Unsupervised Anomalous Vertices Detection Utilizing Link Prediction Algorithms

Dima Kagan[*1], Michael Fire[†2,3] and Yuval Elovici[‡1]

[1]Department of Software and Information Systems Engineering,
Ben-Gurion University of the Negev
[2]Department of Computer Science & Engineering, University of
Washington
[3]The eScience Institute, University of Washington

October 27, 2016

Abstract

In the past decade, complex network structures have penetrated nearly every aspect of our lives. The detection of anomalous vertices in these networks can uncover important insights, such as exposing intruders in a computer network. In this study, we present a novel unsupervised two-layered meta classifier that can be employed to detect irregular vertices in complex networks using solely features extracted from the network topology. Our method is based on the hypothesis that a vertex having many links with low probabilities of existing has a higher likelihood of being anomalous. We evaluated our method on ten networks, using three fully simulated, five semi-simulated, and two real world datasets. In all the scenarios, our method was able to identify anomalous and irregular vertices with low false positive rates and high AUCs. Moreover, we demonstrated that our method can be applied to security-related use cases and is able to detect malicious profiles in online social networks.

Project Links: Website ▶ Data ▶ Code

Keywords: Complex Networks; Anomaly Detection; Cyber Security; Link Prediction; Social Networks; Machine Learning

1 Introduction

Complex networks are defined as systems in nature and society whose structure is irregular, complex, and dynamically evolving in time with thousands, millions,

---

[*]kagandi@post.bgu.ac.il   [†]fire@cs.washington.edu   [‡]elovici@post.bgu.ac.il

or even billions of vertices and edges [6, 9]. These systems reside in every part of our daily life [34, 23], such as electrical power grids, metabolic networks, food webs, the Internet, and co-authorship networks [30, 34]. Analyzing the unique structures of these networks can be very useful in research domains. For example, an analysis of network structures can reveal through which vertices a computer virus will spread most quickly in a computer network [7], or which vertex malfunction will affect more houses in a power grid [36].

Many studies have shown that vertices which deviate from normal behavior may hide important insights [3, 10, 24, 31, 32]. For instance, Bolton et al. [10] showed that fraudsters in e-commerce behave differently from the expected norm. Fire et al. [24] observed that fake profiles and bots in social networks have a higher probability of being connected to a greater number of communities than benign users. Nobel and Cook [31] showed that a graph-based technique is applicable for network intrusion detection. Over the years, studies have offered diverse solutions for anomaly and outlier detection in graph-based structures [4]. These studies have utilized various graph features, such as vertices, dyads, triads, and communities, to detect the fake profiles' behavioral patterns. For example, if we take a social graph, anomalous vertices can be malicious or fake profiles. These profiles do not represent a real person; they do not have real friends and connections in the social network. Therefore, the structure of their connections can indicate whether the vertices are malicious or benign.

In this study, we introduce a novel generic unsupervised learning algorithm for the detection of anomalous vertices, utilizing a graph's topology. The algorithm consists of two main iterations. In the first iteration, we create a link prediction classifier based only on the graph's topology. This classifier is able to predict the probability of an edge existing in the network with high accuracy (see Section 3.1). In the second iteration, we generate a new set of meta-features based on the features created by the link prediction classifier (see Section 3.2). Then, we utilize these meta-features and construct an anomaly detection classifier. Intuitively, the algorithm is based on the assumption that a vertex having many edges with low probabilities of existing has a higher likelihood of being anomalous.

We evaluated our anomaly detection algorithm on three types of complex networks: fully simulated networks, real world networks with simulated anomalous vertices, and real world networks with labeled anomalous vertices. In all the evaluated scenarios, our anomaly detection algorithm demonstrated results that were better than a random guessing algorithm and with low false positive values (see Section 4). For example, in the fully simulated network scenario we obtained an average AUC of 0.993 and FPR of 0.024, and in the online social network cases (Academia, Flixster, and Yelp) we had an average AUC of 0.996 and FPR of 0.004. We also explored the algorithm effects on a non-security related case. We found that we were able to detect a special group of people in the graph with an AUC of 0.931 and FPR of 0.15.

These results demonstrate that the presented algorithm can successfully detect malicious users in complex networks in general, and in online social networks
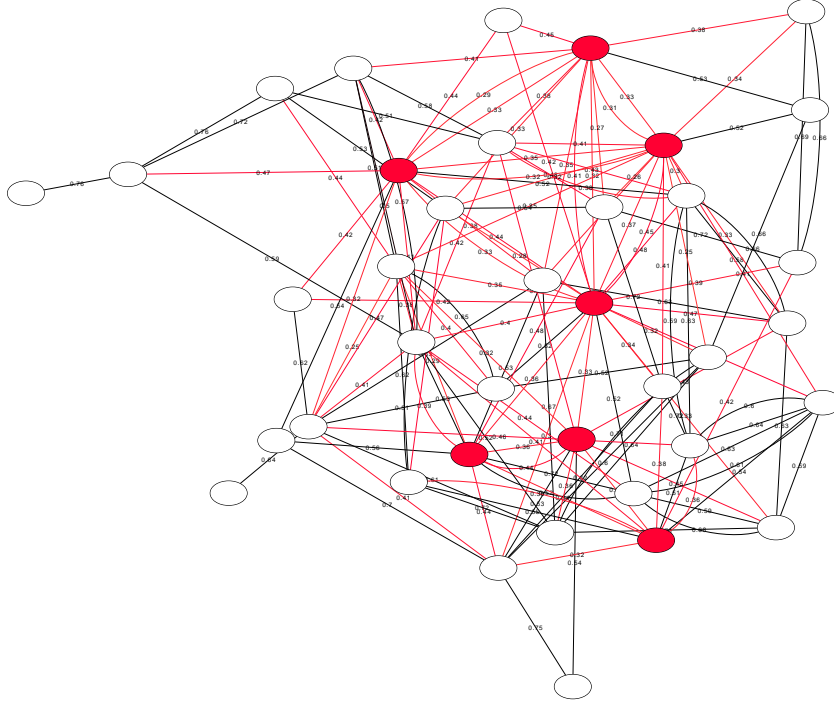
2

Figure 1: Boys' friendship network, where the red vertices represent the boys who are the most central individuals in the friendship network, and the red edges are the edges that are connected to the red vertices.

in particular. Moreover, we showed that this algorithm can be applicable as a general anomaly detection algorithm in additional domains.

