Anomaly detection in dynamic networks: a survey

Stephen Ranshous,[1,2] Shitian Shen,[1,2] Danai Koutra,[3] Steve Harenberg,[1,2] Christos Faloutsos[3] and Nagiza F. Samatova[1,2*]

Anomaly detection is an important problem with multiple applications, and thus has been studied for decades in various research domains. In the past decade there has been a growing interest in anomaly detection in data represented as networks, or graphs, largely because of their robust expressiveness and their natural ability to represent complex relationships. Originally, techniques focused on anomaly detection in static graphs, which do not change and are capable of representing only a single snapshot of data. As real-world networks are constantly changing, there has been a shift in focus to dynamic graphs, which evolve over time.

In this survey, we aim to provide a comprehensive overview of anomaly detection in dynamic networks, concentrating on the state-of-the-art methods. We first describe four types of anomalies that arise in dynamic networks, providing an intuitive explanation, applications, and a concrete example for each. Having established an idea for what constitutes an anomaly, a general two-stage approach to anomaly detection in dynamic networks that is common among the methods is presented. We then construct a two-tiered taxonomy, first partitioning the methods based on the intuition behind their approach, and subsequently subdividing them based on the types of anomalies they detect. Within each of the tier one categories—community, compression, decomposition, distance, and probabilistic model based—we highlight the major similarities and differences, showing the wealth of techniques derived from similar conceptual approaches. © 2015 The Authors. *WIREs Computational Statistics* published by Wiley Periodicals, Inc.

How to cite this article:
WIREs Comput Stat 2015. doi: 10.1002/wics.1347

Keywords: anomaly detection; dynamic networks; outlier detection; graph mining; dynamic network anomaly detection; network anomaly detection

INTRODUCTION

A network[a] is a powerful way to represent a collection of objects and the relationships or connections among them. Examples include global financial systems connecting banks across the world, electric power grids connecting geographically distributed areas, and social networks that connect users, businesses, or customers using relationships such as friendship, collaboration, or transactional interactions. These are examples of *dynamic networks*, which, unlike static networks, are constantly undergoing changes to their structure or attributes. Possible changes include insertion and deletion of vertices (objects), insertion and deletion of edges (relationships), and modification of attributes (e.g., vertex or edge labels).

An important problem over dynamic networks is *anomaly detection*—finding objects, relationships, or

*Correspondence to: samatova@csc.ncsu.edu

[1]Department of Computer Science, North Carolina State University, Raleigh, NC, USA

[2]Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA

[3]Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Conflict of interest: The authors have declared no conflicts of interest for this article.

points in time that are unlike the rest. There are many high-impact and practical applications of anomaly detection spanning numerous domains. A small sample includes: detection of ecological disturbances, such as wildfires[1,2] and cyclones[3]; intrusion detection for individual systems[4] and network systems[5–7]; identifying abnormal users and events in communication networks[8,9]; and detecting civil unrest using twitter feeds.[10]

The ubiquitousness and importance of anomaly detection in dynamic networks has led to the emergence of dozens of methods over the last 5 years (see Tables 3 and 4). These methods complement techniques for static networks,[11–14] as the latter often cannot be easily adopted for dynamic networks. When considering the dynamic nature of the data, new challenges are introduced, including:

- New types of anomalies are introduced as a result of the graph evolving over time, for example, splitting, disappearing, or flickering communities.
- New graphs or updates that arrive over time need to be stored and analyzed. Storing all the new graphs in their entirety can vastly increase the size of the data. Therefore, typical offline analysis, where multiple passes over the data are acceptable and all of the data are assumed to fit into memory, becomes infeasible. Conversely, storing only the most recent graph or updates restricts the analysis to a single point in time.
- Graphs from different domains, such as social networks compared to gene networks, may exhibit entirely different behavior over time. This divergence in evolution can lead to application-specific anomalies and approaches.
- Anomalies, particularly those that are slow to develop and span multiple time steps, can be hard to differentiate from organic graph evolution.

No dedicated and comprehensive survey of anomaly detection in dynamic networks exists, despite the growing importance of the topic because of the increasing availability of network data. Although anomaly detection has been surveyed in a variety of domains,[15–19] it has only recently been touched upon in the context of dynamic networks.[20–22]

In this survey, we hope to bridge the gap between the increasing number of methods for anomaly detection in dynamic networks and the lack of their comprehensive analysis. First, we give a broad overview of the related work in graph mining, anomaly detection, and the high-level approaches used in the papers we discuss. We then introduce four different types of anomalies that these algorithms detect, namely, anomalous vertices, edges, subgraphs, and events. We continue with an extensive overview of the existing methods based on the proposed taxonomy, illustrated in Figure 5 that takes into account their underlying design principles, such as those based on graph communities, compression, decomposition, distance metrics, and probabilistic modeling of graph features. Each taxonomic group is then subcategorized further based on the types of anomalies detected. Finally, we end with a more in-depth discussion of the methods that have code publicly available (see Table 6), highlighting pros and cons of each.

BACKGROUND

Anomaly, or outlier, detection is a problem that spans many domains. Chandola et al.[16] provide an excellent overview, taxonomy, and analysis of a multitude of techniques (e.g., classification, clustering, and statistical) for a variety of domains, expanding the work of Hodge et al.[23] and Agyemang et al.[24] As our focus will be on graphs, it is important to have a basic understanding of graph theory. West et al.[25] and Balakrishnan et al.[26] both offer comprehensive and approachable introductions to graph theory, covering all of the basics required for this survey and well beyond. Cook and Holder[27] show how the theoretical concepts can be applied for graph mining, and recently Samatova et al.[28] provide an overview of many graph mining techniques as well as implementation details in the R programming language.[b] Owing to space limitations we cannot provide an introduction to the fundamentals of the types of methods we discuss, so we provide references for introductory and overview material for each of them in Table 1.

It is important to note that in many domains the data are naturally represented as a network, with the vertices and edges clearly defined. However, in

TABLE 1 | Introductory and Overview References

Method	References
Community detection	Fortunato,[29] Lancichinetti and Fortunato,[30] Reichardt and Bornholdt,[31] Harenberg et al.[32]
MDL and compression	Rissanen,[33–35] Grünwald[36]
Decomposition	Golub and Reinsch,[37] Klema and Laub,[38] Kolda and Bader[39]
Distance	Gao et al.,[40] Rahm and Bernstein,[41] Cook and Holder[27]
Probabilistic	Koller and Friedman,[42] Glaz[43]

some cases, how to represent the data as a network is unclear and can depend on the specific research question being asked. The conversion processes used in specific domains are outside the scope of this survey, and we assume all data are already represented as dynamic networks.

TYPES OF ANOMALIES

In this section, we identify and formalize four types of anomalies that arise in dynamic networks. These categories represent the output of the methods, not the implementation details of how they detect the anomalies, e.g., comparing consecutive time points, using a sliding window technique. Note that often times in real-world graphs, (e.g., social and biological networks) the graph's vertex set V is called a set of nodes. However, to avoid confusion with nodes in a physical computer network, and to align with the abstract mathematical representation, we call it a set of vertices.

Because the graphs are assumed to be dynamic, vertices and edges can be inserted or removed at every time step. For the sake of simplicity, we assume that the vertex correspondence and the edge correspondence across different time steps are resolved because of unique labeling of vertices and edges, respectively. We define a *graph series* $\mathbf{G}$ as an ordered set of graphs with a fixed number of time steps. Formally, $\mathbf{G} = \{\mathbf{G}_t\}_{t=1}^{T}$, where T is the total number of time steps, $\mathbf{G}_t = (V_t, E_t \subseteq (V_t \times V_t))$, and the vertex set V_t and edge set E_t may be plain or attributed (labeled). Graph series where $T \to \infty$ are called *graph streams*. The full set of notations can be found in Table 2. In the following subsections, we start with an intuitive explanation of the problem definition, then give a general formal definition of the anomaly type, continue with some applications, and conclude with a representative example.

TABLE 2 | Notation Summary

Symbol	Meaning
$\mathbf{G}$	Graph series with a fixed number of time points
$\mathbf{G}_t$	Snapshot of the graph series at time t
$\mathbf{G}^s$	The sth graph segment, a grouping of temporally adjacent graphs
I_t	Vertex partitioning at time t, separating all the vertices into disjoint sets
V_t	Vertex set for the graph at time point t
v_i	Vertex i
E_t	Edge set for the graph at time point t
$e_{i,j}$	Edge between v_i and v_j
c_0	Threshold value for normal versus anomalous data

Type 1: Anomalous Vertices

Anomalous vertex detection aims to find a subset of the vertices such that every vertex in the subset has an 'irregular' evolution compared to the other vertices in the graph. Optionally, the time point(s) where the vertices are determined to be anomalous can be identified. What constitutes irregular behavior is dependent on the specific method, but it can be generalized by assuming that each method provides a function that scores each vertex's behavior, e.g., measuring the change in the degree of a vertex from time step to time step. In static graphs, the single snapshot allows only intragraph comparisons to be made, such as finding vertices with an abnormal egonet density.[11] Dynamic graphs allow the temporal dynamics of the vertex to be included, introducing new types of anomalies that are not present in static graphs. A high-level definition for a set of anomalous vertices is as follows.

Definition 1. *(Anomalous vertices).* Given $\mathbf{G}$, the total vertex set $V = \cup_{t=1}^{T} V_t$, and a specified scoring function $f : V \to \mathbb{R}$, the set of anomalous vertices $V' \subseteq V$ is a vertex set such that $\forall v' \in V'$, $|f(v') - \hat{f}| > c_0$, where $\hat{f}$ is a summary statistic of the scores $f(v)$, $\forall v \in V$.

An example of an anomalous vertex is shown in Figure 1. Two time steps are shown, both with 10 vertices and 2 communities. In this example, anomalous vertices are those that have a substantial change in their community involvement compared to the rest of the vertices in the graph. As v_6 is the only vertex that has any change in its community involvement, it is labeled as an anomaly. Formally, we can measure the change in community involvement between adjacent time steps t and $t-1$ by letting $f(v) = \sum_{i=1}^{|C|} c_{i,v}^{t-1} \oplus c_{i,v}^{t}$, where $C = \{c_1, c_2, \ldots, c_{|C|}\}$ is the set of communities, $c_{i,v}^{t} = 1$ if v is part of community c_i at time step t and 0 otherwise, and $\oplus$ is the xor operator.

The scoring function f will depend on the application. In the example shown in Figure 1, the vertices were scored based on the change in community involvement. However, if the objective is identifying computers on a network that become infected and part of a botnet, then an appropriate scoring function might be measuring the change in the number of edges each vertex has between time steps, or the change in the weights of the edges.

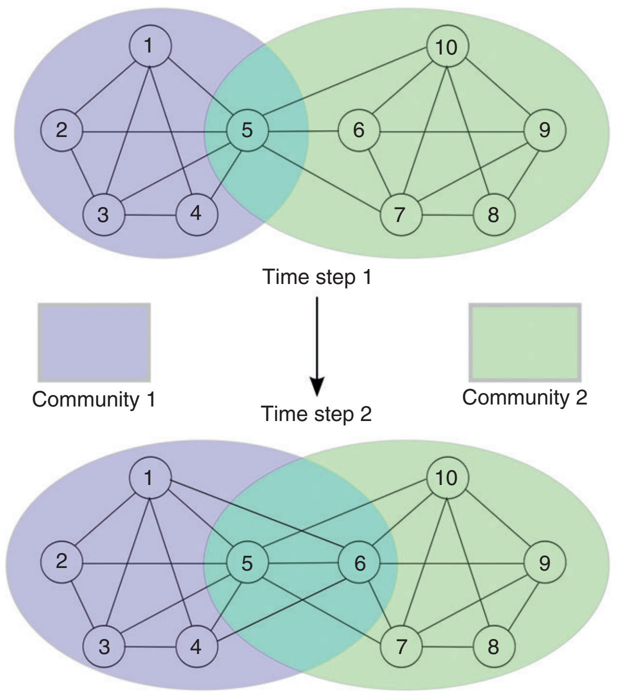

FIGURE 1 | Example of an anomalous vertex. A dynamic graph with two time steps showing a vertex, v_6, that is found to be anomalous based on its change in community involvement. In time step 1, v_6 is only part of community 2, whereas in time step 2 it is part of both communities 1 and 2. As no other vertex's community involvement changes, v_6 is deemed anomalous.

Typical applications of this type of anomaly detection are identifying vertices that contribute the most to a discovered event (also known as attribution), such as in communication networks,[44] and observing the shifts in community involvement.[45,46]

Type 2: Anomalous Edges

Similar to vertex detection, edge detection aims to find a subset of the edges such that every edge in the subset has an 'irregular' evolution, optionally identifying the time point(s) where they are abnormal. Again, this concept can be generalized by assuming each method employs a function that maps each edge in the graph to a real number, low values indicating unusual behavior. In a static graph, a distribution of the edge weights can be found, and each edge can be assigned a score according to the probability of its weight. However, because of the temporal nature of dynamic graphs, two new main types of irregular edge evolution can be found: (1) abnormal edge weight evolution,[47] where the weight of a single edge fluctuates over time and has inconsistent spikes in value, and (2) appearance of unlikely edges in a graph between two vertices that are not typically connected or part of the same community.[48,49] Anomalous edge detection can be formally defined as follows.

