Social Network Analysis and Mining (2018) 8:27
https://doi.org/10.1007/s13278-018-0503-4

ORIGINAL ARTICLE

Generic anomalous vertices detection utilizing a link prediction algorithm

Dima Kagan[1] · Yuval Elovichi[1] · Michael Fire[2]

Received: 1 November 2017 / Revised: 18 February 2018 / Accepted: 21 March 2018 / Published online: 5 April 2018
© Springer-Verlag GmbH Austria, part of Springer Nature 2018

Abstract
In the past decade, graph-based structures have penetrated nearly every aspect of our lives. The detection of anomalies in these networks has become increasingly important, such as in exposing infected endpoints in computer networks or identifying socialbots. In this study, we present a novel unsupervised two-layered meta-classifier that can detect irregular vertices in complex networks solely by utilizing topology-based features. Following the reasoning that a vertex with many improbable links has a higher likelihood of being anomalous, we applied our method on 10 networks of various scales, from a network of several dozen students to online networks with millions of vertices. In every scenario, we succeeded in identifying anomalous vertices with lower false positive rates and higher AUCs compared to other prevalent methods. Moreover, we demonstrated that the presented algorithm is generic, and efficient both in revealing fake users and in disclosing the influential people in social networks.

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, or even billions of vertices and edges (Albert and Barabási 2002; Boccaletti et al. 2006). These systems can be found in every part of our daily life (Strogatz 2001; Fire and Guestrin 2016), such as electrical power grids, metabolic networks, food webs, the Internet, and co-authorship networks (Strogatz 2001; Newman 2003). Analyzing the unique structures of these networks can be very useful in a variety of research domains. For example, an analysis of network structures can reveal how a computer virus will propagate most quickly in a computer network (Balthrop et al. 2004), or which vertex malfunction in a power grid will affect more houses (Wang and Chen 2003).

Many studies have shown that vertices which deviate from normal behavior may hide important insights (Bolton and Hand 2002; Noble and Cook 2003; Akoglu et al. 2010; Papadimitriou et al. 2010; Fire et al. 2012). For instance, Bolton and Hand (2002) showed that in e-commerce fraudsters behave differently from the expected norm. Fire et al. (2012) 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. Hooi et al. (2016) noted that fraudsters tend to create unusually large and dense regions in the adjacency matrix of the graph. Noble and Cook (2003) 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 (Akoglu et al. 2015). These studies have utilized various graph features, such as vertices, dyads, triads, and communities, to detect the fake profiles' behavioral patterns. For example, in the case of a social graph, anomalous vertices can represent malicious or fake profiles. Because illegitimate profiles do not represent a real person and they do not have real friends and connections in the social network, 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,

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s13278-018-0503-4) contains supplementary material, which is available to authorized users.

✉ Dima Kagan
kagandi@post.bgu.ac.il

Yuval Elovichi
elovici@post.bgu.ac.il

Michael Fire
fire@cs.washington.edu

[1] Ben-Gurion University of the Negev, Beersheba, Israel

[2] University of Washington, Seattle, USA

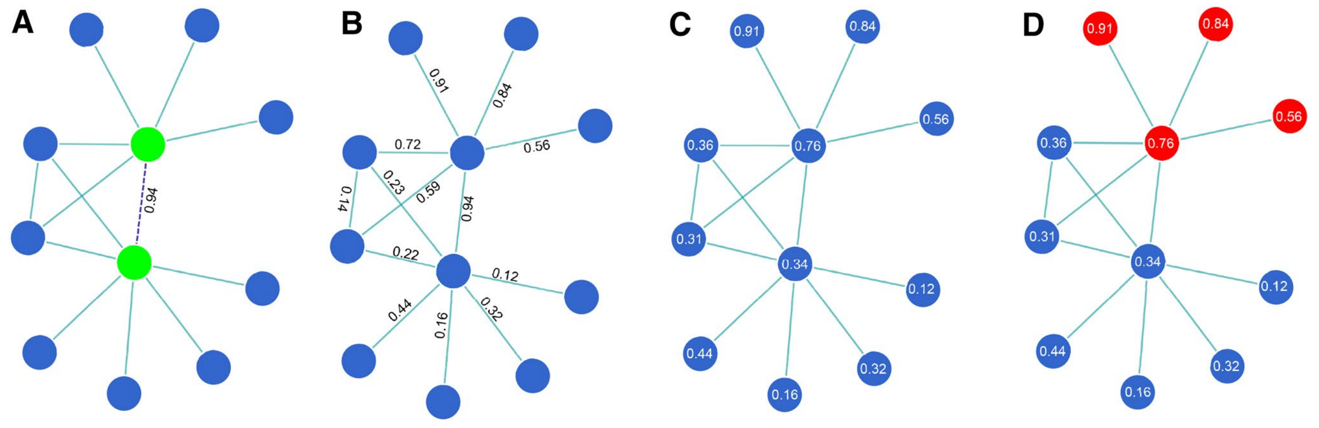

Fig. 1 Algorithm overview. **a** The link prediction classifier is trained to calculate the probability that an edge does not exist in the graph. For example, the classifier can predict that the probability of an edge between the two green vertices not existing is 94%. **b** We utilize the link prediction classifier to predict the probability of each edge not existing. **c** We calculate for each vertex the average probability that the vertex edges do not exist. **d** Vertices that have the highest average probability (the red vertices) are inspected (color figure online)

utilizing a graph's topology (see Fig. 1). The algorithm consists of two main iterations. In the first iteration, we create a link prediction classifier which is 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 Sect. 3.2). In the second iteration, we generate a new set of meta-features based on the features created by the link prediction classifier (see Sect. 3.3). Then, we utilize these meta-features and construct an anomaly detection classifier. Intuitively, the algorithm is based on the assumption that a vertex that has many edges with low probabilities of existing (improbable edges) 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. Our study results indicate that the proposed algorithm can successfully detect malicious users in complex networks in general, and more specifically, in online social networks. Moreover, we showed that this algorithm may be applicable as a generic anomaly detection algorithm in additional domains. The principal contributions of this study are as follows:

1. We successfully incorporate a link prediction technique into an anomaly detection model, which requires almost no prior knowledge of the graph (see Sect. 3.2.2).
2. We propose seven new features which are found to be good predictors for anomaly detection (see Sect. 3.3).
3. We conduct an extensive experimental evaluation of the proposed methods on three types of data: three fully synthetic datasets, five semi-synthetic datasets, and two real datasets (see Sect. 5).
4. The results demonstrate that our algorithm can detect anomalies in networks of different types and sizes and that it also performs better than other methods we tested. In addition, the proposed method performs well with incomplete data in contrast to methods that utilize features such as community structure (see Sect. 6).
5. This study is reproducible; we published all of its code and data online, including the real-world datasets containing labeled fake profiles. This can be used as an open framework to help future vertex anomaly detection algorithms compare their results (see Sect. 9).

The remainder of this paper is organized as follows. In Sect. 2, we present an overview of relevant studies. In Sect. 3, we describe how we constructed our algorithm. We introduce the datasets we have utilized to evaluate the method in Sect. 4. Section 5 provides a description of the experiments we preformed to evaluate the presented method. In Sect. 6, we present our results. Section 7 contains a discussion about the results obtained, and lastly, in Sect. 8, we present our conclusions.

2 Literature overview

The detection of anomalies is an extremely useful ability in many domains 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. (1998) used an anomaly detection method in order to spot intrusions in UNIX systems. Another example is the work of Fawcett and Provost (1997), who developed a fraud detection method by profiling users' behavior and detecting deviations. Anomaly detection is widely used in diverse fields such as the medical area, sensor networks. (Chandola et al. 2009).

The analysis of graphical data has grown in popularity (Eberle and Holder 2007) over the past two decades, and this has been accompanied by increasing amounts of research on anomaly detection in complex networks. However, many insights remain to be discovered, particularly in the structure-based method subgenre of anomaly detection. One of the first studies that combined complex networks and anomaly detection was conducted by Noble and Cook (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. Noble and Cook's second method was to divide the graph into subgraphs and rank them by an anomaly measure, which they defined for each subgraph.

Sun et al. (2005) proposed a method for detecting abnormal vertices in bipartite graphs. They calculated normality scores (based on the neighborhood relevance score), where vertices with a lower normality score had a higher likelihood of being anomalous. Eberle and Holder (2007) proposed detecting fraud by discovering modifications, insertions, and deletions in graphs. Unlike Noble and Cook, Eberle and Holder looked for substructures that, while similar to normative substructures, are still different. The idea behind this approach is that fraudsters may try to masquerade as legitimate entities, which makes them look very similar to legitimate users.

Papadimitriou et al. (2010) presented a method that can identify anomalies in a web graph that may occur as a result of malfunctions. They proposed identifying outliers by comparing the similarity scores of two consecutive graphs against some threshold. Akoglu et al. (2010) proposed a feature-based method to spot strange vertices in weighted graphs. In order to detect anomalies, they defined a score that measures "distance to fitting line" that uses several features that they proposed. They considered vertices with the highest scores as outliers. Fire et al. (2012) proposed Stranger, a method for detecting fake profiles in online social networks based on anomalies in a fake user's social structure. They trained their classifier by simulating a fake profile that randomly sends friendship requests to other users in the network. Recently, Hooi et al. (2016) presented FRAUDAR, a method for detecting "camouflaged" malicious accounts. FRAUDAR utilizes density-based metrics in order to detect malicious accounts in bipartite networks.

In this work, we rely on a link prediction algorithm which is central to our anomaly detection method. Link prediction is defined as the discovery of hidden or future links in a given social network. The link prediction problem was first introduced by Liben-Nowell and Kleinberg (2007) 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 was provided by Kaggle. Cukierski et al. (2011) proposed a method that is based on supervised machine learning and uses 94 different features. They discovered that a Random Forest classifier performed best out of all of the supervised machine learning methods evaluated. In addition, they found that EdgeRank (rooted PageRank) (Brin and Page 2012) was the highest scoring feature out of the 94 features used.

Fire et al. (2011) 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 a smaller subset of features can be used to obtain results with relatively high AUCs (Fire et al. 2011). Later, Fire et al. demonstrated that in many cases the benefits of using a large number of features is insignificant, and that by only using computationally efficient features, it is possible to get highly accurate classifications (Fire et al. 2013).

3 Methods

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 conducted in the past several years indicate that many malicious users present different behavioral patterns than benign users (Boshmaf et al. 2011; Cao et al. 2012; Fire et al. 2012; Ferrara et al. 2016). Boshmaf et al. (2011) described how fake profiles connect randomly to other users in order to establish an influential position or fame. Stringhini et al. (2010) noticed that many spammers on Facebook choose to connect to other users according to their victims' names. Moreover, Fire et al. (2012) described how fake profiles will likely connect to many communities. For example, with fake profiles there is increased likelihood that neighbors will not have any mutual characteristics (e.g., the same workplace, language). Ferrara et al. (2016) noted that social bots on Twitter tend to grow their social circles following random accounts.