The remainder of this paper is organized as follows. In Section 2, we present an overview of relevant studies. In Section 3, we describe how we constructed our algorithm. In Section 4, we present our results. In Section 5, we discuss the obtained results. Lastly, in Section 6 we present our conclusions.

2 Literature Overview

Anomaly detection is the discovery of items or behaviors that do not conform to the rest of the data. The detection of anomalies is an extremely useful ability because irregularities can be discovered without any prior knowledge or help of an expert. Detecting aberrations is very common and crucial in the cyber-security field. For instance, Hofmeyr et al. [28] used an anomaly detection method in order to spot intrusion in UNIX systems. Another example is the work of Fawcett and Provost [22], who developed a fraud detection method by profiling users'

behavior and detecting deviations. Additionally, anomaly detection is broadly used in diverse fields such as the medical area, sensor networks, etc. [14].

In 1999, Barabási and Albert [8] presented one of the first works in the modern complex network domain. They introduced the Barabási-Albert model (BA model), which is a minimal model that can generate scale-free networks. This study greatly influenced future works since many of the observed networks in nature and online are scale-free networks. Research of anomaly detection in complex networks has occurred only during the past two decades, as analyzing graphical data has grown in popularity [18]. There are still many undiscovered insights, however, especially in the structure-based method subgenre.

One of the first works that combined complex networks and anomaly detection was by Noble and Cook [31] in 2003. Noble and Cook proposed two methods for detecting anomalies in graphs. The first method was based on the concept that substructures reoccur in graphs, which means anomalies are substructures that occur infrequently. They also took into account that large substructures occur infrequently, and they developed a heuristic that considered the frequency and size of each substructure. Noble and Cook's second method was to divide the graph into subgraphs and calculate an anomaly measure, which they defined for each subgraph. Then they ranked the subgraphs, and the ones with the highest scores were considered more likely to be anomalous.

In this work, we rely on a link prediction algorithm as central to our anomaly detection method. Link prediction is defined as the discovery of hidden or future links in a given social network [29]. The link prediction problem was first introduced by Liben-Nowell and Kleinberg [29] in 2003 when they studied co-authorship networks and tried to predict future collaborations between researchers. They proved that future links can be predicted with reasonable accuracy from network topology alone.

In 2011, there was a surge in publications on link prediction due to the Kaggle IJCNN 2011 Social Network Challenge.[1] The challenge was to predict edges in an online social network that Kaggle provided. Cukierski et al. [16] proposed a method based on supervised machine learning and used 94 different features. They discovered that a Random Forest classifier performed best out of all the supervised machine learning methods evaluated. In addition, they found that EdgeRank (rooted PageRank) [12] was the highest-scoring feature out of the 94 they had used.

Fire et al. [25] analyzed which topological features are more computationally efficient. They tested a total of 53 different features that were divided into five subsets, using ten different datasets of online social networks. They discovered that it is sufficient to use a smaller subset of features to get results with relatively high AUCs [25]. Later, Fire et al. demonstrated that in many cases the benefits of using a large number of features is insignificant, and by using just computationally efficient features, it is possible to get highly accurate classifications [26].
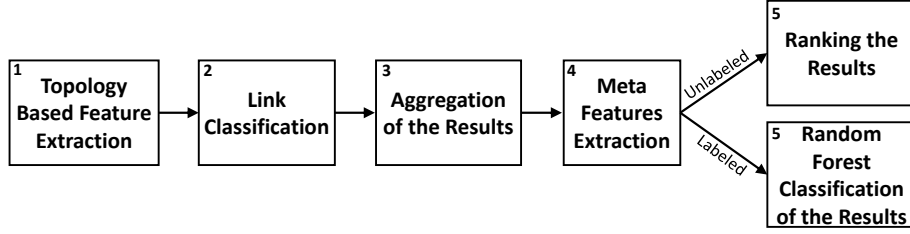
---

[1] https://www.kaggle.com/c/socialNetwork

Figure 2: The process of identifying anomalous vertices in a graph.

3   Methods and Experiments

In this study, we utilize graph topology to develop a novel generic method for identifying anomalous vertices in complex networks. The primary advantage of using graph structure is that topology-based methods are generic and can be utilized on most graph-based data.

Studies from the past several years indicate that many malicious users present different behavioral patterns than benign users [11, 13, 24]. Boshmaf et al. [11] described how fake profiles connect randomly to other users in order to establish an influential position or fame. Stringhini et al. [33] noticed that many spammers on Facebook chose to connect to other users according to their victims' names. Moreover, Fire at el. [24] described how fake profiles will likely connect to many communities. For instance, among fake-profile neighbors there is a high likelihood of finding many people without any mutual characteristics, such as working place, speaking language, etc.

Motivated by the observed behavioral patterns of fake profiles, we developed a method to generate examples for our link classifier. We generated positive examples by randomly selecting non-existing edges and negative examples by selecting existing ones. Then, for each of these edges we extracted features that were based on the network's topology (see Subsection 3.1), and we used the feature set to train a link prediction classifier (see Figure 2). Next, we aggregated the results from the link classifier for each vertex and created an additional set of features (see Subsection 3.2). Then, we extracted the second set of features and used them to build a meta classifier that identifies anomalous vertices in the graph. Lastly, we evaluated the meta classifier on simulated fake vertices and on real world anomalies (see Subsection 3.4).

