A Survey on Anomaly detection in Evolving Data

[with Application to Forest Fire Risk Prediction]

Mahsa Salehi
Faculty of Information Technology
Monash University
Victoria 3800, Australia
mahsa.salehi@monash.edu

Lida Rashidi
Department of Computing and information
Systems, University of Melbourne
Victoria 3000, Australia
rashidi.l@unimelb.edu.au

ABSTRACT

Traditionally most of the anomaly detection algorithms have been designed for 'static' datasets, in which all the observations are available at one time. In non-stationary environments on the other hand, the same algorithms cannot be applied as the underlying data distributions change constantly and the same models are not valid. Hence, we need to devise adaptive models that take into account the dynamically changing characteristics of environments and detect anomalies in 'evolving' data. Over the last two decades, many algorithms have been proposed to detect anomalies in evolving data. Some of them consider scenarios where a sequence of objects (called data streams) with one or multiple features evolves over time. Whereas the others concentrate on more complex scenarios, where streaming objects with one or multiple features have causal/non-causal relationships with each other. The latter can be represented as evolving graphs. In this paper, we categorize existing strategies for detecting anomalies in both scenarios including the state-of-the-art techniques. Since label information is mostly unavailable in real-world applications when data evolves, we review the unsupervised approaches in this paper. We then present an interesting application example, i.e., forest fire risk prediction, and conclude the paper with future research directions in this field for researchers and industry.

1. INTRODUCTION

Anomalies are data points that are inconsistent with the distribution of the majority of data points [6].

Traditionally the problem of anomaly analysis has been considered by statistics community, where the outcome has been a number of invaluable publications [14; 34; 56]. According to the definition by Barnett and Lewis, an anomaly is "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data" [14]. Hawkins defined an anomaly as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" [34].

Anomalies are also known as rare events, abnormalities, deviants or outliers [1]. Anomaly detection has received considerable attention in the field of data mining due to the valuable insights that the detection of unusual events can provide in a variety of applications. For example, in environmental monitoring of a wireless sensor network, outlier detection techniques can be applied to detect faulty sensors or interesting behavioural patterns.

The availability of data that is used for the task of anomaly detection varies based on the properties of the dataset. In a static dataset, the whole observations of objects are available and the anomalies are detected with regards to the whole dataset. There are comprehensive surveys of outlier detection in static data with respect to different scenarios that are presented in [22; 8; 16; 23; 42; 22]. On the other hand, all the observations may not be available at once and instances can arrive sequentially. The observations in the latter group are called *data streams* and they can be represented as a dynamic vector. In addition, objects may possess causal/non-causal relationships such as friendship, citation and communication links. This type of relational dataset can be represented as a graph. Similar to data streams, all the observations of objects (nodes in a graph) may not be available at once and they can arrive sequentially. Moreover, the relationships (edges between nodes in a graph) may change over time. This type of relational dataset is represented in an *evolving graph*. Although various approaches have been developed for anomaly detection, many of them as described in [22; 9] focus specifically on the task of anomaly detection in static data or static graph. However, detecting anomalies in dynamic datasets is a less explored area of research. In this paper, we focus on both evolving scenarios, i.e., data streams and evolving graphs, and provide a categorization of different techniques in each scenario. Note that there are three general application scenarios for outlier detection, i.e., supervised, semi-supervised and unsupervised. Since supervised and semi-supervised scenarios are rare to happen in real-world applications due to the lack of label information regarding the outlierness of some observations, we only survey the unsupervised scenarios, which does not rely on label information in this paper. Figure 1 represents a categorization of different anomaly detection approaches in evolving data.

We also present a real application as an example of anomaly detection in evolving data. We conclude the paper with providing interesting future research directions.

2. ANOMALY DETECTION IN DATA STREAMS

[1]In this paper, we use the terms outlier detection and anomaly detection interchangeably

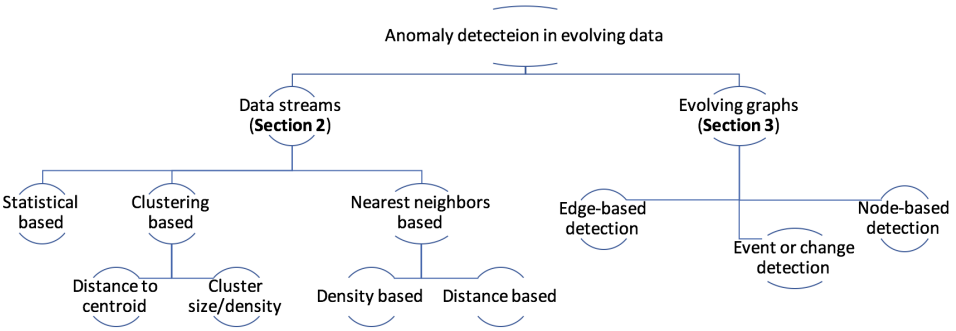


Figure 1: A categorization of existing anomaly detection approaches in evolving data

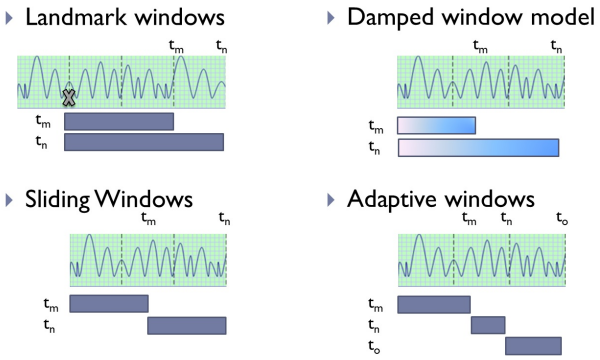


Figure 2: Different windowing techniques

In this section we first start with a definition of data streams and then we survey the literature in the area of anomaly detection in data streams and different techniques.

2.1 Data streams

A data stream is a sequence of data points with three main characteristics. A data stream has a *continuous* flow. Hence, the processing time of the algorithm is a challenge. In addition, the volume of data delivered by a stream continually increases. In other words, the number of incoming data points is *unbounded*. Therefore, memory storage is another challenge. Finally, data streams can change over time. The solution to deal with these data stream challenges is to limit the number of processed data points, called data windowing. We categorize windowing techniques into four groups [67; 18]. Figure 2 shows this categorization.

1. **Landmark windows**: In this windowing technique, a fixed point in the data stream is defined as a landmark and processing is done over data points between the landmark and the present data point.

2. **Sliding windows**: A sliding window size w is considered in this technique. It processes the last w data points in the stream.