Motivated by the difference between the observed behavioral patterns of fake profiles and benign users, 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

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

ones. Then, for each of these edges, we extracted features that were based on the network's topology (see Sect. 3.2), and we used the feature set to train a link prediction classifier (see Fig. 1). Next, we aggregated the results from the link classifier for each vertex and created an additional set of features (see Sect. 3.3). We then extracted the second set of features and used them to build a meta-classifier that identifies anomalous vertices in the graph.

3.1 Problem definition

Generally, an anomaly or outlier defined as "an observation that differs so much from other observations as to arouse suspicion that it was generated by a different mechanism" (Akoglu et al. 2015). Moreover, Noble and Cook (2003) stated in their work, "it remains difficult to give a general, formal definition of what an anomaly is," and they considered an anomaly as "a surprising or unusual occurrence." The definition of anomaly is very general and there are many domain-specific definitions. In this study, we see anomalous vertices simply as vertices which deviate from the normal behavior. More specifically, we see an anomalous vertex as a vertex with edges that deviate from the normal behavior. Formally, we define the problem, given a graph $G = <V, E>$, a vertex $v \in V$ will be flagged as an anomaly/outlier if $score(v) > threshold$.

3.2 Constructing a link prediction classifier

As described in Sect. 2, in the last decade, researchers have proposed various methods for predicting links in graphs (Al Hasan et al. 2006; Liben-Nowell and Kleinberg 2007; Cukierski et al. 2011; Brin and Page 2012; Fire et al. 2013). Moreover, researchers have demonstrated that link prediction classifiers can predict links with a high level of precision on a wide range of complex networks (Fire et al. 2013). In this study, we constructed a topology-based link prediction classifier based on the works of Cukierski et al. (2011) and Fire et al. (2013). We extracted 19 different features,[2] 16 of which are used for directed graphs (all of the edge-based features except for *Transitive Friends* and *Adamic-Adar Index* are used for both directed and undirected graphs) and eight are used for undirected graphs. Prior to describing how the features were used, we provide the following definitions. Let $G := (V, E)$ be a graph where V is a set of the graph's vertices and E is the set of the graph's edges. Let $v \in V$; then $\Gamma(v)$ 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.

3.2.1 Feature extraction

- *Total Friends* is the number of distinct friends between two vertices v and u.

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

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

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

  For a directed graph, we define three variations of Common Friends:

  $$Common\,Friends_{in}(v, u) := |\Gamma_{in}(v) \cap \Gamma_{in}(u)|,$$
  $$Common\,Friends_{out}(v, u) := |\Gamma_{out}(v) \cap \Gamma_{out}(u)|, \text{ and}$$
  $$Common\,Friends_{bi}(v, u) := |\Gamma_{bi}(v) \cap \Gamma_{bi}(u)|.$$

- *Jaccard's Coefficient* measures similarity between two groups of items (Cukierski et al. 2011; Fire et al. 2013; Liben-Nowell and Kleinberg 2007).

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

- *Preferential Attachment Score* is based on the idea that the rich get richer in social networks (Liben-Nowell and Kleinberg 2007).

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

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

  $$Transitive\,Friends(v, u) := |\Gamma_{out}(v)| \cap |\Gamma_{in}(u)|$$

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

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

- *Adamic-Adar Index* is a similarity measure for undirected graphs which measures how strongly two vertices are related (Liben-Nowell and Kleinberg 2007) .

  $$Adamic\text{-}Adar\,Index := \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log |\Gamma(w)|}$$

The kNN weight[3] features are general neighborhood and similarity-based features (Cukierski et al. 2011). They are based on the principle that as the number of friends goes up, the value of each individual friend decreases.

[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 (Fire et al. 2013).

[3] This not the standard kNN acronym, but a set of weight functions defined by Cukierski et al. (2011).

- *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 combinations 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) $kNNW_8(v,u) := w_{out}(v) \cdot w_{out}(u)$.
- *Undirected kNN Weights* are defined similarly, but only for the neighbors. Let $v, u \in V$, where the weight of v edges 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 combinations: (a) $kNNW_9(v,u) := w(v) + w(u)$ and (b) $kNNW_{10}(v,u) := w(v) \cdot w(u)$.

3.2.2 Classifier construction

Similar to Fire et al. (2013), we trained the link classifier on the same number of negative and positive examples, which were existing edges and non-existing edges, respectively. The negative examples represent real users and were selected randomly (degree distribution) from all of the existing edges in the graph. The positive examples were selected as non-existing edges between two random vertices which were sampled uniformly to represent the edges of a malicious user a in social network. After obtaining a set of positive (non-existing edges) and negative (random edges) examples, for each entry, we extracted all of the features described in Sect. 3.2.1. Finally, we used the Random Forest algorithm to construct the link prediction algorithm for our training sets. We chose the Random Forest algorithm because previous link prediction studies (Cukierski et al. 2011; Fire et al. 2013) demonstrated that, in most cases, it performs better than other classification algorithms at predicting links. Our 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 (Boshmaf et al. 2011; Cao et al. 2012; Fire et al. 2012).

```
input : Graph $G$, Number of vertices to sample $N$, Node label $Label$, Minimal
        number of friends $MinFriends$
output: Edges of Selected Vertices
SelectedEdges ← Set();
while $N > 0$ do
    RandomVertex ← SampleNodes($G$,1);
    if RandomVertex = $Label$ and $|\Gamma$(RandomVertex)$| > MinFriends$ then
        TempEdges ← Set();
        foreach $Node\ u\ in\ \Gamma$(RandomVertex) do
            if $|\Gamma(u)| > MinFriends$ then
                TempEdges ← TempEdges + (RandomVertex, $u$);
            end
        end
        if |TempEdges| $> MinFriends$ then
            SelectedEdges ← SelectedEdges + TempEdges;
            N ← N − 1;
        end
    end
end
return SelectedEdges;
```

Algorithm 1: Sampling vertices from a graph.

