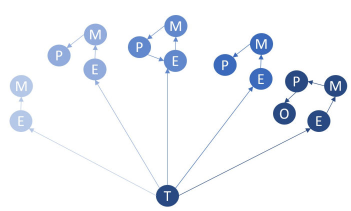
TEMPO

Management and Processing of Temporal Networks

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

D5.2: Outlier Detection Algorithms in Time Evolving Networks



Computer Engineering & Informatics Department
University of Patras
Greece
30/11/2025

D5.2: Outlier Detection Algorithms in Time Evolving Networks

Konstantinos Christopoulos and Kostas Tsichlas

November 30, 2025

Abstract

In this deliverable we discuss two solutions for anomaly detection in dynamic graphs. The first one, which was published in [6], is a local community-based event detection algorithm in a streaming environment. The second one, which is not published yet since it is not concluded by the time of the writing of this report, leverages deep learning methods for event detection. We describe both methods and provide experimental results. Unfortunately, the complexity in the development of the T-Janusgraph system, did not allow us to adapt and test these algorithms due to limited time, but we intend to do so in the near future.

1 Introduction

The ability to detect unusual patterns, or anomalies, is critical across diverse fields ranging from cybersecurity and finance to infrastructure monitoring. While traditional anomaly detection (AD) has focused on static data, real-world interactions are inherently dynamic and best represented by time-evolving networks (also known as temporal graphs). These networks, characterized by the continuous appearance and disappearance of nodes and edges, pose a significant challenge for traditional AD methodologies. Identifying anomalies in this dynamic context often requires detecting sudden or unexpected changes in the network's relational structure over time. A temporal network can be modeled either as a sequence of static graph snapshots or as a network with explicit time annotations on its components. In either representation, the constantly shifting topology leads to transformations in strongly connected sub-graphs, or communities. These changes (such as the emergence of a new community, the vanishing of an existing one, or the rapid growth or contraction of a group) are often the strongest indicators of anomalous behavior, which are consequently categorized as community-based anomalies (CBA). This work addresses the challenge of identifying these structural anomalies by focusing on the evolution of local community structures.

This paper presents a two-pronged approach to Outlier Detection Algorithms in Time Evolving Networks, initially focusing on a community-based approach before reviewing modern deep learning methodologies. In the first part, we integrate and adapt an existing dynamic local community detection algorithm specifically to monitor the status of a community centered around a "seed" node. By establishing a formal definition for a consistent community and using size deviation metrics, we formally define six pivotal events (Growth, Contraction, Birth, Vanish, New "Expanded," and New "Shrank") that characterize the evolutionary behavior of a Local Community (LC). These events are then utilized to define and detect instant and sustained community change anomalies. Furthermore, we provide a preliminary experimental evaluation of this approach using synthetic datasets generated by the RDyn tool.

In the second part, the paper pivots to the state-of-the-art in anomaly detection, discussing the application of Deep Learning (DL) methods to dynamic graphs, specifically utilizing the DyGED (Dynamic Graph Event Detection) framework. DL models, particularly those leveraging Graph Neural Networks (GNNs) and temporal components like LSTMs or attention mechanisms, excel at capturing the dual spatial and temporal complexities of evolving graph data. This section provides a rudimentary discussion on the DyGED architecture, its variants, and preliminary results demonstrating its high effectiveness in detecting structural macro-dynamic events, such as community merge and split anomalies, in controlled synthetic environments.

The remainder of this work is structured as follows. Section 2 begins with a review of related work on community-based anomaly detection, followed by the formal presentation of our proposed local community-based anomaly detection method and its preliminary results. Section 3 introduces the Deep Learning architectural primitives for Dynamic Graph Anomaly Detection, detailing the application and empirical validation of the DyGED framework.

2 Local Community-Based Anomaly Detection in Graph Streams

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

In general, a static network is represented as $G = (V, E)$, where V is the set of vertices (entities) and E is the set of edges (interactions/relations between entities). An edge can be directed, such as the connection between two people where one sends an email to another, or undirected, such as the connection between two collaborating peers. Lastly, edges among nodes can be associated with weights (e.g., frequency of interactions) or nodes can be associated to weights (e.g., specific properties of nodes). In many cases, real-world networks are dynamic/temporal, in the sense that new edges or nodes appear and existing edges or nodes disappear. A temporal network can be represented either as a sequence of static graphs (snapshots) or as a network with time annotations on its nodes/edges that represent its time evolution. The former approach requires the specification of the size of the time window that defines the time instances of snapshot construction. The latter representation is related to events, like edge/node insertion or deletion or its existence interval. The time annotation may have different aspects/interpretations depending on the application. The following aspects in order of generality can be used:

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

2. **Valid Time:** it signifies the time period in which an object is valid [3, 4] (i.e. the time interval that an object existed in the graph). Transaction time can be emulated in the valid time setting by restricting updates to intervals that begin on the time moment of the update. Valid time is the time during which a fact is true in the real world.

As numerous changes such as the appearance or disappearance of edges and nodes occur in a temporal graph over time, the statuses of strongly connected sub-graphs or communities undergo transformations with the network's evolution. Instances of these changes include the emergence of new communities, the vanishing of existing ones, and the division of existing communities into multiple entities. Such alterations in the community structure can signal anomalous behavior. Techniques that identify anomalies by monitoring community evolution are recognized as community-based anomaly detection methods.

Finally, within the literature, an exploration reveals the categorization of anomalies into four distinct types, each delineated by its unique nature and characteristics.

