mathematics

MDPI

Article

Dynamic Community Detection Method of a Social Network Based on Node Embedding Representation

Bo Zhang [1,2,3], Yifei Mi [1], Lele Zhang [1], Yuping Zhang [1], Maozhen Li [1,2,4], Qianqian Zhai [1] and Meizi Li [1,*]

[1] College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai 200234, China
[2] Institute of Artificial Intelligence on Education, Shanghai Normal University, Shanghai 200234, China
[3] Shanghai Engineering Research Center of Intelligent Education and Bigdata, Shanghai Normal University, Shanghai 200234, China
[4] Department of Electronic and Electrical Engineering, Brunel University London, Uxbridge UB8 3PH, UK
* Correspondence: limeizi@shnu.edu.cn

Abstract: The node embedding method enables network structure feature learning and representation for social network community detection. However, the traditional node embedding method only focuses on a node's individual feature representation and ignores the global topological feature representation of the network. Traditional community detection methods cannot use the static node vector from the traditional node embedding method to calculate the dynamic features of the topological structure. In this study, an incremental dynamic community detection model based on a graph neural network node embedding representation is proposed, comprising the following aspects. A node embedding model based on influence random walk improves the information enrichment of the node feature vector representation, which improves the performance of the initial static community detection, whose results are used as the original structure of dynamic community detection. By combining a cohesion coefficient and ordinary modularity, a new modularity calculation method is proposed that uses an incremental training method to obtain node vector representation to detect a dynamic community from the perspectives of coarse- and fine-grained adjustments. A performance analysis based on two dynamic network datasets shows that the proposed method performs better than benchmark algorithms based on time complexity, community detection accuracy, and other indicators.

Keywords: graph neural network; node embedding; dynamic community detection; incremental; modularity

MSC: 91D30

Citation: Zhang, B.; Mi, Y.; Zhang, L.; Zhang, Y.; Li, M.; Zhai, Q.; Li, M. Dynamic Community Detection Method of a Social Network Based on Node Embedding Representation. *Mathematics* **2022**, *10*, 4738. https://doi.org/10.3390/math10244738

Academic Editor: Pasquale De Meo

Received: 19 November 2022
Accepted: 9 December 2022
Published: 13 December 2022

1. Introduction

In social networks, there are some closely related groups of people. The interactions and connections between these people are close and frequent. Such groups of people are often clustered for analysis. In the field of social network research, the composition of these groups is generally used. Groups are called communities. It is very meaningful to discover communities in social networks, and much hidden information can be mined from them. However, with the increase in the complexity of interpersonal relationships, the complexity and scale of social network structures are also changing, and the research on the community structure of complex networks has become particularly important. Group relations in social networks present high dynamic characteristics, and much work focuses on the detection of dynamic communities and their structures. The core difficulty lies in computational cost and performance because of the complexity and dynamics of the group relation structure [1,2]. With the development of deep learning and graph neural networks [3,4], Zhang et al. proposed a framework based on network embedding methods to identify influential nodes and capture comprehensive information to distinguish nodes [5];

Munikoti Sai et al. proposed a framework based on extensible universal graph neural networks (GNN) for identifying critical nodes/links in large complex networks [6]; Wu et al. proposed a session-based graphical neural network recommendation (SR-GNN) aims to predict user behavior based on anonymous conversations [7]; Sun Jianyong et al. proposed a graph neural network coding method for a multi-objective evolutionary algorithm (MOEA) to deal with community detection in complex attribute networks [8]. Researchers have gradually focused on expressing network nodes with low-latitude, high-density spatial vectors, thereby maintaining the structure and feature information of the original network. Learned feature vectors are represented such as by graph-based classification, clustering, and link prediction [9–11]. As a research direction of graph neural networks, graph embedding has also attracted attention [12]. Research on network representation learning technology based on graph embedding and community detection through vector feature methods has attracted much interest. However, the static feature vector obtained by the classical graph embedding algorithm still cannot meet the discrete and real-time characteristics of social network community structure detection. These problems are the focus of this study.