3.3 Detecting anomalous vertices

After constructing a link prediction classifier for each graph, we utilized the classifier to build an unsupervised anomaly detection algorithm. We used the trained classifier to calculate (for all the edges of the inspected vertices) the classifier's confidence that an edge does not exist. Using this metric, we calculated seven features that we used to identify anomalies.

3.3.1 Anomaly detection features

The intuition behind the anomaly detection features is that these features are fast to compute and easy to understand. Intuitively, the Abnormality Vertex Probability feature reflects the problem definition (see Sect. 3.1). The other features are designed to help in detecting special cases. For instance, STDV-based features can indicate vertices that drastically change their behavior; in the cyber-security domain, they can help detect compromised vertices in the network.

To create meta-features, we define the probability of an edge not existing as $p(v, u)$, where $v, u \in V$ and $(v, u) \in E$. We also define $EP(v) := \{p(v, u) | u \in \Gamma(v)\}$, which is the list of all the probabilities of v edges.

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 was previously described.

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

2. *Edges Probability STDV* is the standard deviation of a set of vertex v edges' probability of not existing. If we focus on online entities, a high standard deviation can indicate that at some point the vertex was compromised.

   $$Edges\,Probability\,STDV(v) := \sigma(EP(V))$$

3. *Edges Probability Median* is the median of a set of vertex v edges' probability of not existing. The advantage of median over mean is that it is not as sensitive to unusually large or small values.

   $$Edges\,Probability\,Median(v) := median(EP(V))$$

4. *Edge 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.

   $$Edge\,Count(v) := |\Gamma(v)|$$

5. *Sum Edge Label* is the number of vertex v edges that 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 set to 0.8. The goal of this feature is to detect cases where vertices have many anomalous edges, most of which are only slightly above the *threshold*, resulting in a relatively low P.

   $$Sum\,Edge\,Label(v) := \sum_{u \in \Gamma(v)} EdgeLabel(v, u)$$

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

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

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

   $$Mean\,Predicted\,Link\,Label(v) : = \frac{1}{|\Gamma(v)|} \sum_{u \in \Gamma(v)} Edge\;Label(v, u)$$

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

   $$Predicted\,Label\,STDV(v) : = \sigma(\{Edge\;Label(v, u) | u \in \Gamma(v), u, v \in V\})$$

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 of 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 were labeled or partially labeled. In such cases, we performed additional classification using the Random Forest algorithm on the data and created a meta-classifier.

3.3.2 Test set generation

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 uniformly selecting one random vertex, *RandomVertex* (line 3). Next, we check if it has more neighbors than the minimal amount required (*MinFriends*). In addition, for labeled graphs we also check if *RandomVertex* has the desired label, in order to ensure that the test set has positive examples (line 4). Then, we select all of the *RandomVertex* neighbors that also have more than *MinFriends* neighbors. This constraint is used to ensure that the neighbors of *RandomVertex* also were crawled and that the link features to be extracted are meaningful (lines 6–9). If *RandomVertex* has more than *MinFriends* neighbors that have more than *MinFriends* neighbors, then the selected edges are added to the test set (lines 11–14).

The goal of these steps was to select only edges between vertices which were likely fully crawled and had neighbors in the graph (i.e., have more than *MinFriends* neighbors, which we set at three because it is the minimal number of vertices where we have a majority of edges of one of the class). Vertices that have a small number of neighbors are less relevant, since there is not enough information to determine their behavior (Fire et al. 2012). 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, we set *MinFriends* to three.

3.3.3 Training set generation

Later, we sampled the link prediction classifier training set that was described in Sect. 3.2. The edge sampling for the link classifier worked as follows: Let *test-vertices* be a set of all of the vertices that were selected by Algorithm 1, and if $(v, u) \in E$ is an edge, then (v, u) can be part of the link classifier training set, if and only if $u, v \notin$ *test-vertices*.

4 Social network datasets

Academia.edu[4] is a social platform for academics to share and follow research. We crawled Academia.edu graph during 2011.

ArXiv[5] is an ePrint service used 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.[6]

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 itself was generated by observing students, interviewing pupils and parents, and analyzing school essays (Heidler et al. 2014). Delitsch found that there were 12 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 contain 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 socioeconomically deprived. This is probably the first-ever collected social network dataset (Heidler et al. 2014).[7]

Dblp[8] is the online reference for bibliographic information on major computer science publications. We used the Dblp dataset to build a co-authorship graph.[9]

Flixster[10] is a social movie site which allows users to share movie reviews and discover new movies. We collected the data using a dedicated crawler in 2012.