3.1   Constructing a Link Prediction Classifier

As we described in Section 2, in the last decade researchers have proposed various methods for predicting links in graphs [5, 12, 16, 26, 29]. Moreover, researchers have demonstrated that link prediction classifiers can predict links with high precision on a wide range of complex networks [26]. In this study, we constructed a topology-based link prediction classifier based on the works of

Fire et al. [26] and Cukierski et al. [16]. We extracted 19 different features,[2] 16 of which are for undirected graphs and 8 are for directed graphs. Prior to describing how the features were used, we will define several notions. Let $G := (V, E)$ be a graph where V is a set of the graph's vertices and E the set of the graph's edges. $\Gamma(v)$ is defined as the neighborhood of vertex v, while $\Gamma_{in}(v)$, $\Gamma_{out}(v)$, and $\Gamma_{bi}(v)$ are defined as the inbound, outbound, and bidirectional set of neighbors, respectively.

$$\Gamma(v) := \{u | (u, v) \in E \ or \ (v, u) \in E\}$$
$$\Gamma_{in}(v) := \{u | (u, v) \in E\}$$
$$\Gamma_{out}(v) := \{u | (v, u) \in E\}$$
$$\Gamma_{bi}(v) := \{u | (u, v) \in E \ and \ (v, u) \in E\}$$

3.1.1 Edge-Based Features

1. **Total Friends** is the number of distinct friends between two vertices v and u. Let $v, u \in V$; then $Total\text{-}Friends$ of v and u will be defined as the number of vertices in the union of v friends with u friends.

$$TotalFriends(v, u) := |\Gamma(v) \cup \Gamma(u)|$$

2. **Common Friends** represents the number of common friends between two vertices v and u.

$$CommonFriends(v, u) := |\Gamma(v) \cap \Gamma(u)|$$

For a directed graph we will define three variations of Common Friends: $CommonFriends_{in}(v, u) := |\Gamma_{in}(v) \cap \Gamma_{in}(u)|$, $CommonFriends_{out}(v, u) := |\Gamma_{out}(v) \cap \Gamma_{out}(u)|$, and $CommonFriends_{bi}(v, u) := |\Gamma_{bi}(v) \cap \Gamma_{bi}(u)|$.

3. **Jaccard's Coefficient** is one of the most known link prediction features [16, 26, 29]. It measures similarity between two groups of items. Jaccard's Coefficient is defined as the ratio between $CommonFriends$ and $TotalFriends$.

$$JaccardsCoefficient(v, u) := \frac{|\Gamma(v) \cap \Gamma(u)|}{|\Gamma(v) \cup \Gamma(u)|}$$

4. **Preferential Attachment Score** is another well-known feature [26]. It is based on the idea that in social networks the rich get richer. The Preferential Attachment is defined as the multiplication of the number of friends of two vertices v and u.

$$PreferentialAttachment(v, u) := |\Gamma(v)| \cdot |\Gamma(u)|$$

---

[2] In a large dataset, computing dozens of features can last several hours or even several days; to avoid extremely long computations we used only computationally efficient features [26].

5. **Transitive Friends** for vertices v and u in a directed graph G calculates the number of transitive friends of u, v and v, u.

$$TransitiveFriends(v, u) := |\Gamma(v)_{in}| \cap |\Gamma_{out}(u)|$$

6. **Opposite Direction Friends** for a directed graph G indicates whether reciprocal connections exist between vertices v and u.

$$OppositeDirectionFriends(v, u) := \begin{cases} 1, & \text{if } (u, v) \in E \\ 0, & \text{otherwise} \end{cases}$$

7. **Adamic-Adar Index** is a similarity measure for undirected graphs which measures how strongly two vertices are related [29]. Higher scores will be given to edges that have rare connections [29].

$$AdamicAdarIndex := \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log |\Gamma(w)|}$$

3.1.2 kNN-Based Features

The kNN weight features are general neighborhood and similarity-based features [16]. They are based on the principle that as the number of friends goes up, the value of each individual friend is decreasing.

8. **Directed kNN Weights** are defined by two notations. Let $v, u \in V$, where v edges' weight will be $w_{in}(v) := \frac{1}{\sqrt{1+|\Gamma_{in}(v)|}}$ and $w_{out}(v) := \frac{1}{\sqrt{1+|\Gamma_{out}(v)|}}$, inbound and outbound, respectively. The weight of the connection between v and u can be measured using eight permutations of these weights: (a) $kNNW_1(v, u) := w_{in}(v) + w_{in}(u)$; (b) $kNNW_2(v, u) := w_{in}(v) + w_{out}(u)$; (c) $kNNW_3(v, u) := w_{out}(v) + w_{in}(u)$; (d) $kNNW_4(v, u) := w_{out}(v) + w_{out}(u)$; (e) $kNNW_5(v, u) := w_{in}(v) \cdot w_{in}(u)$; (f) $kNNW_6(v, u) := w_{in}(v) \cdot w_{out}(u)$; (g) $kNNW_7(v, u) := w_{out}(v) \cdot w_{in}(u)$; and (h) $kNNW_8(v, u) := w_{out}(v) \cdot w_{out}(u)$.

9. **Undirected kNN Weights** are defined similarly, but only for the neighbors. Let $v, u \in V$, where v edges' weight will be $w(v) := \frac{1}{\sqrt{1+|\Gamma(v)|}}$ and the weight of the connection between v and u will be measured by two permutations: (a) $kNNW_9(v, u) := w(v) + w(u)$ and (b) $kNNW_{10}(v, u) := w(v) \cdot w(u)$.