3. **Damped window model**: In sliding window technique, only the last w data points are processed and the previous data points are ignored completely. Whereas in damped window model, a weight is assigned to each data point in such a way that the old data points get smaller weights. Therefore, the most recent data points are used with higher weights.

4. **Adaptive windows**: In all previous models, the number of data points that are being processed are fixed. In adaptive techniques, the window size w would change as the data stream evolves. In this technique, the more the data points evolves, w becomes smaller. In contrast, if data points remain constant, the value of w increases.

2.2 Categorisations and techniques

We can categorize the outlier detection techniques in data stream and in unsupervised scenarios to three main groups: statistical based, clustering based and nearest neighbor based approaches. In the following subsections we give an overview of the techniques in each group along with their properties.

2.3 Statistical Based Techniques

In *statistical based* approaches, the aim is to learn a statistical model for a normal behavior of a dataset. Thereafter, the observations that are not (or has low probability to) fit into that model is declared as outliers. In terms of data streams, Yaminshi et. al. [62; 61] assign an outlierness score to the already built Gaussian mixture model, which shows how the incoming data point is deviated from the model. However, in these set of approaches, *a priori* knowledge regarding the underlying distribution of the dataset is required, which is almost always unavailable when data evolves over time.

2.4 Clustering Based Techniques

Clustering data streams has become an interesting topic in recent years and has drawn the attention of many researchers. Clustering algorithms can be used to find anomalies in data streams. They can be categorised into two groups: 1) One group of techniques are proposed in which the anomalies are assumed to fall into the clusters with small number of data points or low density. 2) In the second group of clustering based methods the distance of data points to their nearest cluster centroids are used to detect anomalies. In the following paragraphs we discuss the techniques in each group separately.

Many algorithms have been proposed in the first group. In 1996, a clustering algorithm called BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) was proposed for large databases [66]. It considers the fact that memory can not support large databases and is much smaller than the number of data points. The method uses the landmark window model and clusters the data points incrementally and dynamically in a single scan of the data. In ad-

dition, BIRCH is the first algorithm that handles noise and detects outliers in the data stream. It works based on a concept called a *Cluster Feature (CF)*. It is a triplet consisting of three value, that provides useful information to describe a cluster. In other words, it is a way of summarizing a cluster. A *Cluster Feature* is defined on the following three values:

$$CF =< n, LS, SS > \quad (1)$$

where n is the number of D-dimensional data points in a cluster, LS is the linear sum of n data points and SS is the squared sum of n data points.

The algorithm starts to create clusters and stores only the CF value for each cluster, which is more memory efficient. The CFs are stored in a height-balanced tree (B+ tree). The process of building a tree is a top-down approach and this algorithm is considered as a hierarchical clustering. For each incoming data point BIRCH finds the point's appropriate location in the leaf, then it updates the CF values and finally the path to the root of the tree should be updated. BIRCH scans the CF tree regularly and stores low density leaves as outliers.

In [7], for the first time a clustering algorithm called CluStream was proposed which clusters *evolving* data streams. This algorithm also uses the CF concept which was used in [66], but extends it by adding the temporal characteristics of clusters. Therefore, for each cluster, it considers the relevant time stamps and stores the sum of relevant time stamps in addition to the squared sum of time stamps along with the triplet CF which has been defined above. This algorithm comprises of two steps: online and offline. In the online step the micro-clusters are created, while in the offline step macro-clusters are generated by applying a k-means algorithm on micro-clusters. For each incoming data point the micro clusters are updated accordingly: 1. The new data point can be assigned to one of the micro-clusters, or 2. The new data point is far from the existing clusters, therefore a new micro-cluster is created and one of the previous micro-clusters is identified as an outlier micro-cluster and is deleted (based on micro-cluster points' time stamps) or merged with another micro-cluster. Hence, outliers are not identified by the size of the micro-clusters, rather they are identified by not being active in the recent stream activity.

Many algorithms have been proposed to improve CluStream in terms of quality of clusters for evolving data streams. DenStream [20] specifically focuses on building arbitrary shaped clusters. Therefore, it uses a density-based clustering approach (DBSCAN [29]) to cluster micro-clusters. This algorithm is applicable in noisy datasets and can build normal micro-clusters as well as outlier micro-clusters similar to [7]. It uses a damped window model and weights the recent data points with higher values. Later on, a grid-based density approach is proposed based on two online and offline steps, called DStream [24]. In the online step each data point is assigned to a grid cell and low density grid cells are identified as outliers and removed. Here low density grid cells refer to either the gird cells with low number of data points or the grid cells with high number of data points while their effect have been reduced by a decay factor. In the offline step grids are clustered using a density-based clustering algorithm. This approach is time and memory efficient and clusters arbitrary shaped clusters. While [20] and [24] are based on a damped window model, SDstream [54] is proposed based on a sliding window model. Similar to previous approaches it uses a two step approach, except that it only keeps the last w data points.

While in all of the above approaches outliers are assumed to fall into the clusters with small number of data points or low density, in the remainder of this subsection we survey the papers in which the distance of data points to their nearest cluster centroids are used to detect anomalies in data streams. In [13], a new outlier detection algorithm called AnyOut is proposed to detect anytime outliers in streaming data. In anytime outlier detection, data streams have varying arrival rates and at a given time t, outlierness value for the current data point p_t should be computed until the arrival of next data point. AnyOut builds a specific tree structure called ClusTree [41] suitable for anytime clustering. In this tree structure, CFs are stored in each node as well as a buffer for anytime insertion of clusters. Anyout investigates the Clustree in a top-down fashion and finds the closest CF to p_t at each level. Once the next data point p_{t+1} arrives, an outlierness score is computed as the distance of p_t to its nearest cluster feature centroid.

In [28], the data stream is divided into data chunks and k-means is used for each data chunk. Then the data points which are far enough from clusters centroids are identified as 'candidate' outliers. Hence, the outlierness value of data points might change depending on the upcoming data chunks.

Moshtaghi et. al. proposed a hyperellipsoidal clustering approach to model the normal behavior of the system, where the points outside this clustering boundary will be detected as anomalies [46]. It incrementally updates the hyperellipsoids' parameters (mean and covariance matrix) to keep track of changes in the underlying distributions of data streams.

Another algorithm that falls into the second category is eTSAD [47], a new clustering algorithm that models data streams with a set of elliptical fuzzy rules. Similar to the previous approach the parameters of fuzzy clustering model are updated for each incoming data point to detect outliers and regime changes in data streams.

