to if one wants to rapidly spread information throughout the entire network. The closeness centrality can also be useful to determine the person that is most central in a certain sub-graph, to identify the persons that are closely connected to everyone else in that group.

Lastly, an interesting centrality measure for the purpose of investigations is the PageRank centrality [39]. Created by Google founder Larry Page, the PageRank centrality is a measure that is based on random walks. Intuitively, the PageRank algorithm generates a probability distribution that represents the likelihood that a specific person will arrive at any particular object in the network by randomly navigating through the network. In other words, a node is considered important if there exists a link from another important node or if the node has many incoming links. PageRank centrality is useful as it does not only take into account one-hop away neighbors, but also two, three, or k-hop neighbors with decreasing weight. To compute the PageRank centrality, consider a set of nodes V, where each node $v \in V$ is initialized with a ranking factor $PR(v_i)$ of $\frac{1}{n}$. Then, in each iteration the value $PR(v)$ is updated and distributed over all nodes v is connected to. It is repeatedly computed as follows.

$$PR(v_i) = \frac{1-a}{n} + a \sum_{p_j \in \text{indeg}(v_i)} \frac{PR(v_j)}{\text{outdeg}(v_j)} \tag{3.1}$$

Where a is a damping factor that describes if an individual stops navigating through the network and stays on the node. Note that in each iteration, each node gets a value of at least $\frac{1-a}{n}$ that represents the probability that a random surfer starts over on a new node. a is commonly set to 0.85. As we can infinitely update the PageRank values, the algorithm is set to converge once there is no significant difference in the updated PageRank value. Note that there are many variants of the PageRank centrality. For example, the Eigenvector centrality is equal to the PageRank centrality if the graph is undirected and unweighted.

3.2.3 Clustering based

Another approach to identify anomalous behaviour in static real-world networks is to cluster the nodes in the network. Nodes or sub-graphs can then be considered anomalous if (1) the structure of a cluster, such as the density, is different compared to the other clusters in the network, or (2) if the node does not belong to a cluster or is a link between multiple clusters. Note that it is sometimes claimed that normal community detection methods are not suitable for the purpose of anomaly detection as the results of community detection algorithms are affected by the presence of anomalies [11].

Other techniques that solely cluster the nodes from a local perspective can be used to identify clusters and anomalies. As an example, the Structural Clustering Algorithm for Networks (SCAN) is a method that clusters nodes together if a pair of nodes share at least a common set of neighbors. Nodes that do not belong to any cluster and connect to multiple clusters can then be considered hubs in the network [40].

In the field of dynamic network-based anomaly detection, a wide variety of methods utilize clustering techniques. For example, Tang et al. identify persons of significance in a community to model the evolution of significant communities over time [41], while Wang et al., utilize a clustering-based approach to identify whether a sudden change in the network structure is caused internally in a community or globally by a group of communities [42].

3.2.4 Factorization based

Factorization based methods construct an adjacency matrix or matrix with features of the graph for each time step, to further decompose the matrix to capture the underlying structure of the data. As an example, Tong et al. introduced the novel technique Non-negative residual Matrix Factorization (NrMF) for bipartite graphs. Bipartite graphs are graphs where the vertices are decomposed in two disjoint sets (V_1, V_2), where there only exist relationships between the one and the other set. An example of such a network is a user to movie rating network.

NrMF captures underlying correlations of the graph by decomposing the bipartite adjacency matrix with the unique property of imposing non-negativity constraints on the residual matrix R [33]. Mathematically, the low-rank approximation of the adjacency matrix is presented as $A = \tilde{A} + R = FS + R$, where $F = n_1 \times g$ and $S = g \times n_2$ reveal the community structure of the graph, while the residual matrix R is an indicator of which nodes do not obey the underlying patterns in the data and can therefore be considered anomalous. By imposing the non-negativity constraint on the residual matrix, one can

use this approach to detect four anomalous patterns: (1) strange connection between two remotely connected communities, (2) port scanning, a node from V_1 that connects to many other nodes in V_2, (3) ddos, a node from V_2 that connects to many other nodes in V_1 and (4) bipartite core, where a group in V_1 and a group in V_2 are strongly connected to each other.

3.2.5 Event detection

In the field of dynamic network-based anomaly detection, events can be detected by investigating the difference in graph structure between time step t and $t+1$. A common approach is to extract features of the graph at each time step and compute the similarity of the network with respect to previous time steps. One then sets a threshold on the similarity to flag a certain network as anomalous.

As an example, the algorithm NetSimile firstly extracts f features of each network, such as the number of neighbors of each node, the clustering coefficient, and the number of incoming and outgoing edges in the egonet, to generate a $n \times f$ matrix [38]. The authors found that the $n \times f$ matrix can be reduced by aggregating the n vectors to one vector by solely capturing statistical measures of the vectors such as the median, mean, standard deviation, skewness and kurtosis[1] of each feature for the graph. A variety of distance measures can then be used to measure the similarity between the current and past networks.

Koutra et al. found that one could also measure the change in flow of the network by computing a similarity score between two graphs with the same number of nodes by measuring the difference of affinity of each node i to j in the graphs [37]. The node affinity is computed using Fast Belief Propagation [43], a belief propagation algorithm that is guaranteed to converge. It then measures the distance between the node affinities of the current and earlier networks to provide an indication of the similarity of the graph.

A shortcoming of these methods is that they are biased towards flagging events when the network becomes sparser or denser than the previous time step. To solve this issue, La Fond et al. introduced a variety of statistics that are normalized over the size of the network, and is therefore able to flag size independent events [44].

[1] The kurtosis is a measure that describes whether the distribution is light- or heavy-tailed.

3.3 Evaluation of anomaly detection algorithms

As we described earlier, one of the main challenges in anomaly detection is what should be considered normal or anomalous. Another problem of network-based anomaly detection is that there is no available labeled data set to evaluate anomaly detection techniques. According to [7], there are five evaluation methods that are commonly used in the field of anomaly detection and we describe each one of them below shortly.

- *Internal evaluation* - This type of evaluation utilizes the statistical distribution of the anomaly scores to quantify to what extent a node does not belong to the distribution. This is an internal evaluation because it solely clarifies the outlier score of the model that is used and cannot be used to compare results with other models.
- *Qualitative evaluation* - Qualitative evaluation is described as finding explanations for why the anomaly was detected by performing an in-depth analysis on a real-world data set. This approach may require incorporation of domain knowledge to determine whether the anomalies should truly be considered anomalous.
- *Synthetic graph injection* - In many fields of network science, such as community detection, there is the need to be in control of the network structure to properly evaluate how the technique performs on different network structures. One is able to quantitatively evaluate the anomaly detection technique by generating a network and evaluate on the manually imputed anomalies that the model detected.
- *Anomaly injection* - Contrary to the previous approach, anomaly injection focuses on imputing anomalies in real-world data sets, instead of synthetic data, to quantitatively evaluate on the detected anomalies. A limitation of this method is that the real-world data set may contain anomalies by itself and therefore obscure the performance of uncovering anomalies.
- *Validation by external source* - The last evaluation method triangulates multiple data sources to evaluate whether an identified anomaly truly demonstrates anomalous behavior. As an example, one can identify a fraudulent node by considering only the graph structure, and then use meta information about the node to observe whether the node should be considered anomalous.

3.4 Contributions

As the field of network-based anomaly detection has had a lot of attention in the last decade, a variety of complementary surveys have been conducted that describe the difference between the anomaly detection techniques on a high level [7, 30, 31]. A group of algorithms, that are further discussed in the next Chapter, identify similar anomalies.

The contribution of this thesis is threefold. Firstly, we identify and utilize methods that uncover similar anomalies in a static network, and report on the performance of each of these methods on synthetic and real-world data sets. Secondly, we introduce a new method, named CADA, and demonstrate that the community-aware approach performs better than previous approaches in uncovering the node anomaly. CADA is also described in a dedicated paper [45]. and (3) we show how dynamic network-based anomaly detection methods can be used to identify events in consecutive graphs over time. Furthermore, we show to what extent these anomaly detection algorithms can assist law enforcement and investigations in rapidly organizing the data to answer the two of the golden W's: who and when by identifying suspicious nodes or suspicious moments of interests that should be further investigated.

Chapter 4

Methods

As we have provided information about anomaly detection techniques for networks in general, this Chapter provides an extensive description of the methods that are further used in the paper. These techniques are chosen as they are prominent in the field of network-based anomaly detection [7, 30]. In Section 4.1 we address how to detect anomalies in networks at one point of time, also known as static network-based anomaly detection. In Section 4.2 we describe how to detect events in networks that evolve over time.

4.1 Identifying persons of interest in a static network

In this section we describe how network-based anomaly detection algorithms can support domain experts in identifying anomalous behavior in a static context. It encompasses an extensive description of four static anomaly detection algorithms that are investigated. It begins with Oddball, an algorithm that identifies four types of anomalies and is also used as a baseline in terms of identifying anomalies in static networks.

One of the identified anomalies, is the *node anomaly*, the node that connects to many nodes that are not connected to each other. There are many other types of node anomalies, but from now on we refer to this anomaly as the node anomaly. As a follow-up on Oddball, we describe two other methods that can be used to detect the node anomaly. Then, we propose our new community-aware approach CADA to identify the node anomaly. In some cases, we use the well-known Zachary's Karate Club Network to illustrate the technique with an example. The network is illustrated in Figure 4.2.

4.1.1 Oddball

ODDBALL analyses the ego-network for each node in the network [2]. The ego-network is the direct neighborhood of a node, and the connections between those nodes. The authors extracted multiple features of each ego-network and identified four features that follow a power law and accurately describe normal behavior in networks. The extracted features are (1) the number of neighbors of ego-network i, (2) the number of edges in ego-network i, (3) the total weight of ego-network i, and (4) the principal eigenvalue of the weighted adjacency matrix of ego-network i. By illustrating the features in two-dimensional space, they found that the combination of two features sometimes followed so-called power laws. We mention the three observed power laws below.

1. Egonet Density Power Law (EDPL): the number of nodes n_i and the number of edges m_i of the ego-network i follow a power law ($m_i \propto n_i^{\alpha_d}$).
2. Egonet Weight Power Law (EWPL): the total weight w_i and the number of edges m_i of the ego-network i follow a power law (($w_i \propto m_i^{\alpha_w}$).
3. Egonet $\lambda_{w,i}$ Power Law (EPPL): the principal eigenvalue of the weighted adjacency matrix $\lambda_{w,i}$ and the total weight w_i of ego-network i follow a power law ($\lambda_{w,i} \propto w_i^{\alpha_p}$).