The community detection algorithm based on network node representation has three core deficiencies. (1) Factors considered in random walk-based graph embedding methods are relatively simple. The attribute information of the nodes themselves is less considered, resulting in a low-quality corpus generated by walk sequences. (2) Most community detection algorithms use modularity, topology, or density-based division methods, which only consider the relationships between nodes and ignore potential many-to-many relationships between nodes and communities. (3) Traditional dynamic community detection algorithms tend to analyze node attributes or increments to achieve detection results. The accumulation of single detection errors will lead to a great difference between community structures and actual situations.

To solve the above problems, this study proposes a dynamic community detection method in social networks based on random walk node embedding. In node representation learning, in addition to the topological structure between nodes, attributes of the node itself can be considered, which can enrich the features of node representation. We calculate the probability that each node belongs to each community and integrate it into the node representation vector to detect overlapping communities and discover many-to-many relationships between nodes and communities. The node representation strategy proposed in this paper has been described in more detail and experimental results in the previous work. This work is a follow-up study based on the previous research results [13]. In the dynamic community detection phase, we divide the community in a coarse-grained manner according to the degree of modularity, and incrementally train the node vectors in the changed part of the network to perform fine-grained detection so as to reduce the cumulative error and improve the accuracy of community division.

The highlights of the paper are as follows.

(1) We propose a node representation algorithm that integrates the local topology of the network and influences the attribute information of the node itself; improves the feature representation ability of the node embedding feature vector, so that the resulting nodes contain richer network information, and the semantic quality is also improved; combines local node embedding and global community embedding; and realizes the mining of the many-to-many relationships between the node and community.

(2) We propose a dynamic community discovery algorithm based on the node representation algorithm. A modularity increment is defined, a preliminary judgment is made on the impact of the addition and withdrawal of nodes on the community based on the topology, and an incremental training node method is adopted for different judgment results to reduce the error of community division. Experimental results demonstrate that the proposed community detection algorithm performs well in terms of time complexity and community division accuracy, and the time complexity of the training algorithm is reduced.

The remainder of this study is organized as follows. Section 2 introduces related work. Section 3 describes the proposed algorithms, and Section 4 explains their operation. Experimental results are presented in Section 5. Section 6 summarizes our work and provides guidance for further research.

2. Literature Review

2.1. Node Representation Learning

Graph embedding is the training of neural networks to represent an information network as a set of potential node-embedding vectors. Nowadays, most of the multi-graph embedding work focuses on learning the node representation of homogeneous networks [14]. Social networks are considered to be homogeneous networks when only user-to-user relationships are considered. Therefore, we focus on node representation learning in this section. Node representation learning is applied to many common tasks, such as node classification, community detection, link prediction and so on. Node representation learning is the process of learning node representation in a digital vector format, which can capture the characteristics of network nodes [15].

Traditional machine learning classification methods usually map the attributes of samples to classification labels. Nevertheless, the traditional machine learning algorithm is not suitable for social networks because of the lack of node attribute information in the actual network. The Word2vec model proposed by Tomas Mikolov et al. [16] has become a tool in the field of natural language processing. It is characterized by vectorization of all words so that relationships between them can be measured quantitatively, inspiring its application to discrete social network graphs. In 2014, Perozzi et al. [17] proposed the DeepWalk algorithm, which applied Word2vec technology in deep learning to the node vector representation of discrete network graphs. It uses the random walk sequence of nodes constructed in the network to simulate the text generation process, and then generates the random walk sequence of nodes. It uses the Skip-gram model and a hierarchical softmax function to model each node in the random walk sequence to maximize its likelihood, and the random gradient descent function to learn the parameters. Upon convergence, the loss function learns the vector representation of the nodes.

The traversal strategy of the DeepWalk algorithm uses a complete random walk, which cannot fully integrate the topology information of network nodes. In 2016, Grover et al. [18] proposed node2vec and a biased random walk strategy to control the direction of a random walk by defining an offset function. It combines the advantages of the depth- and breadth-first search, and considers the local and global topology information of nodes. However, node2vec only covers the data information of the topology based in the network, and not the attribute information of the node itself, which improves the quality of the corpus formed by the wandering sequence and affects the accuracy of community detection.