Rather than modeling data streams by only one set of cluster profiles and updating it over time, in [58] an ensemble-based algorithm has been proposed for evolving data streams in which a pool of clustering models are generated. Thereafter, for each incoming data point an outlierness values is being calculated based on a consensus decision using only the relevant set of clustering models. Similar to [46], a hyperellipsoidal clustering approach is used to model the normal behaviour of the data streams in each data chunk and the distance between clusters' centroids are being used for detecting anomalies. A more time/memory efficient anomaly detection algorithm has been proposed in [25], where the notion of active clusters are introduced. The input data streams is divided into different data chunks and for each incoming data chunk active clusters in the current data chunk are identified. Also, if there exists any emerging distributions, the model which consists of a set of hyperellipsoidal clusters are updated as well.

Finally, while all of the above algorithms detect anomalies by modeling data streams based on clustering, none of them can identify anomalies in real-time. [26] proposes a new approach for discovering temporal evolution of clusters while detecting anomalies in real-time.

2.5 Nearest Neighbors Based Techniques

In clustering based approaches, the outlierness of an observation is evaluated considering the clustering models that are build. However, there is another group of outlier detection techniques in which outlierness scores are assigned to data points considering their nearest neighbors which lead to higher resolution analysis. This group of approaches have gained popularity as they are easy to implement and intuitive.

The first trace of distance based outliers can be find in database community. Let P a dataset with n data points. According to the definitions given in [36; 38; 37], a data point $p \in P$ is a distance-based outlier, DB-outlier, if and only if at least a fraction ϵ of the whole number of data points are in a distance larger than radius r. More specifically, a data point p is a DB-outlier if and only if:

$$|\{q \in P | d(p,q) > r\}| \geq \epsilon n \tag{2}$$

where d is a distance function between two data points. These outliers are also called global outliers, as they consider all n existing data points to define outlierness for each data point.

While the output of the above definition is a binary decision on the outlierness of each data point, in [51], a score of distance-based outlierness has been proposed based on a given parameter k (a number of nearest neighbors). For a data point $p \in P$, this score is called k-*distance*(p) and it is computed as the distance between a data point p and its k^{th} nearest neighbor (k^{th}-NN) in P.

Later, Anguilli and Pizzuti defined another score of distance-based outlierness [11]. Given parameter k, for each data point the distance between the data point and its all k *nearest neighbors* (k-NNs) are computed and considered as the outlierness score, specifically:

$$k\text{-}distance - w(p) = \sum_{q \in N_{(p,k)}} d(p,q) \tag{3}$$

where $N_{(p,k)}$ is the set of k nearest neighbors of p.

2.5.1 Distance Based Techniques

In term of streaming data in [10] and [63], a sliding window is used to detect global distance-based outliers in data streams with respect to the current window. The authors in [39] improved the time complexity and memory consumption in comparison to [10] and [63]. Finally in [21], a faster and more general framework called LEAP has been proposed for high volume high dimensional data streams by optimizing the search space. In addition, different types of distance-based outliers [51], [11] have been detected by this framework. Since the outliers are computed with respect to the last n data points, the outliers that are detected by these approaches are called 'global' outliers. Hence these approaches fail to detect outliers in *non-homogeneous* densities.

2.5.2 Density Based Techniques

In contrast to distance-based 'global' outliers, distance-based 'local' outliers (a.k.a density-based outliers) are data points that are outliers with respect to their k nearest neighbors. LOF is a well-known density-based outlier detection algorithm [19]. In this algorithm a score of outlierness, called *LOF* (Local Outlier Factor), is computed for a data point p according to the following definitions, assuming that all data points are available and the number of nearest neighbors is k.

- k-*distance*(p): the distance between a data point p and its k^{th} nearest neighbor (k^{th}-NN).
- *Reachability distance* (*reach-dist*) of a data point p with respect to another data point o:

$$reach\text{-}dist_k(p,o) = max\{k\text{-}distance(o), d(p,o)\} \tag{4}$$

  where $d(p,o)$ is the Euclidean distance between p and o.
- *Local reachability density* (lrd) of a data point p:

$$lrd_k(p) = \left(\frac{1}{k} \sum_{o \in N_{(p,k)}} reach\text{-}dist_k(p,o) \right)^{-1} \tag{5}$$

  where $N_{(p,k)}$ is the set of k nearest neighbors of p.
- *Local outlier factor* of a data point p

$$LOF_k(p) = \frac{1}{k} \sum_{o \in N_{(p,k)}} \frac{lrd_k(o)}{lrd_k(p)} \tag{6}$$

Since the LOF technique achieves good detection accuracy in non-homogeneous densities (often the case in practice) without assumptions regarding the underlying distribution of the dataset, it has become a popular approach and many variants of this technique have been proposed.

Pokrajak et al. [50] were the first to propose an incremental version of the LOF technique (iLOF) that can be used for data streams. All previous versions required the entire dataset to compute *LOF* values for the data points, but in the iLOF technique the outlier factor is computed for each incoming data point. For each incoming data point p, the iLOF finds the k-nearest neighbors (k-NNs) of p, computes the local outlier factor of p based on the outlier factors of its k-NNs, and updates the k-NNs of past data points along with their local outlier factors if needed. Pokrajak et al. showed that only a few data points are required updating and therefore the algorithm was efficient. In order to detect outliers accurately, all past data points need to be retained to compute the outlier factor for each new incoming data point. Therefore, iLOF suffers from large memory requirements, as well as high time complexity.

Recently a memory efficient algorithm called MiLOF has been proposed to address these limitations in density-based outlier detection problems [57]. MiLOF detects outliers in data streams with the same technique as iLOF but with less memory/time consumption with a comparable accuracy to iLOF.

3. ANOMALY DETECTION IN EVOLVING GRAPHS

In this section, we discuss the problem of detecting anomalies in dynamic graphs. The graphs that we consider in this section are snapshots of evolving graphs. Therefore we can consider these graph snapshots to be static entities. Our objective is to detect anomalous behaviour in the graph, which might involve abnormal vertices, edges and subgraphs. This problem is quite similar to anomaly detection in clouds of data points.

Popular anomaly detection techniques can be employed in graph datasets as well. We can extract high level features from the graph and perform anomaly detection techniques on the processed data. The snapshots of the graph can be considered as potential high dimensional data, i.e., after feature extraction, thus we can consider evolving graphs as data streams and deal with the problem of anomaly detection in this type of data using the techniques described in Section 2.

