TEMPO

Management and Processing of Temporal Networks

H.F.R.I. Project No. 03480

D5.1: State of the Art in Outlier Detection on Time-Evolving Graphs

Computer Engineering & Informatics Department
University of Patras
Greece
01/12/2022

State of the Art in Outlier Detection on Time-Evolving Graphs

Kostas Christopoulos and Kostas Tsichlas

December 01, 2022

Abstract

The problem of anomaly detection on static networks has been broadly studied in various research domains. Anomaly detection concerns the identification of objects or connections between objects that differ substantially from the rest. However, a plethora of real-world networks are highly dynamic, in the sense that entities (nodes) as well as relations between them (edges) constantly change. The time dimension affects critically the very definition of anomaly in these cases, and thus care must be taken to properly incorporate it. Many solutions have been proposed in dynamic/temporal networks under various assumptions concerning the modeling of time as well as the emerging anomalies. The problem becomes quite harder when the notion of time is introduced, since various unseen problems in the static case arise.

In the last few years, a few surveys have been conducted regarding anomaly detection in time-evolving networks. In this survey, our objective is to give a comprehensive overview of the current state of the art in anomaly detection in dynamic networks. We begin by providing an overview of the different types of anomalies that can occur in dynamic networks, including node, edge, community/subgraph and event/change point anomalies. Consequently, we classify techniques and methods for anomaly detection in dynamic networks, including feature, community, deep learning, probabilistic and tensor/matrix decomposition methods. Finally, we conclude by outlining future research directions in anomaly detection in dynamic networks.

Keywords: Temporal/Dynamic Graphs, Anomaly Detection, Outlier Detection

1 Introduction

Anomaly detection (henceforward also referred as AD) is a broad field that deals with identifying patterns or observations in data that deviate from what is considered normal or expected. This process is used to identify unusual or unexpected behavior that can indicate a problem or an opportunity for improvement in various fields such as computer science, engineering, finance, and security. AD can be applied to different types of data, including numerical, categorical, time-series, and network data, and it can take many forms, including unusual patterns in data, abnormal behavior in systems, or unexpected changes in patterns over time.

In general, a static network is represented as $G = (V, E)$, where V is the set of vertices (entities) and E is the set of edges (interactions/relations between entities). An edge can be directed, such as the connection between two people where one sends an email to another or undirected, such as the connection between two collaborating peers. Lastly, edges among nodes can be associated with weights (e.g., frequency of interactions) or nodes can be associated to weights (e.g., specific properties of nodes). In many cases, real-world networks are dynamic, in the sense that new edges or nodes appear and existing edges or nodes disappear. As a result, the communities themselves change because of the evolution of the network. The appearance of a new community, the disappearance of an existing and the split of an existing into two or more, are examples of changes in the community structure, which can indicate an anomaly behaviour.

In general, a temporal network can be represented either as a sequence of static graphs (snapshots) or as a network with time annotations on its nodes/edges that represent its time evolution. The former approach requires the specification of the size of the time window that defines the time instances of snapshot construction. The latter representation is related to events, like edge/node insertion or deletion or its existence interval. The time annotation may have different aspects/interpretations depending on the application. The following three aspects have been used in the literature [61]:

1. *Point Networks:* every link among two vertices x and y, which has been created at certain time t, can be represented as a triplet $e = (x, y, t)$.
2. *Time Interval Networks:* the time-interval connection of two nodes is represented as a quadruplet $e = (x, y, t, \Delta t)$. Δt is the duration of the link between the vertices x and y.
3. *Incremental Networks:* edges/nodes can only be added and deletions are forbidden.

In what follows we provide a short description of the structure of this survey. In Section 2 we discuss different models of time as well as frameworks for temporal networks. In the last few years, many surveys in the field of AD in temporal networks have been published. These are discussed briefly in Section 3. The main contribution of this paper is in Section 4, which can be summarized as follows: 1) we provide an updated overview of the current state-of-the-art methods for AD in temporal networks since the last years there are quite a few new related results, and 2) we further classify the approaches regarding the types of anomalies and the techniques that have been used. In section 5, we provide some useful, open-source libraries for anomaly detection. Furthermore, commonly used datasets for experimentation related to the AD problem are presented in Section 6. Finally, we conclude in Section 7.

2 Basic Notions on Temporal Graphs

In this section we discuss preliminary notions related to temporal graphs. We discern mainly the following issues: a) A formalism for temporal graphs, b) The notion of time and c) The model of computation.

2.1 Unified Frameworks

There is a plethora of results on temporal graphs related to concepts, formalisms as well as algorithms. Research has been done on how to unify all these under a common framework.

Time-Varying Graphs (TVGs) [21] are one such notable effort to accomplish this unification. Temporal systems can be described by a TVG $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ as follows:

V : A set of entities (nodes)

E : A set of relations between entities as well as an alphabet L accounting for a set of properties that this relation could have; that is $E \subseteq V \times V \times L$.

$\mathcal{T}$: All these relations are assumed to hold over a time lifespan $\mathcal{T} \subseteq \mathbb{T}$, where the temporal domain $\mathbb{T}$ is assumed to be $\mathbb{N}$ for discrete systems or $\mathbb{R}$ for continuous systems.

ρ : is the presence function $\rho : E \times \mathcal{T} \to \{0, 1\}$, which indicates whether a given edge is valid at some specific time instance (a similar function can be defined for nodes).

ζ : is the latency function $\zeta : E \times \mathcal{T} \to \mathbb{T}$, which indicates the time it takes to cross an edge given the starting time of the traversal (a similar function can be defined for nodes).

In [21] it is shown how TVGs can cope with the notion of time in various network settings. For example, the latency functions can be used to define journeys on such a temporal network. In a journey we require a path from a node to another node such that all time constraints are satisfied.

Stream Graphs and Stream Links [57] is another more recent unified framework for reasoning on temporal graphs. Note that streaming is not related to the notion of streaming as a model of computation.

A simple undirected **stream graph** $S = (T, V, W, E)$ is defined as follows:

V : set of nodes.

T : set of time instants (it can be discrete or continuous)

W : a set of temporal nodes $W \subseteq T \times V$

E : a set of links $E \subseteq T \times V \otimes V$ [1]

In case where there are no temporal nodes we get a stream link. We define by T_v and T_e the set of instances where the node v and the edge e are present in the graph. In [57] there is an extensive discussion about how stream graphs can be used as a formalism to transfer various notions of static graphs to temporal graphs.

2.2 The Notion of Time

The addition of the time dimension in graphs gives rise to temporal graphs. The notion of time may have different aspects/interpretations depending on the application. The following aspects in order of generality can be used:

1. **Transaction Time:** represents the time that an event takes place (i.e., the moment that an edge is inserted or deleted from the graph). In the transaction time setting, updates can only occur in an append-like manner (i.e. an update changes the most recent version).

2. **Valid Time:** it signifies the time period in which an object is valid [15, 16] (i.e. the time interval that an object existed in the graph). Transaction time can be emulated in the valid time setting by restricting updates to intervals that begin on the time moment of the update. Valid time is the time period during which a fact is true in the real world.

3. **Multiple Universe Valid Time:** assumes that the history has a tree-like shape, instead of a linear structure. This is reminiscent of the notion of full persistence in data structures [26], in the sense that the history of the data structure is fully characterized by a tree structure. Similarly, since history is not linear, we require its explicit representation by a history tree. Instead of talking about time that implies a linear evolution we now talk about versions of graphs. New version instances are created by making updates to existing versions of the history tree. For example, let a node v of the history tree correspond to the graph of version v. An update at version v gives a new instance that is represented by node u (with version u) that is a child of v in the version tree. We refer the interested reader for a more detailed analysis to [26]. The crucial point is that navigation in history requires the efficient support of nearest common ancestor queries on the history tree. In this way, searching for a version v in a node is equivalent to finding the version u that exists in this node and is the nearest ancestor among all versions in the node of v. By employing an auxiliary structure for such a tree-like history one can answer efficiently such nearest ancestor queries without further changes to the rest of the structure. Apparently, a special case of this notion of time is the previously mentioned valid time where the history tree degenerates to a single path. We could further generalize this notion, e.g., by allowing merges of graphs of different versions.