Twitter[11] is an undirected online social network where people publish short messages and updates. Currently, Twitter has 310 million monthly active users.[12] According to recent reports, Twitter has a bot infestation problem (Vaas 2014; Hernandez 2015). We used a dedicated API crawler to obtain our dataset in 2014.[13]

Yelp[14] is a web platform to help people find local businesses. In addition, Yelp provides various social capabilities. In 2016, Yelp published a dataset containing a social network of its users.[15]

5 Experimental evaluation

We evaluated our algorithm on ten networks that we categorized into three types of datasets (fully synthetic dataset, real-world dataset with injected anomalies, and real-world dataset with labeled anomalies). In addition, due to hardware limitations, for each dataset, we sampled a test set that contained 900 random existing vertices and 100 anomalous vertices. The test set maintained 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 dataset. 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's performance, we used tenfold cross-validation to measure the TPR, FPR, precision, and AUC for all of the evaluations. In addition, we measured the algorithm's precision at k (precision@k) for $k := 10, 100, 200$, and 500. Finally, we compared our method's performance to the Stranger algorithm (Fire et al. 2012) in detecting anomalies. The

[4] https://www.academia.edu.

[5] https://www.arxiv.com.

[6] https://snap.stanford.edu/data/cit-HepPh.html.

[7] https://github.com/gephi/gephi/wiki/Datasets.

[8] https://www.dblp.com.

[9] http://dblp.uni-trier.de/xml/dblp.xml.gz.

[10] https://www.flixster.com.

[11] https://www.twitter.com.

[12] https://about.twitter.com/company.

[13] We limited the crawler to crawling a maximum of 1000 friends and followers for every profile (see Sect. 9). This limitation is due to the fact that Twitter accounts can have an unlimited number of friends and followers, which in some cases can reach several million.

[14] https://www.yelp.com.

[15] https://www.yelp.com/dataset_challenge.

Table 1 Social network datasets

Network	Directed	Vertices	Links	Date	Labels
Academia	Yes	200,169	1,389,063	2011	No
ArXiv	No	34,546	421,578	2003	No
Class	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

Stranger algorithm is trained on simulated fake users who are inserted into the social networks by using random friendship requests. This process is very similar to the way we generated our test set for the fully and semi-simulated networks. Due to this similarity, we only evaluated the Stranger algorithm on real-world networks.

5.1 Simulation of anomalous vertices

Currently, there is a very limited number of publicly available datasets with known anomalies, and manual labeling is a challenging task (Akoglu et al. 2015). 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 (Boshmaf et al. 2011; Cao et al. 2012; Fire et al. 2012), we generated anomalous vertices by randomly connecting them to other vertices in the network as follows. First, we inserted a new simulated vertex into the graph (line 2). Next, we generated *NeighborsNumber*, the number of edges to be created for the simulated vertex (line 3). Then, we sampled random *NeighborsNumber* vertices from the graph (line 4). Afterward, 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 (Facebook 2015; Vaas 2014).

5.2 Fully simulated network evaluation

To generate fully simulated complex networks, we used the Barabási–Albert model (BA model) (Barabási and Albert 1999), which is a minimal model that can generate scale-free networks. We believe 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 networks that are described in Sect. 4. 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. We used the number of vertices and the average number of edges of the ArXiv, Dblp, and Yelp datasets (see Table 1) to generate the simulated networks. Afterward, we inserted anomalous vertices for the remaining 10% (see Algorithm 2).

Table 2 Machine learning results using fully simulated networks and semi-simulated networks with simulated anomalous nodes

	Network	AUC	TPR	FPR	Precision
Simulation	ArXiv	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
Semi-simulated	Academia	0.999	0.998	2.51×10^{-4}	0.997
	Arxiv	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

```
input  : Graph $G$, having simulated vertex number $N$
output: Graph $G$ with $N$ simulated vertices
1 for $i \leftarrow 1$ to $N$ do
2     SimulatedVerticesNumber $\leftarrow$ AddVertex($G$,$i$,$Fake$);
3     NeighborsNumber $\leftarrow$ Random($G.DegreeDistribution$));
4     RandomVertices $\leftarrow$ SampleVertices($G$,NeighborsNumber);
5     foreach $Vertex\ u\ in$ RandomVertices do
6         AddEdge($v$,$u$);
7     end
8 end
```

Algorithm 2: Adding anomalous vertices to a graph.

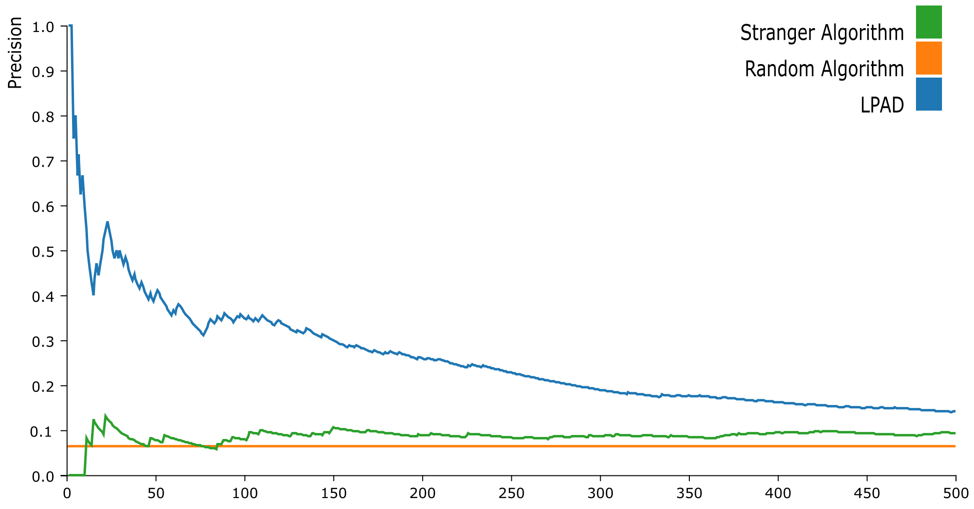

Fig. 2 The blue, green and orange lines represent Twitter precision at K of LPAD (link prediction anomaly), Stranger, and random algorithm, respectively (color figure online)

5.3 Semi-simulated network evaluation

The second dataset type was a semi-simulated network, which is a real-world network with injected simulated anomalous vertices (see Algorithm 2). The evaluation was conducted on the datasets of ArXiv, Dblp, Flixster, and Yelp (see Table 1) with inserted anomalies.