These power laws exhibit normal patterns of the data set. The authors defined four anomaly types that can be identified if an ego-network does not obey one of the power laws, and we define how these may be useful for an investigator:

1. *Star or Near-star:* Nodes that connect to many nodes that are not connected to each other (i.e., a low clustering coefficient, extracted from EDPL). Typical examples of stars are network intruders in physical networks or spammers spreading unwanted advertisements in online social networks. Other examples of star anomalies can be key players in the network as they connect to an extremely large amount of groups.
2. *Clique or Near-Clique:* ego-networks that have an extremely high density (i.e., a high clustering coefficient, extracted from EDPL). It is expected that fraud spreads by word-of-mouth, so it may be useful to find those nodes that are very tightly connected to each other.
3. *Heavy Vicinity:* ego-networks that have an extreme weight compared to the number of edges (extracted from EWPL). This is a follow up on (2), because ego-networks with a high activity may even empower the assumption made in (2).

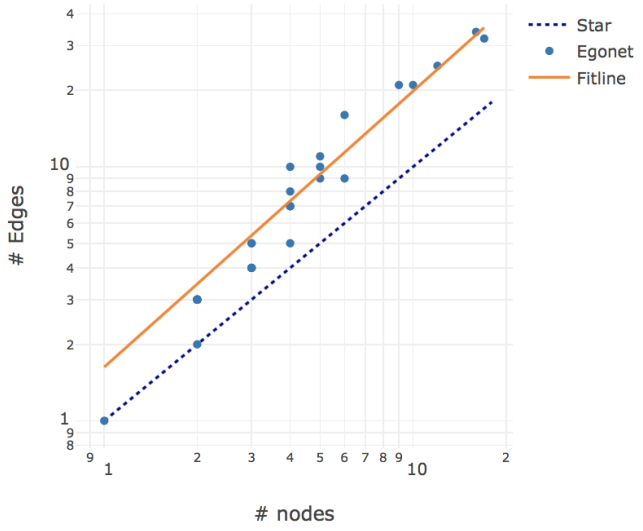


FIGURE 4.1: Egonet Density Power Law on Zachary's Karate Club network. The blue line shows at what point a star ego-network occurs.

The heavy vicinity indicates that there was a high activity of e-mails in the ego-network, and may therefore yield similar results as (4) for denser graphs.

4. *Dominant Pair:* nodes that have an extremely strong tie compared to the rest of the edges in the ego-network (extracted from EPPL). A high activity between nodes persons in the network compared to the other links from that person indicates that something unusual occurs between those two nodes.

As the data points follow a power law, we can fit the function $f(x) = Cx^{\check{\alpha}}$, where C is a constant, and x is the variable of interest where the power law should be fitted on. $\check{\alpha}$ is the power law exponent. The resulting fitting line can be used to measure to distance of each ego-network to the fitting line, which in turn can be used as an indicator of the deviance of a node compared to the rest of the network. Hence, for each ego-network i, where y_i describes the real value of the feature, we compute the anomaly score ob as the distance to the fitting line as follows.

$$ob(i) = \frac{\max(y_i, Cx_i^{\check{\alpha}})}{\min(y_i, Cx_i^{\check{\alpha}})} \log(y_i - Cx_i^{\check{\alpha}} + 1) \tag{4.1}$$

An example of such a plot, the Egonet Density Power Law (EDPL), is illustrated in figure 4.1 on Zachary's Karate Club network. If the node is near or on the 'Star' line, it indicates that the node connect to different nodes that are not connected to each other. To obtain a list of most anomalous nodes, one sorts the list of anomaly scores in descending order.

4.1.2 Structural Clustering Algorithm for Networks

The Structural Clustering Algorithm for Networks (SCAN) is a method developed to detect communities in networks [40]. Many community detection algorithms solely focus on graph partitioning, while not considering (1) hubs, nodes that connect multiple clusters, and (2) outliers, nodes with only one connection in the network. SCAN attempts to achieve this by creating structure-connected clusters, meaning that nodes are clustered together if they share many common neighbors. The structural similarity is defined as follows.

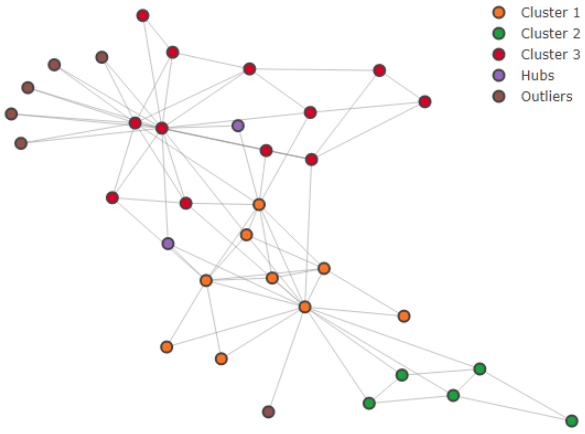


FIGURE 4.2: The results of SCAN with parameter setting $\epsilon = 0.5$ and $\kappa = 2$ on the Karate Club Network.

$$\nu(v, w) = \frac{\Gamma(v) \cap \Gamma(w)}{\sqrt{(}|\Gamma(v)||\Gamma(w)|} \tag{4.2}$$

Where $\Gamma(v)$ denotes the neighborhood of node v, including the node v. Since $\nu(v, w)$ returns a measure between 0 and 1, a threshold ϵ is set to assign cluster membership of a node. Nodes that share a structural similarity of at least ϵ with at least κ neighbors, is defined as a core vertex. The core vertices are used to assign cluster membership. ϵ and κ are two parameters that determine whether the node should be assigned to the cluster of a vertex core. An additional of SCAN is that the algorithms detects anomalous nodes by identifying (1) hubs, nodes that belong to no cluster and connect multiple clusters, and (2) outliers, by identifying nodes that do not belong to any cluster and solely connect to one cluster. An example of SCAN on Zachary's Karate Club Network is given in figure 4.2.

SCAN requires accurate and careful selection of parameters to detect hubs and outliers in an accurate manner. The authors propose to extract the nearest structural similarities of the neighbors for a sample of the nodes. By ordering the nodes by nearest structural similarity, one could identify the turning point at which the nodes tend to cluster together. One is able to use that measure to determine at what point a node should be considered a anomalous. An example of such a plot is given in figure 4.3 for the Karate Club network. Furthermore, the authors recommend to set $\kappa = 2$, which remains an intuitive choice to ensure that the algorithm identifies a vast number of core vertices to further cluster nodes in the network.

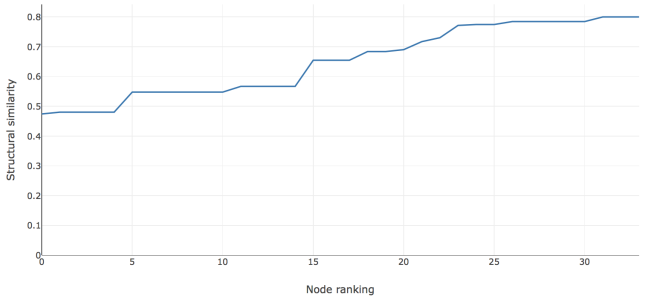


FIGURE 4.3: A nearest structural similarity plot for the neighbors of each node on Zachary's Karate Club network.

One issue of the SCAN algorithm is that it provides a list of hubs, while not providing an indication of the deviance of each hub. In some cases, the number of hubs detected can be of substantial size and an indication of the anomaly score of each hub may be necessary. By testing on synthetic networks, we found that the best way to uncover the node anomaly, was to assign an anomaly score sc to each node by computing the average structural similarity between each hub and its neighbors. A lower average structural similarity indicates that the node does not share any common neighbors with any of its neighbors. We also considered normalizing it solely over the degree and number of communities it is connected to, and chose this measure as it performed best on synthetic data.

4.1.3 Embedding approach

An emerging technique in the field of network science is that of network embedding. Network embedding focuses on embedding each node to a multi-dimensional vector while preserving the network structure in the embedding [46]. In general, network embedding techniques attempt to preserve the global structure to best summarize the network in a set of vectors. Hu. et al approached the problem of network-based anomaly detection by using a slightly different embedding approach. More precisely, they focused on identifying nodes that connect to a number of influential regions by preserving the local linkage structure of each node, instead of the global linkage structure [11]. Those nodes may be considered structural inconsistencies, as they obscure the community structure of the network and therefore decrease the performance of community detection methods. The embedding approach is described as follows.

Given an undirected graph $G = (V, E)$, associate each node $i \in V$ with a R-dimensional vector $\overline{X_i}$, where R is the number of influential regions and item $\overline{X_i^r}$ describes the relationship between node v and region r. As we want to embed each node into a vector, we want to find an embedding where nodes that are connected should have

similar values of $\overline{X_i}$, while disconnected nodes should have distinct values of $\overline{X_i}$. In other words, EMBED attempts to find an embedding of the nodes where the following equation holds true.

$$||\overline{X_i} - \overline{X_j}|| = \begin{cases} 0 & \text{if } (i,j) \in E \\ 1 & \text{if } (i,j) \notin E \end{cases} \tag{4.3}$$

Where $||\overline{X_i}||$ is the Euclidean norm of vector $\overline{X_i}$. The resulting value will always be between 0 and 1 because the authors impose non-negativity constraints on the vector and an upper bound of $\frac{\sqrt{(2)}}{2}$ for $||\overline{X_i}||$. However, this approach is not realistic, since there are many nodes that connect disconnected nodes, and those nodes cannot have similar values of $||\overline{X_i}||$ for both disconnected nodes. Therefore, a so-called stress function $S(\overline{X_i}, \overline{X_j})$ was defined that describes to what extent the ideal embedding is violated. The stress function is described below.

$$S(\overline{X_i}, \overline{X_j}) = \begin{cases} ||\overline{X_i} - \overline{X_j}||^2 & \text{if } (i,j) \in E \\ (||\overline{X_i} - \overline{X_j}|| - 1)^2 & \text{if } (i,j) \notin E \end{cases} \tag{4.4}$$

As this provides an embedding of the local linkage structure, we could formalize this as an objective function that can be minimized. However, as most networks are sparse and thus the quantity of non-edges is relatively high compared to the number of edges, there should be a balancing factor γ that ensures that there is a balance between the edges and non-edges. γ can be estimated by dividing the number of edges by the number of non-edges in the network. An more elegant way is to sample a set of non-edges that is of equal size to the number of edges, to approximately define the objective function Om as follows.