3.1.3 Classifier Construction

The created link classifier relies on the fact that most of the vertices in online social networks are real, and malicious users tend to connect to other profiles randomly [11, 13, 24]. Hence, a vertex that has many random connections has a higher likelihood of being fake. Similar to Fire et al. [26], we trained the

link classifier on the same number of negative examples and positive examples. The negative examples were selected randomly from all the existing edges in the graph, and positive examples were selected as non-existing edges between two random vertices. After obtaining a set of positive (non-existing edges) and negative (random edges) examples, we extracted for each entry all the features described in Sections 3.1.1 and 3.1.2. Finally, we used the Random Forest algorithm to construct the link prediction algorithm for our training sets. We chose the Random Forest algorithm since previous link prediction studies by Fire at el. [26] and Cukierski et al. [16] demonstrated that in most cases it performs better than other classification algorithms in predicting links.

> **input** : Graph G, Number of nodes to sample N, Node label $Label$,
> Minimal number of friends $MinFriends$
> **output**: Edges of Selected Vertices

1 SelectedEdges $\leftarrow$ Set();
2 while $N > 0$ **do**
3   RandomVertex $\leftarrow$ SampleNodes(G,1);
4   **if** RandomVertex $= Label$ and $|\Gamma($RandomVertex$)| > MinFriends$
     then
5   TempEdges $\leftarrow$ Set();
6   **for** E **do**
7   a
8   **end**
9   chNode u in $\Gamma($RandomVertex$)$ **if** $|\Gamma(u)| > MinFriends$ **then**
10   TempEdges $\leftarrow$ TempEdges $+ ($RandomVertex$, u)$;
11   **end**
12   **if** $|$TempEdges$| > MinFriends$ **then**
13   SelectedEdges $\leftarrow$ SelectedEdges $+$ TempEdges;
14   N $\leftarrow$ N $- 1$;
15   **end**
16   **end**
17 end
18 return SelectedEdges;

Algorithm 1: Sampling vertices from a graph.

3.2 Detecting Anomalous Vertices

After constructing a link prediction classifier for each graph, we utilized the classifier to build an unsupervised anomaly detection algorithm using the following steps. First, we created the test set by sampling the edges of random vertices from the graph. The sampling process works as described in Algorithm 1: The algorithm starts by selecting one random vertex, $RandomVertex$ (line 3). Next, we check if $RandomVertex$ has the desired label and if it has more neighbors than the minimal required amount, $MinFriends$ (line 4). Then,

we select all $RandomVertex$ neighbors that also have more than $MinFriends$ neighbors (lines 6-9). If $RandomVertex$ has more than $MinFriends$ neighbors that have more than $MinFriends$, then the selected edges are added to the test set (lines 11-14).

The goal of these constraints was to select only edges between nodes which were fully crawled and had neighbors in the graph. Vertices that have a small number of neighbors (less than three) are less relevant since there is not enough information to determine their behavior [24]. The algorithm continued to run until it added N vertices to the test set. In our experiments, we executed Algorithm 1 twice for each network, the first time to extract positive samples and the second time to extract negative samples. In both cases $MinFriends$ was equal to three.

Later, we created the link prediction classifier training set we described in Section 3.1, where negative examples were existing edges and positive examples were non-existing ones. The edge sampling for the link classifier worked as follows: Let $test$-$vertices$ be a set of all the vertices that were selected by Algorithm 1, and if $(u, v) \in E$ is an edge, then (u, v) can be part of the link classifier training set if and only if $u, v \notin test$-$vertices$. Then, we trained the link prediction classifier on the training set as we described in Section 3.1. Next, we used the trained classifier to calculate the probability of each edge in the test set to be anomalous. Finally, we aggregated all the edges by their source vertex and calculated the 9 new features we had defined.

We have defined the following features for identifying anomalous vertices: Let $p(v, u)$ be the probability of an edge to be existing, where $v, u \in V$, $(v, u) \in E$, and $EP(v) := \{p(v, u) | u \in \Gamma(v) \ and \ u, v \in V\}$ is the set of vertex v edge probabilities to exist.

1. **Abnormality Vertex Probability** is defined as the probability of a vertex v to be anomalous, which is equal to the average probability of its edges not existing. This corresponds with our definition of anomaly which we previously described.

$$P(v) := \frac{1}{|\Gamma(v)|} \sum_{u \in \Gamma(v)} p(v, u)$$

2. **Edges Probability Variance** is the variance of the vertex edges probability not existing. If we focus on online entities, high variance can indicate that at some point the vertex was compromised.

$$EdgesProbabilityVariance(v) := \sigma^2(EP(V))$$

3. **Edges Probability STDV** is the standard deviation of vertex v edges probability not existing.

$$EdgesProbabilitySTDV(v) := \sigma(EP(V))$$

4. **Edges Probability Median** is the median of vertex v edges probability not existing. The advantage of median over mean is that it is not as

sensitive to irregularly large or small values.

$$EdgesProbabilityMedian(v) := median(EP(V))$$

5. **Edged Count** is the number of edges that vertex v has. An extremely low value may indicate that the results for vertex v are statistically insignificant.

$$EdgedCount(v) := |\Gamma(v)|$$

6. **Sum Edge Label** is how many of vertex v edges were labeled as anomalous; in other words, this is the number of edges v with a p higher than a defined *threshold*, which in this work was defined as 0.8. The goal of this feature was to detect cases where vertices had many anomalous edges, but most of them were only a little above the *threshold*, resulting in a relatively low P.

$$SumEdgeLabel(v) := \sum\nolimits_{u \in \Gamma(v)} EdgeLabel(v, u)$$

where we define the function $EdgeLabel(v, u)$ as:

$$EdgeLabel(v, u) := \begin{cases} 0, & \text{if } p(v, u) > threshold \\ 1, & \text{otherwise} \end{cases}$$

7. **Predicted Label Variance** is the variance of v edges classification.