3 Related Work

A very interesting analysis in AD, in both static and temporal networks, is discussed in [1]. Focusing on temporal networks, authors characterize the unusual and abrupt changes (anomalies) in various ways. Such changes include node, linkage, community evolution and distance evolution based anomalies. Moreover, latent embedding methods have been extensively discussed and several applications for each of the aforementioned categories are stated. Similarly, in [52] authors categorize the abnormalities based on whether they involve static or dynamic networks as well as whether the networks are attributed or without attributes.Several methods have been investigated in order to detect different kinds of anomalies, while benefits and cons are discussed along with different methods for each anomaly category.

A rather outdated but important survey is presented in [37]. Three different categories of outlier detection methods regarding temporal networks, are proposed: 1) Graph similarity-based, where various similarity/distance metrics are utilized in order to detect anomalous network snapshots,

[1] $\otimes$ corresponds to an unordered pair of elements.

2) Eigenvector-based approaches, for the purpose of discovering anomalous localized regions, and 3) Community based-methods, where anomalous nodes are detected, by investigating community transition trends in consecutive snapshots. One more comprehensive survey of AD methods for high dimensional big data is discussed in [85]. Although it does not focus on network data, it presents interesting parallel and distributed approaches proposed for big data management. Similarly, the work in [24] gives an overview of deep learning techniques in outlier detection in time series data and provides a list of recent real world applications.

A comprehensive analysis of outlier detection methods in temporal networks that also include a classification of outlier types, is discussed in [78]. The authors identify four types of anomalies: 1) Anomalous nodes, 2) Anomalous edges, 3) Anomalous subgraphs and 4) Network events/changes. In order to detect the aforementioned types of anomalies, five methods (Community Detection, Compression, Matrix/ Tensor Decomposition, Distance Measures and Probabilistic Methods) have been proposed. The common methodology that is used, is a two stage strategy. Firstly, one of the above methods is utilized in order to map the dynamic graph to a vector and then, an anomaly detector is applied to identify the anomalies. Similarly, in [3] the dynamic AD methods are classified, based on the "graph summary" they use and the types of anomalies they detect. In addition, a significant work in outlier detection in high dimensional data streams is presented in [81]. A recent survey, based on deep learning AD methods, is proposed in [64], where an extensive analysis of GNN (Graph Neural Networks) and graph representation techniques are discussed. Current status and challenges in GNN architecture, for both static and dynamic networks, can be found in [53]. Moreover, in [30] a broad research on anomaly detection in multiple directions (like network traffic anomalies, intrusion detection systems, detection methods and systems) is presented.

A very nice survey in graph/node embeddings is presented in [17]. More precisely four types of embedding inputs and outputs are introduced. Moreover, five main embedding techniques (factorization, deep learning, graph kernel, edge reconstruction and generative model), with advantages and disadvantages for each one, are extensively described, and node/edge/graph related applications are discussed. The aforementioned survey is very useful in our work, since most of the detection methods we discuss in the next section utilizes graph embeddings in order to reduce dimensionality. One more survey that describe methods for anomaly detection in social networks is presented in [80]. A comprehensive analysis in event detection methods by tracking the change of community structure is presented in [4]. Lastly, a general framework (SAFARI) for streaming anomaly detection is proposed in [19]. The framework consist of four main components: 1) Data representation, i.e., mean and standard deviation and symbolic aggregate approximation; 2) Learning strategies, i.e., sliding window, landmark window, and uniform reservoir sampling; 3) Nonconformity measures, i.e., nearest neighbors-based, density-based or clustering-based; 4) An anomaly score function. The aforementioned paper is placed in this section because in each of the above components (as it is already described), existed strategies are combined and the type of the detected abnormality depends on the chosen strategy. The experimental evaluation in real world datasets shows that there is not a single method that works best in all cases.

4 Classification and Methods Description

In this section we classify anomaly detection methods in evolving networks. Based on the literature, we can identify four different types of anomalies:

1. **Node anomalies**: These are anomalies that are associated with a specific node in the network, such as a node that has a high degree of connectivity or a node that experiences a change in its behavior with respect to various metrics.

2. **Edge anomalies**: Anomalies of this type occur in the connections between nodes. One example is the sudden appearance and immediate disappearance of a new edge between two nodes or a sudden change in the weight of an existing edge.

3. **Community or subgraph anomalies**: These are anomalies that occur in the formation or structure of groups of nodes (communities) within the network; for instance, a shrunken or a split community.

4. **Change point and event anomalies**: Refers to a sudden change, instantaneous (change point) or enduring (event), in the structure or properties of the network. This can include changes in the number of nodes, edges, or communities, changes in the properties of individual nodes or edges, or changes in the overall structure of the network.

Based on the surveys reviewed in Section 3 and on the most recent papers related to anomaly detection, usually a a two-stage approach is adopted. First, the dynamic network is mapped to a common graph representation in order to generate desired features. In the second stage, an existing outlier detector for arithmetic data is applied and decides whether a certain type of anomaly (node, edge, community or overall structure) exists or not. Thus, anomaly detectors act on the graph representation, which is output from the first stage. Based on the preceding discussion, we identify five different approaches to obtain a common graph representation and derive the generated features from the result of these methods. In what follows, we present these approaches that in fact correspond to the our classification:

1. Graph Feature-based methods
2. Decomposition and Compression-based methods
3. Machine/Deep learning-based methods
4. Community-based methods
5. Probabilistic Model-based methods

Table 1: Overview of the proposed temporal anomaly detection classification.

Temporal Anomaly Detection		
Methods Classification	Description	References
Graph Feature	Metrics to measure the structural features	[73, 36, 67, 72, 95, 88, 99, 50]
Decomposition and Compression	Matrix/Tensor factorization and compression techniques	[22, 48, 46, 71, 45, 41]
Machine/Deep learning	Reinforcement Learning, GCN's, RNN's and other Machine Learning (ML) based techniques	[31, 28, 18, 32, 87, 75, 62, 86, 43, 58, 102, 97, 77, 7, 101, 54, 83, 9, 92, 68]
Community based	Track the evolution of the communities	[69, 34, 5, 94, 39, 23, 65, 59, 103, 60, 35, 104, 55, 49, 70, 98, 33, 44]
Probabilistic Model	Techniques based on statistical models/characteristics	[40, 42, 12, 14, 13, 51, 100, 29, 93, 89, 90, 91]

In the next section, we describe in more detail each of the above classes and we mention some of the most recent and interesting related research papers. Moreover, in the description of the algorithms, we mention the anomaly type and various other characteristics.

4.1 Graph Feature-based Methods

Graph feature-based methods are used in graph analysis to extract relevant information from networks. This information is represented as a set of features (e.g., node centrality, clustering coefficient, distance measures). Feature-based methods are used to transform network data into a more compact and informative representation, making it easier to analyze and understand. The choice of features can have a significant impact on the performance of algorithms; so it is important to select appropriate features for a given problem.

A feature-based method that detects key events, is analyzed in [73], where a number of different graph features are estimated (Betweeness, Closeness, Eigenvector centrality, PageRank, In-degree, Out-degree, etc). By using a vector with average values of these features, authors define the timestamps in which significant changes happen in the network. Compared to the NETSIMILE algorithm [10], the presented approach aggregates metric findings utilizing a wider range of attributes and other statistical values.

DynAnom method in [36], introduces a weighted push mechanism to calculate the PPV_s (Personalized PageRank vectors), and use them in order to obtain node representations by applying dimensionality reduction, utilizing hash functions [76]. Based on these representations, node and graph level anomaly scores are calculated. A recent method (extended version of [66]) that focus on detecting anomalous groups of nodes is presented in [67]. Given a multi-layer network, authors propose a centrality-based anomaly detection approach using a multiobjective optimization technique. Finally, they utilize the ACE score function to calculate the deviations of nodes centrality, and consequently to characterize them as anomalous.