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

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

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

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

In this work, our contribution primarily revolves around integrating an existing dynamic local community detection algorithm [27] into the framework for detecting anomalies. We significantly contributed to adapting and enhancing this algorithm to effectively identify instant and sustained community changes within temporal networks. Furthermore, we conducted preliminary[1] experimentation on synthetic datasets to evaluate the performance of the proposed algorithm. In what follows we provide a short description of the structure of this work. In 2.1 we discuss briefly the related work in community based anomaly detection. Then, in 2.2 we present our proposing local community based anomaly detection method, and in 2.3 we present preliminary results of our model. Finally, we conclude in 2.4.

2.1 Related Work on Community-Based Anomaly Detection Methods

Based on the literature, a typical two-stage approach is used for anomaly detection in dynamic networks. Initially, the network is transformed into a standard graph representation to extract features. Then, existing outlier detection methods are applied to determine anomalies in the graph. We identify five main approaches for generating graph representations and extracting features: Graph Feature-based, Decomposition and Compression-based, Machine/Deep Learning-based, Community-based, and Probabilistic Model-based methods. Given the focus of the current study on community-based methods, we will now highlight recent works in this area.

An event detection method based on the evolution of the community structure in temporal networks, is presented in [21]. The authors assume that large events activate information diffusion, which in turn affects community borders. By applying network aggregation in a certain time period, researchers utilize the InfoMap algorithm to detect communities. Consequently, intra-community and inter-community links are calculated in each time interval, and when the difference between inter-community links and intra-community links is greater or equal to a threshold (mean and standard deviation), then an event is detected. Experiments with the Enron and Boston Marathon bombing networks, show the effectiveness of this method. Similarly, in [10], anomalous events in graph snapshots are detected by utilizing the community boundary nodes.

An improved approach of [21], is discussed in [1]. The authors detect events by combining two different approaches. First, like [21], researchers check communication trends among communities calculating the inter-community and intra-community links. The second approach (community structure-based) is divided into two stages: 1) Check the number of extracted communities in consecutive time steps, examining several buckets in terms of community size and 2) track, in consecutive time steps, the number of central nodes inspecting the ratio changes. Moreover, stages 1 and 2 can be combined, since sometimes, the number of communities and central nodes are affected simultaneously from events. Finally, for the initialization of the method, existing algorithms are used to detect the initial communities and central nodes. Experiments in real datasets show that community structure-based methods are more scalable and faster than others. A relatively recent approach that detects abnormal hosts (nodes with high centrality degree), is described in [25]. In this method, authors focus on discovering collective abnormalities by considering community activities. This approach consists of three steps: First, by utilizing Spark GraphX they create the graph model. Second, by applying a well-known static algorithm, they detect all the host communities. Third, by matching the communities in consecutive time steps, the event type for each community (merge, split, appear, ..etc) is identified and four feature types are estimated. Finally, three different anomaly types are considered: Change in the distance of feature space utilizing the aforementioned four feature types, change in the scale of community and change in the life-cycle of network evolution.

Utilizing co-evolution pattern mining, authors in [13] detect anomalous (target) nodes in multi-typed information networks, i.e., heterogeneous bibliographic information network (HBIN). The proposed algorithm consists of three steps. First, the co-evolution patterns are extracted using relational and evolution constraints on the dataset. Then, an existing multi-typed clustering method

[1]Code available at `https://github.com/kostasada7/Local-Community-Based-Anomaly-Detection-in-Graph-Streams`

is used in order to cluster the patterns. Finally, by estimating the similarity index among target (author) and attribute (paper-count, co-author, and venue) objects, the anomaly score is calculated and the anomalous nodes are identified.

A method that detects anomalous communities based on the community evolution in dynamic graphs, is presented in [5]. To reduce the computational cost, they introduce the notion of graph representatives and community representatives. Taking advantage of these representatives, in each timestamp the algorithm identifies the predecessors and successors for each community, if any. Lastly, exploiting the decision rules, six different types of community-based anomalies are proposed i.e. Grown, Shrunken, Merged, Split, Born, and Vanished community. Experiments in both synthetic and real datasets are conducted exhibiting the efficiency and effectiveness of the method for detecting anomalies. A method that detects community-based change points in snapshots, is analyzed in [20]. In this work, three different network types are considered: 1) static, 2) semi-static and, 3) dynamic. In the first type, nodes are presented in each time slice whereas in semi-static, nodes can be removed. In both cases, the partition method, independently of the community detection algorithm, is constructed using the co-group network. Regarding the third type, the authors use the first two cases to calculate the parameters for an anomaly detection-based streaming method. The experimental evaluation shows that this method is more efficient and less sensitive compared to Generalized Louvain and GraphScope [24].

In [18], a combination of change point detection and community evolution events is proposed. First, the network is divided into snapshots and in order to capture the network change, the change detection range is extended to a time window instead of looking at only two consecutive snapshots. Then, by utilizing matching community measures, the authors find the community evolution event that occurred. Another anomaly detection algorithm, in dynamic attributed networks, is proposed in [28]. There, a function that denotes the anomalous score is utilized. The authors use dynamic graph clustering with a community detection model by ranking nodes based on 1) how close they are to a dense community center and 2) the deviation from current and historical behavioral data, in order to identify anomalies. In [19], authors utilize the sliding window technique to generate multivariate subsequences and apply a modified fuzzy clustering algorithm to detect the structure. Then, the multivariate subsequences, the optimal cluster centers and the partition matrix are reconstructed. By calculating a confidence index, authors quantify a level of anomaly detected in the series and apply Particle Swarm Optimization for the problem of outlier discovery.