Due to the popularity of graphs in representing real-world networks, numerous methodologies have been developed for spotting anomalies in graph data. We discuss anomaly detection in graph data only for *plain* graphs where there are only nodes and edges representing the data, and there are no attributes associated with those nodes and edges. However these techniques can be extended to *attributed* graphs as well. Nodes and/or edges in an attributed graph represent various features. For instance, a node in a social network may have various education levels or interests, and links may have different strengths.

We first need to define the anomaly detection problem given the snapshots of graphs over time. Anomalies are defined as nodes, edges and sub-patterns that deviate from the observed normal patterns in the graphs. Since we are considering plain graphs, the significant information to incorporate in an anomaly detection scheme is the graph structure or topology. The interesting patterns can be detected through two sets of techniques: structural sub-patterns and community-based sub-patterns.

The first category of techniques extracts useful graph centric information such as node degree, in addition to other features, and performs anomaly detection on data points. Therefore, the problem of anomaly detection in graphs is transformed to the well-known problem of spotting outliers in an n-dimensional space.

The graph centric features can be computed from nodes or the combination of two, three or more nodes, i.e., dyads, triads and communities. These features can also be extracted from the combination of all nodes in a more general manner [35]. Network intrusion detection [27] and spotting web spam [15] have utilized graph centric features in their process of anomaly detection.

3.1 Node-based Anomalies

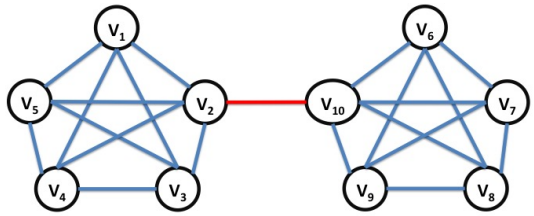
One of the anomalous objects that can be found in a dynamic network is a single vertex or collection of vertices. In each time step, a series of features are extracted from the nodes. The techniques in this category constitute a feature extraction scheme, which measures vertex properties such as the degree, or the ego-net density of a single vertex. The nodes that demonstrate irregular behaviour in comparison to others are detected as anomalous. However it is worth noting that various approaches use a number of different feature extraction techniques based on the assumptions, aim and the domain of the problem they are interested in.

Since we are considering streaming graphs, the context of time is also added to the features extracted from a node. The temporal aspect of the graphs adds another level of complexity to the problem. We define node anomalies as the vertices whose scores are above the average score $\widehat{f}$ as shown in Equation 7. The score is calculated using a function f that yields a real-valued score for each node, $f : V \rightarrow \mathbb{R}$. The average score $\widehat{f}$ is calculated based on the scores of the normal nodes. The anomalous nodes $V' \subset V$ are thus defined as:

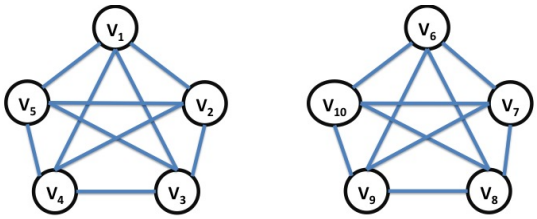
$$\forall v' \in V', |f(v') - \widehat{f}| > c_0 \tag{7}$$

where c_0 is the acceptable deviation calculated from the normal node behaviour. It is worth mentioning that the scoring function f may measure the change in the number of connections that each vertex has between consecutive time steps, or the change in the edge weights. In simple scenarios, anomalous nodes undergo substantial change, which makes their detection easier. However, this is not always the case, for instance, consider the example of a node anomaly shown in Figure 3. The two time stamps t_1 and t_2 are consecutive snapshots of the same network. As you can see, there are two full cliques consisting of nodes $V_{1..5}$ and $V_{6..10}$.

These two full cliques can be considered as communities, such as the students in two different high schools. At time stamp t_1, there is a link between two of the nodes/students from different communities, namely nodes V_2 and V_{10}. However, in the next time stamp, this link is eliminated. Although the degrees of the nodes V_2 and V_{10} have not changed substantially, time stamp t_2 is considered to be abnormal and the two previously mentioned nodes are detected as anomalous. The interesting insight from this example is that detecting abnormal nodes needs a structure-aware feature extraction technique to first spot the discontinuity.



(a) Graph at t_1.



(b) Graph at t_2.

Figure 3: Graph variants where circles and lines represent vertices and edges respectively.

A good technique in detecting anomalous nodes in a community structured network is to determine the level of involvement of a node in communities throughout time. An example of such an approach is [55], where the authors try to model the roles for each node and monitor their role change over time. Their definition of node roles is dissimilar to that of community memberships. The roles in [55] are defined as sets of nodes that are structurally similar to each other, and not necessarily the nodes with many connection within themselves.

They propose a Dynamic Behavioural Mixed-membership Model (DBMM), which uses the concept of feature based roles and can generalize the learned roles to the unobserved

vertices. DBMM focuses on temporal anomalies, where at each time step, a feature extraction procedure takes place that determines node roles. They use regularization to compute the number of roles and also employ transition matrices to calculate the probability of a node changing its role in the upcoming time-step. This approach is consistent with the definition of a node anomaly in Equation 7 in the sense that the node anomaly score is calculated based on the difference of its estimated and true roles. It is worth noting that DBMM can only detect abnormal nodes in each time step and it lacks a more general outcome, i.e., anomalous graph instances.

Another example of node-based anomaly detection in streaming graphs is [33], where the authors introduce the notion of Evolutionary Community Outliers (ECOutlier). ECOutlier refers to the task of detecting vertices that indicate abnormal community-membership evolution in consecutive time-stamps. The authors first define a normal evolutionary trend based on the latent communities discovered in the data. They propose an approach for integrating community matching and anomaly detection, where the problem becomes one of minimizing community matching error while anomalous nodes are ignored by assigning lower weights to them.

One of the properties of ECOutlier [33] is that in each iteration, only the two consecutive time-stamps are compared. This method constructs a belongingness matrix, which is the basis for community matching and anomaly detection. The outcome of this approach at each time-stamp is the anomaly score of each node in the graph.

3.2 Edge-based Anomalies

Another anomalous entity that can be found in a dynamic network is the edge or collection of edges that indicate anomalous evolutionary trends in comparison to the majority of the edges in the network. In contrast to node-based anomalies, the abnormal edges can be directly detected from the edge weight evolution or edge addition/elimination between nodes, which do not belong to the same community or are very unlikely to form a relationship.