In summary, the graph neural network method based on a random walk can obtain a corpus based on a graph structure according to the walk strategy and carry out subsequent research on community detection based on vector representation learning. The shortcoming lies in that the existing walk strategy only covers the data information of the network based on edge topology structure but does not fully consider the attribute information of the node itself. As a result, the quality of the corpus formed by the wandering sequence needs to be improved, thus affecting the accuracy of community detection. To improve the quality of the random walk corpus, this study proposes an improved node representation strategy based on the node2vec algorithm, which combines the network topology and influence attribute information of the node itself to represent and learn the node vector and improve the accuracy of community detection.

2.2. Dynamic Community Detection

There are many types of dynamic community detection algorithms. We discuss the method based on the maximization of modularity, and specifically, the method based on

incremental training nodes. This method regards community detection as an optimization problem. Therefore, we choose modularity as an index of community structure.

A dynamic community detection algorithm based on modularity measures changes in the community structure according to changes in modularity. Li et al. [19] proposed an improved label propagation algorithm named LPA-MNI in this study by combining the modularity function and node importance with the original LPA to solve the problem of community detection. Gerrero et al. [20] proposed a new Pareto-based multi-objective evolutionary algorithm; they analyzed two multi-objective variants involving not only modularity but also the conductance metric and imbalance in the number of nodes of the communities. Lu et al. [21] proposed a regression model for weighting the edges in the network, which maximizes the detection community based on modularity. Kun et al. [22] proposed a local community detection algorithm based on local modularity density. It is used for a specific and personalized community detection tasks; the algorithm consists of two separate stages: the core area detection stage and the local community extension stage. Cheng et al. [23] proposed a community detection method based on a density sequence tree. The core nodes were selected by the density peak model, and the community belonging to other nodes was assigned according to the access order tree.

The main idea of the incremental dynamic community detection algorithm is as follows. Most of the network topology of a complex social network is stable during dynamic evolution, and only a small part of the structure changes. The community detection result corresponding to a snapshot of the network is used as a reference, and only the changed part is recalculated or trained, which can greatly reduce repeated calculation and training, thereby reducing the time complexity. Zhao et al. [24] found that most incremental dynamic community detection algorithms are only suitable for adding a node or edge, so they proposed four types of incremental elements and defined corresponding update strategies. Rossetti et al. [25] proposed a dynamic community detection algorithm that can detect overlapping communities and follow their evolution in a linear iteration process. They dynamically recalculate the identities of node community members upon a new interaction, ensuring a short execution time. For incremental community detection on large dynamic networks, Shang et al. [26] used machine learning classifiers to predict community allocation, aiming to improve efficiency by filtering unchanged nodes to avoid unnecessary processing. Xu et al. [27] proposed a two-stage method. Once the error accumulation degree of incremental clustering exceeds a predefined threshold, the dynamic network snapshot is completely re-divided instead of partially updating the community structure. Wu et al. [28] proposed an incremental and scalable community detection method including both tags and potential interactions between users, which can improve detection accuracy.

In summary, to eliminate the error results caused by community detection based solely on the network topology structure, we adopt the method of comprehensive use of incremental training nodes and community representation vectors to calculate the community affiliation of nodes and adjust the community affiliation of the positioning results in the previous step. Therefore, based on the existing dynamic community detection algorithms, this study proposes an incremental dynamic community detection model based on graph neural network node embedding representation. The modularity is improved for easier realization of the rationality of community division. Incremental training reduces the time complexity and improves the accuracy of community division.

3. Preliminaries

We formally define node embedding and dynamic community detection.

Definition 1 (Social network graph model**).** *The structure model of a social network graph is* ise *as an undirected and unweighted graph* $G(V, E)$*, where* $V = \{v_1, v_2 \cdots v_n\}$ *is the set of nodes* v *in the network,* n *represents the number of nodes, and* $E = \{e_1, e_2 \cdots e_m\}$ *is the set of connected edges* e *between nodes,* m *is the number of edges. Based on this, research on node representation and community detection of networks is conducted.*

Definition 2 (Community structure**).** *Nodes with high similarity in social networks are classified as the same community. A complex network $G(V, E)$ can be divided into several communities $C = \{c_1, c_2, \ldots, c_k\}$. There is a many-to-many relationship between nodes and communities, which can be characterized as $v_i \in c_i,\ \ v_j \in c_i,\ \ i \neq j$. That is, the same node can belong to different communities, and a community can contain different nodes.*

Definition 3 (Dynamic network structure**).** *In the dynamic network $G = \left(G^1,\ G^2, \ldots,\ G^t\right)$, $G^t = (V_t,\ E_t)$ is a snapshot of the network structure at time $t(t = 1, 2, \ldots, T)$, and V_t and E_t are the sets of all nodes and edges, respectively, in the network snapshot at time t.*

According to the above definition, the proposed dynamic network community detection model can be used to model the following problems.