Definition 2. *(Anomalous edges)*. Given **G**, the total edge set $E = \cup_{t=1}^{T} E_t$, and a specified scoring function $f : E \to \mathbb{R}$, the set of anomalous edges $E' \subseteq E$ is an edge set such that $\forall e' \in E'$, $|f(e') - \hat{f}| > c_0$, where $\hat{f}$ is a summary statistic of the scores $f(e)$, $\forall e \in E$.

Figure 2 shows an example of edges that exhibit irregular edge weight evolutions. Over the five time steps there are two anomalous edges, both having a single time point that is unlike the rest of the series. Letting $f(e) = |w_t(e) - w_{t-1}(e)| + |w_t(e) - w_{t+1}(e)|$, where $w_t(e)$ is the edge weight at time step t, each edge is scored based on its current weight compared to the previous and following weight. Abnormally large changes in the weight of an edge results in it is being flagged as anomalous. For example, at time step 2, using the mean change in weights $\hat{f} = 0.43$ as a summary statistic and $c_0 = 0.10$ as a threshold, results in edge (3, 4) being declared anomalous.

While our example used a simple thresholding approach for edge weights, various other scoring functions can be used. Finding unlikely social interactions can be accomplished by modeling the edges between people as the number of times they communicate daily, and then scoring each edge at each time step based on the probability of it (1) being present, and (2) having a particular weight.[50] Another application is identifying anomalous traffic patterns in a graph where vertices represent road intersections and edges represent the

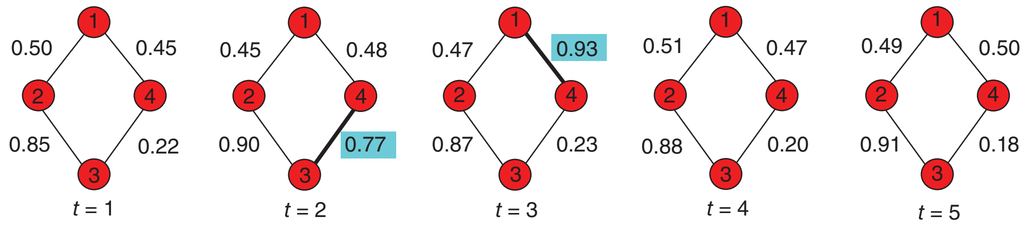

FIGURE 2 | An illustration of anomalous edges that occur because of an irregular pattern of their weight over time, with anomalous edges highlighted. At each time step, a vertex's weight typically shifts by ± 0.05 at most. However, edge (3, 4) has a spike in its weight at time step 2, unlike any other time in the series. Similarly, at time step 3, edge (1, 4) spikes in value. These spikes cause the edges to be considered anomalous.

roads themselves. Projecting the edges into a feature space based on their average speed of traffic throughout the day, the pairwise similarity between every edge can be calculated. Each edge is then scored based on the change in the similarity between itself and every other edge.[47]

Type 3: Anomalous Subgraphs

Finding subgraphs with irregular behavior requires an approach different from the ones for anomalous vertices or edges, as enumerating every possible subgraph in even a single graph is an intractable problem. Hence, the subgraphs that are tracked or identified are typically constrained, for example, to those found by community detection methods. In these cases, matching algorithms are required to track the subgraphs across time steps, such as the community matching technique used in Ref 51. Once a set of subgraphs has been determined, intragraph comparisons or intertime point comparisons can be made to assign scores to each subgraph, e.g., measuring the change in the total weight of the subgraph between adjacent time steps. Typical intragraph comparison methods, such as finding unique substructures in the graph that inhibit compressibility,[14] must be extended to incorporate the additional information gained by using a dynamic network. Instead of finding structures that are unique *within* a single graph, structures that are unique *to* a single graph *within a series of graphs* can be found. Anomalies of this type, unique to dynamic networks, include communities that split, merge, disappear, and reappear frequently, or exhibit a number of other behaviors.

Definition 3. *(Anomalous subgraphs)*. Given G, a subgraph set $H = \cup_{t=1}^{T} H_t$ where $H_t \subseteq G_t$, and a specified scoring function $f: H \to \mathbb{R}$, the set of anomalous subgraphs $H' \subseteq H$ is a subgraph set such that $\forall\, h' \in H'$, $|f(h') - \hat{f}| > c_0$, where $\hat{f}$ is a summary statistic of the scores $f(h)$, $\forall\, h \in H$.

Two specific types of anomalous subgraphs are shown in Figure 3. A *shrunken community*, Figure 3(a), is when a single community loses a significant number of its vertices between time steps. Assuming that a matching of communities between adjacent time steps t and $t-1$ is known, shrunken communities can be identified by measuring the decrease in the number of vertices in the community, $f(h) = |h_t \cap h_{t-1}| - |h_{t-1}|$. Figure 3(b) is an example of a *split community*, when a single community divides into several distinct communities. Split communities will have a high matching score or probability for more than one community in the next graph of the

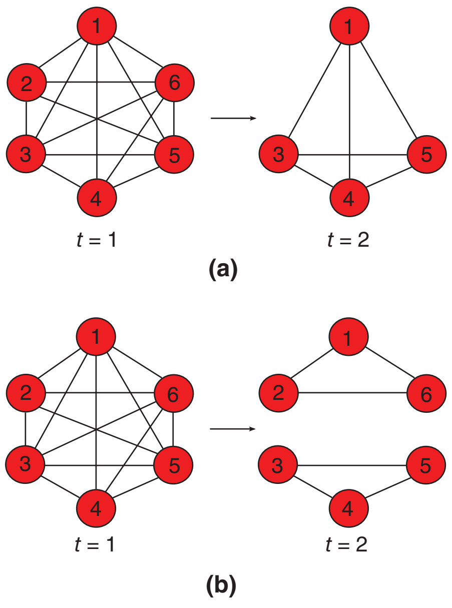

FIGURE 3 | Two different types of anomalous subgraphs. (a) Shrunken community shows a shrunken community, where a community loses members from one time step to the next. (b) Split community shows a split community, where a single community breaks into several distinct smaller communities.

series. Therefore, split communities can be found by examining two adjacent time steps and letting f score each community in $t-1$ as the number of communities it is matched to in t.

What constitutes an anomalous subgraph is heavily dependent on the application domain. Automatic identification of accidents in a traffic network can be done by letting edge weights represent outlier scores, then mining the heaviest dynamic subgraph (HDS).[52] Similarly, changes and threats in social networks can be found by running community detection on a reduced graph composed of suspected anomalous vertices[50] and performing an Eigen decomposition on a residual graph.[53]

Type 4: Event and Change Detection

Unlike the previous three types of anomalies, the two types discussed here are exclusively found in dynamic graphs. We start by defining the problem of event detection, which has attracted much interest in the data mining community. Event detection has a much broader scope compared to the previous three types of anomalies, aiming to identify time points that are significantly different from the rest. Isolated points in time where the graph is unlike the graphs at the previous and following time points represent events. One approach to measuring the similarity of two graphs is comparing the signature vector of summary

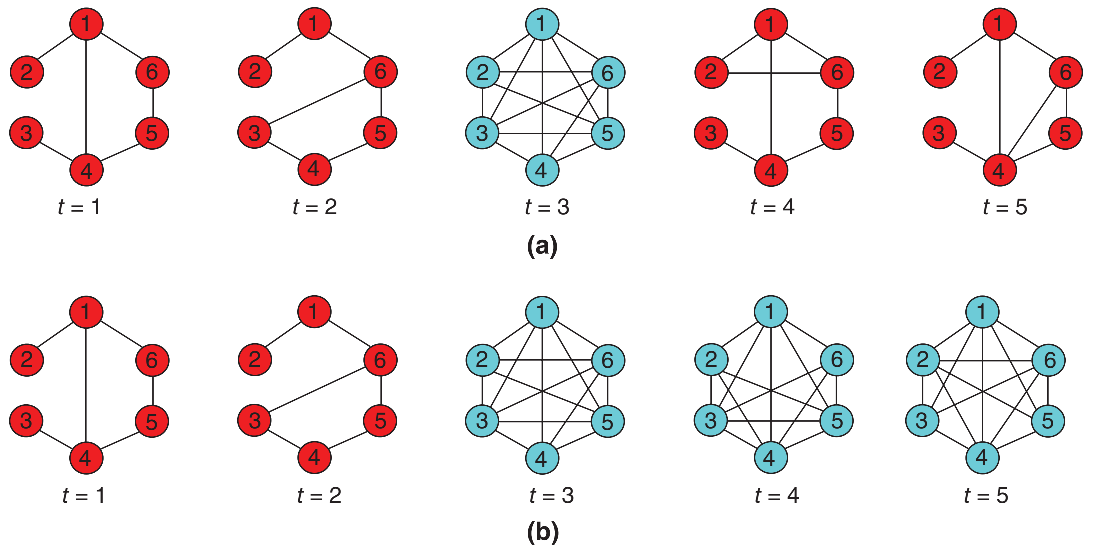

FIGURE 4 | An example of an event (a), a change (b), and the difference between them. In both, the graphs initially have only a few edges. At time step 3, the graphs become highly connected, almost forming a clique. The major difference is observed at the time step following the insertion of the edges. In Figure 4(a) the edges are then removed and the graph returns to a state similar to the one before the insertion, representing an isolated event. In Figure 4(b), however, the graph maintains its new state indicating the sustained shift, or change, in the graph.

values extracted from each graph, such as average clustering coefficient. The task of attribution—finding the cause for the event—is not required as part of event detection, and is often omitted. However, once the time points for events have been identified, potential causes can be found by applying techniques for anomaly detection in static graphs.[11–14]

Definition 4. *(Event detection).* Given $\mathbf{G}$ and a scoring function $f: \mathbf{G}_t \to \mathbb{R}$, an event is defined as a time point t, such that $|f(\mathbf{G}_t) - f(\mathbf{G}_{t-1})| > c_0$ and $|f(\mathbf{G}_t) - f(\mathbf{G}_{t+1})| > c_0$.

One of the simplest similarity metrics for graphs is comparing the number of vertices and edges in them. In Figure 4(a), the event at time step 3 can be found by counting the edges in each graph, $f(G_t) = |E_t|$, and comparing the adjacent time points. The number of edges in each graph, {6, 6, 14, 7, 7}, shows that there is a significant change in the structure of the graph at time step 3, almost forming a clique. Note that this time point is isolated, with the surrounding time points being very dissimilar, indicating an event occurred.

Event detection, while providing less specific information than vertex, edge, or subgraph detection, is highly applicable in many areas. Approximating the data available using a tensor decomposition, then scoring the time point as the amount of error in the reconstruction, has been shown to effectively detect moments in time when the collective motions in molecular dynamics simulations change.[54] Other applications include finding frames in a video that are unlike the others,[55,56] and detecting disturbances in the ecosystem (e.g., wildfires).[1]

Now, we move on to the problem of change detection, which is complementary to event detection. It is important to note the distinction between event and change detection. While events represent isolated incidents, change points mark a point in time where the entire behavior of the graph changes and the difference is maintained until the next change point, Figure 4 illustrates the difference between the two. The distinction between events and change points manifests in the modification of the constraint that the value of the scoring function at t be more than a threshold value away from both $t-1$ and $t+1$, to being more than a threshold value away from *only* $t-1$.

Definition 5. *(Change detection).* Given $\mathbf{G}$ and a scoring function $f: \mathbf{G}_t \to \mathbb{R}$, a change is defined as a time point t, such that $|f(\mathbf{G}_t) - f(\mathbf{G}_{t-1})| > c_0$ and $|f(\mathbf{G}_t) - f(\mathbf{G}_{t+1})| \le c_0$.

An example of change detection is shown in Figure 4(b). Similar to event detection, this change can be identified by comparing the number of edges at adjacent time points. Just as in Figure 4(a), a number of edges are added to the graph, forming nearly a clique. However, at time step 4, instead of the graph reverting back to its original structure, the added edges are persistent and the graph has a new 'normal' behavior. The persistence of the new edges indicates that at $t = 3$ a change was detected, not an event.

One of the most popular applications of change detection is in networks modeling human interactions, such as communication and coauthorship networks. Authors that act as a 'bridge' between different conferences, or switch areas of interest throughout their career, will exhibit a number of change points in their publication records.[57] Additionally, change detection has been applied to communication graphs[44] and network traffic[58] by measuring the change in the Eigen decompositions.