We can define the edge anomalies similar to the vertex anomalies that were described in Equation 7. The edges whose scores are more than the average score $\widehat{f}$ as shown in Equation 8 are considered anomalous. The edge score is calculated using a score function f that yields a real-valued score for each edge, $f : E \to \mathbb{R}$. The average score $\widehat{f}$ is calculated based on the scores of the normal edges. The anomalous edges $V' \subset V$ are thus defined as:

$$\forall e' \in E', |f(e') - \widehat{f}| > c_0 \tag{8}$$

where c_0 is the acceptable deviation from the average normal edge score. A trivial example of the edge score function f is to measure the weight change of an edge from one time-stamp to the next. In simple scenarios, anomalous nodes undergo substantial change, which makes their detection easier. This is shown in Figure 4, where the edge connecting nodes V_3 and V_4 is indicating abnormal evolution in the second time-stamp. Another anomalous edge weight evolution is captured in the edge connecting nodes V_1 and V_4 at the third time-stamp.

In addition to the edge weight evolution, we can model the edges based on their corresponding nodes' interactions. The score function of each edge f in this context is the probability of an edge weight at a given time-stamp. An example of such an approach is in [45], where the authors assign an anomaly score to every edge with respect to the probability of observing that edge weight at a given time-stamp. This method is called NetSpot,which uses the edge anomaly score to detect significant anomalous regions, i.e., subgraphs, within a dynamic graph. The outcome of the NetSpot approach [45] is a collection of anomalous subgraphs and their corresponding time windows.

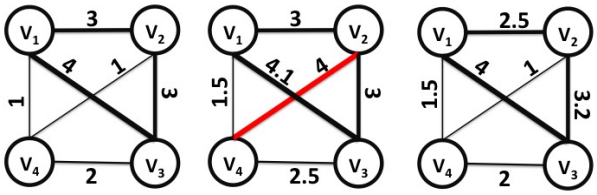


Figure 4: An illustration of an anomalous edge that occurs due to an irregular weight evolution pattern, with the anomalous edge highlighted in red. At each time stamp, a vertex weight typically changes by ± 0.05 at most. However, edge $(1, 3)$ has a spike in its weight at time stamp 2, unlike any other time in the series.

3.3 Events or Changes

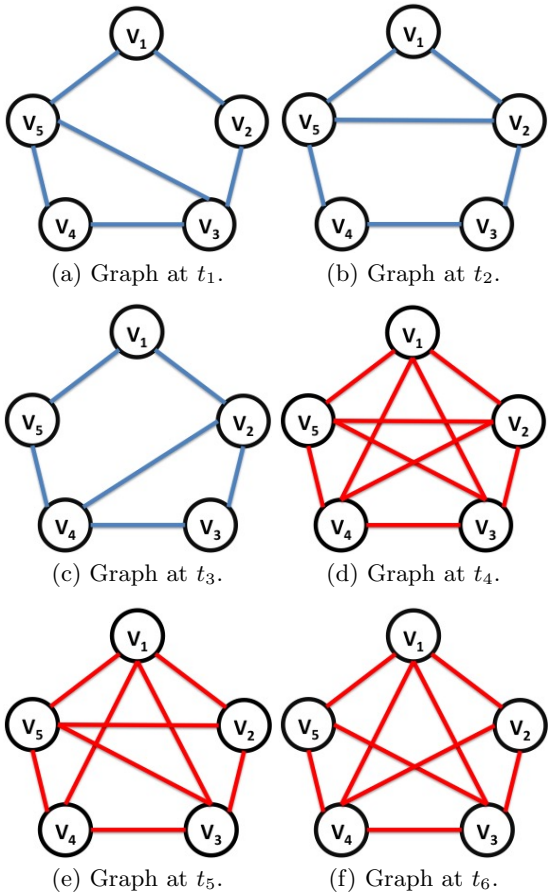
Another type of anomalies in dynamic graphs is changes or events. These are the most important categories of anomalies in networks since they can often be associated with important phenomenon in the real world. Unlike the node- and edge-based anomalies, this type of anomalous behaviour can only be found in dynamic networks. Event detection can be defined as time stamps when the graph structure is different from that of the other time stamps. This definition is much broader than the previously defined types of anomalies and contains edge or node anomalies as well. To address the problem of event detection, we need to measure the graph score for each time stamp in the dynamic network. Examples of graph scores are average clustering coefficient, average vertex degree, and so on. Each graph can be summarized through this scoring function f. We can define an event as a time stamp when the current graph score $f(G_t)$ is different from the previous and next graph scores, i.e., $f(G_{t-1})$ and $f(G_{t+1})$, as shown in Equation 9. The graph score is calculated using a score function f that yields a real-valued score for each graph, $f : G_t \to \mathbb{R}$. The average score $\widehat{f}$ is calculated based on the scores of the normal edges. An anomalous time stamp t is thus defined as:

$$|f(G_t) - f(G_{t-1})| > c_0 \quad and \quad |f(G_t) - f(G_{t+1})| > c_0 \tag{9}$$

where c_0 is the acceptable deviation from the average normal graph score. A simple example of a graph score is to calculate the number of edges or vertices at each time stamp and use it as a basis of comparison between two graphs. Note that the task of event detection in dynamic networks merely identifies an anomalous time-stamp. It does not provide any attribution to the underlying cause of anomalous behaviour.

Another type of anomaly detection in dynamic networks is change detection, which is closely related to events. While events occur suddenly and in an isolated manner, changes indicate the time-stamps, where the graph structure changes

and this change is maintained afterwards until another change point is encountered. Figure 5 shows an example of a change point in a dynamic network. At the fourth time-stamp, the structure of the network has changed to a semi-full clique and it remains the same afterwards.



(a) Graph at t_1. (b) Graph at t_2.

(c) Graph at t_3. (d) Graph at t_4.

(e) Graph at t_5. (f) Graph at t_6.

Figure 5: Example of a time sequence of graphs that contains a change point at t_4.

One of the approaches for event detection in dynamic networks is [49]. The authors in [49] introduce a technique called ParCube, which uses tensor decomposition to efficiently represent the data using sparser latent factors. ParCube is a parallelisable approach for fast tensor decomposition and can be applied on large-scale datasets. Since the data is stored in a tensor, ParCube can manage attributed graphs as well as plain graphs.

The authors in ParCube employ a random sampling strategy to select slices of the original tensor and thereafter perform tensor decomposition on these more manageable slices. The random sampling and decomposition on different slices can be done simultaneously. They then merge the smaller decompositions and combine them to retrieve the overall tensor decomposition outcome. ParCube is used for event detection by computing the reconstruction error of the tensor decomposition from one time-stamp to another.