$$Om = \sum_{(i,j)\in E} ||\overline{X_i} - \overline{X_j}||^2 + \sum_{(i,j)\notin E} (1 - ||\overline{X_i} - \overline{X_j}||)^2, \; \gamma = \frac{m}{\binom{n}{2} - m} \tag{4.5}$$

The optimal embedding can then be found by using gradient descent. Since gradient descent is not scalable for large networks, the authors adopt a mini-batch gradient descent approach and solve the minimization as a nonlinear programming problem. Hence, the gradient descent method iteratively updates the embedding of each node i by

$$\overline{X_{i,t+1}} \leftarrow \overline{X_i} - \Lambda_t \cdot \nabla Om \tag{4.6}$$

Here, Λ_t is the step size of the tth iteration. A t of 50 is sufficient. The authors further propose to approximately represent Om by removing β and equal the number of existent edges (E) and non-existent edges E_s to balance the optimization, leading to the following ∇Om.

$$\nabla Om = \sum_{(i,j)\in E} 2||\overline{X_i} - \overline{X_j}|| \cdot \nabla||\overline{X_i} - \overline{X_j}|| + \sum_{(i,j)\in E_s} 2(||\overline{X_i} - \overline{X_j}|| - 1) \cdot \nabla||\overline{X_i} - \overline{X_j}|| \quad (4.7)$$

To further enhance the process, the authors propose to, (2) to initialize the vectors with equal embedding values if nodes belong to the same partition according to network partitioning method METIS, and (3) reduce the number of dimensions in vectors of the embedding to $k+$, where k is the average degree and β a toleration factor for the number of regions node anomalies connect to. According to the authors, $\beta = k/4$ is sufficient. After successful minimization of the network embedding, we represent the correlation of node i with r regions as follows:

$$\overline{NB(i)} = (y_i^1, ..., y_i^R) = \sum_{(j)\in NB(i)} (1 - ||\overline{X_i} - \overline{X_j}||) \cdot \overline{X_j} \quad (4.8)$$

The EMBED anomaly score $em(i)$ of node i can then be computed as follows.

$$em(i) = \sum_{j=1}^{R} \frac{y_i^k}{y_i^*} \quad (4.9)$$

Here, y_i^* is the maximum of $(y_i^1, ..., y_i^R)$. Embed runs in $O(t \cdot m \cdot (k + \beta))$, where t is the iteration threshold of gradient descent, again with m the number of edges and k the average degree. A drawback of EMBED is that the results are dependent on the number of dimensions R, that are chosen, and as such the approach is not parameter-free.

4.1.4 CADA

The methods discussed above, mainly focus on identifying the node anomalies from a local perspective, or are not parameter-free. Although it is sometimes postulated that the presence of anomalies might affect the performance of community detection algorithms [11], one may wonder to what extent it truly does so. In particular because community detection algorithms typically find an optimum of some function or quality

metric, and as such can deal quite well with imperfect divisions of a network into communities, for example because of anomalous nodes. The efficiency of modern community detection methods times has furthermore improved drastically over the years (they run in $O(m)$), allowing us to extensively investigate their performance for networks with more obvious as well as more obfuscated community structures, as we will do in Section 6. The proposed **Community-Aware Detection of Anomalies** algorithm consists of two steps.

First, CADA (*cd* in short) assigns each node to a particular community using an out-of-the-box community detection method [21]. In this paper, we employ two well-known community detection algorithms that both scale linearly in the number of edges and as such run in $O(m)$: the Louvain algorithm [22] and the Infomap approach [24]. These are extensively described in Section 2.3. We refer to CADA$_L$ or cd_L when Louvain is used as community detection method, and to CADA$_I$ or cd_I when Infomap is used as community detection method. Both assign each node to a community, and can handle undirected and directed networks (Louvain would ignore link direction), and can incorporate weights.

The second step of CADA is to assign an anomaly score to each node, based on the communities each node connects to. The anomaly score describes to what extent the neighbors of a node belong to a diverse number of communities, while not strongly belonging to one of them. Thus, for each node i, we create a vector g_i, where g_i^c represents the number of neighboring nodes of node i that belong to community c. g_i^* represents the maximum number of neighboring nodes that belong to the same community. We can then compute an anomaly score for each node as follows:

$$cd(i) = \sum_{j=1}^{c} \frac{g_i^j}{g_i^*} \tag{4.10}$$

4.2 Identifying moments of interest in dynamic networks

Contrary to static networks, dynamic networks are networks that change over time. Research in the field of anomaly detection in dynamic networks has seen increased interest in recent years, as one is able to define normal behavior of a node and its connections over time. This makes it a powerful approach to identify points in time where a node, sub-graph, or entire network behaves considerably different compared to other moments in time. While a variety of techniques exist in the field of dynamic network-based anomaly detection, we will focus on one approach in particular: event

detection. Event detection is useful for the investigator, because it can uncover moments of critical importance in the network that is under investigation.

The problem of event detection can be described as follows. Given a graph stream, or a sequence of graphs, $G_1, G_2, ..., G_T$, where T is the total number of time steps in the network, identify graphs in time where the network has changed significantly compared to previous points in time. Note that choosing the width of each time step t between G_i and G_{i-1} with $0 < t < T$ is crucial to correctly identify anomalous points in time, because events can be very context specific. For example, in a corporate e-mail network, one can expect to find low e-mail activity in the weekends, causing inconsistencies in various statistics of the graphs in the graph stream if observed on a daily basis. Moreover, by comparing graphs on a monthly basis, the width may be to wide to accurately understand on what day or week the anomaly occurred.

4.2.1 EdgeDiff

A straightforward approach to detect events in networks, is to observe the difference in edge count of the network over time. We refer to this approach as EdgeDiff. One can expect that at an important moment in time, the activity in the network increases, leading to a growth in the number of edges at that moment in time.

Although this may be a simple and effective approach, it may oversee crucial information that can only be discovered by monitoring other graph statistics over time. As an example, consider an communication network of an organization. At a certain moment in time, the CEO is replaced by a newcomer. The newcomer decides to change its board of directors, and higher-level management. While the activity of communication between the old and new board of directors may still be relatively similar, the change of management leads to a change in the global structure of the network, obscuring the old network structure.

Hence, we introduce two measures that indicate to what extent two graphs are similar to each other for the purpose of event detection: NetSimile [38], and DeltaCon [43]. Lastly, we discuss how these measures can be compared to detect anomalous events in time.

4.2.2 NetSimile

NetSimile was the first to measure the similarity between two graphs [38]. It consists of three steps: (1) feature extraction, (2) feature aggregation, and (3) comparison. We discuss each step below shortly.

The first step, feature extraction, computes several features f on a local basis for each node. NetSimile computes the following features for each node, but note that NetSimile is not limited to these features:

- k_i: Degree of node i
- e_i: Clustering coefficient of node i
- K_i: Average number of two-hop away neighbors of node i
- $\widetilde{e_i}$: Average clustering coefficient of the neighboring nodes of node i
- m_i: Number of edges in the ego-network of node i
- $m_{i,out}$: Number of outgoing edges from ego-network i
- nb_i: Number of neighbors of ego-network i

The feature extraction results in a $n \times f$ matrix F_{G_t} for each graph G_t in the graph stream. The second step, feature aggregation, summarizes each feature of the graph matrix F_{G_t} to five values, namely the median $\tilde{x}$, mean μ, standard deviation σ, skewness $skew$, and kurtosis $kurt$. While the former three are relatively straightforward, the latter two can be described as follows. The skewness provides an indication of asymmetry of the distribution.

$$skew = \frac{\sum_{i=0}^{n}(Y_i - \mu)^3/n}{\sigma^3} \tag{4.11}$$

Here, Y_i denotes the ith value in the distribution. The kurtosis describes to what extent the distribution is light tailed or heavy tailed. Mathematically, it is denoted as follows.

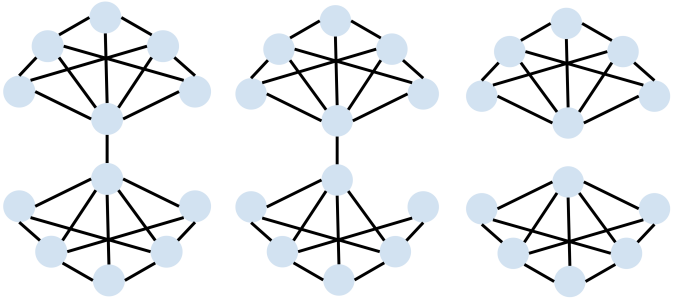
$$kurt = \frac{\sum_{i=0}^{n}(Y_i - \mu)^4/n}{\sigma^4} \tag{4.12}$$

The feature summarization step results in a signature vector S_{G_t} with $fn = f \times 5$ ordinates for each graph in the graph stream. The last step is to measure the similarity between the signature vectors S_{G_t} and $S_{G_{t-1}}$ for the entire graph stream. The authors found that the Canberra Distance suited best for comparison, as it is sensible for small changes near zero. The Canberra distance D_{Can} is described as follows:

$$D_{Can}(S_t, S_{t+1}) = \sum_{i=1}^{fn} \frac{|S_{t,i} - S_{t+1,i}|}{S_{t,i} + S_{t+1,i}} \tag{4.13}$$

4.2.3 DeltaCon

DeltaCon was constructed to measure the similarity in the node affinity of each node to every other node in the G_t and G_{t-1}. The rationale behind the method is that connections between nodes that connect multiple communities should be accounted for more as a similarity difference compared to sole connections in the network. As an example, consider the three graphs in Figure 4.4. While these are nearly similar, graph (b) is different to graph (a), because one connection is missing in one of the two communities. Similarly, in graph (c) one connection is missing. However, the connection links both communities, and as such the difference between graph (a) and graph (c) should be accounted for more compared to graph (a) and graph(b).



(a) Sub graph (b) Sub graph (c) Sub graph

Figure 4.4: Example graphs for DeltaCon. Figure (a) illustrates a normal network, figure (b) illustrates a change in a sub-graph of network (a), while figure (c) illustrates a change that causes the two sub-graphs to become separate. DeltaCon penalizes for changes that lead to a change in the global structure of the network.