$$PredictedLabelVariance(v) := \sigma^2(\{EdgeLabel(v, u) | u \in \Gamma(v), (u, v) \in V\})$$

8. **Mean Predicted Link Label** is the percent of v edges that were labeled as anomalous.

$$MeanPredictedLinkLabel(v) := \frac{1}{|\Gamma(v)|} \sum\nolimits_{u \in \Gamma(v)} EdgeLabel(v, u)$$

9. **Predicted Label STDV** is the standard deviation of v edges classification.

$$PredictedLabelSTDV(v) := \sigma(\{EdgeLabel(v, u) | u \in \Gamma(v), (u, v) \in V\})$$

We used the described features in two ways. The first usage scenario was with data that did not have any labels. In this case we ranked all the vertices by the different features and manually examined the top and bottom vertices, which had the highest and lowest likelihood of being anomalous. The second scenario was when the data was labeled or partially labeled. In such cases we performed additional classification using the Random Forest algorithm on the data and created a meta classifier.

3.3 Anomalous Vertices Simulation

Currently, there is a very limited number of publicly available datasets with known anomalies, and manual labeling is a challenging task [4]. To deal with these issues and evaluate the proposed anomaly detection algorithm on various types of networks, we used simulated anomalous vertices (see Algorithm 2) for different scenarios. Similar to previous studies [11, 13, 24], we generated anomalous vertices by connecting them randomly to other vertices in the network in the following way: First, we inserted a new simulated vertex into the graph (line 2). Next, we generated $NeighborsNumber$, the number of edges that would be created for the simulated vertex (line 3). $NeighborsNumber$ is the number of neighbors of a vertex that was sampled from the graph such that a vertex with more neighbors had a higher likelihood of being chosen. Then, we sampled random $NeighborsNumber$ vertices from the graph (line 4). Afterwards, we connected the newly inserted vertex to the sampled random vertices (lines 5-7). The number of anomalous vertices in each graph was set to 10%, which represents an estimation of the percentage of fake vertices in an average social network [1, 2].

> **input** : Graph G, having simulated vertex number N
> **output**: Graph G with N simulated vertices

1 **for** $i \leftarrow 1$ **to** N **do**
2 $\quad$ SimulatedVerticesNumber $\leftarrow$ AddVertex($G,i,Fake$);
3 $\quad$ NeighborsNumber $\leftarrow |\Gamma($Random$(G.edges)[0])$—;
4 $\quad$ RandomVertices $\leftarrow$ SampleVertices(G,NeighborsNumber);
5 $\quad$ **foreach** $Vertex\ u\ in$ RandomVertices **do**
6 $\quad\quad$ AddEdge(v,u);
7 $\quad$ **end**
8 **end**

Algorithm 2: Adding anomalous vertices to a graph.

3.4 Evaluation

We evaluated our algorithm on three main cases and on ten networks. The first network type was a fully simulated complex network. We generated the graph using the Barabási-Albert model [8], believing it should give a good indication of the performance of the method on various types of complex networks. The generated networks were constructed according to the number of vertices and the average number of edges of a real world network to make them as close as possible to actual networks. First, we generated BA networks that were 90% of the size of the real network. Afterwards, we inserted anomalous vertices for the remaining 10% (see Algorithm 2). The second network type was a semi-simulated network, that is, a real world network with injected simulated anomalous vertices (see Algorithm 2). The third network type we tested our method on was a real world network with labeled anomalous vertices.

Due to hardware limitations, for each network we sampled a test set that contained 900 random existing vertices and 100 anomalous vertices. The test set kept the same 1:10 ratio between anomalous and normal vertices. To reduce the variance of the results, we ran the algorithm 10 times on each network. The more evaluations we performed, the smaller the variance in the results. We evaluated the algorithm on the average result of these experiments. To measure the algorithm performance, we used 10-fold cross validation to measure all the evaluations' true positive rate (TPR), false positive rate (FPR), precision, and AUC. In addition, we measured the algorithm's precision at k (precision@k) for k := 10, 100, 200, and 500.

3.5   Social Network Datasets

We evaluated our algorithm on seven different datasets of various scales, ranging from a graph with 53 vertices to a graph with over 5.3 million vertices (see Table 1):

Table 1: Social Network Datasets

Network	Is Directed	Vertices Number	Links Number	Date	Labeled
Academia	Yes	200,169	1,389,063	2011	No
ArXiv HEP-PH	No	34,546	421,578	2003	No
CLASS OF 1880/81	Yes	53	179	1881	Yes
DBLP	No	1,665,850	13,504,952	2016	No
Flixster	No	672,827	1,099,003	2012	No
Twitter	Yes	5,384,160	16,011,443	2012	Yes
Yelp	No	249,443	3,563,818	2016	No

1. *Academia.edu*[3] is a social platform for academics to share and follow research, and to follow other researchers' work. Using our dedicated crawler, we crawled most of the Academia.edu graph during 2011 (see Section 7).
2. *ArXiv*[4] is an e-print service in fields such as physics and computer science. We used the ArXiv HEP-PH (high energy physics phenomenology) citation graph that was released as part of the 2003 KDD Cup.[5]
3. *Boys' Friendship (CLASS OF 1880/81)* is a dataset which contains the friendship network of a German school class from 1880-81 that was assembled by the class's primary school teacher, Johannes Delitsch. The dataset

---

[3] https://www.academia.edu   [4] https://www.arxiv.com
[5] https://snap.stanford.edu/data/cit-HepPh.html

12

itself was generated from observing students, interviewing pupils and parents, and analyzing school essays [27]. Delitsch found that there were 13 outliers out of 53 students, which Heidler et al. defined as students who did not fit perfectly into their predicted position within the network structure. The data contains three types of outliers: "repeaters," who were four students who often led the games; "sweets giver," a student who bought his peers' friendship with candies; and a specific group of seven students who were psychologically or physically handicapped, or socio-economically deprived. This is probably the first-ever primarily collected social network dataset [27].[6]

4. *DBLP*[7] is the online reference for bibliographic information on major computer science publications. We used a version of the DBLP dataset to build a co-authorship graph where two authors are connected if they published a publication together.[8]

5. *Flixster*[9] is a social movie site which allows users to share movie reviews, discover new movies, and communicate with others. We collected the data using a dedicated crawler during 2012 (see Section 7).