According to the adjacency matrix, the initial social network graph structure model is formally expressed as

$$G^{t_1} = \left[Y\left(v_1^{t_1}\right), Y\left(v_2^{t_1}\right), \ldots, Y\left(v_{n_1}^{t_1}\right)\right] \tag{1}$$

where n_1 is the number of nodes in the network at the initial time t_1, and the column vector $Y\left(v_i^{t_1}\right)$ lists all the nodes connected with node v_i. Then, the node representation strategy can be expressed as

$$G'^{t_1} = \left[Y\left(v_{fc_1}^{t_1}\right), Y(v_{fc_2}^{t_1}), \ldots, Y(v_{fc_{n_1}}^{t_1})\right]' \tag{2}$$

where $v_{fc_i}^{t_1}$ is the vectorized presentation of node v_i. The subscripts f and c indicate that the vector contains the attributes of the node and community belonging information.

The network structure evolves with time, and the output of the structure model of the final network snapshot is

$$G' = \left[G'^{t_1}, G'^{t_2}, \ldots, G'^{t_T}\right] = \begin{bmatrix} Y\left(v_{fc_1}^{t_1}\right), & Y\left(v_{fc_1}^{t_2}\right), & \ldots, & Y(v_{fc_1}^{t_T}) \\ Y(v_{fc_2}^{t_1}), & Y(v_{fc_2}^{t_2}), & \ldots, & Y(v_{fc_2}^{t_T}) \\ \vdots & \vdots & \ldots & \vdots \\ Y(v_{fc_{n_1}}^{t_1}), & Y(v_{fc_{n_2}}^{t_2}), & \ldots, & Y(v_{fc_{nm}}^{t_T}) \end{bmatrix}, \tag{3}$$

where T is the number of network snapshots, n_x is the total number of nodes in the network corresponding to snapshot t_x and $Y(v_{fc_i}^t)$ is the vectorized representation of the ith node in the network snapshot at time t. Our goal is to learn a mapping function, $f:\ G^{t_1} \longrightarrow G'$.

We propose a node representation strategy and define a total objective function that can be divided into two sub-objective functions for the vectorization representation of the attribute information and community belonging information of the training node. After the training of the model, both the topology information and community belonging information of a node can be obtained. Additionally, we build a dynamic community detection method to incrementally train nodes and community vectors in each snapshot according to the situation, so as to obtain the community belonging of each node in the snapshot of the dynamic network.

4. Proposed Model

We describe the proposed dynamic community detection model. We detail the node representation method that incorporates influence. We discuss how to locate the changed community structure based on the change of modularity in dynamic community detection, and how to use the corresponding training method for node vectors according to different positioning results, so as to make a fine-grained community division. Figure 1 shows the framework of the algorithm. The information contained in the node vector is enriched in the node representation learning stage where the influence information is integrated. In the static community detection stage, the many-to-many relationship between nodes

and communities is integrated to improve the accuracy of community detection. The right represents the dynamic community detection model, and the community belonging situation of the node obtained by the left algorithm is regarded as the initial community structure of the dynamic community detection model. The concepts of cohesion coefficient and common modularity are combined to improve the calculation method of modularity according to the stage of community structure with incremental modularity. The coarse-grained positioning divides the community structure with changes. According to the coarse-grained division, the dynamic community detection of new nodes and dynamic community detection of removing old nodes are performed, respectively. The incremental vector training method is used for the unchanged network structure. Through the method of vector training according to the situation, the community structure divided in the previous step can be fine-grained adjusted, error of the detection result is reduced, accuracy of the community structure detection is improved, and real network community structure is restored as much as possible.