5.4 Real-world network evaluation

The third dataset type we tested our method on was a real-world network with labeled anomalous vertices. We evaluated the graphs of the Boys' Friendship (referred to as Class) and Twitter datasets (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 of the profiles (without the edges) in the Twitter dataset, approximately 1 year after the initial crawling took place. Similar to Thomas et al. (2011), we considered all of 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;[16] the most common reasons for suspension are spam, the account being hacked or compromised, and abusive tweets or behavior.[17] As a result, we labeled all of the accounts which were suspended as malicious, and we considered them to be anomalous vertices in the graph. In addition, we filtered all of the verified accounts from the dataset. A verified account is an account of public interest, primarily those of celebrities, politicians, etc.[17] These were filtered because most of their connections do not represent regular users, and many times they are managed by some kind of third party (Plante 2014).

In the Class dataset, we observed that nearly all of the psychologically or physically handicapped students (all 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 socioeconomically 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 Sect. 3.3. In addition, every execution we tested contained only 10 vertices.

Table 3 Comparison of the current method (LPAD) with Strangers (Fire et al. 2012) on the Class dataset (see Sect. 4)

Method	AUC	TPR	FPR	Precision
LPAD	0.91	0.889	0.15	0.964
Strangers	0.714	0.439	0.006	1

6 Results

We evaluated our topology-based anomaly detection method on three scenarios. First, we evaluated the method on fully simulated networks with simulated anomalous vertices using a tenfold cross-validation. As can be seen in Table 2, for the three fully simulated networks, 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, with an average AUC of 0.99 and FPR of 0.021 (see Table 2). Third, we evaluated

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

[17] https://support.twitter.com/articles/15790.

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 Fig. 2. 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 Class network. We found that six out of the seven students (precision@7 = 0.875) with the lowest *MeanPredictedLinkLabel* were the ones that Heidler et al. (2014) referred to as the "repeaters" or the socioeconomically deprived and defined as outliers (see Fig. 3). Evaluating the algorithm using tenfold cross-validation and the Random Forest algorithm, where the "repeaters" and socioeconomically deprived students were labeled as a positive class, resulted in an AUC of 0.931, TPR of 0.91, and FPR of 0.15. In addition, we discovered that our method detects anomalies much better than Strangers (Fire et al. 2012) in the Class dataset (see Table 3).

To determine which of the new features we proposed in Sect. 3.3 have more influence, we analyzed their importance using Weka's information gain attribute selection algorithm. From the results in Tables 4 and 5, we can see that for both simulated and semi-simulated scenarios the most influential feature is *AbnormalityVertexProbability*.

7 Discussion

Upon analyzing the results presented in Sect. 6, 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 of the tested scenarios. In the security domain, a false positive is one of the most important metrics; online operators try to avoid false positives to ensure that legitimate users are not blocked. Similarly, many social network operators prefer to sacrifice a true positive rather than have a relatively high false positive rate (Cao et al. 2012).

In the fully simulated network cases, we can see that the simulation results are correlated with the size of the networks. As can be seen in Tables 1 and 2, we obtained better results for the larger datasets. More 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 dataset 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 (Hernandez 2015). These undiscovered malicious users translate into high false positives rates. By 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 largely retweeted content from other users. Because of the many unsuspended malicious accounts in Twitter (Hernandez 2015), 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 (see Fig. 2), we were able to detect fake Twitter profiles with precision at 100 of 35%; this performance is considerably better than either Strangers or a random algorithm, which in this case resulted in approximately 8.0 and 6.4% precision, respectively. One of the reasons we believe our method's performance in this case is much better than Stranger's is due to the data being incomplete. The Stranger algorithm is based mainly on community features, which are more sensitive to incomplete data than edge-based features.

The Class network results confirmed the previous research (Heidler et al. 2014). The researchers described the "repeaters" as pupils who often led games and were strong, lively, and energetic, especially outside of the classroom. They also mentioned that socioeconomic status exhibited a strong influence on popularity. In their work, the researchers verified that the four "repeaters" and the "sweets giver" had a disproportionately high level of popularity. Our results show that the four "repeaters" had the strongest friendship ties of all the other pupils, which aligns with their findings. The "sweets giver," who also had high popularity, was just ranked in the middle, which also is reasonable, since some boys who looked like his friends only wanted candies, not friendship. In the Class network, the algorithm shows very good results even considering that the network is completely not random. This result indicates that the algorithm can be effectively utilized on various types of network and problems.