METHODS

A common two-stage methodology was found among the papers reviewed. In the first stage, *data-specific features* are generated by applying a mapping from the domain-specific representation, graphs, to a common data representation, a vector of real numbers. A simple example is taking a single static graph as input and outputting its diameter. In the second stage, an *anomaly detector* is applied, taking the output from stage one, and optionally some historical data, and mapping it to a decision on the anomalousness of the data. Anomaly detectors, such as support vector machines (SVMs)[59] or the local outlier factor (LOF) algorithm,[60] are general methodologies that are domain-independent, as they operate on the common data representation that is output from stage one. Hence, stage two consists of identifying outlier points in a multidimensional space.[60–64] The two stages can be formally defined as follows:

$$\text{Stage one} \quad \Rightarrow \quad \Omega : D \rightarrow \mathbb{R}^d,$$

$$\text{Stage two} \quad \Rightarrow \quad f : \left\{ \left[\mathbb{R}^d\right]^*, \left[\mathbb{R}^d\right]^t \right\} \rightarrow \{0, 1\},$$

where D is the domain-specific representation, d is the number of feature dimensions, $[\mathbb{R}^d]^*$ denotes the history of the data, and $[\mathbb{R}^d]^t$ denotes the current time point. Often times, stage one maps the domain-specific data to a single value, $d=1$, such as the graph diameter example. Additionally, stage two can be generalized to mapping to [0, 1] instead of {0, 1}, representing the probability that the data are anomalous or a normalized outlier score assigned to the data.

This two-stage process can be found in many other areas of anomaly or outlier detection, such as text mining and abnormal document detection,[65] and computer vision.[66] However, in this survey we restrict the scope to anomaly detection in dynamic graphs. *The papers we surveyed are interested in finding the best mapping from dynamic graphs to a vector of real numbers*, then applying *existing* anomaly detectors to identify the anomalies. Equivalently, they are proposing methodologies for stage one, and applying existing methodologies for stage two. A complete list of the methods surveyed is provided in Tables 3 and 4, with the complexity notation provided in Table 5.

Community Detection

Community-based methods track the evolution of communities and their associated vertices in the graphs over time.[105–107] While the data-specific features generated by the methods discussed in this section are all derived using the community structure of the graphs, the various approaches differ in two main points: (1) in the aspects of the community structure they analyze, e.g., the connectivity within each community versus how the individual vertices are assigned to the communities at each time step, and (2) in the definitions of communities they use, e.g., *soft* communities where each vertex has a probability of belonging to each community versus *disjoint* communities where each vertex is placed into at most one community in the graph. Moreover, based on how the community evolution is viewed, it can be applied to the detection of different anomaly types. For example, a rapid expansion or contraction of a community could indicate that the specific *subgraph* for that community is undergoing drastic changes, whereas a drop in the number of communities by two corresponds to an abnormal event.

Example

In Ref 67, given a weighted directed graph series $\boldsymbol{G}$ and a vertex partitioning similarity function $Sim(I_s, I_t)$, the set of change points is defined as a set of time points $T : \{t \in T | Sim(I_s, I_t) \leq c_0\}$. The similarity function is defined as

$$Sim\left(I_s, I_t\right) = \frac{P_{I_s}\left(I_t\right) + P_{I_t}\left(I_s\right)}{2N}, \tag{1}$$

where $P_{I_s}\left(I_t\right) = \sum_{\mathrm{p}} \max_{c \in \mathrm{I_s} \cap \mathrm{I_t}} \left| I_{s_p} \cap c \right|$, I_s is the partitioning of the current graph segment $\boldsymbol{G^s}$ and I_t is the partitioning of the newly arrived graph $\boldsymbol{G_t}$, N is the number of vertices, p is the community index in a partitioning, c is the common community between the two partitionings. Mapping the similarity of two graphs to a single score via vertex partition similarity is the extraction of the data-specific feature, and the anomaly detector is the application of a threshold value. If the similarity is above a threshold c_0, then $\boldsymbol{G_t}$ will be put into the current graph segment $\boldsymbol{G^s}$, otherwise, $\boldsymbol{G_t}$ starts a new graph segment $\boldsymbol{G^{s+1}}$. The graph

TABLE 3 | Summary of Methods

Paper	V	E	SG	E/C	Year	Complexity[1]
Community						
Duan[67]				√	2009	$\mathcal{O}(n^3t)$
Aggarwal[68]			√		2012	$\mathcal{O}(mt)$
Chen[69]			√		2012	Exp.
Gupta[70]	√				2012	$\mathcal{O}(nk^2t)$
Gupta[71]	√				2012	$\mathcal{O}(ntk2^t)$
Chen[3]			√		2013	Exp.
Ji[46]	√				2013	$\mathcal{O}(n^3t)$
Rossi[72]	√				2013	$\mathcal{O}(mt)$
Araujo[73]			√		2014	$\mathcal{O}(mt)$
Compression						
Chakrabarti[74]		√			2004	$\mathcal{O}(mt)$
Sun[75]				√	2007	$\mathcal{O}(mt)$
Decomposition						
Ide[76]	√			√	2004	$\mathcal{O}(n^2t)$
Lakhina[77]				√	2004	$\mathcal{O}(tm^2)$
Sun[78]	√		√	√	2006	$\mathcal{O}(nr^3t)$
Sun[79]	√		√	√	2006	$\mathcal{O}(rt\lvert\mathcal{X}\rvert^{N_{\mathcal{X}}})$
Sun[80]	√		√	√	2007	$\mathcal{O}(nct + c^3t)$
Kolda[81]	√		√	√	2008	$\mathcal{O}(\lvert\mathcal{X}\rvert r^3t)$
Tong[82]	√		√	√	2008	$\mathcal{O}(mct + c^3t)$
Akoglu[44]	√			√	2010	$\mathcal{O}(n^2t)$
Ishibashi[58]	√				2010	$\mathcal{O}(n^2t)$
Jiang[83]	√			√	2011	$\mathcal{O}(nt^2 + n^2)$
Koutra[57]	√		√	√	2012	$\mathcal{O}(mt)$
Miller[84]				√	2012	$\mathcal{O}(mt)$
Papalexakis[85]	√		√	√	2012	$\mathcal{O}(mt)$
Miller[86]	√			√	2013	$\mathcal{O}(mt)$
Yu[87]	√				2013	$\mathcal{O}(mt)$

V, vertex; E, edge; SG, subgraph; C/E, change/event.
[1]The complexity stated is, in some cases, a coarse approximation provided for comparative purposes. The complexity shown is for the entire graph series.

is partitioned using a relevance matrix computed with random walks with restarts and modularity optimization.

Vertex Detection

A group of vertices that belong to the same community are expected to exhibit similar behavior. Intuitively this means that if at consecutive time steps one vertex in the community has a significant number of new edges added, the other vertices in the community would also have a significant number of new edges. If the rest of the vertices in the community did not have new edges added, the vertex that did is anomalous.

Based on this logic, using soft community matching and looking at each community individually, the average change in belongingness (the probability the vertex is part of the community) for each vertex can be found for consecutive time steps. Vertices whose change in belongingness is different from the average change of the vertices in the community are said to be *evolutionary community outliers*.[70]

The changes in a vertex's community belongingness may form a pattern over time, called a soft temporal pattern. In Ref 72, a non-negative matrix factorization approach in combination with the minimum description length (MDL) principle is used to detect automatically vertex roles and build transition

TABLE 4 | Summary of Methods (Cont.)

Paper	V	E	SG	E/C	Year	Complexity[1]
Distance						
Pincombe[88]				√	2005	$\mathcal{O}(m^2t)$
Gaston[89]				√	2006	$\mathcal{O}(n^3t)$
Li[47]		√			2009	$\mathcal{O}(m^2t)$
Abello[48]	√	√	√		2010	$\mathcal{O}(mt)$
Papadimitriou[90]				√	2010	$\mathcal{O}(mt)$
Arackaparambil[91]				√	2011	$\mathcal{O}(mt)$
Le[92]				√	2011	$\mathcal{O}(n^2t)$
Berlingerio[93]				√	2012	$\mathcal{O}(nt\log n)$
He[94]			√		2012	$\mathcal{O}(n^3t)$
Malliaros[95]				√	2012	$\mathcal{O}(n^2t)$
Koutra[96]				√	2013	$\mathcal{O}(mt)$
Mongiov[52]			√		2013	$\mathcal{O}(mt+m\log m)$
Mongiov[97]			√		2013	$\mathcal{O}(mt\log m)$
Probabilistic						
Priebe[8]	√	√			2005	$\mathcal{O}(n^3t)$
Huang[98]		√		√	2006	$\mathcal{O}(n^2t)$
Pandit[99]		√			2007	$\mathcal{O}(mt)$
Hirose[100]		√		√	2009	$\mathcal{O}(n^2t)$
Thompson[101]			√		2009	$\mathcal{O}(n^2t)$
Heard[50]	√	√	√	√	2010	$\mathcal{O}(n^2t)$
Djidjev[102]			√		2011	$\mathcal{O}((n+m)\,t)$
Aggarwal[103]		√		√	2011	$\mathcal{O}(mt)$
Neil[104]			√		2013	$\mathcal{O}(mn^{k-1}t)$

V, vertex; E, edge; SG, subgraph; C/E, change/event.
[1]The complexity stated is, in some cases, a coarse approximation provided for comparative purposes. The complexity shown is for the entire graph series.

models. The community memberships are found slightly differently in Ref 71, where Xmeans is used. However, in both cases, the patterns common across all of the vertices in the graph are extracted, then each vertex's patterns are compared to the extracted ones. If a vertex's patterns are not similar to any of the extracted common patterns, then the vertex is anomalous. Later extended to static networks derived from heterogeneous data sources,[108] the two-step approach in Ref 71 is modified to an alternating iterative approach in Ref 70. Instead of extracting patterns first and then identifying outliers, the patterns and outliers are found in an alternating fashion (pattern extraction → outlier detection → pattern extraction → ⋯) until the outliers discovered do not change on consecutive iterations. By alternating back and forth between pattern extraction and outlier detection, the algorithm accounts for the affect the outliers have on the communities discovered.

In Ref 46, communities manifest in the form of *corenets*. Instead of defining the corenet (community) based solely on density, modularity, or hop distance (as egonets are[11]), the definition is based on the weighted paths between vertices. More formally, a vertex's corenet consists of itself and all the vertices within two hops that have a weighted path above a threshold value. If the edge weight between two vertices is considered the strength of their connection, then intuitively the vertices connected with higher weight edges should be considered part of the same community. Consequently, if a vertex has two neighbors removed, one connected with a high edge weight and the other connected with a low edge weight, then the removal of the vertex connected by the higher edge weight should have more of an impact. At each time step every vertex is first given an outlier score based on the change in its corenet, and the top k outlier scores are then declared anomalous.

TABLE 5 | Algorithm Complexity Notation

Symbol	Meaning
n	Number of vertices in the graph
m	Number of edges in the graph
t	Number of time steps
k	Number of communities
$\mathcal{X}$	Tensor representing the graph
$\lvert\mathcal{X}\rvert$	Size of the largest mode in $\mathcal{X}$
r	Size of the largest mode of the core tensor approximation (e.g., in the Tucker decomposition[113])
$N_{\mathcal{X}}$	Number of modes in $\mathcal{X}$
c	Number of columns sampled for low rank approximation
Exp.	Exponential time complexity

Subgraph Detection

Instead of looking at individual vertices and their community belongingness, entire subgraphs that behave abnormally can be found by observing the behavior of communities themselves over time.

Six different types of community-based anomalies are proposed in Ref 69: shrink, grow, merge, split, born, and vanish. A graph at time t is compared to the graphs at time $t+1$ and $t-1$ using graph and community representatives. The representatives reduce the computation required, providing seeding points for community enumeration. Communities are defined as maximal cliques, hence enumerating all communities is NP-hard, and overlapping communities are allowed. Graph representatives are the set of vertices that have appeared in time step t in addition to $t+1$, $t-1$, or both; a community representative is a vertex that appears in the fewest number of the other communities. A set of rules are then derived based on these representatives that decide whether communities are anomalous, falling into one of the six defined classes, or normal. One additional type of community-based anomaly was proposed recently in Ref 73: comets, or communities that come and go, often periodically. The time-evolving graph is represented by a tensor (3-mode matrix), and the comets are detected by low rank tensor decomposition combined with the MDL principle that allows for parameter-free community search.

Conversely, instead of finding changes, communities that are conserved, or stable, can be identified. Constructing multiple networks at each time step based on different information sources, communities can be conserved across time and networks. Networks that behave similarly can be grouped using clustering or prior knowledge. In Ref 3, if a community is conserved across time steps and the networks within its group, but has no corresponding community in any other group of networks, then the community is defined as anomalous; two communities are considered corresponding communities if they have a certain percentage of their vertices in common.