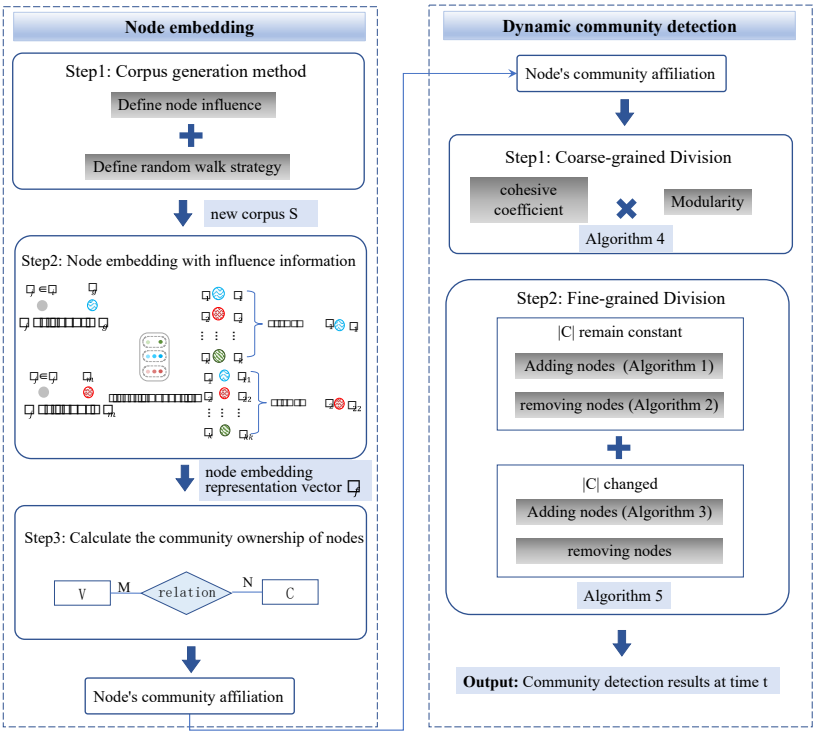


Figure 1. Model frame of proposed algorithm.

4.1. Node Embedding with Influence Information

Referring to node2vec, we define the new walking strategy construction graph corpus and use this to represent and learn all nodes in a network. We calculate the community belonging probabilities of nodes and allocate community belonging so as to obtain the community structure under the initial state of the dynamic network.

We quantify the node influence from the two indicators of degree centrality and agglomeration coefficient.

Degree centrality represents the influence of a node. The greater the degree, the higher the degree centrality of the node, and the greater its influence in the network. For an undirected graph with n nodes, the degree centrality of node v_i is

$$C_{deg} = \sum_{j=1}^{n} x_{v_i,v_j} (v_i \neq v_j). \quad (4)$$

C_{deg} represents the number of nodes directly connected to node v_i. x_{v_i,v_j} indicates the existence of the connection between v_i and v_j. If there is an edge between v_i and v_j, then

$x_{v_i,v_j} = 1$; if there is no edge between v_i and v_j, it is not a pair of neighbor nodes, then $x_{v_i,v_j} = 0$. To eliminate the influence of complex network scale on degree centrality, it is normalized as

$$C'_{deg} = \frac{C_{deg}}{n-1}, \tag{5}$$

after which the degree centrality is between 0.0 and 1.0.

The clustering coefficient is expressed as the probability of edge connection between any two neighbor nodes. According to the ternary closure principle in social networks, a common neighbor node of two unconnected nodes will increase the probability of a future edge connection between them. The aggregation coefficient expresses the node connection probability as

$$CC_{v_i} = \frac{T_i}{C_k^2} = \frac{2T_i}{k(k-1)}, \tag{6}$$

Among them, k represents the number of all neighbor nodes, T_i represents the actual number of edges between all neighbor nodes of v_i, C_k^2 represents the number of edges connected between all neighbor nodes of v_i when any two nodes in all neighbor nodes of v_i are assumed to have edges.