In [72], a distribution-free framework for anomaly detection in dynamic networks is presented. To begin with, the data representation of each network snapshot is given as a linear object. Then, local and global topological graph summary statistics (e.g., number of nodes, edges, betweenness centrality) of each snaphot are utilized to find the univariate characterization. Consequently, a change point detection method, based on efficient scores, is adopted and by using the sieve bootstrap method (estimating the distribution of a statistic of interest), authors approximate the asymptotic distribution of the test statistic.

In [95], an anomalous event detection method in dynamic graphs, with high accuracy and speed, is proposed. Authors classify the anomalies into two types, AnomalyS (presence of edge among two nodes) and AnomalyW (number of appearances of each edge). By utilizing a modified PageRank [96] metric, they define the node score functions ScoreS and ScoreW respectively. Subsequently, they define the metrics AnomRankS and AnomRankW, by calculating the first (p') and second (p'') derivative for each score function. As a consequence, the AnomRankS (similarly for AnomRankW) is computed as a $(n \times 2)$ matrix $a_s = [p'_s \; p''_s]$ and is equal to $||a_s||_1$. Finally, $||a||_1 = max(||a_s||_1, ||a_w||_1)$ is returned as the anomaly score.

The proposed method in [88], detects anomalous changes in evolving social graphs. The SeaDM algorithm consists of two main components: ESCA (Eigenvector Spreading Centrality Algorithm) and OEOA (Optimized Eigenvector Outlier Algorithm). The former, constructs the evolution state of the network, while the latter detects the optimized observation vector of the evolution state. Finally, by combining these two components, the change state is evaluated and detected. An approach called NetWalk that learns network representations incrementally in order to detect vertex/edge anomalies in dynamic graphs, is described in [99]. By using random walks, authors construct a network walk set, and consequently a novel clique embedding method is utilized. Finally, they deploy the streaming k-means clustering method to detect if a newly arrived node (or edge) belongs to a cluster or is an anomaly.

4.2 Decomposition and Compression-based Methods

Decomposition methods break down a complex data structure into smaller, simpler components. These components constitute the generated features. Common examples of decomposition methods include matrix decomposition, such as Singular Value Decomposition (SVD), eigendecomposition, and tensor decomposition. Compression-based methods (Minimum Description Length - MDL principle and compression techniques), aim to reduce the size of data by removing redundant information. In this case, features are derived from the encoding cost of the graph and its substructure. Examples of compression-based methods include principal component analysis (PCA) and the MDL principle.

A novel approach called F-FADE [22], detects anomalies of either a single type of interactions (edges) or a batch of them with different types. By utilizing frequency factorization of the most frequent appeared edges, they calculate the node embeddings and update them each time. Consequently, for each arrived edge, an anomaly score is estimated based on the likelihood of the observed frequencies.

[48] proposes a scalable change point detection method (LADdos) in dynamic networks. Authors combine an already known anomaly detection method, LAD [47], with the network/spectral density of states framework [25]. Thus, initially, LAD detects anomalies from low dimensional embeddings (signature vector), using the singular values obtained from the SVD method applied on the Laplacian matrix. Consequently, authors model the distribution of the obtained singular values utilizing

efficient methods (e.g., the Kernel Polynomial Method), in order to scale LAD. The effectiveness of the method is the same to LAD, but it is two orders of magnitude faster. A node anomaly detection method that uses spectral factorization to create graph embeddings aiming at reducing dimensionality, is proposed in [46]. By using Procrustes analysis in each time-step, they calculate the profile features of the last embedding and estimate the change scores for nodes between the current and the last embedding.

In [71], they describe a change detection method, called CODEtect, to detect anomalies in node-labeled multi-graph databases based on the MDL [8]. The gist of the method is to uncover important network motifs that can encode the graph concisely, and then, the anomaly score can be derived from its compression length. To achieve this, they present novel lossless encoding schemes and efficient search algorithms. Experiments show that CODEtect outperforms GNN-based baseline methods.

A comparative study on multi-view dynamic graphs that uses tensors (with multiple slices for each time instance) to represent network information, is presented in [45]. They apply seven different embedding strategies combined with a moving window approach in order to estimate the profile embedding and the change scores for each node. Among all these strategies, authors conclude that tensor-based spectral embedding and spectral embedding based on generalized orthogonal Procrustes Analysis [27], show the best results in detecting node-based anomalies.

An interesting approach in both community and anomaly detection in urban temporal networks, is presented in [41]. Community detection is used to reduce data dimensionality and noise, and at the same time to preserve the information of the topological structure. Subsequently, decomposition techniques are employed to learn the network representation and further decrease the data dimensionality. Lastly, utilizing Gaussian Mixture Models (GMM), they calculate the probabilistic score for each point. Based on these scores, nodes are classified as anomalous or not. Experiments exhibited that the presented method outperforms baseline approaches based on mobility and spatial aggregation.

4.3 Machine/Deep Learning-based Methods

Deep learning methods such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are also used for anomaly detection. They can be used in both supervised and unsupervised modes. In supervised deep learning, the model is trained on labeled data and can be used to classify new instances as normal or anomalous. In unsupervised deep learning, the model is trained on unlabeled data and the reconstruction error is used as an anomaly score to identify instances that deviate from the main trend. In this case, the NN-graph representation has a crucial role in deriving the specific features.

[31] detect bots in Tweeter network, using a Multi-Agent method. A three layered architecture is designed in which collector, classifier and referee agents are involved. Collectors gather, record and send information to three classifiers. Then, classifiers (learning agents) extract the features and by utilizing different ML algorithms classify the receiving data. Lastly, the referee is responsible for counting the votes of the classifier agents. If the referee determines that a Twitter profile has received a lot of votes, they will report the profile to Twitter (an anomaly) and monitor whether the profile is banned.

A novel and recent method based on PCA (Principle Component Analysis) and machine learning techniques, is discussed in [28]. The PCA-NN (Neural Network) algorithm propose an efficient way to discover anomalous activities and communities in Facebook network. Using StrGNN (structural Graph Neural Networks), an anomalous edge detection method based on the subgraph structural changes related to the target nodes, is presented in [18]. The proposed approach consists of three phases. Firstly, the d-hop enclosing subgraph of a target edge is extracted and node label annotation on this subgraph is performed. Consequently, combining the GCN method (projection into embedding space) with the graph pooling method (fixed size feature extraction), the importance score for each node is calculated. Lastly, in order to obtain temporal information they utilize the GRU model (Gated Recurrent Units) and use a context dependent negative sampling strategy. Experimental results establish the effectiveness of the proposed algorithm, with no false negatives. One more deep learning technique using Graph Auto-encoders, is described in [32]. Utilizing random walks, authors extract the graph structures capturing the temporal information, as well. Then, by using auto-encoders, the node embeddings are learned. In the last step, the k communities of the embedded graph are detected and when the distance of a node from a community center is greater than a threshold, the node is referred as anomalous.

Based on a dynamic deep learning architecture, authors in [87] propose a node anomaly detection method (DeepFriend), by training link features in online social networks. By introducing the WalkPool function in the hidden layer, researchers achieve better graph performance, by finding suitable pooling value in order to efficiently estimate the parameters in every layer. Experiments show that the proposed approach has better performance and accuracy, when compared to Gradient Boosting, Logistic Regression, and Support Vector machine learning approaches.

A walk-based embedding technique (Pikachu) in dynamic/streaming graphs that detect anomalous edges is discussed in [75]. They emphasize on the temporal order of events in order to understand and predict the network behaviour. To begin with, network is split in snapshots and then each snapshot is traversed, in increasing time, using a temporal walk method. Then, the skip Gram model and the GRU auto-encoder are utilized, in order to capture the short and long term temporal dependency, respectively. Lastly, by finding the probability distribution for each node taking into account its neighbours, the authors conclude that the smaller the probability the more likely an edge is abnormal. Pikachu performs better than the other state of the art methods by considerably reducing the false positive rate.