Unlike Refs 3, 69, 73 that consider only the structure of the network, in social networks there is often more information available. For example, in the Twitter user network, clusters can be found based on the content of tweets (edges), as well as the users (vertices) involved. If the fraction of the tweets (edges) added during the recent time window for a cluster is much larger than the fraction of tweets (edges) added anytime before the window, then this influx is declared as an evolution event for that cluster.[68] A cluster that experiences an evolution event is marked as an anomalous subgraph at the time when the evolution event occurs. Although the authors did not use labeled datasets in Ref 73, the proposed algorithm can be applied to such data by appropriately incorporating the information in the tensor.

Change Detection

Changes are detected by partitioning the streaming graphs into coherent segments based on the similarity of their partitionings (communities). The beginning of each segment represents a detected change.

The segments are found online by comparing the vertex partitioning of the newest graph to the partitioning found for the graphs in the current, growing, segment. Vertex partitioning can be achieved with many methods, but in Ref 67 it is done using the relevance matrix computed by random walks with restarts and modularity maximization. When the partitioning of the new graph is much different from the current segment's, a new segment begins, and the time point for the new graph is output as a detected change. The similarity of two partitions is computed as their Jaccard coefficient, and the similarity of two partitionings is the normalized sum of their partition similarities. A similar approach, GraphScope,[75] based on the same idea but using the MDL principle to guide its partitioning and segmenting will be discussed later.

Compression

The methods discussed in this section are all based on the MDL principle.[33] The MDL principle and compression techniques based on this principle exploit patterns and regularity in the data to achieve a compact graph representation.[109] Applying this principle to graphs is done by viewing the adjacency matrix of a graph as a single binary string, flattened in row or column major order. If the rows and columns of the matrix can be rearranged such that the entropy of the binary string representation of the matrix is minimized, then the compression cost (also known as encoding cost[34]) is minimized. The data-specific features are all derived from the encoding cost of the graph or its specific substructures; hence, anomalies are then defined as graphs or substructures (e.g., edges) that inhibit compressibility.

Example

In Ref 75, given unweighted and undirected graph series $\boldsymbol{G}$ and segment encoding cost function c, the set of change points is defined as a set of time points $T : \left\{t \in T | c\left(\boldsymbol{G}^s \cup \left\{\boldsymbol{G}_t\right\}\right) - c\left(\boldsymbol{G}^s\right) \geq c\left(\boldsymbol{G}_t\right)\right\}$, where $\boldsymbol{G}^s$ is the current graph segment, and $\boldsymbol{G}_t$ is the newly arrived graph at time point t. The main difference between GraphScope[75] and the example from the Community Detection section by Duan et al[67] is the scoring function used for partitioning and change detection. Equivalently, the two methods generate different data-specific features. Duan et al.[67] used community driven metrics, computed by random walks with restarts and modularity optimization, to partition and segment the graphs, whereas, GraphScope[75] is guided by the MDL principle, using encoding cost as a scoring function. While both methods use a threshold system as an anomaly detector, GraphScope uses a dynamic threshold, based on the current graph segment, instead of a fixed threshold as in Ref 67.

Edge Detection

An edge is considered anomalous if the compression of a subgraph has higher encoding cost when the edge is included than when it is omitted.

In Ref 74, a two-step alternating iterative method is used for automatic partitioning of the vertices. Vertex partitioning can be done by rearranging the rows and columns in the adjacency matrix. In the first step, the vertices are separated into a fixed number of disjoint partitions such that the encoding cost is minimized. The second step iteratively splits the partitions with high entropy into two. Any vertex whose removal from the original partition would result in a decrease in entropy is removed from that partition and placed into the new one. Once the method has converged, meaning steps 1 and 2 are unable to find an improvement, the edges can be given outlierness scores. The score for each edge is computed by comparing the encoding cost of the matrix including the edge to the encoding cost if the edge is removed. Streaming updates can be performed using the previous result as a seed for a new run of the algorithm, thus avoiding full recomputations.

Change Detection

The main idea is that consecutive time steps that are very similar can be grouped together leading to low compression cost. Increases in the compression cost mean that the new time step differs significantly from the previous ones, and thus signifies a change.

Akin to Ref 67, to detect changes in a graph stream, consecutive time steps that are similar can be grouped into segments. When considering the next graph in the stream, it can either be grouped with the graphs in the current segment, or it can be the beginning of a new segment. The decision to start a new segment in Ref 75 is made by comparing the encoding cost of the current segment without the next graph to the encoding cost of the segment if the next graph were included. If the vertex partitioning for the new graph is very similar to the vertex partitioning of the graphs in the segment then the encoding cost will not change much. However, if the partitions are very different, the encoding cost would increase because of the increase in entropy. Changes in the graph stream are the time points when a new segment begins. This method is also parameter-free.

Matrix/Tensor Decomposition

These techniques represent the set of graphs as a tensor, most easily thought of as a multidimensional array, and perform tensor factorization or dimensionality reduction. The most straightforward method for modeling a dynamic graph as a tensor is to create a dimension for each graph aspect of interest, e.g., a dimension for time, source vertices, and destination vertices. For example, modeling the Enron email dataset can be done using a 4-mode tensor, with dimensions for sender, recipient, keyword, and date. The element (i, j, k, l) is 1 if there exists an email that is sent from sender i to recipient j with keyword k on day l, otherwise, it is 0.

Similar to compression techniques, decomposition techniques search for patterns or regularity in the data to exploit. All of the data-specific features generated by these methods are derived from the result of the decomposition of a matrix or tensor. Unlike the community methods, these typically take a more global view of the graphs, but are able to incorporate more information (attributes) via additional dimensions in a tensor. One of the most popular methods for matrices (2-mode tensors) is singular value decomposition (SVD),[37] and for higher order tensors (≥ 3 modes) is PARAFAC,[110] a generalization of SVD. The main differences among the decomposition based methods are whether they use a matrix or a higher order tensor, how the tensor is constructed (what information is stored), and the method of decomposition.

Example

In Ref 44, given the 3-D $T \times N \times F$ tensor, where T denotes the number of time steps, N denotes the number of vertices in a graph, F denotes the number of features extracted for each vertex, a scoring function $Z(t)$, and the time window W, a set of time points for change detection is a set of time steps $T^{'}: \{t \in T^{'} | Z(t) \geq c_0\}$, where c_0 is a dynamic threshold, $Z(t) = 1 - u(t)^T r(t-1)$, $u(t)$ is the principal Eigenvector (or 'Eigen-behavior') of the vertex-to-vertex feature correlation matrix computed at time point t, and $r(t-1)$ is the 'typical Eigen-behavior', calculated using the last W Eigenvectors. Each time points score, $Z(t)$, can be thought of as the similarity between the current graph, and the previous W graphs. Using the Z scores that are output as the data-specific features, the anomaly detector is a simple heuristic of choosing the top k values and labeling them as the anomalies. Similar to the examples in the Community Detection and Compression-based sections, this method incorporates the historical values into the data-specific feature generation instead of the anomaly detector. However, unlike the previous two examples, it considers the entire graph at once to compute its score, instead of building up the score based on substructures or partitions, and uses a fixed sliding window of graphs over the time series instead of a dynamic segmenting process.

Vertex Detection

Matrix decomposition is used to obtain activity vectors per vertex. A vertex is characterized as anomalous if its activity changes significantly between consecutive time steps.

Owing to the computational complexity of performing principal component analysis[111] (PCA) on the entire graph, it is computationally advantageous to apply it locally. The approach used in Ref 87 is to have each vertex maintain an edge correlation matrix M, which has one row and column for every neighbor of the vertex. The value of an entry in the matrix for vertex i, $M(j, k)$, is the correlation between the weighted frequencies of the edges (i, j) and (i, k). The weighted frequencies are found using a decay function, where edges that occurred more recently have a higher weight. The largest Eigenvalue and its corresponding vector obtained by performing PCA on M are summaries of the activity of the vertex and the correlation of its edges, respectively. The time series formed by finding the changes in these values are used to compute a score for each vertex at each time step. Vertices that have a score above a threshold value are output as anomalies at that time.

Event Detection

There are two main approaches: (1) Tensor decomposition approximates the original data in a reduced dimensionality, and the reconstruction error is an indicator of how well the original data is approximated. Sub-tensors, slices, or individual fibers in the tensor that have high reconstruction error do not exhibit behavior typical of the surrounding data, and reveal anomalous vertices, subgraphs, or events. (2) Singular values and vectors, as well as Eigenvalues and Eigenvectors are tracked over time in order to detect significant changes that showcase anomalous vertices.

Using the reconstruction error as an indicator for anomalies has been employed for detecting times during molecular dynamics simulations where the collective motions suddenly change,[54] finding frames in a video that are unlike the others,[56] and identifying data that do not fit any concepts.[112]

To address the *intermediate blowup* problem—when the input tensor and output tensors exceed memory capacity during the computation—memory-efficient tucker (MET) decomposition was proposed,[81] based on the original Tucker decomposition.[113] The Tucker decomposition approximates a higher order tensor using a smaller core tensor (thought of as a compressed version of the original tensor) and a matrix for every mode (dimension) of the original tensor. Similarly, methods have been developed for offline, dynamic, and streaming tensor analysis,[79] in addition to static and sliding window based methods.[78] These extensions allow the method to operate on continuous graph streams as well as those with a fixed number of time points. Compact matrix decomposition[80] (CMD) computes a *sparse* low rank approximation of a given matrix. By applying CMD to every adjacency matrix in the stream, a time series of reconstruction values is created and used for event detection. Colibri[82] and ParCube[85] can be used in the same fashion and provide a large increase in efficiency. The PARAFAC decomposition has been shown to be effective at spotting anomalies in tensors as well.[57]

Instead of finding the difference between the graph reconstructed from the approximation obtained by a decomposition, a probabilistic model can be used. The Chung-Lu random graph model[114] is used in Ref 84 Taking the difference between the real graph's adjacency matrix and the expected graph's forms a residual matrix. Anomalous time windows are found by performing SVD on the residual matrices—on which a linear ramp filter has been applied—and by analyzing the change in the top singular values. The responsible vertices are identified via inspection of the right singular vectors. More accurate graph models that also consider attributes are proposed in Ref 86.

Going away from comparing an expected or approximate model of the graph to the real model to find the deviation, events can be identified via significant changes in the decomposed space. Specifically, by performing PCA on the data, the calculated Eigenvectors can be separated into 'normal' and 'anomalous' sets by examining the values of the data projected onto each Eigenvector. At each time step the Eigenvectors are examined (via projection of the data) in descending order of their corresponding Eigenvalues, and the first Eigenvector to contain a projected point outside 3 standard deviations of the rest of the values, and every Eigenvector thereafter, constitute the anomalous set. The second step is then to project the data onto its normal and anomalous subspaces. Once this is done, events are detected when the modifications in the anomalous components from the previous time step to the current are above a threshold value.[77] Expanding on this method, joint sparse PCA and graph-guided joint sparse PCA were developed to localize the anomalies and identify the vertices responsible.[83] The responsible vertices are more easily identified by using a *sparse* set of components for the anomalous set. Vertices are given an anomaly score based on the values of their corresponding row in the abnormal subspace. As a result of the anomalous components being sparse, the vertices that are not anomalous receive a score of 0. Owing to the popularity of PCA in traffic anomaly detection, a study was performed identifying and evaluating the main challenges of using PCA.[115]

Change Detection

The activity vector of a graph, $\mathbf{u}(t)$, is the principal component, the left singular vector corresponding to the largest singular value obtained by performing SVD on the weighted adjacency matrix. A change point is when an activity vector is substantially different from the 'normal activity' vector, which is derived from previous activity vectors.

The normal activity vector, $\mathbf{r}(t-1)$, is the left singular vector obtained by performing SVD on the matrix formed by the activity vectors for the last W time steps. Each time point is given a score $Z(t) = 1 - \mathbf{r}(t-1)^{\mathrm{T}}\mathbf{u}(t)$, which is intuitively a score of how different the current activity is compared to normal, where a higher value is worse. Anomalies can be found online using a dynamic thresholding scheme, where time points with a score above the threshold are output as changes.[76] The vertices responsible are found by calculating the ratio of change between the normal and activity vectors. The vertices that correspond to the indexes with the largest change are labeled anomalous. Similar approaches have used the activity vector of a vertex-to-vertex feature correlation matrix,[44] and a vertex-to-vertex correlation matrix based on the similarity between vertex's neighbors.[58]

Distance Measures

Using the notion of distance as a metric to measure change is natural and widely used.[23,60,116,117] Two objects that have a small difference in a measured metric can be said to be similar. The metrics measured in graphs are typically structural features, such as the number of vertices. Once the summary metrics are found for each graph, the difference or similarity, which are inversely related, can be calculated. The variation in the algorithms lies in the metrics chosen to extract and compare, and the methods they use to determine the anomalous values and corresponding graphs.