In addition, for the purpose of identifying anomalous events in graph streams, and by using a novel definition of anomaly score based on the history of the actions of nodes, a community-based method is presented in [11]. In [29], a network of snapshots is constructed. The weight of each edge is defined as the similarity score between the corresponding snapshots. Then, by detecting the network communities, they check whether each community consists of similar snapshots and whether two consecutive snapshots belong to different clusters. Based on these checks, a change point anomaly is identified.

A novel method that detects changes in labeled and directed heterogeneous stream graphs is presented in [17]. By using the graph substructures and GED (Graph Edit Distance), they construct the network embedding. Consequently, cluster construction (using the k-Medoid algorithm) is performed and the incoming graphs are compared to the clusters to spot potential anomalies with respect to communities. One more recent research on the field of community and anomaly detection is presented in [15]. For each snapshot, they use the Louvain algorithm and the LPA to detect communities. Then, to discover anomalies, they do the following: 1) they utilize a bipartite graph to capture the community evolution between two consecutive time steps, and then, 2) from the constructed evolutionary paths, they detect possible community abnormalities. Other papers with similar results can be found in [22],[26],[9],[14]

2.2 Preliminaries and Problem Formulation

A dynamic network $G = (V, E_t)$ consists of a node set $V = 1, ..., n$ and a set of edges with time stamps E_t. These edges represent interactions between nodes at particular time points t, where $t = 1, 2, ..., n$, and are generated by an interaction streaming source (ISS). The ISS generates stream updates, such as edge insertions or deletions, among both new and existing nodes in the network. Consequently, the communities within the network also undergo changes as the network evolves. This study treats G as an unweighted and undirected network. The neighbors of a node v

are nodes directly connected to v, and the degree of a node is the count of its neighbors. To uncover anomalies based on community structure, our primary objective is to detect the community centered around the seed node. This seed could represent a node of particular importance based on external information or a node with distinctive topological features. We define a Local Community (LC) as the community to which the seed node belongs. Thus, a network G can be partitioned into the LC and the remaining network U, where $G - LC = U$.

In a greedy local community detection method, various quality metrics measure the quality of a local community. In this study, the chosen metric is f_{monc} (henceforth referred to as fitness score), which quantifies the sum of internal community edges divided by the total sum of internal and external community edges [12]. An internal edge denotes a connection between two nodes within the local community, while an external edge signifies a connection between a community node and a node within its neighborhood. In Equation 1 below, k_{in}^C and k_{out}^C represent the internal and external edges of community C, respectively.

$$f_{monc}(C) = \frac{2k_{in}^C + 1}{2k_{in}^C + k_{out}^C}, \tag{1}$$

Below, we delve into a more detailed explanation of the LCD framework. Given a static network G and an initial seed U_0, our objective at each step is to incorporate a new member node into C. Initially, the community comprises solely the seed, denoted as $C(U_0)$, with a corresponding fitness score of $f_C = \frac{1}{k_{out}^C}$. Subsequently, the algorithm scans through all community neighbors to identify the node whose potential inclusion maximizes the community's fitness score. Upon identifying such a node, the algorithm evaluates whether its estimated fitness score surpasses that of the current community state (i.e., without the new node). If this condition holds, the new node is integrated into the community. This process iterates, with each new node addition resulting in an increment in the fitness score of C, leading to the final fitness scores arranged in ascending order. For example, consider a scenario where a community C initially comprises only the seed U_0. Upon exploring the neighbors of U_0, if U_1 is identified as offering the highest fitness score for C among all neighbors of U_0, and the addition of U_1 to C enhances its fitness score (i.e., $f_{U_1} > f_{U_0} = f_C$), then U_1 is incorporated into C, resulting in a new community fitness score of $f_C = f_{U_1}$. This iterative process continues until no new candidate node can further enhance the fitness score of C.

In this study, we incorporate the dynamic version of the previously described static algorithm. This dynamic algorithm [27] allows us to identify evolving local communities, with the primary objective of detecting anomalies based on community structures. The static algorithm is now augmented with dynamic capabilities, enabling continuous adjustments to the local community around a seed node as the graph evolves. This dynamic approach provides communities with robust fitness scores and significant overlap with those generated by the static algorithm, eliminating the need for rerunning the static algorithm whenever the graph undergoes modifications.

2.2.1 Problem Formulation

A Local Community (LC) denotes the community to which the seed node belongs. These seed nodes act as the defining nodes for the community under examination. Given a dynamic network G and a continuous flow of edges generated by the ISS, our aim is to estimate the changes in the status of LC between successive time steps. In dynamic networks, the potential of a local community undergoes alterations over time, particularly regarding its size. Specifically, the number of nodes constituting our community of interest may fluctuate between consecutive time steps. Using deviations in size as a guide, we define six events that characterize the evolutionary behavior of the LC. However, before exploring these events, it is crucial to establish a definition of a consistent community as time evolves.

Definition 1. ***Characteristics of a consistent community in a temporal network:*** *With a parameter d representing a small percentage, the C^i community remains unchanged at the i^{th} time step if C^{i-1} shares at least $(1 - d)$ of its members with C^i and the absolute difference in their sizes is equal to or less than d times the size of C^{i-1}.*

$$||C^i| - |C^{i-1}|| \leq d|C^{i-1}| \text{ and } |C^i \cap C^{i-1}| \geq (1 - d)|C^{i-1}| \tag{2}$$

The relation above implies that a community maintaining consistency should demonstrate minimal change over time. However, to differentiate between unstable and consistent communities, we introduce a slight relaxation parameter d, which permits the addition or removal of peripheral nodes within the community.