Isolation forest (IForest [62]) is an anomaly detection method in static networks that is based on splitting a dataset in order to isolate observations. IForest is trained on a dataset by constructing a series of isolated trees; each of these trees contains only a random subset of the data points. Each isolated tree has to decide if an observation is an anomaly or not. An isolation Forest Algorithm for stream data is described in [86], which utilizes sliding window semantics to address the problem of streaming data. Each time, they consider the anomaly rate of every window to decide whether to discard the current iForest detector. In [43], an ensemble-based online outlier detection method, called PCB-iForest, is proposed. More precisely, two variants of existing static methods [38, 82] are introduced, $PCB-iForest_{EIF}$ and $PCB-iForest_{IBFS}$. For the former, authors adapt an already existed static iForest-based method for streaming data, while for the latter, they modify a feature scoring function to adapt it to a streaming scenario.

In [102], an edge anomaly detection method is presented that utilizes the GCN model (structural and content feature of nodes are considered). The state of the nodes over a short window is calculated, and the output of the GCN model is received by GRU in order to find the latent state of nodes at the next timestamp. Finally, the anomaly probabilities based on the latent state of nodes is computed and consequently, the margin loss function is utilized in order to deal with the unlabeled anomaly data. This method outperforms, significantly, other well-known competitors.

A deep learning method is presented in [97]. In this research work, the authors utilize an adaptive model pool (existing deep anomaly detection methods) and look for abnormal data points in a complex data stream. Concept-driven inference and concept drift-aware update are the two main techniques used. While the former combines models to estimate the anomaly score, the latter, based on the model pool contribution to the final anomaly score, updates the model pool if it is assessed as inadequate.

A distributed anomaly detection method, which is based on a Graph Neural Network architecture, is described in [77]. The authors propose a multi-agent system, considering a three layered IoT infrastructure, strengthened by the GNN technology. Several experiments in real and synthetic datasets are conducted and the proposed method seems to outperform all the existing state of the art techniques. This method can be applied in scenarios concerning smart electricity grids, food chains and internet of medical devices.

One more deep learning method is described in [7]. In this paper, authors perform random walks on graph nodes to obtain the structural information and consequently feed them to an auto-encoder in order to generate the node embeddings. Then, the anomaly score for each node/edge is estimated based on its distance to tghe cluster centers. The more the node or edge is further from its closest cluster, the more it is considered as abnormal. Finally, the reservoir sampling algorithm is utilized to maintain the structure of the growing network.

[54] proposes a generic deep learning model that detects events in dynamic networks. The proposed architecture is divided into two learning graph embedding stages, based on micro and macro dynamics. For the former, a GCN model and the v-Att pooling operator is utilized in order to capture dynamics at a node level. For the latter, the produced embedding is fed to a modified RNN (LSTM). Consequently, using the t-Att pooling operator, dynamics at graph level are captured.

Moreover, three different variants of the proposed model are discussed and experiments shows that the presented model and its variants outperform the baselines.

A novel, change point detection method using siamese Graph Neural Networks (s-GNN) is proposed in [83]. A pair of graph (snapshots) is fed to the s-GNN and the siamese encoder estimates the two embeddings. Consequently, the two embeddings are processed by a similarity module in order to learn the graph similarity function. Authors point out the prominence of choosing an appropriate initialisation of the node features matrix, when the dynamic network is not attributed. Finally, random and windowed scheme sampling mechanisms are designed for the training and validation set respectively, for the purpose of satisfying the double objective (graph similarity function and change point detection).

An edge anomaly detection method (AnoMulY) in multiplex networks that manage multiple types of connections between nodes, is proposed in [9]. Each time, the GNN architecture is used in order to learn node embeddings with the same relation type. Consequently, through the attention mechanism, all information from different relation types are summarized in a weighted manner. Finally, in the last phase, the training process takes place by using the negative sampling approach. Regarding the experiments, the aforementioned method shows significant results by detecting anomalous behaviours in human brains, predicting diseases and disorders.

A multi-scale reconstruction method, called M-VGRAE, that detects anomalous nodes and edges in stochastic dynamic networks is described in [92]. First, normal nodes and edges are processed to obtain snapshots for training. In order to extract the multi scale spatial and temporal features, they use GNN and RNN layers respectively. In the next step, the reconstruction probabilities are estimated through the trained M-VGRAE and if the probability of a node or edge is lower than a threshold, it is considered to be an anomaly. Experimental results in four real datasets show that M-VGRAE outperforms the baseline methods. An unsupervised feature detection method to detect bots (Retweet - Buster) in Tweeter network, is discussed in [68]. First, authors arrange the data as a retweet time series and compress it in order to make it compact. Consequently, the unsupervised feature extraction is taking place. By using deep learning techniques (LSTM encoder), they achieve dimensionality reduction and then a density-based clustering algorithm (HDBSCAN [20]) is applied. At the end of the HDBSCAN process, all the users that belong to huge clusters are labeled as bots.

4.4 Community-based Methods

Community-based anomaly detection methods are a type of graph-based anomaly detection that use the structure of a network to identify abnormalities. These methods are based on the idea that anomalies tend to be located in regions of the graph that have different topological characteristics than the rest of the graph. As a consequence, all the data specific features are derived from the community structure.

An event detection method based on the evolution of the community structure in temporal networks, is presented in [69]. The authors assume that large events activate information diffusion, which in turn affect community borders. By applying network aggregation in a certain time period, researchers utilize the InfoMap algorithm to detect communities. Consequently, intra-community and inter-community links are calculated in each time interval, and when the difference between inter-community links and intra-community links is greater or equal to a threshold (mean and standard deviation), then an event is detected. Experiments with the Enron and Boston Marathon bombing networks, show the effectiveness of this method. Similarly, in [34], anomalous events in graph snapshots are detected by utilizing the community boundary nodes.

An improved approach of [69], is discussed in [5]. The authors detect events by combining two different approaches. First, like [69], researchers check communication trends among communities calculating the inter-community and intra-community links. The second approach (community structure based) is divided into two stages: 1) Check the number of extracted communities in consecutive time steps, examining several buckets in terms of community size and 2) track, in consecutive time steps, the number of central nodes inspecting the ratio changes. Moreover, stages 1 and 2 can be combined, since sometimes, the number of communities and central nodes are affected simultaneously from events. Finally, for the initialization of the method, existed algorithms are used to detect the initial communities and central nodes. Experiments in real datasets show that community structure based methods are more scalable and faster than others. A relative recent approach that detects abnormal hosts (nodes with high centrality degree), is described in [94]. In this method, authors focus on discovering collective abnormalities by considering community activities. This approach consists of three steps: First, by utilizing Spark GraphX they create the graph model. Second, by applying a well-known static algorithm, they detect all the host communities. Third, by matching the communities in consecutive time steps, the event type for each community (merge, split, appear, ..etc) is identified and four feature types are estimated. Finally, three different anomaly types are considered: Change in the distance of feature space utilizing the aforementioned four feature types, change in the scale of community and change in the life-cycle of network evolution.

Utilizing co-evolution pattern mining, authors in [39] detect anomalous (target) nodes in multi-typed information networks, i.e., heterogeneous bibliographic information network (HBIN). The proposed algorithm consists of three steps. First, the co-evolution patterns are extracted using relational and evolutional constraints on the dataset. Then, an existed multi-typed clustering method is used in order to cluster the patterns. Finally, by estimating the similarity index among target (author) and attribute (paper-count, co-author, and venue) objects, the anomaly score is calculated and the anomalous nodes are identified.