With this intuition, the authors propose to use Fast Belief Propagation [43], a variant of Random Walks with Restarts that is based on maximum likelihood estimations. Furthermore, the underlying technique takes into account k-hop away neighbors as well with decreasing weight and therefore suits the intuition that nodes that function as bridge in the network should be accounted for more in terms of similarity. The $n \times n$ node affinity matrix B is defined as follows.

$$B = [b_{ij}] = [I + \zeta^2 K - \zeta A]^{-1} \tag{4.14}$$

Here, I is the identity matrix, K the diagonal matrix with the degree of node i on the K_{ii} dimension, and A the adjacency matrix of the graph. ζ captures the influence between neighbors. Among the many measures that were investigated, the authors propose to use the Matusita distance because it is similar to the Euclidean distance, while it usually gives better results because it even detects small changes in the graph. The Matusita distance D_{mat} is defined as follows.

$$D_{Mat} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(\sqrt{b_{t,ij}} - \sqrt{b_{t+1,ij}})^2} \tag{4.15}$$

Here, $s_{t,ij}$ is the node affinity of node i to node j of the affinity matrix s on timestep t. Note that we obtain the similarity between consecutive graphs by $sim = \frac{1}{1+d_{Mat}}$.

4.2.4 Anomaly detection in graph time series

The similarity measures and the difference in number of edges of consecutive graphs provide how the networks differ to each other over time. By using these methods, we obtain a set of data points for each similarity measures. Now, the challenge is to flag data points that deviate significantly from the rest of the distribution. Recall that we discussed how to flag data points as anomalous from data distributions in Section 2.5. The authors of NetSimile and DeltaCon chose a similar approach. That is, that data points that exceed the $\tilde{x} + 3\sigma$ (for NetSimile), or $\tilde{x} - 3\sigma$ (for DeltaCon) are flagged as anomalous. Here, $\tilde{x}$ is the median, and σ the standard deviation.

Chapter 5

Experimental setup

In this Chapter we describe how we answer the problem statement and research questions. Throughout this research, it became clear that it is difficult to evaluate network-based anomaly detection algorithms for a variety of reasons. Therefore, we narrow down the scope to one specific anomaly in static anomaly detection: *the node anomaly*, the node that is unaware of the global structure of the graph. But before we are able to extensively review the anomaly detection algorithms, we discuss the data sets that are used in this research in Section 5.1. In Section 5.2 we discuss the settings of each anomaly detection algorithm that is discussed in the previous Chapter. Lastly, we discuss how the algorithms are compared to each other in Section 5.3.

5.1 Data sets

As we described in Section 3.3, there are no available labeled data sets to evaluate network-based anomaly detection techniques. Therefore, we described five mechanisms to evaluate the algorithms, and consider two of them most suitable for evaluation purposes: (1) synthetic graph injection, and (2) qualitative evaluation. These two are chosen because they allow to quantify and qualify the performance of each anomaly detection algorithm. We describe the construction of the data sets below.

5.1.1 Synthetic data sets

We generate synthetic networks with ground-truth community structure ranging from 1,000 to 500,000 nodes with the LFR benchmark graph [27]. The LFR benchmark is chosen because it satisfies many real-world properties. Moreover, the range of parameters to tune in the LFR benchmark makes the benchmark extremely suitable for the

TABLE 5.1: Parameter settings for generating LFR benchmark networks.

Parameter	Description	Setting
n	The number of nodes	from 10^3 to $5 \cdot 10^5$
d	Average degree	$2 \cdot n^{1.15}/n$
k_{max}	Maximum degree	$n^{1/(t_1-1)}$
λ_1	minus exponent for degree distribution	3
λ_2	minus exponent for community size distribution	2
ρ	Mixing parameter for topology	from 0.1 to 0.6

generation of unweighted, weighted, undirected, and directed networks. The parameters are illustrated in Table 5.1, and chosen to cover a wide variety of network structures. We generated 10 networks of 100,000 nodes for each mixing parameter between 0.1 to 0.6, and 10 networks for each size ranging from 1,000 to 500,000 with a mixing parameter of 0.4.

Following the approach suggested in [11] we employ two generative processes to insert anomalies in the synthetic networks, that reflect on the node anomaly.

Random anomaly is inspired by the fact that infiltrating nodes are not aware of the global network structure and therefore connect to random nodes in the network. The anomalies are injected by adding $n/100$ nodes that connect to x random existing nodes, where x is between k and k_{max}. For each inserted node, the value of x is set by drawing a value from the the same power law degree distribution as that of the synthetic network.

Replaced anomaly first generates $n + h$ nodes with the LFR benchmark. The goal is to replace h nodes to obtain $n + n/100$ nodes. We randomly select x existing nodes in the network that have a degree lower than $2 \cdot k$. An anomaly is injected by rewiring all edges from the x nodes to the new anomaly. The x nodes are then removed from the network. the value of x ranges from 2 to 21, with an increment of 1, until $n + n/100$ nodes are obtained.

5.1.2 Real-world data sets

Ranshous et al. provide a clear overview of fifteen data sets that can be used in either a static or dynamic context [31]. The Enron data set is the main focus of our investigations, because it closely reflects to real-world data in the field of electronic discovery. The Enron data set is made available by the Federal Energy Regulatory Commission as a part of the investigations of the Enron scandal. It consists of e-mail boxes from about 150 former Enron employees that had a senior position. During our experiments, we noticed that

there were some issues in reproducing work of other papers, such as [2, 38, 43], as they do not elaborate on the way the network data set was extracted. We therefore describe precisely how the data set should be processed to obtain similar results.

We extracted all one-to-one and one-to-many e-mails on a weekly basis, including CC and BCC, where the nodes represent e-mail addresses and the edges represent whether two nodes have sent an e-mail to one another from a preprocessed database, published at http://www.ahschulz.de/enron-email-data/. An e-mailbox often consisted of multiple e-mail addresses belonging to the same person. All those e-mail addresses are normalized to the foremost e-mail address of that e-mailbox. Furthermore, the Enron data contains a lot of duplicates. For example, if an e-mail was sent from one to the other e-mail box, the e-mail belongs to both e-mail boxes. Hence, we deleted duplicate e-mails if the from, to, and message identifier were equal. We generated a network for each week, starting from July 1999 to December 2002.

Lastly, it is important to note that one can obtain different kinds of communication networks. One possibility is to include all e-mails that were sent from or to one of the e-mail boxes. This results in a network with many nodes that only sent one e-mail, and includes irrelevant and spam e-mails. However, it may also contain relevant information about e-mail addresses that interacted with the senior employees of the network. We therefore chose to create two Enron data sets, (1) Enron_{core}, that only consists of the e-mail addresses of the senior employees from the e-mailboxes, and (2) Enron_{total}, that consists of all encountered e-mails addresses in the data set.

Furthermore, to support our results in static anomaly detection, three other real-world data sets were chosen to qualitatively assess the performance of the anomaly detection algorithms. *Douban* (http://socialcomputing.asu.edu/datasets/Douban) is a Chinese recommendation website where a link exists between two users if they had an explicit friendship connection. *Amazon* (http://snap.stanford.edu/data/amazon0601.html) is a co-purchasing network where a link exists between two articles if these products are frequently purchased together on Amazon. *DBLP* (http://projects.csail.mit.edu/dnd/DBLP) is a collaboration network based on co-authorship between mostly computer scientists, extracted from the popular DBLP listing website. See Table 5.2 for an overview of the data sets. The number number of messages is only relevant for the Enron network as the weight of the links is also included in the analysis.

TABLE 5.2: Properties of the real-world network data sets.

Data set	Description	n	m
$Enron_{core}$	E-mail network	149	1,853
$Enron_{total}$	E-mail network	75,377	293,200
Douban	Social network	154,907	327,162
Amazon	Co-purchase network	403,394	2,443,408
DBLP	Co-authorship network	1,412,414	5,947,085

5.2 Settings of anomaly detection algorithms

In this section we describe the parameter settings for each algorithm. We firstly discuss the settings of the proposed static anomaly detection algorithms. Then, we discuss how the parameters are tuned to consider a graph in time as anomalous in Section 5.2.2.

5.2.1 Static algorithm settings

The static anomaly detection algorithms rely on different concepts of machine learning, such as clustering and network embedding. Therefore, some algorithms require parameter selection to detect anomalies. We discuss each algorithm and its settings below shortly.

ODDBALL is completely parameter free. However, the distance to the fitting line needs to be computed to measure the anomaly score of each node. For the purpose of identifying the node anomaly, we only need to identify the stars, i.e., the nodes below the fitting line of the EDPL. Thus, we can redefine Equation 4.1 as follows.

$$ob(i) = \frac{Cx_i^{\check{\alpha}}}{y_i} \log(y_i - Cx_i^{\check{\alpha}} + 1) \tag{5.1}$$

For SCAN, we choose two parameters, κ and ϵ. Recall, that κ and ϵ can be selected by identifying the turning point in the structural similarity plot as was shown in Figure 4.3. We noticed that the structural similarity plot was not always helpful in selecting a good value for ϵ. We finally set κ to 2, and ϵ to 0.3 on the synthetic data sets to maximize performance. This provided us with a list of hubs, and the hubs were ordered in ascending order of average structural similarity with the neighbors of each hub in the network as an indicator of deviance. Hence, hubs with lowest average structural similarity were considered most anomalous.

EMBED relies on the number of dimensions R. As mentioned in the paper, $R = n/500$ has shown to provide good results for various graphs. However, this does not scale

properly with network size. Hence, we set $R = \sqrt{n}$ to let the algorithm scale with all network sizes.

CADA is completely parameter free. However, one can tune the performance of CADA by tuning parameters of the out-of-the-box community detection algorithms. As an example, one can tune the resolution parameter of the Modularity maximization algorithm Louvain. We set the resolution parameter to 0.1.

5.2.2 Dynamic algorithm settings

There are no parameters to tune for the dynamic algorithms. However, we must decide when a certain moment in time should be considered anomalous. Recall that, DELTACON and NETSIMILE both suggest to solely observe the median of the network and flag moments as anomalous when the similarity between consecutive graphs exceeds a threshold of the median and z standard deviations $x \pm z \cdot \sigma$. The authors set z to 3.

We approach the problem in a similar fashion, in which we tune z to maximize performance. Furthermore, note that this approach may flag two consecutive graphs in time as anomalous, because one of them was considerably different compared to the previous and upcoming graph. This leads to a low similarity for two consecutive graphs. In this case, we only flag the first consecutive graph as anomalous. Note that for DELTACON, graphs are flagged anomalous if the similarity is below $x - z \cdot \sigma$, while data points for NETSIMILE and EDGEDIFF are considered anomalous if the similarity is below $x - z \cdot \sigma$. Furthermore, an anomaly is considered correct if the anomaly is detected $\pm$ 10 days from the ground-truth event.