Hence, having established the local community LC and outlining a consistent community with Definition 1 , we proceed to formally define the six pivotal events as follows:

1. **Growth** - The C^i community is labeled as Growth at the i^{th} time step, if C^{i-1} share at least $1-d$ of its members with C^i, and its size is less than C^i:

$$|C^i| - |C^{i-1}| > 0 \text{ and } |C^{i-1}| \neq 1 \text{ and } |C^i \cap C^{i-1}| \geq (1-d)|C^{i-1}| \tag{3}$$

2. **Contraction** - The C^i community is labeled as Contraction at the i^{th} time step, if C^i share at least $1-d$ of its members with C^{i-1}, and its size is greater than C^i:

$$|C^i| - |C^{i-1}| < 0 \text{ and } |C^i| \neq 1 \text{ and } (1-d)|C^i| \leq |C^i \cap C^{i-1}| \tag{4}$$

The above scenarios aim to demonstrate that a local community can either grow or shrink significantly. Nevertheless, a substantial base of shared nodes remains stable over time.

3. **New "Expanded"** - The C^i community is labeled as New "Expanded" at the i^{th} time step, if C^{i-1} share less than $1-d$ of its members with C^i, its size is equal to or greater than C^{i-1}, and C^{i-1} contains more than one member:

$$|C^i| - |C^{i-1}| \geq 0 \text{ and } |C^{i-1}| \neq 1 \text{ and } |C^i \cap C^{i-1}| < (1-d)|C^{i-1}| \tag{5}$$

4. **New "Shrank"** - The C^i community is labeled as New "Shrank" at the i^{th} time step, if C^i share less than $1-d$ of its members with C^{i-1}, its size is greater than C^i, and C^i contains more than one member:

$$|C^i| - |C^{i-1}| < 0 \text{ and } |C^i| \neq 1 \text{ and } (1-d)|C^i| > |C^i \cap C^{i-1}| \tag{6}$$

Events 3 and 4 indicate that there are a few identical shared members (≥ 2) in local community at consecutive time steps. Consequently, we label them as "New Shrank" or "New Expanded".

5. **Vanish** - The C^i community that contains only one member, is labeled as vanish at the i^{th} time step, if C^{i-1} contains more than one member and share only one member with C^i:

$$|C^{i-1}| > 1 \text{ and } |C^i| = 1 \tag{7}$$

This event indicates that the new community consists only of the seed node while the community of the previous time step $i-1$ contains more than one member (seed node).

6. **Birth** - The C^i community that contains more than one member, is labeled as birth at the i^{th} time step, if C^{i-1} contains only one member:

$$|C^{i-1}| = 1 \text{ and } |C^i| > 1 \tag{8}$$

This event indicates that the community of the previous time step $i-1$ consists only of the seed node while the new community contains more than one member.

Given the definitions mentioned above, we proceed to delineate the types of anomalies based on the evolution of the community. Specifically, within a sequence of time steps, we seek to identify: 1) Instant community change and 2) sustained community change.

Definition 2. ***Instant community change:*** *This refers to a sudden alteration in the community size, lasting only for the current time step.*

Definition 3. ***Sustained community change:*** *This denotes a change in the community size that persists for at least two consecutive time steps.*

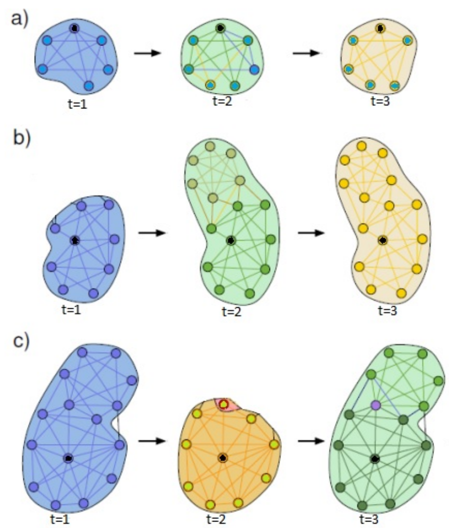


Figure 1: Most common anomaly scenarios in the local community evolution: a) No anomaly, b) sustained community change and c) instant community change. The black node in each of these communities constitutes the seed node (Figure from [2]).

In this work, time steps are considered as batches of updates. Specifically, following a defined number of edge insertions/deletions around the current community, we assess whether our local community (LC) undergoes an instant change, a sustained change, or remains consistent. In what follows, we provide an overview of local community evolution over time, illustrating three distinct scenarios, see Fig.1. In the first scenario, the local community remains consistent, showing minimal fluctuations in size over consecutive time steps. This indicates a state of immutability or stability within the community. Moving on to the second scenario, we observe growth/expansion in the local community between two consecutive time steps, followed by stability in size in the subsequent time step. This transition reflects a sustained community change. Finally, the third scenario depicts fluctuations in community size across all time steps, indicating instant community changes characterized by both contractions/shrink and expansions/growth in size within single time steps.

2.3 Experiments

2.3.1 Datasets

The synthetic datasets employed in our experiments are generated using RDyn [23], an approach designed to produce dynamic networks that exhibit properties akin to real-world networks. These datasets include time-dependent ground truth communities with adjustable quality, allowing for both the merging and splitting of communities. The RDyn generator operates based on two key user-defined parameters: the number of nodes in the dynamic network and the number of iterations. Each iteration comprises a batch of actions involving edge insertion or deletion, with the number of actions varying between iterations. In this study, we refrain from utilizing an initial graph as our starting point. Instead, we carry out experiments in a fully streaming fashion, commencing solely from the seed. In our preliminary experiment, we used three different datasets generated by the RDyn generator, and its fundamental characteristics are outlined in Table 1.