According to the overall results, we can clearly state that our method can detect malicious profiles that act according to a 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 research targeted employees at specific organizations (Elyashar et al. 2014). 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 proposed 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 social group examined. Moreover, we believe that the

Table 4 InfoGain values for different features for semi-simulated networks (darker color represents higher InfoGain)

	Abnormality Vertex Probability	Probability Median	Sum Link Label	Mean Predicted Link Label	Probability STDV	Predicted Label STDV	Edge Count
Academia	0.47	0.47	0.37	0.39	0.29	0.1	0
Arxiv	0.11	0.1	0.1	0.11	0.02	0.01	0.01
Dblp	0.34	0.23	0.33	0.32	0.23	0.15	0.04
Flixster	0.21	0.19	0.2	0.21	0.05	0.01	0.05
Yelp	0.18	0.14	0.27	0.17	0.25	0.06	0.06
Mean	0.34	0.31	0.33	0.32	0.2	0.05	0.03
STDV	0.18	0.19	0.15	0.16	0.12	0.05	0 .02

Table 5 InfoGain values for different features for fully simulated networks (darker color represents higher InfoGain)

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

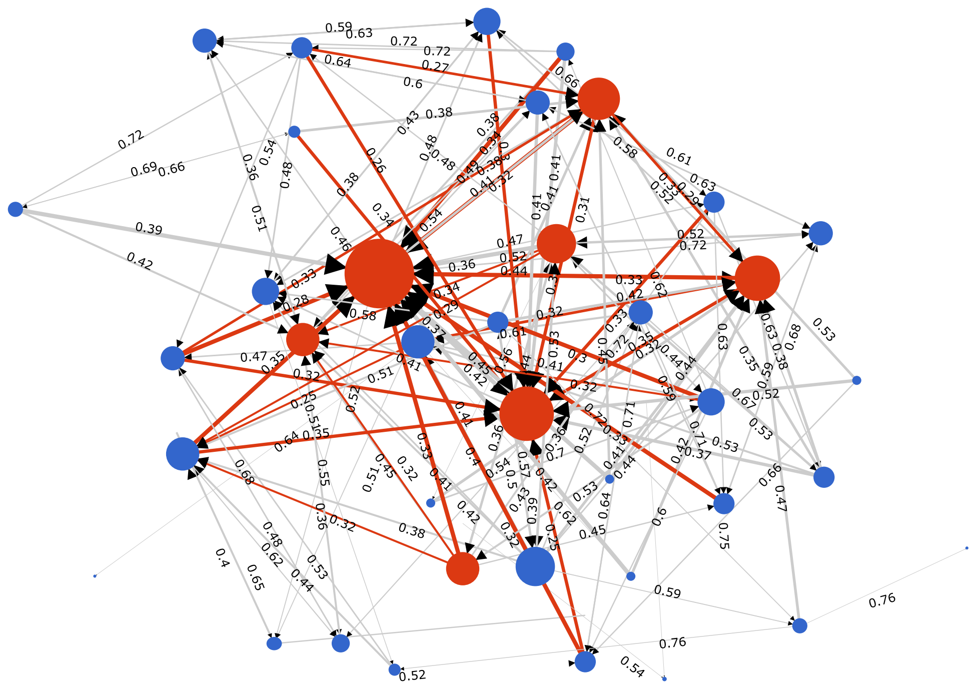

Fig. 3 The Class network, where the red vertices represent the anomalous vertices (the boys who are the most central individuals in the friendship network), and the red edges are the edges that have the lowest probabilities of being fake. The graph demonstrates that almost all the red edges connected to the red vertices (color figure online)

presented method could be used to detect hijacked profiles, if the hijacker starts to connect randomly to other vertices in the network.

A possible attack that could be tried against our method is to create many fake accounts with a small number of neighbors (also referred to as a Sybil attack[18] (Douceur 2002). In such a case, our algorithm will skip these accounts since it will consider them as irrelevant. Regardless, such an attack is inefficient in many types of networks; for instance, in Facebook, friendship is a result of mutual agreement between two users. Facebook users mostly have access to their friends information; hence, a user with a small number of friends has almost no impact in the network. Moreover, vertices that have very small numbers of neighbors relative to other vertices in the network tend to look suspicious.

There is always a big concern with models that are using synthetic data, namely whether the data truly represent the real world. With synthetic data, there is always a risk of creating a "self-fulfilling prophecy." We model the behavior of anomalies in this study based on observations made in previous studies (Stringhini et al. 2010; Boshmaf et al. 2011; Cao et al. 2012; Fire et al. 2012; Ferrara et al. 2016). In addition, there is already an example of a method (Fire et al. 2012) that is using a similar technique to generate training that is able to detect real-world malicious users in social networks. Moreover, we demonstrate how our method is able to detect real-world anomalies in Twitter and in the Class networks.

8 Conclusions

The ability to detect anomalies has become increasingly important in understanding complex networks. We present a novel generic method for detecting anomalous vertices based on features extracted from the network topology. The proposed 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. Overall, the evaluation results demonstrate that our anomaly detection model performed well in terms of AUC measure. 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 also showed that the presented method outperforms other anomaly detection methods. We believe that the presented method has considerable potential for a wide range of applications, particularly in the cyber-security domain.

As future for directions, we plan to investigate using the algorithm for other types of networks, such as bipartite and weighted graphs. In the near future, we are planning to investigate what happens to the network properties when random vertices and edges are attached. Furthermore, we intend to show that the method can be utilized for the detection of hijacked accounts in online networks. We also think it could be interesting to investigate what scale of a Sybil attack would need to be executed so that it is no longer possible to distinguish between fake and real vertices. Finally, we would like to test the usage of information theoretic measures in the future.

9 Availability

This study is reproducible research. Therefore, the anonymous versions of the social network datasets and the study's code, including implementation, are available on the project's website[19] and repository.[20]

Acknowledgements We would like to thank Carol Teegarden and Robin Levy-Stevenson for editing and proofreading this article to completion. We also thank the Washington Research Foundation Fund for Innovation in Data-Intensive Discovery, and the Moore/Sloan Data Science Environment Project at the University of Washington for supporting this study. Finally, we would like to thank the anonymous reviewers for their helpful comments.