5.3 Perform comparative experiments

In this section, we define how we evaluate and compare the anomaly detection algorithms in a static (see Section 5.3.1) and dynamic context (see Section 5.3.2).

5.3.1 Static anomaly detection

We will qualitatively and quantitatively evaluate the static anomaly detection algorithms. Combining both mechanisms gives us a fair performance overview of each algorithm.

5.3.1.1 Evaluation on synthetic data sets

As it is hard to quantitatively evaluate on unlabeled data sets, we can only evaluate on synthetic graphs. The output of what is considered an anomaly can vary per technique. For example, one technique provides an anomaly score for each node, while the other technique labels whether the node is anomalous or not. To fairly evaluate and compare the algorithms with each other, we have defined an anomaly score for each node as is described in Section 6.1. Therefore, we can sort the list of anomaly scores to obtain the first k most anomalous nodes and evaluate on the difference between them.

We perform the quantitative evaluation by firstly imputing a fixed number of $i = 0.01n$ anomalies in each synthetic graph. We flag nodes as anomalous as follows. We firstly run $Cada_I$ on each data set. Then we compare the 1% most anomalous nodes according to each algorithm with the ground-truth anomalies by computing the F1-score. The F1-score is based on recall and precision. Recall is the fraction of true positives (discovered anomalies) divided by the total number of ground-truth anomalies (that we inserted), while precision is equal to the true positives divided by the number of nodes that are flagged anomalous. The F_1-score is then:

$$F_1 = 2 \cdot \frac{precision \times recall}{precision + recall} \tag{5.2}$$

5.3.1.2 Evaluation on real-world data sets

The qualitative evaluation is twofold: (1) we provide a case study of the nodes on the Enron e-mail network and the DBLP network, and (2) we compare how many common nodes are found in the top 1% most anomalous nodes of the considered real-world networks. Note that this could be slightly more than 1% because nodes could have the same anomaly score, in which case we included all nodes with the same score as the node at the exact cutoff. For two sets of discovered node anomaly sets A_1 and A_2, we use the Jaccard similarity, defined as follows:

$$J(A_1, A_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|} \tag{5.3}$$

5.3.2 Dynamic comparative experiments

To evaluate on the dynamic anomaly detection algorithms, we use ground-truth data of events that happened in the last years of Enron. Hence, we can check whether anomalous graphs that are discovered by any of the methods, correspond to the ground-truth events.

Performance of each method can be further described in terms of recall, precision, and F_1, as described in Section 5.3.1.1. Furthermore, we observe the distribution of each similarity measure over time, to examine whether we are missing crucial items that should actually be flagged anomalous.

Note that, the significance of events is merely subjective, as a variety of newspapers highlight different moments as important in the last years of Enron. We selected the most important events that follow the timelines of the NY times[1], The Guardian[2], and the movie Enron: The smartest guys in the room. The ground-truth events are listed in Table 5.3.

TABLE 5.3: Ground-truth events from the Enron scandal between June 1999 and February 2002.

Date	Description
May 5, 2000	Operation Death Star announced
July 19, 2000	Enron announces deal with Blockbuster
August 23, 2000	Enron hit all time high
December 13, 2000	Enron announces Skilling will take over as CEO
February 12, 2001	Skilling takes over as CEO
March 5, 2001	Bethany McLean releases article
April 17, 2001	The asshole call
May 14, 2001	Mintz sends memorandum to Skilling on LJM paperwork
June 22, 2001	Skilling hit in the face by a pie in California
August 14, 2001	Skilling resigns as CEO while stating company is doing well
October 16, 2001	Enron reports gigantic loss
October 23, 2001	Lay supports Fastow during conference call with analists
October 24, 2001	Fastow fired
November 19, 2001	Enron restates its third-quarter earnings and discloses to restructure
February 4, 2002	Lay resigns from board
February 7, 2002	Skilling testifies before congress

[1] https://www.nytimes.com/2006/01/18/business/worldbusiness/timeline-a-chronology-of-enron-corp.html

[2] https://www.theguardian.com/business/2006/jan/30/corporatefraud.enron

Chapter 6

Results

In this Chapter we illustrate and describe the results of the experiments of graph-based anomaly detection algorithms. In Section 6.1, we report how each of the static anomaly detection algorithms performs on both synthetic and real-world networks. Then, we follow up with the detection of events in consecutive graphs in Section 6.2.

6.1 Static comparative experiments

This section encompasses results of the four discussed static anomaly detection algorithms on real-world networks (see Section 6.1.1) and synthetic networks (see Section 6.1.2).

6.1.1 Results on real-world data

We firstly illustrate an anomaly found on ENRON$_{total}$, using ODDBALL, with the EDPL, to measure the distance to the fitting line for each node. Illustrated in Figure 6.1, the foremost node anomaly is Kenneth Lay, CEO of Enron and key player in the ENRON scandal.

All other methods did not flag Kenneth Lay as an anomalous node. As a matter of fact, EMBED, SCAN and CADA gave Kenneth Lay a low anomaly score. One reason why this could be the case is that Kenneth Lay connects to a large number of persons that have no further connections in the graph, which causes CADA and EMBED to classify neighboring nodes to the same group. Furthermore, there is a lot of room for interpretation to fully comprehend the anomalies on the Enron data set, and we have no extensive ground-truth information about the nodes in the network.

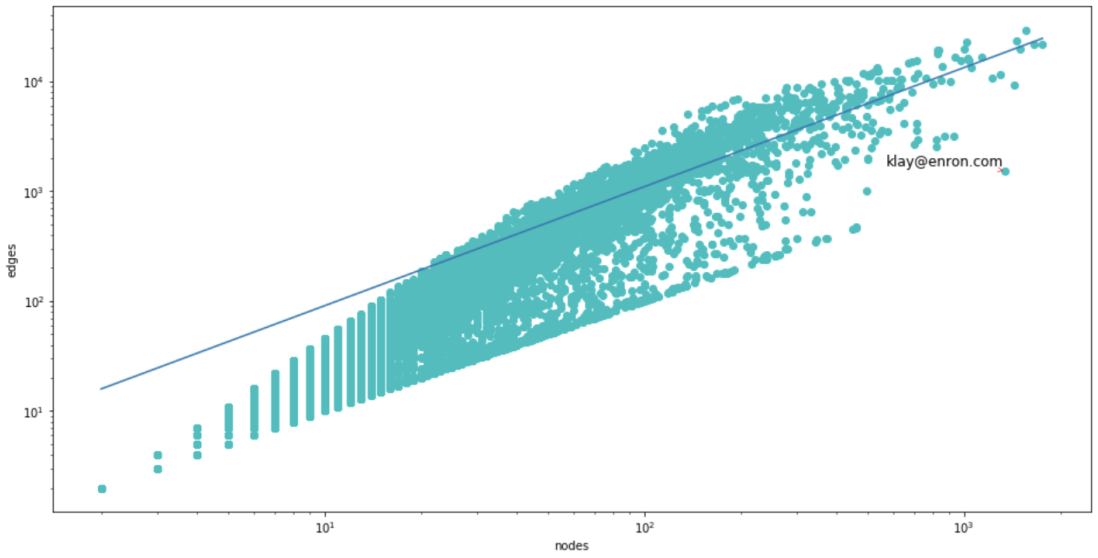


FIGURE 6.1: ODDBALL Egonet Density Power Law on the ENRON$_{total}$ network, where CEO Kenneth Lay, key player in the ENRON Scandal, is flagged as an anomaly.

Therefore, we decided to zoom in on the DBLP data set, and two noteworthy findings were detected by CADA. These anomalies were solely identified by CADA with Infomap, but ignored by ODDBALL, SCAN, and EMBED. First, we found authors that have published with many different authors, such as prof. dr. H. Vincent Poor, who was president of the IEEE Information Theory Society, and at the time of the data set has published with over 400 authors from all over the world. Second, many discovered anomalies were actually authors with the same name, such as 63 distinct authors named 'Wei Liu' that collaboratively published with over 1332 different authors. Other such authors were 'Jing Li', 'Yan Zhang', and 'Yu Zhang'. This illustrates that apart from outliers such as authors with extremely large number of publications, also errors in the underlying data can efficiently be identified using anomaly detection techniques.

After a careful check, we found that there is a small overlap in the anomalies that were detected by each method. Therefore, we also zoomed in on the agreement between different methods on real-world data sets, of which results in the form of Jaccard similarity are reported in Table 6.1.

6.1.2 Evaluation on synthetic data sets

To quantitatively evaluate the static anomaly detection algorithms, we generated LFR benchmark graphs and imputed Random Anomaly and Replaced Anomaly in the synthetic graphs, as described in Section 5.1.1. We firstly evaluate the performance on synthetic networks with different degrees of community structure, by illustrating the

TABLE 6.1: Jaccard similarity of the most anomalous nodes on *DBLP*, *Amazon*, *Douban*, $Enron_{total}$, and $Enron_{core}$ (left to right, up to down). ODDBALL (ob), EMBED (em), CADA Louvain (cd_L), and CADA Infomap (cd_I).

	sc	em	cd_L	cd_I
ob	0.0	0.02	0.02	0.02
sc	-	0.09	0.16	0.22
em	-	-	0.22	0.24
cd_L	-	-	-	0.24

	sc	em	cd_L	cd_I
ob	0.03	0.11	0.11	0.17
sc	-	0.06	0.06	0.09
em	-	-	0.17	0.20
cd_L	-	-	-	0.21