A method that detects anomalous communities based on the community evolution in dynamic graphs, is presented in [23]. To reduce the computational cost, they introduce the notion of graph representatives and community representatives. Taking advantage of these representatives, in each timestamp the algorithm identifies the predecessors and successors for each community, if any. Lastly, exploiting the decision rules, six different types of community based anomalies are proposed i.e. Grown, Shrunken, Merged, Split, Born and Vanished community. Experiments in both synthetic and real datasets are conducted exhibitng the efficiency and the effectiveness of the method for detecting anomalies. A method that detects community-based change points in snapshots, is analyzed in [65]. In this work, three different network types are considered: 1) static, 2) semi-static and 3) dynamic. In the first type, nodes are presented in each time slice whereas in semi-static, nodes can be removed. In both cases, the partition method, independently of the community detection algorithm, is constructed using the co-group network. Regarding the third type, authors use the first two cases to calculate the parameters for an anomaly detection-based streaming method. The experimental evaluation shows that this method is more efficient and less sensitive compared to Generalized Louvain and GraphScope [84].

In [59], a combination of change point detection and community evolution events is proposed. First, the network is divided into snapshots and in order to capture the network change, the change detection range is extended to a time window instead of looking at only two consecutive snapshots. Then, by utilizing matching community measures, the authors find the community evolution event that occurred. Another anomaly detection algorithm, in dynamic attributed networks, is proposed in [103]. There, a function that denotes the anomalous score is utilized. The authors use dynamic graph clustering with a community detection model by ranking nodes based on 1) how close they are to a dense community center and 2) the deviation from current and historical behavioral data, in order to identify anomalies. In [60], authors utilize the sliding window technique in order to generate multivariate subsequences and apply a modified fuzzy clustering algorithm to detect the structure. Then, the multivariate subsequences, the optimal cluster centers and the partition matrix are reconstructed. By calculating a confidence index, authors quantify a level of anomaly detected in the series, and apply Particle Swarm Optimization for the problem of outlier discovery.

In addition, for the purpose of identifying anomalous events in graph streams, and by using a novel definition of anomaly score based on the history of the actions of nodes, a community based method is presented in [35]. In [104], a network of snapshots is constructed. The weight of each edge is defined as the similarity score between the corresponding snapshots. Then, by detecting the network communities, they check as to whether each community consists of similar snapshots and whether two consecutive snapshots belong to different clusters. Based on these checks, a change point anomaly is identified.

A novel method that detects changes in labeled and directed heterogeneous stream graphs is presented in [55]. By using the graph substructures and GED (Graph Edit Distance), they construct the network embedding. Consequently, cluster construction (using the k-Medoid algorithm) is performed and the incoming graphs are compared to the clusters to spot potential anomalies with respect to communities. One more recent research on the field of community and anomaly detection is presented in [49]. For each snapshot, they use the Louvain algorithm and the LPA to detect communities. Then, in order to discover anomalies the do the following: 1) they utilize a bipartite graph to capture the community evolution between two consecutive time steps, and then, 2) from the constructed evolutionary paths, they detect possible community abnormalities. Other papers with similar results can be found in [70, 98, 33, 44]

4.5 Probabilistic Model-based Methods

Probabilistic model-based anomaly detection methods are approaches that use probability theory, scan statistics or distribution properties to model and to identify unusual data points. These methods are based on the assumption that the data points that belong to the normal class follow a certain probability distribution, while the data points that belong to the abnormal class do not.

In [40], they develop a probabilistic graph model aiming at discovering outbreaks in temporally dependent networks. Two moving window-based monitoring strategies are developed to detect changes in networks utilizing a random graph model. The first, is a multi-path dependent LRT, while the second is the multivariate MEWMA control chart. Both strategies are evaluated in synthetic and real datasets. The multi-path dependent LRT outperforms MEWMA with respect to efficiency and effectiveness. A comparison is also conducted with the approach presented in [42]. The outcome of this experiment shows that the proposed approach performs better.

Given a stream of edges over time, a recent approach, called MIDAS [12], that uses the Count-Min Sketch (CMS) data structure and utilizes the Chi-squared statistic, identifies micro-cluster anomalies and avoids false positive anomalous edges. Furthermore, an improved version of MIDAS is proposed, MIDAS-F [14], that identifies future anomalies. Similarly to [12], authors in MStream [13] utilize the Chi-squared statistic, but first, in order to address the problem of multi-feature edge streams, they use a number of hash functions in two ways. They use hash functions that are applied to a single feature (FeatureHash) as well as to all features of a record (RecordFeature) simultaneously. Finally, they combine MStream with principal component analysis, information bottleneck or autoencoder so that anomalies can be detected fast and efficient.

A statistical feature-based method (oddnet) to detect anomalous subgraphs in dynamic/temporal networks, is presented in [51]. As a first step, authors map each graph to the feature space and then in each feature time series the best ARIMA model is fitted. Consequently, the residuals of the fitted ARIMA are considered, and robust PCA is used in order to reduce the dimensional space. Finally, an existed anomaly detection algorithm (lookout) is used and abnormalities are discovered. Experimental results both in synthetic and real datasets manifest the dominance of oddnet compared to two other anomaly detection methods.

A nice method to detect changes in streams of attributed graphs is presented in [100]. To begin with, stream of graphs are generated by a stochastic process. Then the generic graph is embedded into a vector (dissimilarity representation), by calculating the dissimilarity among the current graph and the prototype graphs. Consequently, as time goes by, a vector of embedded graph streams is generated and each time a change detection process, based on the statistical hypothesis test, is applied in order to infer possible anomalies (changes). Experimental results show that the method is comparable to the state of the art graph stream approaches.

Both change point and localization detection is discussed in [29]. By assuming first a fixed number of nodes, they use test statistics in order to examine different cases of change point detection. Moreover, by utilizing the CUmulative SUM (CUSUM) statistic, they calculate the position of the change. In addition, the framework is changed assuming that the number of nodes are not fixed. In particular, by introducing the graphon model in combination with the CUSUM statistic, authors detect changes in the network. Experiments are conducted in real and synthetic datasets for different scenarios, e.g., stochastic block model. In [93], a novel statistical-based method is developed in order to model dynamic social networks. They take into consideration the birth, death and lifetime for both individual features and network entities. They also model the network dynamics, which enables them to detect anomalous edges and nodes.

Utilizing the MCMC sampling method, authors in [89] estimate edge probabilities between nodes in each snapshot. Then, by using distance measures, they compare the probability distribution among consecutive snapshots and finally, when the score between two dissimilar snapshots is above a threshold a change point is identified.

Utilizing the Degree Corrected Stochastic Block Model (DCSBM), the authors in [90] model and monitor dynamic graphs that exhibit structural changes. More precisely, the goal of the proposed framework is to detect any possible changes in the distribution of a dynamic graph sequence that is generated from DCSBM. First, for the generated sequence of graphs, a statistic is estimated based on the topological characteristics of graph, e.g., connectivity of nodes. Thereafter, the mean and the variance of this statistic is calculated, creating the tolerance region $R(\hat{\mu}, \hat{\sigma}^2)$. Consequently, when a new graph is inserted, the two proposed monitoring strategies, Shewhart and EWMA, detect sudden large and small network changes respectively, by checking if the new statistics lies outside of the limits of the tolerance region. Regarding the experiments, the proposed framework models a variety of dynamic networks with structural changes, and detects relevant changes in a real dynamic system.

5 Anomaly Detection Libraries

An anomaly detection library is a collection of algorithms and tools for detecting anomalies in data. Such libraries typically provide a set of pre-implemented algorithms for detecting anomalies in various types of data, such as time series data, graph data, and image data. These libraries also provide tools for data pre-processing, such as normalization and scaling, and visualization, making it easier to understand and interpret the results of the anomaly detection algorithms. By using an anomaly detection library, data scientists and researchers can save time and effort in developing their own anomaly detection algorithms and can focus on analyzing and interpreting the results. These libraries also provide a convenient and streamlined tool for working with data and performing anomaly detection, making it easier to detect and understand anomalies in large and complex data sets. Python and other libraries, followed by a small description, can be found below.