In addition to ParCube, there are other approaches that use tensor decomposition as the basis of event detection. For instance the authors in [12] introduce a technique called Com2, which detects community-based anomalies in a dynamic network. The intuition behind this approach is to determine comets or communities that appear and disappear periodically. The authors use low-rank tensor factorization in combination with Minimum Description Length (MDL) to detect communities in a temporal environment.

Another event detection technique is introduced in [17], where the authors propose an approach called NetSimile, which extracts structural features from each graph. These features comprise the signature vector for each graph in the dynamic network setting. The problem of finding the graph similarity between two graphs is reduced to finding the distance of their corresponding signature vectors. NetSimile uses the Canberra distance between the pairs of signature vectors. The features that NetSimile captures include ego-net properties, node degree, clustering coefficient and so on.

Finally the last techniques that we discuss in this category are [40; 52; 53], which were introduced recently. All of these approaches compare consecutive graphs based on their structural features. The authors in [40] introduce an approached called Delta Connectivity (DeltaCon), where the underlying intuition is to calculate node affinities, i.e., the influence that nodes have on each other, by using Fast Belief propagation. DeltaCon uses the pairwise node affinities to measure graph similarity. The authors in [52] propose a scalable technique that creates a compact yet structure-aware feature set for each graph using a matrix permutation technique called Amplay. The matrix permutation step can be used as a heuristic for determining the Maximal Independent Sets (MISs) in a graph. The resulting feature set includes the rank of each node in a graph and this rank ordering is used by rank correlation for comparing a pair of graphs. Finally, [53] proposes a structure aware graph embedding scheme based on random projection and exploits the Johnson and Lindenstrauss lemma to provide a theoretical proof of its performance. They capture features pertaining to community structure or topological proximity of nodes in a graph.

Table 1 provides a comparison between dynamic graph anomaly detection schemes.

4. APPLICATION: FOREST FIRE RISK PREDICTION

In this section we fist review a few methods currently used for forest fire danger assessment and then we show how anomaly detection techniques can be used to predict the risk of forest fire.

Natural disasters are a prevalent reality around the world. Roughly 102 million people worldwide were affected by natural disasters in 2014 alone [4], with a global annual economic loss estimated at over $300B [5]. Researchers link natural disasters with climate change and statistics show that in 2014, 87% of worldwide natural disasters were climate-related [4]. Some countries, including Australia, experience forest fires, locally known as bushfires, as the most damaging disasters. In the Australian state of Victoria, forest fires pose the largest annual risk to the safety of residents [3]. For example, in the recent 'Black Saturday' Victorian forest fire event of February 2009, over 1.1 million acres burnt, 173 people lost their lives and over 400 people were injured [2]. Another notable impactful event was 'Ash Wednesday' in February 1983, with 75 fatalities and over 1 million acres burnt in the states of Victoria and South Australia. The

Table 1: Comparison of dynamic graph anomaly detection schemes. The graph properties are summarized in the first four columns, the type column corresponds to the category of anomaly detection, i.e., whether the method detects node, edge or subgraph anomaly. The last column denotes the time complexity of each method where high complexity means nonlinear in regards to the number of edges.

Method	Graph Properties				Type	Complexity
	Plain	Attributed	Weighted	Unweighted		
[45]	✓	×	✓	×	Subgraph-Region	Linear in #edges
[33]	✓	×	✓	✓	Community Membership	Linear in #edges
[12]	✓	×	✓	✓	Change	Linear in #edges
[55]	✓	×	✓	✓	Node-Role Membership	Linear in #edges
[40]	✓	✓	✓	✓	Change	Linear in #edges
[17]	✓	×	×	✓	Change	Linear in #edges
[49]	✓	✓	✓	✓	Change	Parallel implementation

impact of these disasters is multi-dimensional, from social and psychological to economic and environmental.

The reality described above makes a strong case for forest fire research, not only in Australia but also around the world. One of the most pressing problems in this context is the ability to predict the risk of a forest fire event. By knowing the risk, government agencies, communities and individuals can be better informed so they can take the most appropriate measures to mitigate and prepare for of forest fire events, if and when they eventuate.

In Australia, the McArthur Forest Fire Danger Index (FFDI) [43] is used by the national Bureau of Meteorology to draw Fire Danger Index maps [1], which in turn are used by state forest fire authorities to determine Fire Danger Ratings (FDR). FFDI is calculated based on temperature, humidity, wind speed, dryness and fuel weight as shown in Equations 10 and 11 [48]:

$$FFDI_\gamma = 2e^{-0.45+0.987*ln(DF_\gamma)-0.0345H_\gamma+0.0338T_\gamma+0.0234V_\gamma} \tag{10}$$

$$DF_\gamma = \frac{0.191(I_\gamma+104)(N_\gamma+1)^{1.5}}{3.52(N_\gamma+1)^{1.5}+Pre_\gamma-1} \tag{11}$$

where H_γ is relative humidity, T_γ is air temperature, V_γ is average wind velocity in the open at the height of 10 m, $FFDI_\gamma$ is forest fire danger index and DF_γ is drought factor which uses precipitation observations Pre_γ, N_γ is time since rain and I_γ is soil's moisture content all at time γ.

The output FFDI values are further interpreted based on a defined categorisation. Figure 6 depicts the six different categories. Using the above equations and these categories, the Australian emergency management agencies find the relevant category and broadcast it daily.

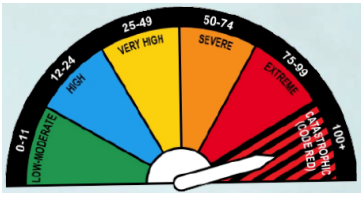


Figure 6: Forest fire Danger Categories in Australia

The main drawback of operational forest fire rating systems including FFDI is that the danger index is calculated based on the most current weather observations available, using a static model, and not customised to the particular location where the danger index is used. This determines a coarse variation between the danger indices of consecutive days and is not indicative of the temporal development of the forest fire danger conditions, which can be observed prior to forest fire events.