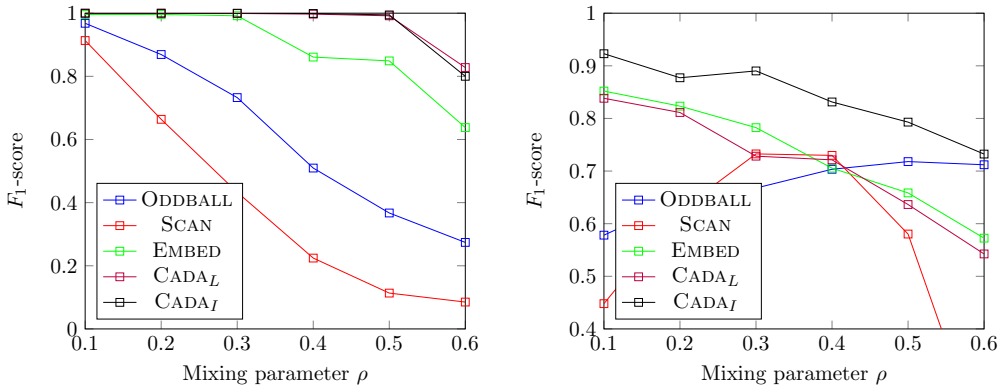
	ob	em	cd_L	cd_I
ob	0.1	0.00	0.00	0.00
sc	-	0.09	0.09	0.11
em	-	-	0.37	0.43
cd_L	-	-	-	0.34

	sc	em	cd_L	cd_I
ob	0.07	0.03	0.02	0.02
sc	-	0.07	0.04	0.07
em	-	-	0.25	0.23
cd_L	-	-	-	0.15

	ob	em	cd_L	cd_I
ob	0.15	0.0	0.36	0.0
sc	-	0.0	0.0	0.0
em	-	-	0.07	0.03
cd_L	-	-	-	0.0

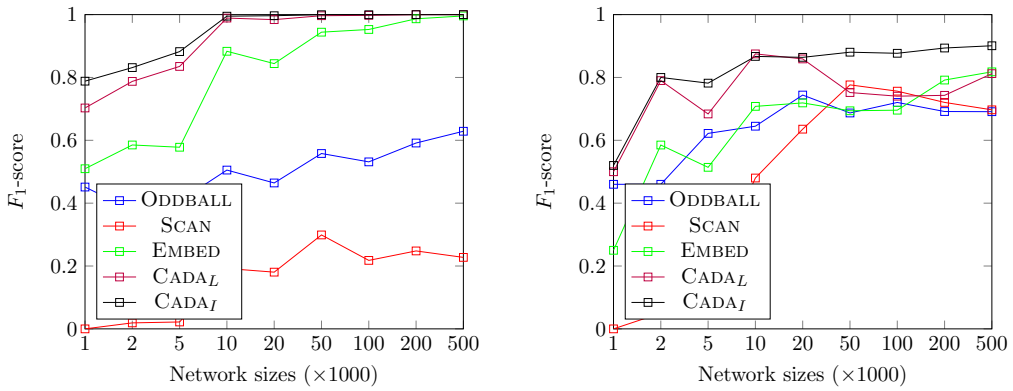
performance of each algorithm on networks with different mixing parameters (a higher mixing parameter means a less present community structure) in Figure 6.2.

Besides evaluating on networks with different community structures, another evaluation metric is to assess the performance on networks with varying network sizes. The results on networks with network sizes of $1,000$ to $500,000$ nodes with a fixed mixing parameter of 0.4 are illustrated in Figure 6.3.



(a) RandomAnomaly (b) ReplacedAnomaly

FIGURE 6.2: F_1-score for different values of the mixing parameter for networks with 100,000 nodes.



(a) RandomAnomaly (b) ReplacedAnomaly

FIGURE 6.3: F_1-score for different network sizes with fixed mixing parameter 0.4.

6.2 Results on dynamic networks

In this section, we identify anomalous moments in time by observing the similarity of the consecutive graphs on a weekly basis with three different similarity indicators, EDGEDIFF, NETSIMILE and DELTACON. We illustrate the number of edges at each time stamp in Figure 6.4.

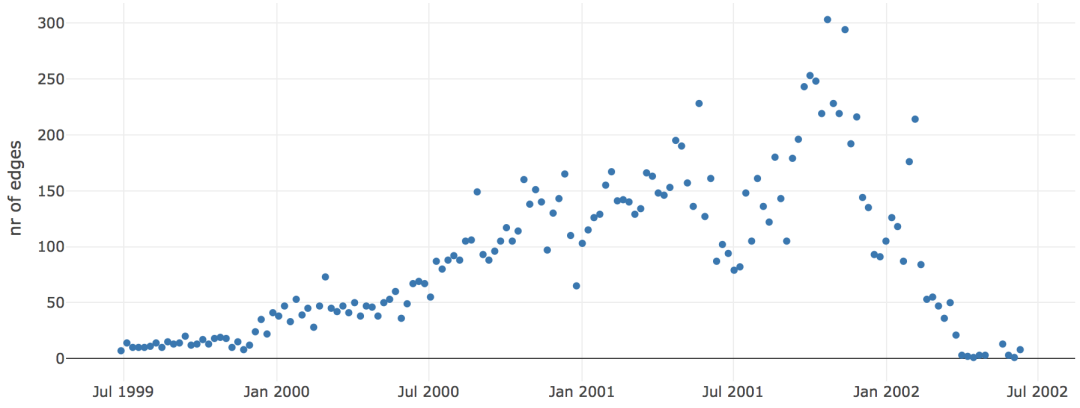
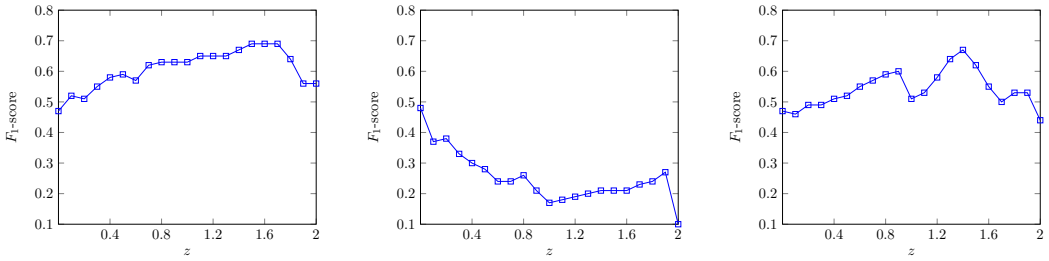


FIGURE 6.4: The number of new edges at each time stamp of Enron$_{core}$ over the time span of June 1999 to July 2002

First, we quantify the performance for different settings of the number of times z that the standard deviation σ should deviate from the median $\tilde{x}$. The results are illustrated in Figure 6.5. The illustration shows that EDGEDIFF and DELTACON perform best on the real-world Enron data set.

$F1$-scores indicate how well the graph is performing, but we are also interested in whether the metrics show to identify similar anomalies. To further qualify the performance, we illustrate the flagged anomalous events for each similarity metric in Figure 6.6. The value



(a) EdgeDiff (b) NetSimile (c) DeltaCon

FIGURE 6.5: F_1-score for each similarity metric for various z on the Enron$_{core}$ network, where z represents the number of times that the standard deviation should diverge from the median to consider a data point as anomalous.

of z was selected from Figure 6.5, so that the threshold maximizes $F1$-score. Hence, we set z to 1.5, 0.0, and 1.4 on EdgeDiff, NetSimile, and DeltaCon respectively.

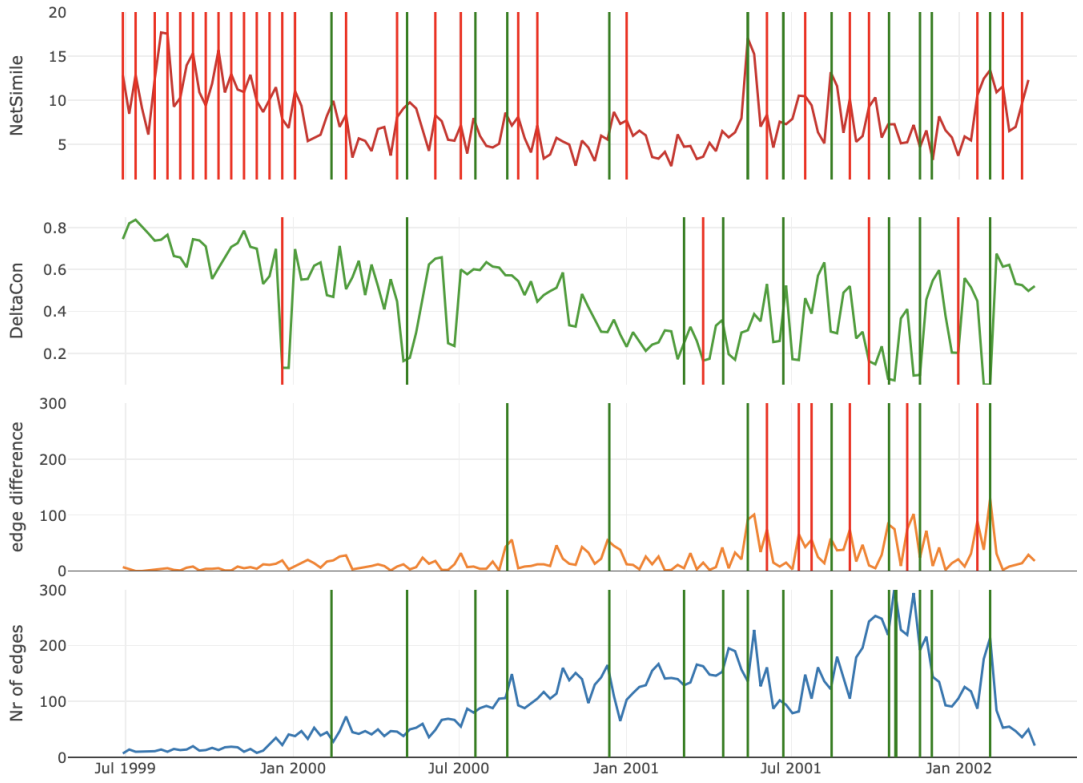


FIGURE 6.6: Events of Enron between June 1999 and February 2002. Note that the lowest figure illustrates the edge count over time, with the ground-truth events shown with the green lines. For the other three measures, the green line is an event that is correctly flagged as an anomaly, while the red line is an event that is incorrectly flagged as an anomaly.

Chapter 7

Discussion

In the previous Chapter we reported results of both static and dynamic anomaly detection algorithms. In this Chapter, we discuss the results obtained in the previous Chapter. In Section 7.1, we discuss the results of the static anomaly detection algorithms. In Section 7.2, we discuss the results of the dynamic anomaly detection algorithms.

7.1 Static anomaly detection algorithms

To accurately evaluate on the static network-based anomaly detection algorithms, we performed experiments on both synthetic networks and real-world networks. The results on real-world network data is discussed in Section 7.1.1, and the results on synthetic network data is discussed in Section 7.1.2. In Section 7.1.3, we discuss how each method is beneficial for a domain expert.