2.3.2 Experimental Results

In assessing our proposed framework, we compare the outcomes of our community-based anomaly detection approach with the ground truth communities generated by the synthetic dataset generator. For our evaluation, we focus on precision, recall, and the F1 score as the appropriate metrics. Precision represents the ratio of correctly identified anomalies to the total number of identified anomalies, while recall indicates the proportion of relevant anomalies that were successfully retrieved. The F1 score, being the harmonic mean of precision and recall [8], is preferred over a simple average as it penalizes extreme values.

Table 1: Synthetic datasets containing information about the number of nodes, iterations, initial/-final edges, and actions performed.

Synthetic Datasets	Nodes	Iterations	Final # of edges	Actions
$SD1$	1000	1000	3907	52257
$SD2$	2000	1000	10483	85025
$SD3$	5000	1000	22588	152144

In our experimentation, we employ the dynamic algorithm proposed by Zakrzewska et al. [27] to identify the community surrounding the seed node. The selection of the seed node is based on its degree. In our experiments, we used various high-degree nodes and explore different values for the parameter d. Ultimately, we fix the user-defined parameter d to 0.1. It's worth noting that multiple experiments have been conducted utilizing diverse ground-truth communities and seeds. In what follows, we provide representative results for each dataset.

After selecting a seed node based on predefined criteria, our approach employs the dynamic algorithm to detect the community surrounding this seed node across various time intervals. Subsequently, we apply the set of six criteria to identify potential events within these communities, flagging them as event instances. This procedure is executed for both the detected communities and the ground-truth communities, facilitating a comparative analysis during periods of abnormal community behavior. In Table 2, we summarize the detected events.

Table 2: Types and quantities of events present in both the detected and ground truth communities. A = Our method and B = Ground Truth.

Datasets		Event Types					
		Growth	Contraction	New "Expanded"	New "Shrank"	Vanish	Birth
SD1	A	2	2	1	1	0	1
	B	4	4	0	0	0	1
SD2	A	1	1	0	1	0	1
	B	2	2	0	0	0	1
SD3	A	2	2	3	0	0	1
	B	2	2	1	0	0	1

The table presented above details the individual events that were retrieved from the detected and the ground-truth community, in all three synthetic datasets. Regarding the first dataset, we have two Growth, two Contraction, one New "Expanded", one New "Shrank", and one Birth event for the detected community. On the other hand, for the ground truth community, we get four Growth, four Contraction, and one Birth event. At the outset of the process, we presume that both the ground truth and detected communities comprise solely of the seed node. At the initial time step when we compare both communities, their size exceeds 1, prompting us to record a Birth event for both the ground truth and the detected community. Notably, no Vanish events are observed. This observation is straightforward to interpret as the selected seeds are high-degree nodes, thus exhibiting cohesion within their periphery. We observe that the events identified in the detected community closely resemble those in the ground truth. In particular, we found one Growth/New "Expanded" and one Contraction/New "Shrank" event less, when compared to the ground truth. This disparity could be ascribed to the nature of the local community detection algorithm we utilized, wherein an edge update (deletion/insertion) in proximity to the seed may result in a sudden decrease/increase in community size. Nonetheless, the overall evolution of the community mirrors that of the ground truth.

Comparable trends emerge in the remaining synthetic datasets, SD2 and SD3. To elaborate, SD2 exhibits one less detected event compared to the ground truth, while SD3 has two more events than the ground truth. In the SD2 dataset, the variation is highlighted in Growth/New "Expanded" events, while in SD3, it's observed in New "Expanded" events. Essentially, these differences are not significant, as a thorough examination of the dataset reveals that the nature of the employed local community detection algorithm leads to these minor deviations. As mentioned earlier, certain edge updates within the 1-hop vicinity of the seed node lead to such size community fluctuations.

Table 3: Evaluation of the detected events. E = Exact matches and A = Approximate matches.

SD1	F1 score	Precision	Recall
E	59%	72%	50%
A	70%	86%	60%

SD2	F1 score	Precision	Recall
E	75%	100%	60%
A	75%	100%	60%

SD3	F1 score	Precision	Recall
E	62%	50%	80%
A	62%	50%	80%

Upon evaluating the detected events, we observe that our model achieves a reasonably high and consistent precision rate across all three datasets. More precisely, in the SD1 we reach a precision rate of 72% and an overall F1 score of almost 59%. However, the recall rate is slightly lower 50%, compared to precision, as there are instances where our detected events do not precisely match those in the ground truth. As mentioned above, this discrepancy can be attributed to the local community detection method used. In SD2, we achieve a precision rate of 100% and an overall F1 score of nearly 75%. On the other hand, the recall reaches a rate of 60%.

Table 4: The types and quantity of anomalies present in both the detected and ground truth communities.

		Anomaly Types	
		Instant	Sustained
SD1	Our method	1	6
	Ground Truth	1	8
SD2	Our method	0	4
	Ground Truth	0	5
SD3	Our method	2	6
	Ground Truth	1	5

Lastly, in SD3, our precision rate reaches 50% with an overall F1 score of almost 62%. Notably, the recall rate significantly surpasses precision, standing at 80%, since our detected events nearly align with those in the ground truth. Moreover, expanding the time window for both detected and ground truth events leads to a marked improvement in F1 score for SD1. Specifically, if a detected event, not directly aligned with a ground truth event, can be matched to ground truth events within a time interval of 2 time steps (rather than 1), the F1 score experiences a 11% enhancement, reaching 70%.