Example

In Ref 96, given a graph series $\boldsymbol{G}$, a distance function d that computes the distance between two matrices, and a function to calculate the vertex affinity matrix S, where s_{ij} indicates the influence vertex i has on vertex j, a set of time points for detected events is $T : \{t \in T | d(\boldsymbol{G}_t, \boldsymbol{G}_{t+1}) \le c_0\}$, where

$$d\left(\boldsymbol{G}_t, \boldsymbol{G}_{t+1}\right) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sqrt{S_{t,ij}} - \sqrt{S_{t+1,ij}}\right)^2}, \quad (2)$$

and c_0 is a threshold value. Similar to the example given in the Community Detection section,[67] the data-specific feature of interest here is a measure of finding the distance (inversely related to similarity) between two graphs. Thus, each adjacent graph pair is mapped to a single real value, creating a time series. The anomaly detector, the quality control moving average threshold, is unique compared to the previous examples, as it considers the historical values of the data-specific features. The previous three sections (Community Detection, Compression, and Decomposition) incorporated this information into their data-specific feature generation, using windowing or segmenting techniques.

Edge Detection

If the evolution of some edge attribute (e.g., edge weight) differs from the 'normal' evolution, then the corresponding edge is characterized as anomalous.

In Ref 47, a dynamic road traffic network whose edge weights vary over time is studied. The similarity between the edges over time is computed using a decay function that weighs the more recent patterns higher. At each time step, an outlierness score is calculated for each edge based on the sum of the changes in similarity scores. Edges with the highest score, chosen using either a threshold value or top-k heuristic, are marked as anomalous at that time step.

Viewing the network as a stream of edges, meaning the network does not have a fixed topology as the road traffic network did, the frequency and persistence of an edge can be measured and used as an indicator of its novelty. The persistence of an edge is how long it remains in the graph once it is added, and the frequency is how often it appears. In Ref 48, set system discrepancy[118] is used to measure the persistence and frequency of the edges. When an edge arrives, its discrepancy is calculated and compared to the mean discrepancy value of the active edge set. If the weighted discrepancy of the edge is more than a threshold level greater than the mean discrepancy, the edge is declared novel (anomalous). Using the novel edges detected, a number of metrics can be calculated for various graph elements (e.g., vertices, edges, and subgraphs). Individual graph elements can then be identified anomalous via comparison of the calculated metrics for that element with the overall distribution and change of the metric.

Subgraph Detection

A subgraph with many 'anomalous' edges is deemed anomalous.

Contrasting with Ref 46 where the edge weights are considered the strength of a connection, in Ref 52 the edge weights are considered as outlier scores, or how 'anomalous' that edge is at that time. Every edge at every time step is given its own anomaly score, which is a function of the probability of seeing that particular edge weight on that particular edge given the distribution of weights for that edge over all the graphs in the series. Alternatively, the output of an existing method that assigns outlier scores for edges in a network[47,48,50,74,119] could be used as the input to this method. The latter approach allows[52] to be applied to any network that is capable of having outlier scores assigned to edges (e.g., weighted, directed, and attributed). Once every edge is assigned an outlier score, significant anomalous regions (SARs)—fixed subgraphs over a window of time—are mined from the sequence, analogous to the problem of finding the HDSs. An alternating iterative approach based on Ref 120—first finding an optimal time window for a fixed subgraph, then finding an optimal subgraph for the fixed time window, repeating until no improvement is found—is constructed to approximate a solution to the NP-hard problem. This work was later extended to allow the subgraphs to smoothly evolve over time, where vertices can be added or removed between adjacent time steps.[97] A similar approach mines weighted frequent subgraphs in network traffic, where the edge weights correspond to the anomaly contribution of that edge.[94]

Event Detection

Provided a function $f(G_i, G_j)$ that measures the distance between two graphs, a time series of distance values can be constructed by applying the function on consecutive time steps in the series. Anomalous values can then be extracted from this time series using a number of different heuristics, such as selecting the top k or using a moving average threshold.

Extracting features from the graphs is a common technique to create a summary of the graph in a few scalar values, its signature. *Local* features are specific to a single vertex and its egonet[11] (the subgraph induced by itself and its one-hop neighbors), such as the vertex or egonet degree. *Global* features are derived from the graph as a whole, such as the graph radius. The local features of every vertex in the graph can be agglomerated into a single vector, the signature vector, of values that describe the graph using the moments of distribution (such as mean and standard deviation) of the feature values. In Ref 93, the similarity between two graphs is the Canberra distance, a weighted version of the L_1 distance, between the two signature vectors. A similar approach is used in Ref 92 to detect abnormal times in traffic dispersion graphs. Instead of an agglomeration of local features, it extracts global features from each graph, introducing a dK-2 series[121–123]-based distance metric, and any graph with a feature value above a threshold is anomalous.

As an alternative to extracting multiple features from the graph and using the signatures, the pairwise vertex affinity scores may be used. Pairwise vertex affinity scores are a measure of how much each vertex influences another vertex, and can be found using fast belief propagation.[124] In Ref 96, the scores are calculated for two consecutive time steps, and the similarity between the two graphs is the rooted Euclidean distance (Matusita distance) between the score matrices. The changes in the vertex affinity score are shown to accurately reflect and scale with the level of impact of the changes. For example, removing an edge that connects two otherwise disconnected components, a 'bridging edge', results in a higher score than removing an edge that does not affect the overall structure of the graph. A moving threshold is set on the time series of similarity scores using quality control with individual moving range. The exponential weighted moving average has also been used as a way to dynamically set the threshold, tested on distribution features extracted from Wikipedia revision logs.[91]

Complementary to feature similarity, one can look at the structural differences between graphs to identify the magnitude of change. These methods focus on the function that defines the distance between graphs instead of finding the optimal features to use as summaries. Many metrics have been developed and tested to quantify the differences between graphs. In Ref 88, 10 different distance functions were evaluated on TCP/IP traffic graphs with known anomalies (ground truth). Box-Jenkins autoregressive moving average (ARMA) modeling[125] was used to create a 'normal' model of the feature values, and the time points with residuals, calculated as the difference between expected and real value, exceeding a threshold are flagged. The 10 distance functions tested were weight, maximum common subgraph (MCS) weight, MCS edge, MCS vertex, graph edit distance, median graph edit distance, modality, diameter, entropy, and spectral. Of these 10 distance functions, the MCS-based methods performed the best; however, finding the MCS between two graphs is an NP-hard problem. More recently in Ref 90, five different distance scoring functions, all with a linear complexity, were tested on web graphs with specific types of anomalies: missing subgraph, missing vertices, and connectivity change. The five scoring functions were: vertex/edge overlap, vertex ranking, vector similarity, sequence similarity, and signature similarity. The method that performed the best was the signature similarity, which is done by extracting a signature vector from each graph and finding the distance between them using SimHash.[126] Unlike the signature vectors discussed above that create summaries of the entire graph in a few scalar values, the signature here includes every vertex and edge. Specifically, the features used were each vertex and its PageRank value, and each edge (u, v) with a weight equal to u's PageRank value divided by the number of outlinks from u. Weighting the vertices with their 'quality', measured by PageRank, ensures that the removal of a high-quality vertex will have more of an impact than the removal of a low quality vertex. A fixed threshold was set to find graphs with abnormally low similarity scores.

Instead of finding structural difference between two consecutive graphs, events can be detected using the time series of robustness values for each graph. Robustness is a measure of the connectivity of the graph. A graph with a high robustness will retain the same general structure and connectivity even when some vertices or edges are removed. Finding events is then finding when the robustness value changes significantly.[95] A similar approach is given in Ref 89, but using a variant of the graph diameter. The graph diameter used is the average of all vertex eccentricities (the eccentricity of a vertex v is the longest shortest path from v to any other vertex), instead of the typical definition of using the maximum vertex eccentricity. Both the time series of graph diameter changes and the graph diameters themselves are shown to be effective methods for detecting events.

Probabilistic Models

With a foundation in probability theory, distributions, and scan statistics, these methods typically construct a model of what is considered 'normal' or expected, and flag deviations from this model as anomalous. The type of model used, how it is constructed, what it is modeling, and the method for determining outliers is what differentiates these approaches.

Example

In Ref 98, given a graph series $\boldsymbol{G}$ and a likelihood scoring function L, a set of time points for change detection is a set of time steps $T : \{t \in T | log(L(t))/|\theta| \leq c_0\}$, where $L(t) = \prod_{\theta \text{appear in time t}} P(\theta)^{c(\theta)}$, θ denotes sender–recipient tuples, $P(\theta)$ indicates the probability of all possible sender–recipient tuples (emails from a single sender to a list of receivers), and $c(\theta)$ is the number of occurrences of a particular sender–recipient tuple in time t. Thus, each time step is mapped to a single value, the average log-likelihood, indicating the predictability of the emails during that time step, high values indicating normal behavior. Abnormal time steps are identified via inspection of the plot of log-likelihood values. In general, the data-specific features that are output are often probabilities, or likelihoods, of certain structures or events occurring.

Typically, models or distributions are constructed using past data to derive an 'expected' case for the next time step. Anomaly detectors in probabilistic methods do not always perform a hard mapping from features to anomalies, where everything is either normal or abnormal, but can provide a probability that the structure or event is anomalous.

Vertex Detection

There are two main approaches: (1) Building scan statistics time series and detecting points that are several standard deviations away from the mean, (2) vertex classification.

Scan statistics are often called 'moving window analysis', where the local maximum or minimum of a measured statistic is found in specific regions of the data. In a graph, a scan statistic can be considered as the maximum of a graph invariant feature, such as the number of edges, found for each vertex and its neighborhood in the graph. In Ref 8, they use a variable 'scale' for the measured statistic; each vertex has the number of edges in its 0, 1, and 2 step neighborhood measured. The local statistic for each vertex is then normalized using the mean and standard deviation of its recent values, and the scan statistic of a graph is the maximum of all of the normalized local statistics. Normalizing the values accounts for the history of each vertex, meaning the statistic for each vertex is relative to its own past instead of the values of the other vertices. This ensures that the largest change measured in the graph is not tied directly to the magnitude of change, but the ratio. Building a standardized time series of the scan statistic values, any value that is five standard deviations away from the mean of the series is considered an event (hence, events are detected as well). The vertex most responsible is identified as the one that was chosen for the scan statistic value for the entire graph.

Similar to the use of neighborhoods for scan statistics, the Markov random field model (MRF) is used to uncover the states for vertices and infer the maximum likelihood assignment by a belief propagation algorithm, where a vertex is labeled based on the vertices it is connected to. In Ref 99, anomalies (fraudsters) are uncovered in an online auction network by discovering bipartite cores, which are posited to be the interaction behavior between fraudsters and accomplices. It incrementally updates the model as new edges are added, taking advantage of the fact that an edge insertion or removal will affect only a local subgraph. In the propagation matrix a vertex can be in one of three states: fraudster, accomplice, or honest. Vertices that are assigned the label of fraudster are anomalous.

Edge Detection

Communications (edges) are modeled using a counting process, and edges that deviate from the model by a statistically significant amount are flagged.

One way to model the relationship between vertices is considering the communication between them as a counting process. In Ref 50, a Bayesian discrete time counting process is used to model the number of communications (edge weights) between vertices, and is continuously updated as new graphs are considered. Predictive p-values, based on the learning of the distribution of the counts, are calculated for the edges of the newest observation and subsequently used for flagging anomalous vertex pairs (edges). Moreover, as new graphs are considered, both sequential—comparing the new graph to the model based on history—and retrospective analysis—adjusting the history based on the newest graphs—are performed. The retrospective analysis helps alleviate the insufficient data problem, when decisions are made early on with insufficient history to be accurate.

Subgraph Detection

Fixed subgraphs (e.g., paths and stars), multigraphs, and cumulative graphs are used to construct models on the expected behaviors. Deviations from the models signify an anomalous subgraph.

To identify hacker behaviors in a network,[104] combines scan statistics with a Hidden Markov Model (HMM) for edge behavior. Unlike Ref 8 that used neighborhoods, the local scan statistics are based on two graph shapes, the k-path and star. Comparing the scan statistics of a sliding window to its past values, and using an online thresholding system, local anomalies are identified. The local anomaly is the union of all subgraphs (representing the k-paths or stars) that led to statistically significant test statistics.

Another method to model a dynamic network, instead of using a series or stream of graphs, is to have a single multigraph where parallel edges correspond to communication between vertices at two different time steps. The initial multigraph is decomposed into telescoping subgraphs (TSGs) for each time window.[102] A TSG satisfies two conditions: (1) for any two edges that share a vertex, the edge that begins communication *first* finishes communication *last*; (2) there exists a vertex r, the root, that has no incoming edges and has a path to every vertex in the TSG. TSGs that have a low probability of appearing (according to their size and historic connection patterns) are output as anomalies.