- **Pygod [63]** Pygod is a valuable resource for researchers and data scientists working with graph data and outlier detection. It provides a convenient and streamlined tool for working with graph data. It supports a wide range of methods for anomaly detection.
- **Anomalib [2]** The Anomalib library is a free and open source framework for training, deploying, and building deep-learning-based anomaly detection models. It includes a suite of tools for comparing multiple anomaly detection algorithms on any dataset quickly and reproducibly.
- **Anomalykits [74]** AnomalyKits is a valuable resource for researchers and data scientists working with time series data and anomaly detection. It provides a convenient and streamlined tool for working with time series data and performing anomaly detection, and supports a wide range of algorithms for anomaly detection.
- **tegdet [11]** TEGDET (An Extensible Python Library for Anomaly Detection Using Time-Evolving Graphs) is a specialized library for outlier detection in dynamic networks. It is built on top of popular machine learning and graph analysis libraries, such as NetworkX, scikit-learn, and PyTorch, and provides various methods for outlier detection in time-evolving graphs.

6 Datasets

Datasets are commonly used in research, machine learning, and data analysis for training and evaluating models, testing hypotheses, and exploring patterns and trends in the data. They can be either public or private, depending on the source and the intended use of the data. Synthetic generators and real datasets are two types that are used for different purposes in research and development. In the next two subsections we mention some of the most significant repositories of these two types.

6.1 Real Datasets

There is a plethora of real datasets that are used to experiment with proposed approaches. The most of these datasets, followed by a small description, can be found below.

- **SNAP** (`https://snap.stanford.edu/snap/`) A large repository of networks of various types among which one can find temporal networks as well. It also provides a Python API.
- **SocioPatterns** (`http://www.sociopatterns.org/datasets/`) It contains social networks (people/animals), some of which have also temporal information.

- **ADRepository** (`https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets`) ADRepository is a collection of datasets, some of them suitable for anomaly detection in dynamic networks.

- **Outlier Detection DataSets (ODDS)** (`http://odds.cs.stonybrook.edu/`) This website provides access to a collection of datasets for outlier detection and anomaly detection in various domains, including dynamic networks.

6.2 Synthetic Generators

The RDyn synthetic generator for dynamic networks with communities is given in [79]. This is a dynamic network generator suitable for modeling community dynamics. RDyn imposes power-law distributions on the degree of nodes as well as on the community sizes and can also be parameterized in order to plant communities that optimize totally different quality measures. The suggested method, first off all generates the community size and degree distribution and utilizes both of them to connect every node to a partition. Then it models network dynamics by repeatedly adding and removing links. When RDyn recognizes a stable state (really high quality partition), it gives an output as ground truth and generates community perturbations (splits and merges), changing node-community connections of chosen communities.

Another synthetic generator for social graphs is described in [6]. The LDBC Social Network Benchmark (LDBC SNB) has two components: a synthetic generator for social graphs as well as a suite of benchmark queries. LDBC SNB is meant to be a believable look-alike of all the aspects of a social network site. LDBC SNB includes the Interactive workload that consists of user-centric transactional-like interactive queries, and also the Business Intelligence Workload, which incorporates analytic queries to reply to business essential queries. At first, a graph analytics workload was conjointly enclosed within the roadmap of LDBC SNB, but this was finally delegated to the Graphalytics benchmark project, that was adopted as a formal LDBC graph analytics benchmark. LDBC SNB and Graphalytics joined, target a broad variety of systems of totally different nature and characteristics. LDBC SNB and Graphalytics are intended to capture the essential options of those scenarios while abstracting away details of specific business deployments.

Finally, in [56], the LFR Benchmark network is proposed. They utilize features of real networks (e.g., heterogeneity), in order to generate graphs. The experimental results show that these features contribute to a harder test of the existing methods.

7 Conclusion

The current survey provides a comprehensive overview of the current state-of-the-art in the field of anomaly detection in dynamic networks. The increasing use of dynamic networks in various domains, such as network security, transportation, and social networks, highlights the importance of this problem and the need for effective and efficient anomaly detection methods. The survey findings reveal that various techniques have been proposed to address this challenge, including graph representation, deep learning, community and probabilistic methods.

The future of this field holds great promise, and it is anticipated that ongoing research and advancements will lead to more effective and efficient methods for detecting anomalies in dynamic networks. This will not only benefit various domains where dynamic networks are used but also contribute to the overall improvement of the field. Further research in this area will play a crucial role in addressing the challenges and in taking advantage of the opportunities in the field of anomaly detection in dynamic networks.

References

[1] Charu C Aggarwal. An introduction to outlier analysis. In *Outlier analysis*, pages 1–34. Springer, 2017.

[2] Samet Akcay, Dick Ameln, Ashwin Vaidya, Barath Lakshmanan, Nilesh Ahuja, and Utku Genc. Anomalib: A deep learning library for anomaly detection. *arXiv preprint arXiv:2202.08341*, 2022.

[3] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29(3):626–688, 2015.

[4] Rıza Aktunç. Event detection via tracking the change in community structure, communication trends, and graph embeddings. 2022.

[5] Riza Aktunc, Pinar Karagoz, and Ismail Hakki Toroslu. Event detection via tracking the change in community structure and communication trends. *IEEE Access*, 10:109712–109728, 2022.

[6] Renzo Angles, János Benjamin Antal, Alex Averbuch, Altan Birler, Peter Boncz, Márton Búr, Orri Erling, Andrey Gubichev, Vlad Haprian, Moritz Kaufmann, et al. The ldbc social network benchmark. *arXiv preprint arXiv:2001.02299*, 2020.

[7] Monika Bansal and Dolly Sharma. Density-based structural embedding for anomaly detection in dynamic networks. *Neurocomputing*, 2022.

[8] Andrew Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *IEEE transactions on information theory*, 44(6):2743–2760, 1998.

[9] Ali Behrouz and Margo Seltzer. Anomaly detection in multiplex dynamic networks: from blockchain security to brain disease prediction. *arXiv preprint arXiv:2211.08378*, 2022.

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

[11] Simona Bernardi, José Merseguer, and Raúl Javierre. tegdet: An extensible python library for anomaly detection using time-evolving graphs. *arXiv preprint arXiv:2210.08847*, 2022.

[12] Siddharth Bhatia, Bryan Hooi, Minji Yoon, Kijung Shin, and Christos Faloutsos. Midas: Microcluster-based detector of anomalies in edge streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3242–3249, 2020.

[13] Siddharth Bhatia, Arjit Jain, Pan Li, Ritesh Kumar, and Bryan Hooi. Mstream: Fast anomaly detection in multi-aspect streams. In *Proceedings of the Web Conference 2021*, pages 3371–3382, 2021.

[14] Siddharth Bhatia, Rui Liu, Bryan Hooi, Minji Yoon, Kijung Shin, and Christos Faloutsos. Real-time anomaly detection in edge streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(4):1–22, 2022.

[15] Nieves R Brisaboa, Diego Caro, Antonio Farina, and M Andrea Rodriguez. Using compressed suffix-arrays for a compact representation of temporal-graphs. *Information Sciences*, 465:459–483, 2018.

[16] Luiz FA Brito, Bruno AN Travençolo, and Marcelo K Albertini. A review of in-memory space-efficient data structures for temporal graphs. *arXiv preprint arXiv:2204.12468*, 2022.

[17] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[18] Lei Cai, Zhengzhang Chen, Chen Luo, Jiaping Gui, Jingchao Ni, Ding Li, and Haifeng Chen. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the 30th ACM international conference on Information & Knowledge Management*, pages 3747–3756, 2021.

[19] Ece Calikus, Sławomir Nowaczyk, Anita Sant'Anna, and Onur Dikmen. No free lunch but a cheaper supper: A general framework for streaming anomaly detection. *Expert Systems with Applications*, 155:113453, 2020.

[20] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[21] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. In *Proceedings of the 10th International Conference on Ad-Hoc, Mobile, and Wireless Networks*, ADHOC-NOW'11, page 346–359, Berlin, Heidelberg, 2011. Springer-Verlag.