Since the detected anomalies are contingent on the identified events, we observe that our model detects quite similar anomalies across all three datasets, compared to the ground truth. Upon analyzing the results in Table 4, we notice a difference in instant change anomalies in SD3. Conversely, sustained change anomalies differ across all three datasets. Finally, we see an average of nearly 7 anomalies in both the ground truth and detected communities.

2.4 Conclusion

Our aim in this study is to conduct a thorough analysis of recent advancements in community-based anomaly detection and to explore foundational concepts concerning temporal networks. Additionally, we propose a well-established algorithm designed for dynamic local community detection, with the objective of identifying anomalies through the evolution of communities over time. Moreover, initial tests are performed using synthetic datasets. Building upon this work, we aim to expand our findings along the following dimensions: 1. Broadened experimental assessment encompassing both synthetic and real datasets. 2. Identification of node anomalies within or surrounding the community. 3.

Investigations involving historical graphs, which introduce a temporal aspect, enabling analysis of the evolution and fluctuations in connections between entities across various time spans.

3 Detecting Anomalies in Dynamic Graphs Using Deep Learning Methods

This is a recent work that has not been published yet (until the time of writing) and it concerns deep learning methods that have stormed almost all fields of science. Although not published yet, the project is publicly available in github[2]. Our main goal, when we decided to adopt such an apporach, was to look at deep learning methods for anomaly detection in dynamic/temporal graphs aiming at integrating in the future such methods with the T-Janusgraph system. In the following, we are providing a rudimentary discussion on related work, define our approach and the problem that we looked at, and finally provide preliminary experimental results.

3.1 Introduction

Anomaly Detection (AD), sometimes referred to as outlier detection, is the process of identifying unusual patterns, behaviors, or data points that deviate significantly from what is considered normal, typical, or expected within a dataset. In the context of dynamic graphs, this task frequently translates into event detection, where an "event" is defined as a short-lived or sudden deviation from the graph's normal behavioral patterns across a sequence of snapshots. The applications of Dynamic Graph Anomaly Detection (DGAD) are critical across various domains for ensuring system integrity, security, and timely decision-making. In cybersecurity, DGAD models are crucial for identifying malicious activities, such as zero-day attacks or Advanced Persistent Threats (APTs), which often manifest as anomalies in network traffic or system behavior patterns that traditional signature-based systems might miss. In the financial sector, DGAD is indispensable for fraud detection, flagging fraudulent transactions or money laundering schemes by monitoring sudden, atypical changes within transaction networks. Furthermore, social media platforms utilize these techniques to detect non-authentic coordinated activities, spam, or the spread of misinformation, maintaining platform integrity. In manufacturing and infrastructure, DGAD on sensor data enables predictive maintenance by identifying unusual patterns indicative of equipment malfunctions before critical failures occur. The widespread importance of anomaly detection underscores the criticality of developing robust and scalable deep learning methodologies capable of handling the temporal complexities of dynamic data.

The field of DGAD has seen a rapid evolution, moving beyond foundational statistical and traditional machine learning methods to embrace deep learning architectures. Recent survey works have systematically organized DGAD methodologies into a structured taxonomy. The overarching framework for DGAD approaches typically organizes methods based on their underlying mechanism for capturing the relational structure and complex interactions present in dynamic networks. These groups include: Traditional machine-learning models, Matrix transformations (such as tensor decomposition), Probabilistic approaches, and Deep-learning approaches. Deep learning (DL) has emerged as the dominant methodological category within this framework [7]. This prominence is explained by the fundamental advantages DL offers over traditional methods in modeling complex data structures. DL models, particularly those based on Graph Neural Networks (GNNs), excel at non-linear representation learning. For DGAD, these methods can inherently capture both the complex relational structure of the graph (spatial dependencies) and the intricate evolution patterns over time (temporal dynamics), often simultaneously, using components like Recurrent Neural Networks (RNNs) or attention mechanisms. This joint modeling capability surpasses the limitations of simpler linear or statistical models, which often fail when faced with high-dimensional, non-Euclidean data that changes over time. This capacity to synthesize both spatial and temporal dependencies is what makes Deep Learning the mandated technology for current and future DGAD endeavors.

One way to characterize anomalies is by the domain they affect: a) *Structural Anomalies*: These are deviations in the topological or relational properties of the graph, such as the unexpected formation of communities (like the merge/split events), or the sudden appearance of unusual bridge connections between previously unconnected groups. Detecting these often involves analyzing graph metrics like centrality, clustering coefficients, or community structure over time, b) *Temporal Anomalies:* These anomalies specifically concern deviations from expected patterns over time, such as sudden, unexpected spikes or drops in network activity. Time-series analysis models, such as ARIMA or

[2] Code available at `https://github.com/kostasada7/Anomaly-Detection-in-Dynamic-Graphs-Using-Deep-Machine-Learning-Methods`

seasonal decomposition, are commonly used here to establish a baseline expectation against which current activity is measured, and c) *Behavioral Anomalies:* These anomalies represent an unexpected change in the typical activity profile of an entity, such as a user suddenly initiating an excessive volume of transactions or a sensor reporting readings far outside its established normal range. Our work falls in the domain of structural anomalies.

3.2 Deep Learning Architectural Primitives for DGAD

The DyGED (Dynamic Graph Event Detection) model, proposed in [16], serves as a central reference point in deep learning-based DGAD. Its architecture systematically addresses the dual challenge of modeling complex spatial structure and sequential temporal dependencies, providing a high-performance blueprint for macro-dynamic event detection. In the following, we discuss the pipeline of DyGED, which we used for our case.