Likewise, a cumulative graph, which includes every edge seen up until the current time step, is used in Ref 101. Edge weights in the graph are calculated using a decay function where more recent edges weigh more. A normal behavior for subgraphs is defined by identifying *persistent patterns*, here found by constructing a graph where each edge is weighted by the fraction of time it is in the graph and iteratively extracting the heaviest weight connected components. As the cumulative graph evolves the extracted subgraphs are monitored, comparing their current activity to the expected activity, which is based on the recent behavior.

Event Detection

Deviations from the models of the graph likelihood or the distribution of the Eigenvalues reveal when an event occurs.

Recently, a new reservoir sampling method was proposed in Ref 103 to maintain structural summaries of a graph stream. The online sampling method manages multiple distinct partitionings of the network to provide a statistically significant summary. As a new graph is added to the stream, every edge has its likelihood calculated based on the edge generation models of the different partitions. Once the edge likelihoods have been found an global graph likelihood is calculated as the geometric mean of the edge likelihood values. A similar edge generation model approach was used in Ref 98, where the probability that and edge exists between vertices i and j is stored in a matrix, $M(i, j)$. The estimated probabilities are calculated using expectation maximization, and subsequently used to give potential scores to every sender–recipient pair. Each sender–recipient tuple, which is an email from one sender to multiple recipients, then has its likelihood computed based on the estimated distribution of all possible sender–recipient tuples. A single score is calculated for each graph as the average of the log-likelihood scores. In both of these methods a single score is calculated for each graph based on the likelihoods of the individual edges within it, and the individual edges (or tuples) responsible are those that have the lowest likelihood.

Similar to estimating the distribution of possible sender–recipient tuples, the distribution of the Eigenvalues and compressed Eigen equations is calculated in Ref 100. Based on the assumption that each vertex has a time series of a vertex-local feature (e.g., summed edge weight), a vertex-to-vertex correlation matrix can be generated for each time step. The Eigen equation of the correlation matrix is compressed by keeping the largest Eigenvalues and a set of low dimension matrices (one matrix for each vertex). By learning the distributions of the Eigenvalues and matrices, both events and the vertices responsible can be identified. When the Eigenvalues deviate from the expected distribution an event has occurred, and the vertices whose matrices deviate from the matrix distribution[127] the most are considered responsible.[100]

DISCUSSION

With the wealth of algorithms discussed, the decision of which to use can be difficult. The choice depends on the types of graphs being analyzed (plain, attributed, directed, undirected, etc.), the desired types of anomalies, and the size of the graphs—influencing the allowed complexity, required parallelism, and whether streaming analysis is required. It is worth noting, however, that using multiple methods in parallel or in conjunction with one another may yield superior results compared to using a single method. Using publicly available datasets (Table 7), each algorithm can be applied to data from a variety of domains, comparing the found anomalies, their sensitivity, and their scalability.

While each method has its own advantages and disadvantages, for brevity we will give an overview of only those that have publicly available code (see Table 6). We note that the available software listed is not comprehensive. We listed all software that was said to be available in their papers, as well as the software of the authors that responded to our email asking about the availability of their code. The methods are again partitioned by the types of anomalies they detect so that a fair qualitative comparison can be made. Note that a quantitative comparison among the available methods was impractical because the methods mostly detect different types of anomalies. Of the few intra-type comparisons that exist, coordinating the different outputs (e.g., anomalous vertices in the graphs versus anomalous vertices in communities of the graphs) and attaining a valuable comparison would be unlikely.

Type 1 (Vertices) Methods

The earliest work with available software is the online method by Heard et al.,[50] which models the frequency of the connections between vertices as a counting process and uses Bayesian learning and predictive p-values to identify anomalies.[71,70,50] In addition to operating on graph streams, it leverages the sparsity of the network by only examining edges that appear in the graphs. A key advantage of this technique is that it performs both sequential analysis, where new graphs are analyzed using the history of the graph stream, in addition to retrospective analysis, where

TABLE 6 | Methods with Open Source Code, the Types of Graphs they Work on, the Language They Are Written in, and the Output Format

Code	Directed	Undirected	Weighted	Unweighted	Plain	Attributed	Parameterized	Streaming	Language	Output
BAYESIAN[50]	√	√	√		√		√	√	MATLAB	[0, 1] per time step and [0, 1] per edge per time step
DELTACON[96]		√	√	√		√		√	MATLAB	[0, 1] per time step
NETSPOT[52]		√	√		√		√		Java	Subgraphs with time intervals
CTOUTLIER [71][1]	√	√	√	√	√	√	√		Java	[0, 1] per vertex for all time steps
ECOUTLIER[70][1]	√	√	√	√	√	√	√	√	Java	[0, 1] per vertex per community per time step
PARCUBE[85]	√	√	√	√	√	√	√	√	MATLAB	Tensor factors for all time steps

[1]These methods operate on the community belongingess matrices of graphs, therefore they are applicable to any graph that can have communities extracted.

the history is updated based on the new graphs that arrive. One example of when this could be useful is in the initial stages of the analysis when very little history is available. When the algorithm terminates the output is very easy to interpret, labeling vertices/edges/subgraphs explicitly as anomalous or not. The flexibility of the final part of the algorithm, when analysis is performed on only the identified anomalous portions of the graph and their neighbors, opens many possibilities—anomalous community detection, identifying the most important or influential anomalous vertex, and many more.

Similar to Ref 50, ECOutlier[70] can be performed on graph streams. Each new graph will have a community detection algorithm run on it, and then the community matching and anomaly detection can be performed. However, it compares only two adjacent time points in the graph stream—the newly added graph and the one immediately preceding it. An interesting feature of this method is that it operates on the community belongingness matrices of a graph, and is therefore applicable to graphs of any type (plain, attributed, directed, etc.). The output from this method is not as straightforward as Ref 50, at each time step providing a matrix that has the outlier score for each vertex in regards to each community. A related method, CTOutlier,[71] has the same appeal of working on community belongingness matrices. A distinguishing factor is that CTOutlier operates on the sequence of belongingness matrices, requiring all of them to be in memory at once for a complete analysis. As a direct result of this, it is able to perform community pattern extraction at an arbitrary scale, removing the restriction of comparing only adjacent time points. For long sequences the length of the patterns should be restricted to 2, as the number of potential patterns grows exponentially with the number of time steps. A single score for each vertex is output, representing how anomalous the community evolution for each vertex is over the entire sequence.

Type 2 (Edges) Methods

Only the Bayesian learning technique by Heard et al.[50] identifies anomalous edges explicitly.[50] However, tensor methods, such as ParCube,[85] can find the reconstruction error at arbitrary granularities, even on a per cell basis. Therefore, they are capable of identifying anomalous edges; however, the reconstruction error must then be maintained for each edge, instead of each tensor, or the user must provide individual edges of interest to examine. A more practical approach for using a tensor-based method could be to identify the time points that are considered anomalous, and then examine the reconstruction error for the cells in those time points individually.

Type 3 (Subgraphs) Methods

NetSpot[52] is designed to find optimal anomalous subgraphs.[52] Its alternating optimization approach—fixing a subgraph and finding the optimal time window, then fixing the time window and finding the optimal subgraph within the window, and repeat—restricts the method to operating on an entire graph series at a time. The output from the algorithm is a set of highly anomalous subgraphs and their corresponding time windows, solving the problem of attribution and requiring no further analysis on the part of the user. Additionally, using their novel seed generation algorithm for mining heavy subgraphs, it is shown to scale linearly with the size of the network in terms of both vertices and time slices. While the authors assign outlier scores to the edges in the network in their own way, it would be interesting to utilize an algorithm specifically designed to assign outlier scores to edges in a network.[48,74,50,47] Preprocessing the networks using an auxiliary edge scoring method allows a mapping from networks that do not fit the requirements of being plain, weighted, and undirected to networks that do.

As mentioned above, tensor methods are capable of identifying anomalous subgraphs using the reconstruction error for particular portions of the tensor. Without user guidance this is prohibitively expensive, as the number of possible subgraphs is exponential on the number of vertices in the graph. However, if the user is interested in a particular set of vertices, or a particular vertex and its neighbors, the error can be tracked for just that region of the graph.

Type 4 (Events/Changes) Methods

DeltaCon[96] provides a graph similarity scoring function based on a number of desirable principles.[85,96] It has been shown to effectively distinguish between, and correctly mark as more important, the removal of 'bridging' edges and edges with higher weights, compared to edges that do not affect the overall structure if removed or have a low weight. Anomalies are found by finding the similarity between two adjacent graphs in the stream, flagging those that are sufficiently different from their immediate neighbors as outliers. While not as robust as methods such as Ref 50 that perform both sequential and retrospective analysis, there is a commensurate decrease in the run time, scaling linearly with the number of edges in the networks.

The tensor decomposition method ParCube[85] is the only method we discuss that is capable of handling graphs that do not fit in memory. Moreover, it is parallelizable on a shared memory machine and has been designed specifically for sparse tensors. As with other tensor methods, it is able to handle any type of graphs with arbitrary attributes, as long as the attributes can be reduced to a real number (e.g., edge labels being enumerated and mapping them to integers). Using an online threshold, e.g., maintaining the mean and standard deviation of the reconstruction error and setting the threshold to two standard deviations above the mean, ParCube can be applied to graph streams.

CONCLUSION

In this survey, we hoped to bridge the gap between the increasing interest in anomaly detection in dynamic networks and the extensive literature available. A common high-level theoretical formulation was

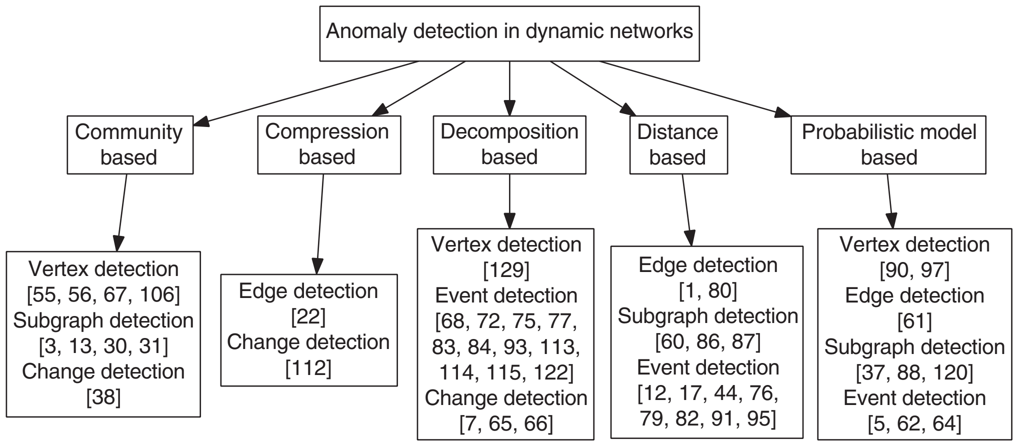
Anomaly detection in dynamic networks
Community based
Compression based
Decomposition based
Distance based
Probabilistic model based
Vertex detection [55, 56, 67, 106] Subgraph detection [3, 13, 30, 31] Change detection [38]
Edge detection [22] Change detection [112]
Vertex detection [129] Event detection [68, 72, 75, 77, 83, 84, 93, 113, 114, 115, 122] Change detection [7, 65, 66]
Edge detection [1, 80] Subgraph detection [60, 86, 87] Event detection [12, 17, 44, 76, 79, 82, 91, 95]
Vertex detection [90, 97] Edge detection [61] Subgraph detection [37, 88, 120] Event detection [5, 62, 64]

FIGURE 5 | The proposed two-tiered taxonomy that illustrates the layout of the survey. Methods are first categorized based on their overarching approach, then subdivided based on the types of anomalies they detect.

found among the methods surveyed. Within this formulation, all of the methods are concerned with exploring various mappings from graphs to real numbers to best identify anomalous behavior using existing anomaly detectors. Although a myriad of approaches have been proposed and explored, they all aim to identify one or more of the four key anomalies we have outlined for dynamic networks: vertices (Figure 1), edges (Figure 2), subgraphs (Figure 3), and events or changes (Figure 4). In conjunction with the types of anomalies identified, we partition the methods into five main categories (Figure 5) derived from the intuition behind each: communities, compression, decomposition, distance, and probabilistic model based.