7.1.1 Results on real-world networks

The results on real-world networks demonstrate that the static anomaly detection algorithms under investigations were capable of identifying anomalous nodes. The first finding was illustrated in Figure 6.1, where Oddball provided a clear overview of the graph in two-dimensional space. Furthermore, Kenneth Lay, key player in the Enron scandal, was flagged as the most anomalous node in the network according to Oddball.

It is rather straightforward why Kenneth Lay was considered an anomaly, as he received e-mails from over 1,000 objects in the network. Of course, Kenneth Lay was in contact with a variety of people, may it be colleagues, journalists, or people that simply tried to

reach the CEO. Unsurprisingly, the high degree with many unrelated objects implies a low clustering coefficient, that results in a high anomaly score according to ODDBALL.

Other techniques that were investigated, were limited in the sense that they erenot able to give a similar two-dimensional overview as ODDBALL. However, by zooming in on the DBLP data set, we were able to give meaning to the anomalies discovered by EMBED and CADA. As a matter of fact, CADA found relevant anomalies that were not discovered by any of the other methods.

Table 6.1 demonstrates that the overlap in anomalous nodes on all real-world data sets under investigation varies a lot. It uncovers a limitation of CADA, as one would expect that the overlap between nodes from CADA$_L$ and CADA$_I$ would be higher than the overlap between nodes from CADA and the other methods. It illustrates, that the technique is dependent on the community detection performance of the network, and unfortunately there is no universal method to detect communities most accurately in each network. A possible step for future research may be to check which community detection algorithm performs best for CADA.

7.1.2 Results on synthetic networks

Results on the LFR benchmark networks show that each method is capable of uncovering node anomalies in the networks. We have illustrated how each algorithm performs in identifying the node anomaly with two methods, namely by (1) varying with the strengths of community structures (see Figure 6.2), and (2) by varying with the graph sizes (see Figure 6.3).

Figure 6.2(a) illustrates that the algorithms perform very well on networks with a strong community structure, that is, F_1-scores of 0.9 or higher for a mixing parameter of 0.1 on the Random Anomaly. Once the mixing parameter increases, the results of both ODDBALL and SCAN, that approach the node anomaly detection from a local perspective, perform worse than the other two methods. The performance of SCAN decreases most significantly, followed by ODDBALL. EMBED and CADA are performing significantly better, where CADA takes the lead with the Louvain and the Infomap community detection method.

Figure 6.2(b), again, illustrates that CADA$_I$ performs best in identifying the Replaced Anomaly on graphs with varying community structure. ODDBALL closes in when the community structure diminishes. It further demonstrates that CADA$_L$ performs worse than CADA$_I$, which is likely due to the resolution limit, which starts to play a larger role for higher values of the mixing parameter [47].

Furthermore, CADA performs best on both anomalies for networks of all sizes, with EMBED closing in, especially once the network becomes larger than $200,000$ nodes. EMBED starts performing better on larger networks as the number of dimensions R may play a smaller role on larger networks. In Figure 6.3(b), CADA with Infomap consistently performs best, but the difference with other methods is smallest.

7.1.3 Uses for the domain expert

While a variety of static network-based anomaly detection techniques exist, many of them focus on one specific anomaly; a node that connects to many communities while not belonging to one of them. We investigated how the anomaly detection techniques can benefit the domain expert, and we discuss each one of them below shortly.

The ODDBALL algorithm defines normal behavior by fitting a line on the data. It can be easily visualized in two-dimensional space and thus assist in navigating through the network. With the visualization it is rather easy to interpret why nodes are anomalous according to the algorithm. The most expensive task of the anomaly detection method is to extract the features from the egonet, with a complexity of $O(n \cdot d^2)$, where d is the average degree of the network. ODDBALL is not only limited in identifying the node anomaly, but is a best-of-suite method that can identify other anomalies by taking other features of the ego-networks into consideration. A clear shortcoming of the Oddball algorithm is that it solely identifies anomalies by observing the local neighborhood of a node, instead of utilizing regional features for the detection of anomalies.

SCAN does not only detect outliers, but also clusters the network from a local perspective. It can therefore inform the domain expert on two aspects: (1) by providing an overview of the groups of clusters in the network, and (2) by providing a list of anomalous nodes. However, the parameter optimization requires a lot of effort and it is difficult to determine whether the parameters are accurately selected. Furthermore, Figure 6.2 and 6.3 illustrates that SCAN flags many false positives as anomalies on the synthetic data set, especially as the mixing parameter increases. Later on it became clear why this is the case: the structural similarity becomes tremendously low for nodes that connect to many other nodes in the network, that is not limited towards nodes that are unaware of the global structure of the network. The time complexity of SCAN is $O(m)$.

EMBED solely focuses on identifying nodes that connect to a number of influential regions. It takes into account regional features (e.g. cluster membership), for identifying such nodes. It does, however, require that one chooses the number of dimensions R to find the optimal embedding. As a rule of thumb the authors claim that $R = n/500$ is a good initial value. However, this does not scale well with networks of various sizes

and different parameter settings gave rather different results. After some tuning, we set $R = \sqrt{(n)}$. EMBED scores well on synthetic data with a variety of network statistics, outperforming ODDBALL and SCAN on most networks, and closing in on CADA on networks with larger sizes. A shortcoming of EMBED is the time complexity, being $O(t \cdot m \cdot (k + \beta))$, and that the method is not completely parameter-free.

CADA, the new method that is proposed, is (1) parameter free, (2) scales linearly with the number of edges, (3) provides a list of groups that cluster together in the network, and (4) incorporates global features of the graph by using highly performing community detection methods to find anomalous nodes. It outperforms all above methods on a variety of synthetic data sets. On real-world data, CADA shows to identify relevant anomalous nodes that were not discovered by the other methods. Hence, the *node anomaly* is most accurately detected by CADA, making it a suitable approach for providing insight into the communities of the network and suitable for identifying anomalous nodes.

7.2 Dynamic anomaly detection algorithms

Time brings a completely new dimension to the analysis of networks. This is beneficial and relevant for the purpose of anomaly detection, because one is able to define how nodes, edges, or (sub-)graphs behave normally over time. While there exist a wide variety of applications in dynamic anomaly detection, we focused on event detection, where we used three different metrics to measure the similarity between consecutive graphs over time.

The similarity metrics were able to identify anomalous points in time that are meaningful in the context of the Enron scandal. Illustrated in Figure 6.5, EDGEDIFF and DELTACON perform well on the data set, while NETSIMILE performs considerably worse. EDGEDIFF performs most consistent manner, while DELTACON shows some fluctuation.

By zooming in on the distributions over time in Figure 6.6, we see that EDGEDIFF flags many anomalous events accurately, while missing other relevant events. DELTACON, that performed similarly to EDGEDIFF in terms of F_1-score, flags different moments in time as anomalous. This indicates that DELTACON is suitable to identify moments in time as anomalous that were elsewise not discovered. Moreover, due to the low similarity of first anomaly detected by DELTACON, we did a careful check. The anomaly flagged is on December 26, 2000. While this is not an event that corresponds with the Enron data set, it is clear that the graph is different compared to previous graphs due to Christmas.

It also became clear why NETSIMILE performs worse. First of all, most anomalous points are detected between July 1999 and January 2000. Due to the low activity in

terms of edges, the small differences between those graph are accounted for more for NetSimile. After February 2000, NetSimile performs better and identifies nearly as much anomalous events as false positives. This also leads to a drawback of our current approach to flag moments in time as anomalous.

By determining the median and standard deviation on the entire distribution of data points, one also takes into account statistics of the future. The network can change over time, with a growing or shrinking number of nodes, causing a change in what can be considered a normal similarity in the graph. As an example, one can expect that once the number of nodes and edges grow in general, the similarity between consecutive graphs becomes lower in general. Considering data points of the graph stream may therefore lead to a higher standard deviation, that causes the techniques to inaccurately flag moments in time as not anomalous. This could be further investigated in future work.

Nonetheless, our results illustrate that graph similarity measures can be used to indicate when events occurred in the network. Therefore, it can support the domain expert by illustrating at which moments in time something happened between the persons of interest in the investigation, that might lead to relevant moments in time that can be acted upon.

Chapter 8

Conclusions

In this Chapter we provide an answer to the three research questions and the problem statement formulated in Chapter 1. In Section 8.1 we provide answers to the research questions. In Section 8.2 we answer the problem statement. Finally, we discuss future work in Section 8.3.

8.1 Answers to the research questions

In this section, we provide an answer to each of the research questions formulated in Chapter 1.

Research question 1: *To what extent can network-based anomaly detection techniques be utilized to identify anomalous behaviour in static real-world networks?*

To answer this research question, we firstly identified and implemented three existing network-based anomaly detection techniques. One of them, demonstrated that anomalous nodes, edges, and sub-graphs could be found by observing various features of the direct neighborhood of each node of the graph in two-dimensional space. Other techniques discovered a similar node anomaly, the node that is unaware of the global structure of the graph. Typical examples of such anomalies are network intruders in physical networks or spammers spreading unwanted advertisement in online social networks.

The previously proposed techniques identified the node anomaly from a local perspective, or were not parameter-free. That motivated us to introduce a new method, named CADA, or Community-Aware Detection of Anomalies. CADA identifies anomalous nodes based on whether they connect to large number of communities, while not belonging to one distinct community themself. As such, it tackles the problem from a global

perspective. An advantage of this community-aware approach is that it scales linearly with the number of edges, improving upon previous techniques. Furthermore, it is parameter free and highly effective. Experiments showed that our proposed community-aware methodology can spot anomalies in both synthetic and real-world data sets that were not discovered by other methods. Furthermore, on all synthetic benchmark data sets, CADA outperformed previous approaches. In addition to uncovering the node anomaly, CADA provides a list of groups of people that tend to cluster together in the network.

Therefore, the answer to the first research question reads as follows. Network-based anomaly detection techniques can be used to gain insight into anomalous behaviour in static real-world networks. The techniques can be used to summarize the graph and uncover nodes that do not obey common patterns in the graph. Furthermore, most techniques also have additional benefits, where our new anomaly detection technique CADA also uncovers groups of people that tend to cluster together in the network. Hence, static network-based anomaly detection techniques can be a powerful technique to provide insight into the network for a domain expert.

Research question 2: *To what extent does the addition of the dynamic component in network-based anomaly detection affect the performance of anomaly detection?*

To answer this research question, we firstly discussed a variety of anomalies that can be detected in networks that change over time. A benefit of dynamic network-based anomaly detection is that one can learn how nodes, edges, or (sub-)graphs behave normally over time. One can utilize this information to discover at what points in time parts of the graph behave considerably different and mark those points as anomalous.