6. *Twitter*[10] is an undirected online social network where people publish short messages and updates. Currently, Twitter has 310 million monthly active users.[11] According to recent reports, Twitter has a bot infestation problem [2, 17]. We used a dedicated API crawler to obtain our dataset in 2014.[12]

7. *Yelp*[13] is a web platform to help people find local businesses. In addition to finding local business and writing reviews, Yelp allows its users to discover events, make lists, and talk with other Yelpers. In 2016 Yelp published several big datasets as part of the Yelp Dataset Challenge; one is a social network of its users.[14]

3.5.1 Dataset Evaluation

First, we evaluated the presented algorithm on the *fully simulated networks*. Generating complex networks using the BA model requires two parameters: the number of vertices to be generated and the number of edges to be created for each vertex. To create networks that represent real networks as closely as possible, we used the number of vertices and the average number of edges of the DBLP, ArXiv, and Yelp datasets (see Table 1) to generate the simulated networks.

Next, we evaluated our algorithm on *semi-simulated networks*. The evaluation was conducted on the graphs of ArXiv HEP-PH, DBLP, Flixster, and Yelp (see Table 1) with inserted anomalies.

Then, we tested the proposed algorithm on *networks with labeled anomalies*. We evaluated the graphs of Twitter and the Boys' Friendship network (see Table 1). The Twitter data, by default, did not have any labels. To create labels that can be considered ground truth, we crawled all the profiles (without the edges) in the Twitter dataset about one year after the initial crawling. Similar to Kurt et al. [35], we considered all the accounts that Twitter operators decided to block as a ground truth. When Twitter labels an account as malicious, it is suspended and an appropriate message is presented. Twitter defines a suspended account as one that violated Twitter's terms of service,[15] and the most common reasons for suspension are spam, the account being hacked or compromised, and abusive tweets or behavior.[16] As a result, we labeled all the accounts which were suspended as malicious, and we considered them to be anomalous vertices in the graph. In addition, we filtered all the verified accounts from the dataset. A verified account is an account of public interest, primarily those of celebrities, politicians, etc.[16] We filtered them since most of their connections do not represent regular users, and many times they are managed by some kind of third party [15].

In the Boys' Friendship network, we observed that all the psychologically or physically handicapped students, except one, did not have neighbors in the graph. This left us with three groups of outliers: the four "repeaters" ; the single "sweets giver"; and two pupils who were socio-economically deprived. Due to the small scale of the dataset, running the anomaly detection algorithm with a small number of repetitions can result in high variance rates. Therefore, to reduce the variance we ran our method 100 times on the dataset and calculated the average of the features presented in Section 3.2. In addition, every execution we tested contained 10 vertices, not 1000 as we described in Section 3.4.

4 Results

We evaluated our topology-based anomaly detection method on three types of complex networks. First, we evaluated the method on fully simulated networks with simulated anomalous vertices using a 10-fold cross validation. We can see in Table 2 that for the three simulations we performed, we obtained high AUCs and low FPRs. Second, we evaluated the proposed method on semi-simulated graphs, i.e., real world networks with injected anomalous vertices. We can see that the algorithm generated especially good results, giving an average AUC of 0.99 and FPR of 0.021 (see Table 3). Third, we evaluated our algorithm on labeled real world data. The first real world dataset was Twitter. The results showing the classifier precision at k average value are presented in Figure 3. We can see that the precision at 10, 50, 100, 200, and 500 was 0.6, 0.4, 0.35, 0.26, and 0.142, respectively. The second real world dataset was the Boys' Friendship network. We found that six out of the seven students (precision@7=0.875) with the lowest Mean Predicted Link Label were the ones that Heidler et al. [27]

---

[15] https://support.twitter.com/articles/18311 [16] https://support.twitter.com/articles/15790

Table 2: Machine Learning Results on Fully Simulated Networks with Simulated Anomalous Nodes

Simulation	AUC	TPR	FPR	Precision
Arxiv HEP-PH	0.991	0.889	0.011	0.904
DBLP	0.997	0.935	0.006	0.993
Yelp	0.993	0.917	0.007	0.937

Table 3: Machine Learning Results on Real World Networks with Simulated Anomalous Nodes

	AUC	TPR	FPR	Precision
Academia	0.999	0.998	$2.51 \cdot 10^{-4}$	0.997
Arxiv HEP-PH	0.997	0.953	0.004	0.965
DBLP	0.997	0.940	0.005	0.995
Flixster	0.992	0.908	0.010	0.990
Yelp	0.996	0.941	0.005	0.958

referred to as either the "repeaters" or the socio-economically deprived, and defined them as outliers (see Figure 1). Evaluating the algorithm using 10-fold cross validation and the Random Forest algorithm, where the "repeaters" and socio-economically deprived students were labeled as a positive class, resulted in an AUC of 0.931, TPR of 0.91, and FP of 0.15. To determine which of the

Table 4: InfoGain Values of Different Features for Real World Networks with Simulated Anomalous Nodes