References

Akoglu L, McGlohon M, Faloutsos C (2010) Oddball: spotting anomalies in weighted graphs. In: Zaki MJ, Yu JX, Ravindran B, Pudi V (eds) Advances in Knowledge Discovery and Data Mining, vol 6119. Springer, Berlin, Heidelberg

Akoglu L, Tong H, Koutra D (2015) Graph based anomaly detection and description: a survey. Data Min Knowl Discov 29(3):626–688

Al Hasan M, Chaoji V, Salem S, Zaki M (2006) Link prediction using supervised learning. In: SDM'06: workshop on link analysis, counter-terrorism and security

Albert R, Barabási AL (2002) Statistical mechanics of complex networks. Rev Mod Phys 74(1):47

Balthrop J, Forrest S, Newman ME, Williamson MM (2004) Technological networks and the spread of computer viruses. Science 304(5670):527–529

Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512

Boccaletti S, Latora V, Moreno Y, Chavez M, Hwang DU (2006) Complex networks: structure and dynamics. Phys Rep 424(4):175–308

Bolton RJ, Hand DJ (2002) Statistical fraud detection: a review. Stat sci 17:235–249

Boshmaf Y, Muslukhov I, Beznosov K, Ripeanu M (2011) The socialbot network: when bots socialize for fame and money. In:

[18] A Sybil attack is when the adversary controls a substantial fraction of the vertices in the system, which are then used to influence and manipulate the system to achieve the end goals of the attacker.

[19] http://data4good.io/dataset.html.

[20] https://github.com/Kagandi/anomalous-vertices-detection.

Proceedings of the 27th Annual Computer Security Applications Conference. ACM, pp 93–102

Brin S, Page L (2012) Reprint of: the anatomy of a large-scale hypertextual web search engine. Comput Netw 56(18):3825–3833

Cao Q, Sirivianos M, Yang X, Pregueiro T (2012) 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. USENIX Association, p 15

Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv (CSUR) 41(3):15

Cukierski W, Hamner B, Yang B (2011) Graph-based features for supervised link prediction. In: The 2011 international joint conference on neural networks (IJCNN). IEEE, pp 1237–1244

Douceur JR (2002) The Sybil attack. In: International workshop on peer-to-peer systems. Springer, pp 251–260

Eberle W, Holder L (2007) Anomaly detection in data represented as graphs. Intell Data Anal 11(6):663–689

Elyashar A, Fire M, Kagan D, Elovici Y (2014) Guided socialbots: infiltrating the social networks of specific organizations' employees. AI Commun 29(1):87–106

Facebook (2015) Facebooks annual report 2015. https://s21.q4cdn.com/399680738/files/doc_financials/annual_reports/2015-Annual-Report.pdf. Accessed 16 Oct 2016

Fawcett T, Provost F (1997) Adaptive fraud detection. Data Min Knowl Discov 1(3):291–316

Ferrara E, Varol O, Davis C, Menczer F, Flammini A (2016) The rise of social bots. Commun ACM 59(7):96–104

Fire M, Guestrin C (2016) Analyzing complex network user arrival patterns and their effect on network topologies. arXiv:160307445

Fire M, Tenenboim L, Lesser O, Puzis R, Rokach L, Elovici Y (2011) Link prediction in social networks using computationally efficient topological features. In: 2011 IEEE third international conference on privacy, security, risk and trust (PASSAT) and social computing (SocialCom). IEEE, pp 73–80

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

Fire M, Tenenboim-Chekina L, Puzis R, Lesser O, Rokach L, Elovici Y (2013) Computationally efficient link prediction in a variety of social networks. ACM Trans Intell Syst Technol (TIST) 5(1):10

Heidler R, Gamper M, Herz A, Eßer F (2014) Relationship patterns in the 19th century: the friendship network in a German boys' school class from 1880 to 1881 revisited. Soc Netw 37:1–13

Hernandez D (2015) Why can't twitter kill its bots? http://fusion.net/story/195901/twitter-bots-spam-detection/. Accessed 16 Oct 2016

Hofmeyr SA, Forrest S, Somayaji A (1998) Intrusion detection using sequences of system calls. J Comput Secur 6(3):151–180

Hooi B, Song HA, Beutel A, Shah N, Shin K, Faloutsos C (2016) Fraudar: bounding graph fraud in the face of camouflage. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 895–904

Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58(7):1019–1031

Newman ME (2003) The structure and function of complex networks. SIAM Rev 45(2):167–256

Noble CC, Cook DJ (2003) Graph-based anomaly detection. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 631–636

Papadimitriou P, Dasdan A, Garcia-Molina H (2010) Web graph similarity for anomaly detection. J Internet Serv Appl 1(1):19–30

Plante C (2014) 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. Accessed 16 Oct 2016

Stringhini G, Kruegel C, Vigna G (2010) Detecting spammers on social networks. In: Proceedings of the 26th annual computer security applications conference. ACM, pp 1–9

Strogatz SH (2001) Exploring complex networks. Nature 410(6825):268–276

Sun J, Qu H, Chakrabarti D, Faloutsos C (2005) Neighborhood formation and anomaly detection in bipartite graphs. In: Fifth IEEE international conference on data mining. IEEE, p 8

Thomas K, Grier C, Song D, Paxson V (2011) Suspended accounts in retrospect: an analysis of Twitter spam. In: Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference. ACM, pp 243–258

Vaas L (2014) 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/. Accessed 16 Oct 2016

Wang XF, Chen G (2003) Complex networks: small-world, scale-free and beyond. IEEE Circuits Syst Mag 3(1):6–20