Although the possibilities of dynamic network-based anomaly detection are endless, we have focused on detecting one anomaly in specific, the event anomaly. We showed that one can use various graph features to quantify the similarity between consecutive graphs over time. Anomalous points in time can then be uncovered by using simple statistical anomaly detection methods on the set of similarities or by observing the similarity plots over time.

Therefore, the answer to the second research question reads as follows. In real-world applications, networks are constantly evolving over time. Incorporating the dynamic component when detecting anomalies can be of critical importance to accurately reduce the amount of manual work that should be done by the domain expert. Hence, event detection on networks can be a vigorous approach to identify activities that have occurred in a network of persons that are under investigation.

8.2 Answer to problem statement

Problem statement: *How can network-based anomaly detection algorithms support domain experts in detecting anomalous behavior in real-world networks?*

In this thesis we investigated whether we can assist domain experts in the field of electronic discovery by using network-based anomaly detection techniques. Our aim was to provide an overview of the current network-based anomaly detection techniques and compare them to each other in a unsupervised setting. We investigated (1) anomaly detection techniques in a network at one moment in time, and (2) the detection of events in graphs over time.

In Chapter 1, we defined when a domain expert is considered supported, which was when at least one of the three conditions is met: (1) the domain expert understands how to evaluate and compare several results from anomaly detection techniques, (2) the domain expert is provided with a list of objects that demonstrate divergent behavior, or (3) the domain expert knows why a certain event in time has occurred.

From our investigations, we may conclude that network-based anomaly detection techniques can be used to support domain experts in the field of electronic discovery to a certain extent. The methods can be used to rapidly summarize statistics of the network to uncover nodes or moments in time that deviate considerably from the expected patterns in the data set. These anomalies can be used as a starting point of the investigation, and therefore enhance the process of an investigator.

However, in some cases, it may be difficult to understand why certain nodes are considered anomalous. Therefore, domain knowledge can be of critical importance to comprehend the data set and accurately flag anomalous nodes or events. Of course, each network-based anomaly detection technique can be used to find such anomalies, but a combination of multiple methods and collaboration with a domain expert may significantly enhance the process of anomaly detection and uncover useful anomalies that can be acted upon in real-world networks.

8.3 Future work

This section provides recommendations for further research. Network science has more to offer than solely network-based anomaly detection, and therefore we have multiple recommendations that can support the domain expert in the field of electronic discovery.

For the e-discovery field, we have three recommendations for future work in the field of network science.

- *Network visualization and navigation:* While lists of anomalous nodes and moments in time are useful for a domain expert, navigating through the networks over time can truly support the domain expert in comprehending the graph. Hence, it allows the domain expert to analyze the network and its most important nodes in an accessible manner.
- *Community evolution:* The research field of community evolution focuses on identifying how significant communities shrink, grow, appear, and dissapear over time. Modeling such behavior may show how certain groups started to appear and how they have collaborated with each other over time.
- *Centrality-based methods:* This methods focus on identifying the most important nodes in the network, that is already done on the Enron data set [48]. Further research may be useful in combination with network visualization and navigation.

For network-based anomaly detection, we also have three recommendations for future work.

- *Node anomaly detection:* An interesting step in future development of CADA is identify which community detection methods are most robust for node anomaly detection, and whether a hybrid method combining both global and local features, may yield more accurate and relevant anomalies.
- *Attributed anomaly detection:* We only took into consideration network nodes and edges. The methods could be enriched by incorporating node and edge attributes, such as the weight of the graph. Hence, developing algorithms that use graph features and information about the nodes can provide new insights into the network.
- *Event detection:* As we mentioned in Section 7.2, a drawback of the current approach is that the median and standard deviation of the entire data set are used. Improvements may be achieved by investigating whether other statistical methods to anomaly detection, such as moving windows, can more accurately flag events as anomalous.

Bibliography

[1] Normal distribution. Normal distribution — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/wiki/Normal_distribution. [Online; accessed 3-July-2018].

[2] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, PAKDD, pages 410–421. Springer, 2010.

[3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009.

[4] A.L. Barabási, N. Gulbahce, and J. Loscalzo. Network medicine: a network-based approach to human disease. *Nature Reviews Genetics*, 12(1):56–68, 2011.

[5] M. J. Keeling, L. Danon, A. P. Ford, T. House, Chris P. Jewell, G. O. Roberts, J. V. Ross, and M. C. Vernon. Networks and the epidemiology of infectious disease. *Interdisciplinary Perspectives on Infectious Diseases*, 2011, 2011.

[6] E. Cotilla-Sanchez, P.D.H. Hines, C. Barrows, and S. Blumsack. Comparing the topological and electrical structure of the North American electric power infrastructure. *IEEE Systems Journal*, 6(4):616–626, 2012.

[7] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.

[8] G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC, pages 1–9. ACM, 2010.

[9] S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. In *Proceedings of the 19th National Information Systems Security Conference*, NISSC, pages 361–370. CSRC.

[10] C. Cortes, D. Pregibon, and C. Volinsky. Communities of Interest. *Intelligent Data Analysis*, pages 105–114, 2002.

[11] R. Hu, C. C. Aggarwal, S. Ma, and J. Huai. An embedding approach to anomaly detection. In *Proceedings of the 32nd International Conference on Data Engineering*, ICDE, pages 385–396. IEEE, 2016.

[12] J.H.M. Janssens. *Outlier selection and one-class classification.* Van Lankveld, 2013.

[13] E. Casey. *Handbook of Digital Forensics and Investigation.* 2010.

[14] J. T. Wells, V. Kanhere, and D. Ph. Principles of Fraud Examination. *Information Systems Journal*, pages 1–2, 2006.

[15] R. Tillman. Reputations and corporate malfeasance: Collusive networks in financial statement fraud. *Crime, Law and Social Change*, 51(3-4):365–382, 2009.

[16] M. S. Beasley, J. V. Carcello, D. R. Hermanson, and T. L. Neal. Fraudulent Financial Reporting. *Committee*, 12:60, 2010.

[17] M.D. Kohn, M.M. Eloff, and J.H.P. Eloff. Integrated digital forensic process model. *Computers & Security*, 38:103–115, 2013.

[18] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[19] J. Travers and S. Milgram. An Experimental Study of the Small World Problem. *Sociometry*, 32(4):425, 1969.

[20] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[21] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75 – 174, 2010.

[22] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 10008(10):6, 2008.

[23] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. 105(4):1118–1123, 2008.

[24] M. Rosvall and C. T. Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLOS ONE*, 6: 1–10, 2011.

[25] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69, 2004.

[26] U. Brandes, D. Delling, M Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[27] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E.*, 78(4), 2008.

[28] F.Y. Edgeworth. Xli. on discordant observations. *Philosophical Magazine Series 5*, 23(143):364–375, 1887.

[29] G. Münz, S. Li, and G. Carle. Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*, 2007.

[30] P. V. Bindu and P. Santhi Thilagam. Mining social networks for anomalies: Methods and challenges. *Journal of Network and Computer Applications*, 68:213–229, 2016.

[31] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova. Anomaly detection in dynamic networks: A survey. *WIREs Comput. Stat.*, 7:223–247, 2015.

[32] R. Hassanzadeh, R. Nayak, and D. Stebila. Analyzing the effectiveness of graph metrics for anomaly detection in online social networks. In *Proceedings of the 13th Conference on Web Information Systems Engineering*, WISE, pages 624–630. Springer, 2012.

[33] H. Tong and C. Lin. Non-Negative Residual Matrix Factorization with Application to Graph Anomaly Detection. In *Proceedings of the 2011 International Conference on Data Mining*, SIAM, pages 143–153, 2011.

[34] R. Kaur and S. Singh. Detecting anomalies in Online Social Networks using graph metrics. In *Proceedings of the 12th Annual IEEE India Conference*, INDICON, pages 1–6. IEEE, 2015.

[35] Deepayan Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *Proceedings of the 2004 Knowledge Discovery in Databases*, PKDD, pages 112–124. Springer, 2004.

[36] G. Wang, S. Xie, B. Liu, and P. S. Yu. Review graph based online store review spammer detection. In *Proceedings of the 11th International Conference on Data Mining*, ICDM, pages 1242–1247. IEEE, 2011.

[37] D. Koutra and Faloutsos C. Vogelstein, J. T. DELTACON: A principled massive-graph similarity function. In *Proceedings of the 2013 International Conference on Data Mining*, volume 10 of *SIAM*, pages 162–170, 2013.

[38] M. Berlingerio, D. Koutra, T. Eliassi-Rad, and C. Faloutsos. NetSimile: A Scalable Approach to Size-Independent Network Similarity. In *Workshop on Information in Networks*, WIN, 2012.

[39] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *World Wide Web Internet And Web Information Systems*, 54(1999-66):1–17, 1998.

[40] X. Xu, N. Yuruk, Z. Feng, and T. Schweiger. SCAN: A Structural Clustering Algorithm for Networks. In *Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining*, SIGKDD, pages 824–833. ACM, 2007.

[41] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th International Conference on Knowledge discovery and data mining*, SIGKDD, pages 677–685. ACM, 2008.

[42] Y. Wang, A. Chakrabarti, D. Sivakoff, and S. Parthasarathy. Hierarchical change point detection on dynamic networks. In *Proceedings of the Conference on Web Science*, WebSci, pages 171–179. ACM, 2017.

[43] D. Koutra, T. Ke, U. Kang, D. H. Chau, H. K. K. Pao, and C. Faloutsos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *Proceedings of the Conference on Machine Learning and Knowledge Discovery in Databases*, PKDD, pages 245–260. Springer, 2011.

[44] T. L. Fond, J. Neville, and B. Gallagher. Designing size consistent statistics for accurate anomaly detection in dynamic networks. *ACM Trans. Knowl. Discov. Data*, 12(4):46:1–46:49, 2018.

[45] T. J. Helling, J. C. Scholtes, and F. W. Takes. A community-aware approach for identifying node anomalies in complex networks. In *Proceedings of the 7th International Conference on Complex Networks*, CI, pages 244–255. Springer, 2019.

[46] C. Peng, W. Xiao, P. Jian, and Z. Wenwu. A survey on network embedding. *Transactions on Knowledge and Data Engineering*, 2018.

[47] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[48] J. S. Hardin, G. Sarkis, and P. C. Urc. Network analysis with the enron email corpus. *Journal of Statistics Education*, 23(2), 2015.