	Abnormality Vertex Probability	Probability Median	Sum Link Label	Mean Predicted Link Label	Probability Variance	Probability STDV	Predicted Label STDV	Predicted Label Variance	Edged Count
Academia	0.47	0.47	0.37	0.39	0.29	0.29	0.1	0.1	0
Arxiv HEP-PH	0.11	0.1	0.1	0.11	0.02	0.02	0.01	0.01	0.01
DBLP	0.34	0.23	0.33	0.32	0.23	0.23	0.15	0.15	0.04
Flixster	0.21	0.19	0.2	0.21	0.05	0.05	0.01	0.01	0.05
Yelp	0.18	0.14	0.27	0.17	0.25	0.25	0.06	0.06	0.06
Mean	0.34	0.31	0.33	0.32	0.2	0.2	0.05	0.05	0.03
STDV	0.18	0.19	0.15	0.16	0.12	0.12	0.05	0.05	0.02

new features we proposed in Section 3.2 have more influence, we analyzed their importance using Weka's information gain attribute selection algorithm. From the results in Tables 4 and Table 5 we can see that the three most influential
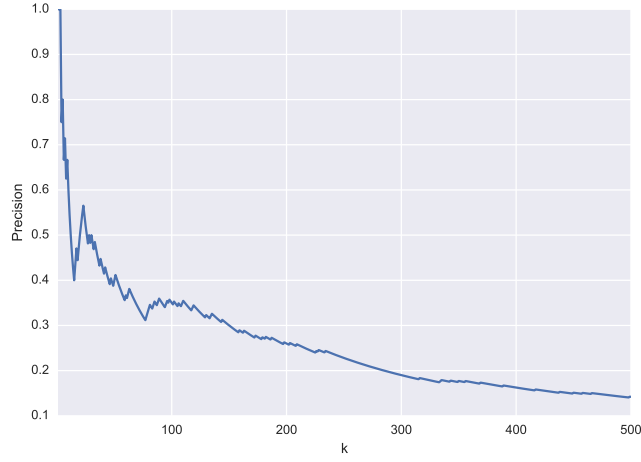
Figure 3: Twitter precision at k results.

Table 5: InfoGain Values of Different Features for Fully Simulated Networks

	Abnormality Vertex Probability	Probability STDV	Probability Variance	Probability Median	Mean Predicted Link Label	Sum Link Label	Edged Count	Predicted Label Variance	Predicted Label STDV
Arxiv HEP-PH	0.18	0.216	0.216	0.092	0	0	0.021	0	0
DBLP	0.165	0.082	0.082	0.146	0.148	0.124	0.036	0.072	0.072
Yelp	0.187	0.223	0.223	0.105	0	0	0.027	0	0
Mean	0.18	0.17	0.17	0.11	0.05	0.04	0.03	0.02	0.02
STDV	0.01	0.08	0.08	0.03	0.09	0.07	0.01	0.04	0.04

features are Abnormality Vertex Probability, Probability Median, and Mean Predicted Link Label.

5   Discussion

Upon analyzing the results presented in Section 4, we can conclude that the proposed anomaly detection algorithm fits well in the network security domain. Our results demonstrate very low false positive rates, on average 0.006 in all the tested scenarios. In the security domain, false positive is one of the most important metrics; many social network operators prefer to miss a true positive rather than have a relatively high false positive rate [13]. An online operator certainly wants to avoid false positives to ensure that legitimate users are not blocked.

In the fully simulated network cases, we can see that the simulation results

were correlated with the size of the networks. We can observe in Table 1 and 2 that we obtained better results for the larger datasets. Specifically, the simulated network that was based on DBLP characteristics was the largest and had the best TPR, while the ArXiv-based simulation was the smallest and had the lowest TPR. From these results, we can clearly see that our algorithm can detect vertices which connect randomly to other vertices in the network, assuming the BA model generates networks that represent real world networks.

In the Twitter case, we strongly believe there are substantial numbers of malicious accounts that Twitter operators have not discovered [17]. These undiscovered malicious users translate into high rates of false positives. From manually sampling the false positives, we discovered that many of these profiles are inactive and their tweets look like generated commercial content, whereas other profiles mostly retweeted other profiles' content. Because of the many unsuspended malicious accounts in Twitter [17], we believe our method would perform better on a fully labeled dataset. Yet even with these issues, we think Twitter is a good indicator of how well our method performs on real world data. According to the Twitter results, we were able to detect fake Twitter profiles with precision at 100 of 35%; that is considerably better than a random algorithm, which results in about 6.4% precision in this case.

The Boys' Friendship network results confirmed the work of Heidler et al. [27]. They described the "repeaters" as pupils who often led at games and were strong, lively, and energetic, especially outside of the classroom. They also mentioned that socio-economic status exhibited a strong influence on popularity. Heidler et al. [27] verified in their work that the four "repeaters" and the "sweets giver" had a disproportionately high popularity. Our results show that the four "repeaters" had the strongest friendship ties of all the other pupils, which aligns with the findings of Heidler et al. The "sweets giver," who also had high popularity, was ranked only in the middle, which also is acceptable since some boys who looked like his friends only wanted candies, not friendship.

Lastly, according to the overall results, we can clearly state that our method can detect malicious profiles that act according to random strategy. We suspect, however, that the method would be less effective on malicious users that have specific targets and strategies; for instance, the bots we developed in our previous works targeted specific organizations' employees [19, 20, 21]. We also found that it is more challenging to detect malicious users on networks like Twitter, where most of the users have some randomness in their behavior. Such properties are more common in undirected networks where a user can follow anyone, without the need for the other side's consent.

Our results also indicate that the presented method can be utilized outside of the security domain. For instance, in a friendship graph, a vertex that has many edges with high probabilities of existing is a marker of a central person in the examined social group. Moreover, we believe that the presented method could be utilized to detect hijacked profiles if the hijacker starts to connect randomly to other vertices in the network.

6   Conclusions

The ability to detect anomalies has become increasingly important in understanding complex networks. This study presents a novel generic method for detecting anomalous vertices based on features extracted from the network topology. The presented method combines cutting-edge techniques in link prediction, graph theory, and machine learning.