Along with FFDI, other approaches to predict forest fire danger also exist around the world. There is a vast body of work modelling the risk of forest fires with diverse focus. Some investigate the likelihood of forest fires, while others study the intensity or effects of forest fires (based on ecological, social or economic values) [44]. To counter the issue of spatially limited data (e.g. received from sparse weather stations), a number of studies look at forest fire danger detection using sensor networks. For example, [64] propose a wireless sensor network where sensor nodes are used to collect the data (e.g. temperature, relative humidity) and submit the information to specific cluster nodes. The cluster nodes will then construct neural networks to produce a weather index showing the likelihood for the weather to cause a forest fire. The paper does not describe the actual model hence the spatiotemporal customisation of the model cannot be assessed. In [31], it is claimed that forest fire risk needs to consider both forest fire behaviour probability and effect. The former depends on the spatial and temporal factors controlling forest fire spread, including fuel and weather. Moreover, the authors investigate the burn probability based on historic forest fire data, and highlight the need to move beyond assumptions of spatial and temporal uniformity while modelling that probability. Their approach is in line with [30] that argues that localised spatial properties (topography, fuel, weather) produce local differences in the forest fire behaviour, hence local differences in forest fire risk.

4.1 Anomaly detection based approach

In this subsection we argue how anomaly detection can be used for forest fire risk prediction. The input data for most of the forest fire danger prediction methods are meteorological observations sourced from weather stations (e.g. temperature, wind speed, wind gust, wind direction relative humidity, precipitations) together with spatial features such

as topography, vegetation type and vegetation density to predict the forest fire danger/risk for the given conditions based on empirical methods. The meteorological observations are examples of evolving data streams. Any of the anomaly detection techniques described in Section 2 can be used to detect forest fire risk. Note here extreme weather conditions can be seen as high risk episodes. In [59] a similar approach is used and it is shown that leveraging anomaly detection approaches is beneficial in predicting risk of forest fire in 3 weather stations in Australia, and as a result giving the decision makers more fine-grained information about the increasing forest fire danger and allowing them to take even better informed decision to protect communities and the environment

If we monitor meteorological features in more than one location then we can consider both meteorological as well as spatial features along with their relationships. Hence we can represent this dataset as an evolving graph. Anomaly detection techniques describes in Section 3 can be potentially used to model this problem and predict the risk of forest fire.

5. CONCLUSIONS

While there exists a considerable number of literature surveys capturing a variety of techniques in detecting anomalies in static data [22; 9], anomaly detection in the presence of evolving data has not been well explored. In this paper, we have considered the problem of detecting anomalies in evolving data and reviewed the existing 'unsupervised' anomaly detection techniques in this domain. We first surveyed techniques proposed for data streams and then focused on more complex scenarios, i.e., evolving graphs. We have also showcased the importance of anomaly detection in dynamic settings through a real-world application example, i.e., forest fire risk prediction.

5.1 Future Work

One of the emerging directions of research in detecting anomalies is their implications in high dimensional data streams or attributed dynamic graphs. Although there have been limited attempts by [60; 65; 32] in data streams, most of the existing algorithms in this survey lose their effectiveness in the presence of high dimensional data. Thus, we need to redesign the current models to be able to detect outlying patterns accurately and efficiently. More specifically, when there is a large number of features, there might exist a set of anomalies that emerge in only a subset of dimensions at a particular period of time. This set of anomalies may appear normal with regards to a different subset of dimensions and/or period of time. Another interesting future research direction is the appearance/disappearance of new/old data dimensions over time. This is an interesting area with many potential applications such as anomaly detection in IoT devices, where the sensors (representing the number of dimensions) can go off/on intermittently over time.

Another emerging area is ensemble analysis. Ensemble analysis is a technique that has been shown to improve the accuracy of many data mining tasks including anomaly detection in static data. Initial attempts on ensemble analysis for anomaly detection in streaming setting are described in [58; 32]. However, this area of research is still unexplored and more accurate models can be proposed based on the bias-variance trade-off.

6. REFERENCES

[1] Australian bureau of meteorology weather stations. http://www.bom.gov.au/vic/forecasts/fire-map.shtml.

[2] Black saturday bushfires. https://en.wikipedia.org/wiki/black saturday bushfires.

[3] Emergency management victoria strategic action plan. https://www.emv.vic.gov.au/plans/strategic-action-plan/.

[4] The human cost of natural disasters 2015: a global perspective, http://reliefweb.int/report/world/human-cost-natural-disasters-2015-global-perspective.

[5] The united nations office for disaster risk reduction, http://www.unisdr.org/archive/42814.

[6] C. C. Aggarwal. *Outlier Analysis*. Springer, 2013.

[7] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *International Conference on Very Large Data Bases (VLDB)*, pages 81–92, 2003.

[8] M. Agyemang, K. Barker, and R. Alhajj. A comprehensive survey of numeric and symbolic outlier mining techniques. *Intelligent Data Analysis*, 10(6):521–538, 2006.

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

[10] F. Angiulli and F. Fassetti. Detecting distance-based outliers in streams of data. In *Internaitonal Conference on Information and Knowledge Management (CIKM)*, pages 811–820, 2007.

[11] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, volume 2, pages 15–26, 2002.

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

[13] I. Assent, P. Kranen, C. Baldauf, and T. Seidl. Anyout: Anytime outlier detection on streaming data. In *Database Systems for Advanced Applications (DASFAA)*, pages 228–242, 2012.

[14] V. Barnett and T. Lewis. *Outliers in statistical data*, volume 3. Wiley New York, 1994.

[15] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. A. Baeza-Yates. Link-based characterization and detection of web spam. In *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 1–8, 2006.

[16] R. J. Beckman and R. D. Cook. Outliers. *Technometrics*, 25(2):119–149, 1983.

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

[18] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*, pages 443–448, 2007.

[19] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *ACM SIGMOD*, volume 29, pages 93–104, 2000.

[20] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SIAM International Conference on Data Mining (SDM)*, pages 328–339, 2006.

[21] L. Cao, D. Yang, Q. Wang, Y. Yu, J. Wang, and E. A. Rundensteiner. Scalable distance-based outlier detection over high-volume data streams. In *International Conference on Data Engineering (ICDE)*, pages 76–87, 2014.

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

[23] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 24(5):823–839, 2012.

[24] Y. Chen and L. Tu. Density-based clustering for real-time stream data. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142. ACM, 2007.

[25] M. Chenaghlou, M. Moshtaghi, C. Leckie, and M. Salehi. An efficient method for anomaly detection in non-stationary data streams. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2017.

[26] M. Chenaghlou, M. Moshtaghi, C. Leckie, and M. Salehi. Online clustering for evolving data streams with online anomaly detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2018.

[27] Q. Ding, N. Katenka, P. Barford, E. Kolaczyk, and M. Crovella. Intrusion as (anti) social communication: characterization and detection. In *Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 886–894. ACM, 2012.