The foundational layer is the Graph Neural Network (GNN) backbone, responsible for processing the spatial structure and generating informative embeddings. In the DyGED pipeline, this role is fulfilled by the Graph Convolutional Network (GCN) layer. The GCN receives the node features (X) and the adjacency matrix (A) of a graph snapshot (G_t). It operates by aggregating information from neighboring nodes to produce an updated node embedding. This process effectively embeds both the attributes of a node and its local neighborhood structure into a continuous vector space. The GCN layer thus serves as the engine for extracting structural features, converting the complex, non-Euclidean graph topology into a set of usable numerical representations, which is the necessary starting point for feature aggregation in the subsequent layers.

DyGED's unique ability to detect graph-level events (macro-dynamics) stems from its pipeline design, which transforms micro-level node changes into a global classification decision. The sequential process is as follows:

1. *GCN Layer*: Extracts node features based on local structure and attributes.

2. *Pooling Layer*: This is a crucial step for event detection, as it aggregates the set of individual node embeddings (micro-dynamics) produced by the GCN into a single, unified vector representation for the entire graph snapshot (macro-dynamics). This transformation is essential because events are defined as changes at the global, snapshot level, rather than individual node or edge outliers.

3. *Temporal Block (LSTM/Attention)*: The sequence of these graph-level representations over time is then processed by the model's temporal component (LSTM and/or Attention) to model the evolution of the macro-state.

4. *MLP Classifier*: Finally, the resulting time-aware graph embedding is fed into a Multi-Layer Perceptron (MLP) for classification, yielding the probability of an event (anomaly) occurring at that time step.

This architectural choice, integrating spatial GCN features with sequential modeling through graph pooling, enables DyGED to effectively address the collective nature of events, such as community merge and split anomalies, as confirmed by extensive experimental evaluation.

The full DyGED model and its DyGED-NA variant incorporate LSTM layers to effectively capture long-term temporal dependencies and the overall evolutionary trajectory of the graph's state. This is necessary because network dynamics are often complex, with current structural changes potentially dependent on graph states far in the past. However, relying on recurrent units for modeling long sequences of large graph snapshots introduces a major computational trade-off. While LSTMs are effective for capturing long-term patterns, the computational overhead required for training them on ever-growing sequences contributes significantly to the execution time, a limitation evident in the DyGED model's higher runtime compared to its simpler variants in empirical tests. This efficiency constraint underscores a major scalability challenge for deploying such models in true high-speed data streaming applications. To enhance sequential modeling beyond simple recurrence, self-attention mechanisms are employed. Temporal self-attention is a technique that dynamically assigns importance weights to different historical time steps, allowing the model to selectively focus on the most relevant past information when making a current prediction. This overcomes the limitation of traditional RNNs that often assign uniform importance across the sequence. DyGED is notable for its pioneering use of both structural and temporal self-attention in the context of event detection on dynamic graphs. This architectural choice ensures that the model can account for application-specific node and time significance, essentially asking, "Which historical moments are most salient right now?". The DyGED-NL variant, which incorporates Temporal Attention but omits the LSTM layer, provides evidence of the mechanism's value. This ablation study indicates that the attention mechanism contributes critical contextual awareness - the ability to identify key moments in time - independent of the long-term memory capacity provided by the LSTM. Furthermore, attention mechanisms are naturally aligned with interpretability efforts, as the learned weights can be leveraged to visually isolate the specific time steps and structural components that drove the anomaly classification.

3.3 Applying DyGED to Temporal Graphs

The core problem we looked at is Event Detection within the domain of Dynamic Graphs. In particular, we were looking at macro-dynamic events that concern merge or splits of communities during the evolution of the network. To achieve this, we employed the DyGED framework along with its three variants, tailored to our needs. The models were designed to address the problem by exploiting both temporal and structural features to identify deviations from normal graph evolution. The general process involves a) the feature extraction by using a GCN layer to learn node embeddings based on graph structure, b) the macro-dynamics representation by applying a pooling layer to aggregate node-level features into a single, unified vector representation for the entire graph snapshot, c) the temporal analysis by processing the sequence of these graph representations using LSTM and/or Attention to model the graph's evolution over time, and finally the classification by using a final Multi-Layer Perceptron (MLP) for binary classification (event or non-event).

Our contributions are centered on extending the rigorous empirical evaluation, analysis, and deployment scope of the pre-existing DyGED framework and its variants (DyGED-CT, DyGED-NL, DyGED-NA). We first provide a novel empirical validation on synthetic data (RDyn). We systematically generate and utilize dynamic graph datasets using the RDyn library. This approach allowed for the creation of evolving graph structures with precisely defined and controlled structural anomalies. In addition, we focus only on structural anomaly detection, that was not done before with DyGED framework (it was crucially based on additional features of the nodes). We specifically investigated DyGED's performance in detecting explicit structural evolution anomalies, such as community merge and split events. The resulting data provided a controlled test of the model's architectural capacity against known structural challenges, demonstrating that the architecture is highly effective in these controlled environments (where AUC scores exceeded 0.96). We also provide a focused analysis and extension of the evaluation metrics beyond the standard implementation, critically examining the trade-offs of the DyGED models in real-world scenarios. We expanded the standard evaluation to include a detailed analysis of the F1 Score, Precision, and Recall for all DyGED variants across the NYC Cab and Twitter Weather datasets. This detailed metric breakdown exposed the critical divergence between the high discriminatory power of the models (high AUC, up to 0.88) and their poor practical reliability (low F1, sometimes as low as 0.21) due to the severe class imbalance in real-world data. This analysis reinforces the necessity for cost-sensitive techniques to address this fundamental challenge in anomaly detection applications.