[22] Yen-Yu Chang, Pan Li, Rok Sosic, MH Afifi, Marco Schweighauser, and Jure Leskovec. F-fade: Frequency factorization for anomaly detection in edge streams. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 589–597, 2021.

[23] Zhengzhang Chen, William Hendrix, and Nagiza F Samatova. Community-based anomaly detection in evolutionary networks. *Journal of Intelligent Information Systems*, 39(1):59–85, 2012.

[24] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. Deep learning for anomaly detection in time-series data: review, analysis, and guidelines. *IEEE Access*, 2021.

[25] Kun Dong, Austin R Benson, and David Bindel. Network density of states. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1152–1161, 2019.

[26] J R Driscoll, N Sarnak, D D Sleator, and R E Tarjan. Making data structures persistent. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, page 109–121, New York, NY, USA, 1986. Association for Computing Machinery.

[27] Ian L Dryden and Kanti V Mardia. *Statistical shape analysis: with applications in R*, volume 995. John Wiley & Sons, 2016.

[28] Ramzi Hamid Elghanuni, Rabab Alayham Abbas Helmi, and Muhammad Irsyad Abdullah. Improved detection of facebook anomalies and abnormalities using graph-based and machine learning techniques. In *AIP Conference Proceedings*, volume 2398, page 050001. AIP Publishing LLC, 2022.

[29] Farida Enikeeva and Olga Klopp. Change-point detection in dynamic networks with missing links. *arXiv preprint arXiv:2106.14470*, 2021.

[30] Gilberto Fernandes, Joel JPC Rodrigues, Luiz Fernando Carvalho, Jalal F Al-Muhtadi, and Mario Lemes Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70(3):447–489, 2019.

[31] Jefferson Viana Fonseca Abreu, Célia Ghedini Ralha, and João José Costa Gondim. A multi-agent approach for online twitter bot detection. 2021.

[32] Peng Gao, Gu Feng, and Fei Liang. Anomaly detection in dynamic graph based on deep graph auto-encoder. In *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, pages 317–320. IEEE, 2022.

[33] Xubo Gao, Qiusheng Zheng, Didier A Vega-Oliveros, Leandro Anghinoni, and Liang Zhao. Temporal network pattern identification by community modelling. *Scientific Reports*, 10(1):1–12, 2020.

[34] Arnab Kumar Ghoshal and Nabanita Das. Anomaly detection in evolutionary social networks leveraging community structure. In *2021 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, pages 1–6. IEEE, 2021.

[35] Arnab Kumar Ghoshal, Nabanita Das, and Soham Das. A fast community-based approach for discovering anomalies in evolutionary networks. In *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 455–463. IEEE, 2022.

[36] Xingzhi Guo, Baojian Zhou, and Steven Skiena. Subset node anomaly tracking over large dynamic graphs. *arXiv preprint arXiv:2205.09786*, 2022.

[37] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and data Engineering*, 26(9):2250–2267, 2013.

[38] Sahand Hariri, Matias Carrasco Kind, and Robert J Brunner. Extended isolation forest. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1479–1489, 2019.

[39] Malik Khizar Hayat and Ali Daud. Anomaly detection in heterogeneous bibliographic information networks using co-evolution pattern mining. *Scientometrics*, 113(1):149–175, 2017.

[40] Hossein Hazrati-Marangaloo and Rassoul Noorossana. Detecting outbreaks in temporally dependent networks. *Quality and Reliability Engineering International*, 35(6):1753–1765, 2019.

[41] Mingyi He, Shivam Pathak, Urwa Muaz, Jingtian Zhou, Saloni Saini, Sergey Malinchik, and Stanislav Sobolevsky. Pattern and anomaly detection in urban temporal networks. *arXiv preprint arXiv:1912.01960*, 2019.

[42] Nicholas A Heard, David J Weston, Kiriaki Platanioti, and David J Hand. Bayesian anomaly detection methods for social networks. *The Annals of Applied Statistics*, 4(2):645–662, 2010.

[43] Michael Heigl, Kumar Ashutosh Anand, Andreas Urmann, Dalibor Fiala, Martin Schramm, and Robert Hable. On the improvement of the isolation forest algorithm for outlier detection with streaming data. *Electronics*, 10(13):1534, 2021.

[44] Thomas J Helling, Johannes C Scholtes, and Frank W Takes. A community-aware approach for identifying node anomalies in complex networks. In *Complex Networks and Their Applications VII: Volume 1 Proceedings The 7th International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2018 7*, pages 244–255. Springer, 2019.

[45] Isuru Udayangani Hewapathirana and Dominic Lee. Combining information from multiple views for vertex-based change detection in dynamic networks: A comparative study. *SN Computer Science*, 3(2):1–29, 2022.

[46] Isuru Udayangani Hewapathirana, Dominic Lee, Elena Moltchanova, and Jeanette McLeod. Change detection in noisy dynamic networks: a spectral embedding approach. *Social Network Analysis and Mining*, 10(1):1–22, 2020.

[47] Shenyang Huang, Yasmeen Hitti, Guillaume Rabusseau, and Reihaneh Rabbany. Laplacian change point detection for dynamic graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 349–358, 2020.

[48] Shenyang Huang, Guillaume Rabusseau, and Reihaneh Rabbany. Scalable change point detection for dynamic graphs. 2021.

[49] Yan Jiang and Guannan Liu. Two-stage anomaly detection algorithm via dynamic community evolution in temporal graph. *Applied Intelligence*, pages 1–19, 2022.

[50] Fei Jie, Chunpai Wang, Feng Chen, Lei Li, and Xindong Wu. Block-structured optimization for anomalous pattern detection in interdependent networks. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1138–1143. IEEE, 2019.

[51] Sevvandi Kandanaarachchi and Rob J Hyndman. Anomaly detection in dynamic networks. *arXiv preprint arXiv:2210.07407*, 2022.

[52] Wasim Khan and Mohammad Haroon. A pilot study and survey on methods for anomaly detection in online social networks. In *Human-Centric Smart Computing*, pages 119–128. Springer, 2023.

[53] Hwan Kim, Byung Suk Lee, Won-Yong Shin, and Sungsu Lim. Graph anomaly detection with graph neural networks: Current status and challenges. *IEEE Access*, 2022.

[54] Mert Kosan, Arlei Silva, Sourav Medya, Brian Uzzi, and Ambuj Singh. Event detection on dynamic graphs. *arXiv preprint arXiv:2110.12148*, 2021.

[55] Sofiane Lagraa, Karima Amrouche, Hamida Seba, et al. A simple graph embedding for anomaly detection in a stream of heterogeneous labeled graphs. *Pattern Recognition*, 112:107746, 2021.

[56] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.

[57] Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8:1–29, 2018.

[58] Gen Li and Jason J Jung. Dynamic relationship identification for abnormality detection on financial time series. *Pattern Recognition Letters*, 145:194–199, 2021.

[59] Huichun Li, Xue Zhang, and Chengli Zhao. Explaining social events through community evolution on temporal networks. *Applied Mathematics and Computation*, 404:126148, 2021.

[60] Jinbo Li, Hesam Izakian, Witold Pedrycz, and Iqbal Jamal. Clustering-based anomaly detection in multivariate time series data. *Applied Soft Computing*, 100:106919, 2021.

[61] Panagiotis Liakos, Katia Papakonstantinopoulou, Theodore Stefou, and Alex Delis. On compressing temporal graphs. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1301–1313. IEEE, 2022.

[62] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[63] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, et al. Pygod: A python library for graph outlier detection. *arXiv preprint arXiv:2204.12095*, 2022.

[64] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[65] Thomas Magelinski and Kathleen M Carley. Community-based time segmentation from network snapshots. *Applied Network Science*, 4(1):1–19, 2019.

[66] Asep Maulana and Martin Atzmueller. Centrality-based anomaly detection on multi-layer networks using many-objective optimization. In *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*, volume 1, pages 633–638. IEEE, 2020.