[28] M. Elahi, K. Li, W. Nisar, X. Lv, and H. Wang. Efficient clustering-based outlier detection algorithm for dynamic data stream. In *International Conference on Fuzzy Systems and Knowledge Discovery*, volume 5, pages 298–304, 2008.

[29] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, volume 96, pages 226–231, 1996.

[30] C. A. Farris, C. Pezeshki, and L. F. Neuenschwander. A comparison of fire probability maps derived from gis modeling and direct simulation techniques. In *Joint Fire Science Conference and Workshop*, pages 131–138, 1999.

[31] M. A. Finney. The challenge of quantitative risk analysis for wildland fire. *Forest Ecology and Management*, 211(1):97–108, 2005.

[32] S. Guha, N. Mishra, G. Roy, and O. Schrijvers. Robust random cut forest based anomaly detection on streams. In *International Conference on Machine Learning*, pages 2712–2721, 2016.

[33] M. Gupta, J. Gao, Y. Sun, and J. Han. 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)*, pages 859–867. ACM, 2012.

[34] D. M. Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[35] K. Henderson, T. Eliassi-Rad, C. Faloutsos, L. Akoglu, L. Li, K. Maruhashi, B. A. Prakash, and H. Tong. Metric forensics: a multi-level approach for mining volatile graphs. In *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 163–172. ACM, 2010.

[36] E. M. Knorr and R. T. Ng. A Unified Notion of Outliers: Properties and Computation. In *ACM International Conference on Knowledge Discovery and Data Minning (KDD)*, pages 219–222, 1997.

[37] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *International Journal on Very Large Data Bases (VLDB)*, 8(3-4):237–253, 2000.

[38] E. M. Knox and R. T. Ng. Algorithms for mining distancebased outliers in large datasets. In *International Conference on Very Large Data Bases (VLDB)*, pages 392–403, 1998.

[39] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos. Continuous monitoring of distance-based outliers over data streams. In *International Conference on Data Engineering (ICDE)*, pages 135–146, 2011.

[40] D. Koutra, N. Shah, J. T. Vogelstein, B. Gallagher, and C. Faloutsos. Delta con: Principled massive-graph similarity function with attribution. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(3):28, 2016.

[41] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, 29(2):249–272, 2011.

[42] H.-P. Kriegel, P. Kröger, and A. Zimek. Outlier detection techniques. In *Tutorial at the International Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[43] A. G. McArthur. Fire behaviour in eucalypt forests. 1967.

[44] C. Miller and A. A. Ager. A review of recent advances in risk analysis for wildfire management. *International journal of wildland fire*, 22(1):1–14, 2013.

[45] M. Mongiovi, P. Bogdanov, R. Ranca, E. E. Papalexakis, C. Faloutsos, and A. K. Singh. Netspot: Spotting significant anomalous regions on dynamic networks. In *Proceedings of the 13th SIAM International Conference on Data Mining (SDM)*, pages 28–36. SIAM, 2013.

[46] M. Moshtaghi, J. C. Bezdek, T. C. Havens, C. Leckie, S. Karunasekera, S. Rajasegarar, and M. Palaniswami. Streaming analysis in wireless sensor networks. *Wireless Communications and Mobile Computing*, 14(9):905–921, 2014.

[47] M. Moshtaghi, J. C. Bezdek, C. Leckie, S. Karunasekera, and M. Palaniswami. Evolving Fuzzy Rules for Anomaly Detection in Data Streams. *IEEE Transactions on Fuzzy Systems*, 2014.

[48] I. Noble, A. Gill, and G. Bary. Mcarthur's fire-danger meters expressed as equations. *Australian Journal of Ecology*, 5(2):201–203, 1980.

[49] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *Machine Learning and Knowledge Discovery in Databases*, pages 521–536. Springer, 2012.

[50] D. Pokrajac, A. Lazarevic, and L. J. Latecki. Incremental local outlier detection for data streams. In *Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 504–515. IEEE, 2007.

[51] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *ACM International Conference in Management of Data (SIGMOD)*, volume 29, pages 427–438, 2000.

[52] L. Rashidi, A. Kan, J. Bailey, J. Chan, C. Leckie, W. Liu, S. Rajasegarar, and K. Ramamohanarao. Node re-ordering as a means of anomaly detection in time-evolving graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 162–178. Springer, 2016.

[53] L. Rashidi, S. Rajasegarar, and C. Leckie. An embedding scheme for detecting anomalous block structured graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 215–227. Springer, 2015.

[54] J. Ren and R. Ma. Density-based data streams clustering over sliding windows. In *International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 5, pages 248–252. IEEE, 2009.

[55] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson. Modeling dynamic behavior in large evolving graphs. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pages 667–676. ACM, 2013.

[56] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*, volume 589. John Wiley & Sons, 2005.

[57] M. Salehi, C. Leckie, J. C. Bezdek, T. Vaithianathan, and X. Zhang. Fast memory efficient local outlier detection in data streams. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3246–3260, 2016.

[58] M. Salehi, C. A. Leckie, M. Moshtaghi, and T. Vaithianathan. A relevance weighted ensemble model for anomaly detection in switching data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 461–473. Springer, 2014.

[59] M. Salehi, L. I. Rusu, T. Lynar, and A. Phan. Dynamic and robust wildfire risk prediction system: an unsupervised approach. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 245–254. ACM, 2016.

[60] S. Sathe and C. C. Aggarwal. Subspace histograms for outlier detection in linear time. *Knowledge and Information Systems*, pages 1–25, 2018.

[61] K. Yamanishi and J.-i. Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 676–681, 2002.

[62] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 320–324, 2000.

[63] D. Yang, E. A. Rundensteiner, and M. O. Ward. Neighbor-based pattern detection for windows over streaming data. In *Advances in DB Tech.*, pages 529–540, 2009.

[64] L. Yu, N. Wang, and X. Meng. Real-time forest fire detection with wireless sensor networks. In *International Conference on Wireless Communications, Networking and Mobile Computing*, volume 2, pages 1214–1217, 2005.

[65] J. Zhang, Q. Gao, and H. Wang. Spot: A system for detecting projected outliers from high-dimensional data streams. In *IEEE International Conference on Data Engineering*, pages 1628–1631. IEEE, 2008.

[66] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, volume 25, pages 103–114. ACM, 1996.

[67] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *International Conference on Very Large Data Bases (VLDB)*, pages 358–369. VLDB Endowment, 2002.