In the following we provide some preliminary experimental results on our own synthetic datasets for all versions of DyGED. In particular, Table 5 provides the AUC and F1 scores for synthetic datasets regarding the detection of merge or splits of communities, demonstrating the effectiveness of the method. The training parameters for all experiments are as follows: $train\_ratio = 0.65$, $learning\_rate = 0.005$, $dropout\_rate = 0.02$, and $num\_hidden = 64$.

Finally, in Figure 2 we provide the validation loss for the synthetic dataset with 500 nodes.

Extensions There is a lot of work that needs to be done in order fully exploit the DyGED framework in a structural CD setting for temporal graphs. We need to conduct further experimental evaluation to optimize model performance by systematically exploring different values for key parameters. Continue the search for an application of alternative datasets to thoroughly examine the models' ability to generalize their predictive performance to new, unseen environments. This could

	DyGED_CT	DyGED_NL	DyGED_NA	DyGED
RDyn (30 nodes, 60 snapshots), num_epochs=50	AUC: 0.973, F1:0.8	AUC: 0.971, F1:0.8	AUC: 0.973, F1:0.8	AUC: 0.975, F1:0.8
RDyn (150 nodes, 100 snapshots), num_epochs=100	AUC: 0.936, F1:0.909	AUC: 0.895, F1:0.909	AUC: 0.902, F1:0.91	AUC: 0.945, F1:0.909
RDyn (500 nodes, 60 snapshots), num_epochs=150	AUC: 0.981, F1:0.863	AUC: 0.962, F1:0.754	AUC: 0.963, F1:0.861	AUC: 0.967, F1:0.868

Table 5: Experimental results regarding synthetic datasets with differents versions of DyGED based only on structural information.

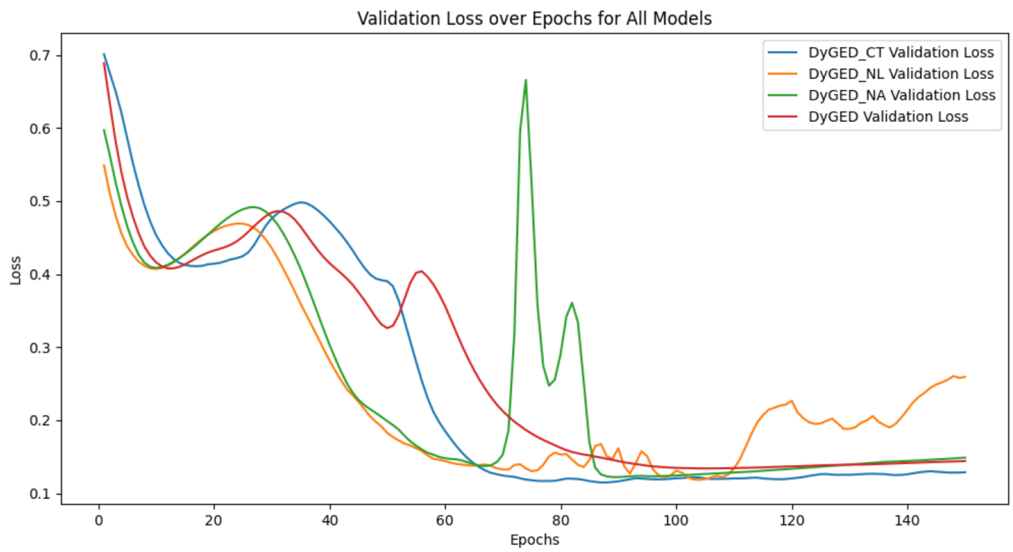


Figure 2: The validation loss for all versions of DyGED over all epochs (150 in total) for the synthetic dataset with 500 nodes.

be accomplished by designing and implementing a new synthetic generator for massive temporal networks with valid intervals, in order to be consistent with the rest of our work in TEMPO. Finally, we would like to extend the study to include distributed dynamic graphs, which would be a crucial step for applying these methods in scalable and real-time operational environments.

References

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

[2] Sitaram Asur, Srinivasan Parthasarathy, and Duygu Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(4):1–36, 2009.

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

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

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

[6] Konstantinos Christopoulos and Konstantinos Tsichlas. Local community-based anomaly detection in graph streams. In Ilias Maglogiannis, Lazaros Iliadis, John Macintyre, Markos Avlonitis, and Antonios Papaleonidas, editors, *Artificial Intelligence Applications and Innovations*, pages 348–361, Cham, 2024. Springer Nature Switzerland.

[7] Ocheme Anthony Ekle and William Eberle. Anomaly detection in dynamic graphs: A comprehensive survey. *ACM Trans. Knowl. Discov. Data*, 18(8), July 2024.

[8] F1 score lemma. F1 score lemma — Wikipedia, the free encyclopedia, 2020.

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

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

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

[12] Frank Havemann, Michael Heinz, Alexander Struck, and Jochen Gläser. Identification of overlapping communities and their hierarchy by locally calculating community-changing resolution levels. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(01):P01023, 2011.

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

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

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

[16] Mert Kosan, Arlei Silva, Sourav Medya, Brian Uzzi, and Ambuj Singh. Graph macro dynamics with self-attention for event detection. In *Deep Learning on Graphs: Methods and Applications (DLG-AAAI'23)*, 2023. Workshop paper.

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

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

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

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

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

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

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

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

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

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

[27] Anita Zakrzewska and David A Bader. Tracking local communities in streaming graphs with a dynamic algorithm. *Social Network Analysis and Mining*, 6:1–16, 2016.

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

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