[67] Asep Maulana and Martin Atzmueller. Many-objective optimization for anomaly detection on multi-layer complex interaction networks. *Applied Sciences*, 11(9):4005, 2021.

[68] Michele Mazza, Stefano Cresci, Marco Avvenuti, Walter Quattrociocchi, and Maurizio Tesconi. Rtbust: Exploiting temporal patterns for botnet detection on twitter. In *Proceedings of the 10th ACM conference on web science*, pages 183–192, 2019.

[69] Pablo Moriano, Jorge Finke, and Yong-Yeol Ahn. Community-based event detection in temporal networks. *Scientific reports*, 9(1):1–9, 2019.

[70] Trishita Mukherjee and Rajeev Kumar. Localized community-based node anomalies in complex networks. In Manoj Thakur, Samar Agnihotri, Bharat Singh Rajpurohit, Millie Pant, Kusum Deep, and Atulya K. Nagar, editors, *Soft Computing for Problem Solving*, pages 679–689, Singapore, 2023. Springer Nature Singapore.

[71] Hung T Nguyen, Pierre J Liang, and Leman Akoglu. Detecting anomalous graphs in labeled multi-graph databases. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2022.

[72] Dorcas Ofori-Boateng, Yulia R Gel, and Ivor Cribben. Nonparametric anomaly detection on time series of graphs. *Journal of Computational and Graphical Statistics*, 30(3):756–767, 2021.

[73] Łukasz Oliwa and Jarosław Koźlak. Anomaly detection in dynamic social networks for identifying key events. In *2017 International Conference on Behavioral, Economic, Socio-cultural Computing (BESC)*, pages 1–6. IEEE, 2017.

[74] Dhaval Patel, Giridhar Ganapavarapu, Srideepika Jayaraman, Shuxin Lin, Anuradha Bhamidipaty, and Jayant Kalagnanam. Anomalykits: Anomaly detection toolkit for time series. In *AAAI*. AAAI Press, 2022.

[75] Ramesh Paudel and H Howie Huang. Pikachu: Temporal walk based dynamic graph embedding for network anomaly detection. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7. IEEE, 2022.

[76] Ştefan Postăvaru, Anton Tsitsulin, Filipe Miguel Gonçalves de Almeida, Yingtao Tian, Silvio Lattanzi, and Bryan Perozzi. Instantembedding: Efficient local node representations. *arXiv preprint arXiv:2010.06992*, 2020.

[77] Aikaterini Protogerou, Stavros Papadopoulos, Anastasios Drosou, Dimitrios Tzovaras, and Ioannis Refanidis. A graph neural network method for distributed anomaly detection in iot. *Evolving Systems*, 12(1):19–36, 2021.

[78] Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(3):223–247, 2015.

[79] Giulio Rossetti. Rdyn: graph benchmark handling community dynamics. *Journal of Complex Networks*, 5(6):893–912, 2017.

[80] David Savage, Xiuzhen Zhang, Xinghuo Yu, Pauline Chou, and Qingmai Wang. Anomaly detection in online social networks. *Social networks*, 39:62–70, 2014.

[81] Imen Souiden, Mohamed Nazih Omri, and Zaki Brahmi. A survey of outlier detection in high dimensional data streams. *Computer Science Review*, 44:100463, 2022.

[82] Guillaume Staerman, Pavlo Mozharovskyi, Stephan Clémençon, and Florence d'Alché Buc. Functional isolation forest. In *Asian Conference on Machine Learning*, pages 332–347. PMLR, 2019.

[83] Deborah Sulem, Henry Kenlay, Mihai Cucuringu, and Xiaowen Dong. Graph similarity learning for change-point detection in dynamic networks. *arXiv preprint arXiv:2203.15470*, 2022.

[84] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 687–696, 2007.

[85] Srikanth Thudumu, Philip Branch, Jiong Jin, and Jugdutt Jack Singh. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7(1):1–30, 2020.

[86] Maurras Ulbricht Togbe, Mariam Barry, Aliou Boly, Yousra Chabchoub, Raja Chiky, Jacob Montiel, and Vinh-Thuy Tran. Anomaly detection for data streams based on isolation forest using scikit-multiflow. In *International Conference on Computational Science and Its Applications*, pages 15–30. Springer, 2020.

[87] Putra Wanda and Huang J Jie. Deepfriend: finding abnormal nodes in online social networks using dynamic deep learning. *Social Network Analysis and Mining*, 11(1):1–12, 2021.

[88] Huan Wang, Qing Gao, Hao Li, Hao Wang, Liping Yan, and Guanghua Liu. A structural evolution-based anomaly detection method for generalized evolving social networks. *The Computer Journal*, 65(5):1189–1199, 2022.

[89] Yu Wang, Aniket Chakrabarti, David Sivakoff, and Srinivasan Parthasarathy. Fast change point detection on dynamic social networks. *arXiv preprint arXiv:1705.07325*, 2017.

[90] James D Wilson, Nathaniel T Stevens, and William H Woodall. Modeling and detecting change in temporal networks via the degree corrected stochastic block model. *Quality and Reliability Engineering International*, 35(5):1363–1378, 2019.

[91] Feiran Xu and Ramin Moghaddass. A scalable bayesian framework for large-scale sensor-driven network anomaly detection. *IISE Transactions*, pages 1–18, 2022.

[92] Chenming Yang, Hui Wen, Bryan Hooi, Yue Wu, and Liang Zhou. A multi-scale reconstruction method for the anomaly detection in stochastic dynamic networks. *Neurocomputing*, 518:482–495, 2023.

[93] Yasser Yasami. Anomaly detection in dynamic complex networks. In *Modern and Interdisciplinary Problems in Network Science*, pages 239–263. CRC Press, 2018.

[94] Xiaoming Ye, Shaojie Qiao, Nan Han, Kun Yue, Tao Wu, Li Yang, Faliang Huang, and Chang-an Yuan. Algorithm for detecting anomalous hosts based on group activity evolution. *Knowledge-Based Systems*, 214:106734, 2021.

[95] Minji Yoon, Bryan Hooi, Kijung Shin, and Christos Faloutsos. Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 647–657, 2019.

[96] Minji Yoon, Jinhong Jung, and U Kang. Tpa: Fast, scalable, and accurate method for approximate random walk with restart on billion scale graphs. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1132–1143. IEEE, 2018.

[97] Susik Yoon, Youngjun Lee, Jae-Gil Lee, and Byung Suk Lee. Adaptive model pooling for online deep anomaly detection from a complex evolving data stream. *arXiv preprint arXiv:2206.04792*, 2022.

[98] Lanlan Yu, Biao Wang, Luojie Huang, Zhen Dai, Yang Yang, Yan Chen, and Ping Li. Detecting change points in dynamic networks by measuring cluster stability. *International Journal of Modern Physics C*, 32(09):2150123, 2021.

[99] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2672–2681, 2018.

[100] Daniele Zambon, Cesare Alippi, and Lorenzo Livi. Concept drift and anomaly detection in graph streams. *IEEE transactions on neural networks and learning systems*, 29(11):5592–5605, 2018.

[101] Zifeng Zhao, Li Chen, and Lizhen Lin. Change-point detection in dynamic networks via graphon estimation. *arXiv preprint arXiv:1908.01823*, 2019.

[102] Li Zheng, Zhenpeng Li, Jian Li, Zhao Li, and Jun Gao. Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn. In *IJCAI*, pages 4419–4425, 2019.

[103] Ruizhi Zhou, Qin Zhang, Peng Zhang, Lingfeng Niu, and Xiaodong Lin. Anomaly detection in dynamic attributed networks. *Neural Computing and Applications*, 33(6):2125–2136, 2021.

[104] Tingting Zhu, Ping Li, Lanlan Yu, Kaiqi Chen, and Yan Chen. Change point detection in dynamic networks based on community identification. *IEEE Transactions on Network Science and Engineering*, 7(3):2067–2077, 2020.