The field of anomaly detection in dynamic networks is relatively young and is rapidly growing in popularity, as indicated by the number of papers published in the past five years. A concise summary of the methods discussed in this survey is given in Tables 3 and 4, showing the type of anomalies detected, year of publication, and time complexity (notation defined

TABLE 7 | Public Datasets

Dataset	Network Type[1]	Description
SNAP	Multiple	A large collection of networks, both static and dynamic, including domains such as network traffic, social networks, and online reviews. See also http://www-personal.umich.edu/mejn/netdata/Newman's collection.
VAST Challenge 2008[2,3]	Social	A who-calls-whom dataset, recording 10 fictional days of calls and 400 unique cell phones (vertices).
PeMS	Vehicle traffic	Provides up to 10 years of real historical data on the road networks in California.
Wikipedia	Web links	Provides an expansive Wikipedia data set that has information such as page links, page views, and page revisions.
Enron	Communication	A who-emails-whom network between about 150 former Enron employees, mostly upper management.
IMDB	Social	Over 100 years worth of movie data, including titles, producers, and actors.
DBLP	Co-author	Computer science co-author network.
US patent citations	Citation	A list of almost 3 million U.S. patents granted between January 1963 and December 1999, and all citations made to these patents between 1975 and 1999.
Climate Reanalysis[4]	Climate	Various climate variables (air temperature, precipitable water) assimilated from 1948 to the present, across the globe in a fixed grid.
Reality Commons	Communication and social	Multiple datasets, each with information on who-calls-whom and proximity based connections.
HEP-Th	Citation network	Consolidated data from the http://www.cs.cornell.edu/projects/kddcup/datasets.htmlKDD 2003 datasets for theoretical high energy physics papers.
Can-o-sleep	Communication	A p2p filesharing network consisting of records of all the mp3 files shared and transferred over 81 days.
KDD Cup 1999[2]	TCP traffic	Labeled network traffic data that is a version of the simulated network data made available by the http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1998data.html1988 DARPA Intrusion Detection Evaluation Program.
Facebook	Social	A list of Facebook wall posts, where a wall posts between two users creates a timestamped edge.
House and Senate Sponsors	Co-sponsorship	The co-sponsorship networks of 280,000 pieces of legislation proposed in the U.S. House and Senate from 1973 to 2004.

[1]Data may not already be in network format.
[2]Synthetic instead of real data.
[3]Many more datasets are available from the 2008, 2009, and 2010 challenges.
[4]See Ref 128 for one example of creating climate networks from raw data.

in Tables 2 and 5). A list of papers that have made their code available to the public is given in Table 6. Moreover, there are many publicly accessible datasets for testing the available methods or for personal use, see Table 7. For further reading on anomaly detection in graphs and other domains we direct the reader to Refs 16–18, 20–22, 28, 129, 130.

NOTES

[a] Throughout this paper we will use the terms network and graph interchangeably.
[b] Information on the R programming language can be found at http://www.r-project.org/

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their excellent feedback and suggestions. We would also like to thank the authors of the work we cited who provided feedback and additional information regarding their papers. This work was supported in part by the DOE SDAVI Institute and the U.S. National Science Foundation (Expeditions in Computing program). In addition, this material is based upon work supported in part with funding from the Laboratory for Analytic Sciences. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Laboratory for Analytic Sciences and/or any agent or entity of the United States government.

REFERENCES

1. Cheng H, Tan P-N, Potter C, Klooster S. A robust graph-based algorithm for detection and characterization of anomalies in noisy multivariate time series. In: *IEEE International Conference on Data Mining Workshops*, Pisa, Italy, 2008, 349–358.
2. Cheng H, Tan P-N, Potter C , Klooster SA. Detection and characterization of anomalies in multivariate time series. In: *Proceedings of the 9th SIAM International Conference on Data Mining (SDM)*, Sparks, NV, 2009, 413–424.
3. Chen Z, Hendrix W, Guan H, Tetteh IK, Choudhary A, Semazzi F, Samatova NF. Discovery of extreme events-related communities in contrasting groups of physical system networks. *Data Min Knowl Discov* 2013, 27:225–258.
4. Zhang T, Zhuang X, Pande S, Lee W. Anomalous path detection with hardware support. In: *Proceedings of the 2005 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*. New York: ACM; 2005, 43–54.
5. Ding Q Katenka N, Barford P, Kolaczyk E , Crovella M. Intrusion as (anti) social communication: characterization and detection. In: *Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Beijing, China, 2012, 886–894.
6. Ghoting A, Eric Otey M, Parthasarathy S. Loaded: link-based outlier and anomaly detection in evolving data sets. In: *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM)*, Brighton, UK, 2004, 387–390.
7. Jing X, Shelton CR. Intrusion detection using continuous time Bayesian networks. *J Artif Intell Res* 2010, 39:745–774.
8. Priebe CE, Conroy JM, Marchette DJ, Park Y. Scan statistics on Enron graphs. *Comput Math Organ Theory* 2005, 11:229–247.
9. Wang H, Tang M, Park Y, Priebe CE. Locality statistics for anomaly detection in time series of graphs. *arXiv*:*1306.0267*, 2013.
10. Chen F, Neill DB. Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: ACM; 2014, 1166–1175.
11. Akoglu L, McGlohon M, Faloutsos C. Oddball: spotting anomalies in weighted graphs. In: *Advances in Knowledge Discovery and Data Mining*. Berlin/Heidelberg: Springer; 2010, 410–421.
12. Eberle W, Holder L. Detecting anomalies in cargo using graph properties. In: *Intelligence and Security Informatics*. Berlin/Heidelberg: Springer; 2006, 728–730.
13. Eberle W, Holder L. Anomaly detection in data represented as graphs. *Intell Data Anal* 2007, 11:663–689.
14. Noble CC, Cook DJ. Graph-based anomaly detection. In: *Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Washington, DC, 2003, 631–636.
15. Barnett V, Lewis T. *Outliers in Statistical Data*, vol. 3. New York: John Wiley & Sons; 1994608 p.
16. Chandola V, Banerjee A, Kumar V. Anomaly detection: a survey. *ACM Comput Surv* 2009, 41:15.
17. Chandola V, Banerjee A, Kumar V. Anomaly detection for discrete sequences: a survey. *IEEE Trans Knowl Data Eng* 2012, 24:823–839.

18. Chandola V, Mithal V, Kumar V. Comparative evaluation of anomaly detection techniques for sequence data. In: *Proceedings of the 8th IEEE International Conference on Data Mining* (*ICDM*), Pisa, Italy, 2008, 743–748.
19. Hawkins DM. *Identification of Outliers*, vol. 11. London: Chapman and Hall; 1980.
20. Akoglu L, Tong H, Koutra D. Graph-based anomaly detection and description: a survey. *Data Min Knowl Discov* 2014, 28:1–63.
21. Bilgin CC, Yener B. Dynamic network evolution: models, clustering, anomaly detection. *IEEE Netw* 2006.
22. Gupta M, Gao J, Aggarwal CC, Han J. *Outlier Detection for Temporal Data*. San Rafael, CA: Morgan & Claypool Publishers; 2014.
23. Hodge VJ, Austin J. A survey of outlier detection methodologies. *Artif Intell Rev* 2004, 22:85–126.
24. Agyemang M, Barker K, Alhajj R. A comprehensive survey of numeric and symbolic outlier mining techniques. *Intell Data Anal* 2006, 10:521–538.
25. Brent West D. *Introduction to Graph Theory*, vol. 2. Upper Saddle River, NJ: Prentice Hall; 2001.
26. Balakrishnan R, Ranganathan K. *A Textbook of Graph Theory*. Berlin/Heidelberg: Springer; 2012.
27. Cook DJ, Holder LB. *Mining Graph Data*. New York: John Wiley & Sons; 2006.
28. Samatova NF, Hendrix W, Jenkins J, Padmanabhan K, Chakraborty A. *Practical Graph Mining with R*. Boca Raton, FL: CRC Press; 2013.
29. Fortunato S. Community detection in graphs. *Phys Rep* 2010, 486:75–174.
30. Lancichinetti A, Fortunato S. Community detection algorithms: a comparative analysis. *Phys Rev E* 2009, 80:056117.
31. Reichardt J, Bornholdt S. Statistical mechanics of community detection. *Phys Rev E* 2006, 74:016110.
32. Harenberg S, Bello G, Gjeltema L, Ranshous S, Harlalka J, Seay R, Padmanabhan K, Samatova N. Community detection in large-scale networks: a survey and empirical evaluation. *WIREs Comput Stat* 2014, 6:426–439.
33. Rissanen J. Modeling by shortest data description. *Automatica* 1978, 14:465–471.
34. Rissanen J. A universal prior for integers and estimation by minimum description length. *Ann Stat* 1983, 11:416–431.
35. Rissanen J. *Minimum-Description-Length Principle*. New York: John Wiley & Sons; 1985.
36. Grünwald P. *The Minimum Description Length Principle*. Cambridge, MA: MIT Press; 2007.
37. Golub GH, Reinsch C. Singular value decomposition and least squares solutions. *Numerische Mathematik* 1970, 14:403–420.
38. Klema V, Laub AJ. The singular value decomposition: its computation and some applications. *IEEE Trans Automatic Control* 1980, 25:164–176.
39. Kolda TG, Bader BW. Tensor decompositions and applications. *SIAM Rev* 2009, 51:455–500.
40. Gao X, Xiao B, Tao D, Li X. A survey of graph edit distance. *Pattern Anal Appl* 2010, 13:113–129.
41. Rahm E, Bernstein PA. A survey of approaches to automatic schema matching. *VLDB J* 2001, 10:334–350.
42. Koller D, Friedman N. *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: MIT Press; 2009.
43. Glaz J. *Scan Statistics*. Berlin/Heidelberg: Springer; 2001.
44. Akoglu L, Faloutsos C. Event detection in time series of mobile communication graphs. *27th Army Sci Conf* 2010, 2:18.
45. Gao J, Liang F, Fan W, Wang C, Sun Y, Han J. On community outliers and their efficient detection in information networks. In: *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining* (*SIGKDD*), Seattle, WA, 2010, 813–822.
46. Ji T, Yang D, Gao J. Incremental local evolutionary outlier detection for dynamic social networks. In: *Machine Learning and Knowledge Discovery in Databases*. Berlin/Heidelberg: Springer; 2013, 1–15.
47. Li X, Li Z, Han J, Lee J-G. Temporal outlier detection in vehicle traffic data. In: *Proceedings of the 25th International Conference on Data Engineering* (*ICDE*), Shanghai, China, 2009, 1319–1322.
48. James A, Tina E-R, Nishchal D. Detecting novel discrepancies in communication networks. In: *Proceedings of the 10th IEEE International Conference on Data Mining* (*ICDM*), Sydney, Australia, 2010, 8–17.
49. Rattigan MJ, Jensen D. The case for anomalous link discovery. *ACM SIGKDD Explor Newsl* 2005, 7:41–47.
50. Heard NA, Weston DJ, Platanioti K, Hand DJ. Bayesian anomaly detection methods for social networks. *Ann Appl Stat* 2010, 4:645–662.
51. Greene D, Doyle D, Cunningham P. Tracking the evolution of communities in dynamic social networks. In: *2010 International Conference on Advances in Social Networks Analysis and Mining* (*ASONAM*), 2010, 176–183.
52. Mongiov M, Bogdanov P, Ranca R, Papalexakis EE, Faloutsos C, Singh AK. Netspot: spotting significant anomalous regions on dynamic networks. In: *Proceedings of the 13th SIAM International Conference on Data Mining* (*SDM*), Austin, TX, 2013.
53. Miller BA, Beard MS, Bliss NT. Eigenspace analysis for threat detection in social networks. In: *2011 Proceedings of the 14th International Conference on Information Fusion* (*FUSION*), Chicago, IL, 2011, 1–7.

54. Ramanathan A, Agarwal PK, Kurnikova M, Langmead CJ. An online approach for mining collective behaviors from molecular dynamics simulations. *J Comput Biol* 2010, 17:309–324.

55. Saligrama V, Chen Z. Video anomaly detection based on local statistical aggregates. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, 2012, 2112–2119.

56. Tran L, Navasca C, Luo J. Video detection anomaly via low-rank and sparse decompositions. In: *Image Processing Workshop (WNYIPW), 2012 Western New York*, New York, NY, 2012, 17–20.

57. Koutra D, Papalexakis EE, Faloutsos C. Tensorsplat: spotting latent anomalies in time. In: *2012 16th Panhellenic Conference on Informatics (PCI)*, Piraeus, Greece, 2012, 144–149.

58. Ishibashi K, Kondoh T, Harada S, Mori T, Kawahara R, Asano S. Detecting anomalous traffic using communication graphs. In: *Telecommunications: The Infrastructure for the 21st Century (WTC)*, Vienna, Austria, 2010, 1–6

59. Suykens JAK, Vandewalle J. Least squares support vector machine classifiers. *Neural Process Lett* 1999, 9:293–300.

60. Breunig MM, Kriegel H-P, Ng RT, Sander J. LOF: identifying density-based local outliers. In: *ACM Sigmod Record*, vol. 29. New York: ACM; 2000, 93–104.

61. Aggarwal CC, Yu PS. Outlier detection for high dimensional data. In: *ACM Sigmod Record*, vol. 30. New York: ACM; 2001, 37–46.

62. Angiulli F, Pizzuti C. Fast outlier detection in high dimensional spaces. In: *Principles of Data Mining and Knowledge Discovery*. Berlin/Heidelberg: Springer; 2002, 15–27.

63. Angiulli F, Pizzuti C. Outlier mining in large high-dimensional data sets. *IEEE Trans Knowl Data Eng* 2005, 17:203–215.