We evaluated our anomaly detection method on ten networks that can be categorized into three scenarios. First, we used our method on three BA model-generated graphs with injected anomalies. Next, we applied it to five semi-simulated graphs, which are real world networks with injected anomalies. Finally, we tested our method on two real world networks with real outliers.

Overall, the evaluation results demonstrate that our anomaly detection model performed well in terms of AUC measures. We demonstrated that in a real-life friendship graph, we can detect people who have the strongest friendship ties. Moreover, we showed that our algorithm can be utilized to detect malicious users on Twitter. We believe that the presented method has considerable potential for a wide range of applications, particularly in the cyber-security domain. In many cyber-security scenarios, there is a high cost for mistakes such as blocking a benign user from the network. Our algorithm can detect malicious users, and our results present a low FPR (average FPR of 0.006), which is ideal for such use cases. Additionally, we demonstrated that the presented method can be applied to networks of different sizes, types, and domains. We also determined that the importance of different features varied over different networks.

We plan to continue to develop and expand the presented method into new directions. First, we plan to test the algorithm on additional domains, such as computer networks, emails, biological networks, etc. Next, we want to develop a version of the algorithm for other types of networks, such as bipartite and weighted graphs. Furthermore, we plan to show that the presented method can be utilized for the detection of hijacked accounts in online networks. In each of these future directions, much can be gained from understanding anomalies in complex networks and their structures.

7   Availability

This study is reproducible research. Therefore, the anonymous versions of the social network datasets and the study's code, including implementation, are available at the project's website and repository.

8   Acknowledgments

References

[1] Facebooks annual report 2015. https://s21.q4cdn.com/399680738/files/doc_financials/annual_reports/2015-Annual-Report.pdf. (Accessed on 10/16/2016).

[2] Lisa Vaas. Good bot, bad bot? 23 million twitter accounts are automated. https://nakedsecurity.sophos.com/2014/08/14/good-bot-bad-bot-23-million-twitter-accounts-are-automated/, August 2014. (Accessed on 10/16/2016).

[3] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining*, pages 410–421. Springer, 2010.

[4] 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.

[5] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM'06: Workshop on Link Analysis, Counterterrorism and Security*, 2006.

[6] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[7] J. Balthrop, S. Forrest, M. E. Newman, and M. M. Williamson. Technological networks and the spread of computer viruses. *Science*, 304(5670):527–529, 2004.

[8] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[9] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.

[10] R. J. Bolton and D. J. Hand. Statistical fraud detection: A review. *Statistical science*, pages 235–249, 2002.

[11] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. The socialbot network: when bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 93–102. ACM, 2011.

[12] S. Brin and L. Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

[13] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 15–15, Berkeley, CA, USA, 2012. USENIX Association.

[14] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[15] Chris Plante. That's not a celebrity you're following on twitter, it's an assistant. http://www.theverge.com/2014/9/8/6121985/celebrity-twitter-adam-levine, September 2014. (Accessed on 10/16/2016).

[16] W. Cukierski, B. Hamner, and B. Yang. Graph-based features for supervised link prediction. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1237–1244. IEEE, 2011.

[17] Daniela Hernandez. Why can't twitter kill its bots? http://fusion.net/story/195901/twitter-bots-spam-detection/, September 2015. (Accessed on 10/16/2016).

[18] W. Eberle and L. Holder. Anomaly detection in data represented as graphs. *Intelligent Data Analysis*, 11(6):663–689, 2007.

[19] A. Elishar, M. Fire, D. Kagan, and Y. Elovici. Organizational intrusion: Organization mining using socialbots. In *Social Informatics (SocialInformatics), 2012 International Conference on*, pages 7–12. IEEE, 2012.

[20] A. Elyashar, M. Fire, D. Kagan, and Y. Elovici. Homing socialbots: intrusion on a specific organization's employee using socialbots. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1358–1365. ACM, 2013.

[21] A. Elyashar, M. Fire, D. Kagan, and Y. Elovici. Guided socialbots: Infiltrating the social networks of specific organizations' employees. *AI Communications*, 29(1):87–106, 2014.

[22] T. Fawcett and F. Provost. Adaptive fraud detection. *Data mining and knowledge discovery*, 1(3):291–316, 1997.

[23] M. Fire and C. Guestrin. Analyzing complex network user arrival patterns and their effect on network topologies. *arXiv preprint arXiv:1603.07445*, 2016.

[24] M. Fire, G. Katz, and Y. Elovici. Strangers intrusion detection-detecting spammers and fake profiles in social networks based on topology anomalies. *Human Journal*, 1(1):26–39, 2012.

[25] M. Fire, L. Tenenboim, O. Lesser, R. Puzis, L. Rokach, and Y. Elovici. Link prediction in social networks using computationally efficient topological features. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 73–80. IEEE, 2011.

[26] M. Fire, L. Tenenboim-Chekina, R. Puzis, O. Lesser, L. Rokach, and Y. Elovici. Computationally efficient link prediction in a variety of social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):10, 2013.

[27] R. Heidler, M. Gamper, A. Herz, and F. Eßer. Relationship patterns in the 19th century: The friendship network in a german boys' school class from 1880 to 1881 revisited. *Social Networks*, 37:1–13, 2014.

[28] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of computer security*, 6(3):151–180, 1998.

[29] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[30] M. E. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

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

[32] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications*, 1(1):19–30, 2010.

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

[34] S. H. Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276, 2001.

[35] K. Thomas, C. Grier, D. Song, and V. Paxson. Suspended accounts in retrospect: an analysis of twitter spam. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 243–258. ACM, 2011.

[36] X. F. Wang and G. Chen. Complex networks: small-world, scale-free and beyond. *Circuits and Systems Magazine, IEEE*, 3(1):6–20, 2003.