64. Parsons L, Haque E, Liu H. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explor Newsl* 2004, 6:90–105.

65. Srivastava AN. Enabling the discovery of recurring anomalies in aerospace problem reports using high-dimensional clustering techniques. In: *Aerospace Conference*, Big Sky, MT, 2006, 17 pp.

66. Pokrajac D, Lazarevic A, Latecki LJ. Incremental local outlier detection for data streams. In: *IEEE Symposium on Computational Intelligence and Data Mining*, Orlando, FL, 2007, 504–515.

67. Duan D, Li Y, Jin Y, Zhengding L. Community mining on dynamic weighted directed graphs. In: *Proceedings of the 1st ACM International Workshop on Complex Networks Meet Information & Knowledge Management*. New York: ACM; 2009, 11–18.

68. Aggarwal CC, Subbian K. Event detection in social streams. In: *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)*, Anaheim, CA, 2012, 624–635.

69. Chen Z, Hendrix W, Samatova NF. Community-based anomaly detection in evolutionary networks. *J Intell Inf Syst* 2012, 39:59–85.

70. Gupta M, Gao J, Sun Y, Han J. Integrating community matching and outlier detection for mining evolutionary community outliers. In: *Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Beijing, China, 2012, 859–867.

71. Gupta M, Gao J, Sun Y, Han J. Community trend outlier detection using soft temporal pattern mining. In: *Machine Learning and Knowledge Discovery in Databases*. Berlin/Heidelberg: Springer; 2012, 692–708.

72. Rossi RA, Gallagher B, Neville J, Henderson K. Modeling dynamic behavior in large evolving graphs. In: *Proceeding of the 6th ACM International Conference on Web Search and Data Mining (WSDM)*, Rome, Italy, 2013, 667–676.

73. Araujo M, Papadimitriou S, Günnemann S, Faloutsos C, Basu P, Swami A, Papalexakis EE, Koutra D. Com2: fast automatic discovery of temporal ('comet') communities. In: *Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, Tainan, Taiwan, 2014, 271–283.

74. Chakrabarti D. Autopart: parameter-free graph partitioning and outlier detection. In: *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, Pisa, Italy, 2004, 112–124.

75. Sun J, Faloutsos C, Papadimitriou S, Yu PS. Graphscope: parameter-free mining of large time-evolving graphs. In: *Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, San Jose, CA, 2007, 687–696.

76. Ide T, Kashima H. Eigenspace-based anomaly detection in computer systems. In: *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Seattle, WA, 2004, 440–449.

77. Lakhina A, Crovella M, Diot C. Diagnosing network-wide traffic anomalies. In: *ACM SIGCOMM Computer Communication Review*, vol. 34. New York: ACM; 2004, 219–230.

78. Sun J, Papadimitriou S, Yu PS. Window-based tensor analysis on high-dimensional and multi-aspect streams. In: *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM)*, Hong Kong, China, 2006, 1076–1080.

79. Sun J, Tao D, Christos F. Beyond streams and graphs: dynamic tensor analysis. In: *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Philadelphia, PA, 2006, 374–383.

80. Sun J, Xie Y, Zhang H, Faloutsos C. Less is more: compact matrix decomposition for large sparse graphs. In: *Proceedings of the 7th SIAM International Conference on Data Mining* (*SDM*), Minneapolis, MN, 2007.

81. Kolda TG, Sun J. Scalable tensor decompositions for multi-aspect data mining. In: *Proceedings of the 8th IEEE International Conference on Data Mining* (*ICDM*), Pisa, Italy, 2008, 363–372.

82. Tong H, Papadimitriou S, Sun J, Yu PS, Faloutsos C. Colibri: fast mining of large static and dynamic graphs. In: *Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining* (*SIGKDD*), Las Vegas, NV, 2008, 686–694.

83. Jiang R, Fei H, Huan J. Anomaly localization for network data streams with graph joint sparse PCA. In: *Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining* (*SIGKDD*), San Diego, CA, 2011, 886–894.

84. Miller BA, Arcolano N, Beard MS, Kepner J, Schmidt MC, Bliss NT, Wolfe PJ. A scalable signal processing architecture for massive graph analysis. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*), 2012, 5329–5332.

85. Papalexakis EE, Faloutsos C, Sidiropoulos ND. Parcube: sparse parallelizable tensor decompositions. In: *Machine Learning and Knowledge Discovery in Databases*. Berlin/Heidelberg: Springer; 2012, 521–536.

86. Miller BA, Arcolano N, Bliss NT. Efficient anomaly detection in dynamic, attributed graphs: emerging phenomena and big data. In: *2013 IEEE International Conference on Intelligence and Security Informatics* (*ISI*), Washington, DC, 2013, 179–184.

87. Yu W, Aggarwal CC, Ma S, Wang H. On anomalous hotspot discovery in graph streams. In: *Proceedings of the 13th IEEE International Conference on Data Mining* (*ICDM*), Dallas, TX, 2013.

88. Pincombe B. Anomaly detection in time series of graphs using Arma processes. *ASOR Bull* 2005, 24:2.

89. Gaston ME, Kraetzl M, Wallis WD. Using graph diameter for change detection in dynamic networks. *Australasian J Combinatorics* 2006, 35:299.

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

91. Arackaparambil C, Yan G. Wiki-watchdog: anomaly detection in Wikipedia through a distributional lens. In: *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*. Washington, DC: IEEE Computer Society; 2011, 257–264.

92. Quoc Le D, Jeong T, Eduardo Roman H, Won-Ki Hong J. Traffic dispersion graph based anomaly detection. In: *Proceedings of the Second Symposium on Information and Communication Technology*. New York: ACM; 2011, 36–41.

93. Berlingerio M, Koutra D, Eliassi-Rad T, Faloutsos C. Netsimile: a scalable approach to size-independent network similarity. *arXiv:1209.2684*, 2012.

94. He W, Guangmin H, Zhou Y. Large-scale ip network behavior anomaly detection and identification using substructure-based approach and multivariate time series mining. *Telecomm Syst* 2012, 50:1–13.

95. Malliaros FD, Megalooikonomou V, Faloutsos C. Fast robustness estimation in large social graphs: communities and anomaly detection. In: *Proceedings of the 12th SIAM International Conference on Data Mining* (*SDM*), Anaheim, CA, 2012, 942–953.

96. Koutra D, Vogelstein J, Faloutsos C. Deltacon: a principled massive-graph similarity function. In: *Proceedings of the 13th SIAM International Conference on Data Mining* (*SDM*), Texas-Austin, TX, 2013.

97. Mongiovi M, Bogdanov P, Singh AK. Mining evolving network processes. In: *Proceedings of the 13th IEEE International Conference on Data Mining* (*ICDM*), Dallas, TX, 2013.

98. Huang Z, Dajun Zeng D. A link prediction approach to anomalous email detection. In: *IEEE International Conference on Systems, Man and Cybernetics*, San Diego, CA, 2006, 1131–1136.

99. Pandit S, Horng Chau D, Wang S, Faloutsos C. Netprobe: a fast and scalable system for fraud detection in online auction networks. In: *Proceedings of the 16th International Conference on World Wide Web*, Banff, Canada, 2007, 201–210.

100. Hirose S, Yamanishi K, Nakata T, Fujimaki R. Network anomaly detection based on Eigen equation compression. In: *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining* (*SIGKDD*), Paris, France, 2009, 1185–1194.

101. Thompson B, Eliassi-Rad T. Dapa-v10: discovery and analysis of patterns and anomalies in volatile time-evolving networks. In: *Notes of the 1st Workshop on Information in Networks* (*WIN*), New York, NY, 2009.

102. Djidjev H, Sandine G, Storlie C, Vander Wiel S. Graph based statistical analysis of network traffic. In: *Proceedings of the Ninth Workshop on Mining and Learning with Graphs*, San Diego, CA, 2011.

103. Aggarwal CC, Zhao Y, Yu PS. Outlier detection in graph streams. In: *Proceedings of the 27th International Conference on Data Engineering* (*ICDE*), Hannover, Germany, 2011, 399–409.

104. Neil J, Hash C, Brugh A, Fisk M, Storlie CB. Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics* 2013, 55:403–414.

105. Tantipathananandh C, Berger-Wolf T. Constant-factor approximation algorithms for identifying dynamic communities. In: *Proceedings of the 15th*

ACM International Conference on Knowledge Discovery and Data Mining (*SIGKDD*), Paris, France, 2009, 827–836

106. Tantipathananandh C , Berger-Wolf T. Finding communities in dynamic social networks. In: *Proceedings of the 11th IEEE International Conference on Data Mining* (*ICDM*), Vancouver, Canada, 2011, 1236–1241.
107. Tantipathananandh C , Berger-Wolf T, Kempe D. A framework for community identification in dynamic social networks. In: *Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining* (*SIGKDD*), San Jose, CA, 2007, 717–726.
108. Gupta M, Gao J, Han J. Community distribution outlier detection in heterogeneous information networks. In: *Machine Learning and Knowledge Discovery in Databases*. Berlin/Heidelberg: Springer; 2013, 557–573.
109. Grünwald P. *A Tutorial Introduction to the Minimum Description Length Principle*. Cambridge, MA: MIT Press; 2005.
110. Harshman RA. *Foundations of the Parafac Procedure: Models and Conditions for an "Explanatory" Multimodal Factor Analysis*. Los Angeles, CA: UCLA; 1970.
111. Jolliffe I. *Principal Component Analysis*. New York: John Wiley & Sons; 2005.
112. Barnathan M, Megalooikonomou V, Faloutsos C, Faro S, Mohamed FB. Twave: high-order analysis of functional MRI. *Neuroimage* 2011, 58:537–548.
113. Tucker LR. Some mathematical notes on three-mode factor analysis. *Psychometrika* 1966, 31:279–311.
114. Chung F, Linyuan L, Van V. Spectra of random graphs with given expected degrees. *Proc Natl Acad Sci* 2003, 100:6313–6318.
115. Ringberg H, Soule A, Rexford J, Diot C. Sensitivity of PCA for traffic anomaly detection. *ACM SIGMETRICS Perf Eval Rev* 2007, 35:109–120.
116. Chen Y, Miao D, Zhang H. Neighborhood outlier detection. *Expert Syst Appl* 2010, 37:8745–8749.
117. Papadimitriou S, Kitagawa H, Gibbons PB, Faloutsos C. Loci: fast outlier detection using the local correlation integral. In: *Proceeding of the 19th International Conference on Data Engineering*, 2003, 315–326.
118. Chazelle B. *The Discrepancy Method: Randomness and Complexity*. Cambridge, UK: Cambridge University Press; 2002.
119. Tong H, Lin C-Y. Non-negative residual matrix factorization with application to graph anomaly detection. In: *SDM*. Mesa, AZ: SIAM; 2011, 143–153.
120. Bogdanov P, Mongiov M, Singh AK. Mining heavy subgraphs in time-evolving networks. In: *Proceedings of the 11th IEEE International Conference on Data Mining* (*ICDM*), Vancouver, Canada, 2011, 81–90.
121. Mahadevan P, Krioukov D, Fall K, Vahdat A. Systematic topology analysis and generation using degree correlations. In: *ACM SIGCOMM Computer Communication Review*, vol. 36. New York: ACM; 2006, 135–146.
122. Sala A, Cao L, Wilson C, Zablit R, Zheng H, Zhao BY. Measurement-calibrated graph models for social network experiments. In: *Proceedings of the 19th International Conference on World Wide Web*. New York: ACM; 2010, 861–870.
123. Wang J, Provan GM. Generating application-specific benchmark models for complex systems. In: *AAAI*, 2008, 566–571.
124. Koutra D, Ke T-Y, Kang U, Horng Polo Chau D, Kenneth Pao H-K, Faloutsos C. Unifying guilt-by-association approaches: theorems and fast algorithms. In: *Machine Learning and Knowledge Discovery in Databases*. Berlin/Heidelberg: Springer; 2011, 245–260.
125. Box GEP, Jenkins G. *Time Series Analysis: Forecasting and Control*. San Francisco, CA: Holden-Day; 1970.
126. Charikar MS. Similarity estimation techniques from rounding algorithms. In: *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*. New York: ACM; 2002, 380–388.
127. Gupta AK, Nagar DK. *Matrix Variate Distributions*, vol. 104. Boca Raton, FL: CRC Press; 1999.
128. Steinhaeuser K, Chawla NV, Ganguly AR. Complex networks as a unified framework for descriptive analysis and predictive modeling in climate science. *Stat Anal Data Min* 2011, 4:497–511.
129. Aggarwal CC. *Outlier Analysis*. Berlin/Heidelberg: Springer; 2013.
130. Gupta M, Gao J, Aggarwal CC, Han J. Outlier detection for temporal data: a survey. *IEEE Trans Knowl Data Eng* 2013, 25:1.