According to the above two indicators, the node influence value is calculated as

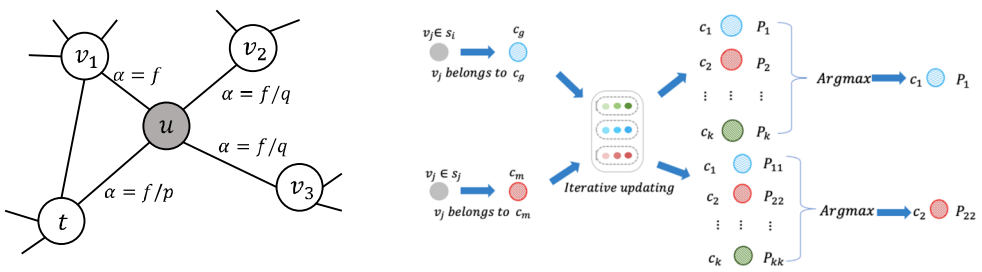
$$f_{v_i} = \beta C'_{deg} + \gamma CC_{v_i}, \tag{7}$$

where β and γ are constants, and $\beta + \gamma = 1$. Thus, the closer f_{v_i} is to 1.0, the greater the influence of node v_i.

The nodes in the network graph should be traversed first when generating the database. Based on the node2vec model, this study improves the swimming control strategy. On the basis of preserving the network topology information and integrating the influence attribute f_{v_i} of the node itself, we characterize the walking-direction-control function α as follows:

$$\alpha_{p,q}(t, v_i) = \begin{cases} \frac{f_{v_i}}{p}, & if\ d_{t,v_i} = 0 \\ f_{v_i}, & if\ d_{t,v_i} = 1 \\ \frac{f_{v_i}}{q}, & if\ d_{t,v_i} = 2 \end{cases}. \tag{8}$$

Using this formula, given a central node, the next node is determined according to the relationship between the previous step and the next step. Suppose t is the previous node, u is the current node, v_i is the next step to traverse the node, and d_{t,v_i} is the step size between t and v_i. Then, starting from t, the current node is u, and the selection of node v_i to traverse in the next step is shown in Figure 2a.



(**a**) (**b**)

Figure 2. Node embedding process. (**a**) Random walk search strategy; (**b**) Node representation learning.

(1) When $d = 0$, t is selected with probability f_{v_i}/p.
(2) When $d = 1$, the node directly connected with it is selected with probability f_{v_i}, which is the influence value of v_i.
(3) When $d = 2$, nodes not directly connected to it are selected with probability f_{v_i}/q.

Because the influence values of nodes are different, so is the probability of selecting the next node, so as to improve the representation quality of nodes.

Given a central node u, according to the above strategy, a walk sequence with length l can be obtained, and the probability of its neighbor nodes can be calculated as

$$P(v_j|v_i) = \begin{cases} \frac{\alpha(v_i,v_j)}{Z}, & if(v_i,\ v_j) \in E \\ 0, & otherwise \end{cases}, \tag{9}$$

where $v_j \in N(v_i)$, $\alpha(v_i,\ v_j)$ is the non-normalized conversion probability between i and j, Z is the normalized constant, and $P(v_j|v_i)$ can be defined by the softmax function as

$$P(v_j|v_i) = \frac{\exp\left(v_j' \cdot v_i\right)}{\sum_{v \in V} \exp(v \cdot v_i)}. \tag{10}$$

The objective function of node attribute representation is

$$L_f(s) = \frac{1}{|s|} \sum_{i=1}^{|s|} logP(v_j|v_i). \tag{11}$$

We extract the network topology and node attribute information, which should be retained by maximizing the objective function. When the objective function converges, the required node embedding representation vector v_f can be obtained.

The initial state of the dynamic network $G = (G^1,\ G^2, \ldots,\ G^t)$ is regarded as that of a static network $G = (V, E)$. The goal is to divide the nodes in V into k sub-community sets, $C = \{c_1, c_2, \ldots, c_k\}$. To fully consider the community attribute information of a node, we make two assumptions.

Assumption 1. *Each node in the network graph has a different probability of belonging to multiple communities.*

Assumption 2. *The central node v_i is known to belong to multiple communities. It is stipulated that v_i belongs to only one community c_i in a particular walk sequence s_i.*

Under these two assumptions, it is unknown whether the neighbor node j of I belongs to community C. Therefore, the key objective function P can be obtained to represent the probability that the neighbor node j belongs to C if the central node belongs to C.

Similar to Equation (10), we use the softmax function to calculate

$$P(v_j|c_i) = \frac{\exp(v_j \cdot c_i)}{\sum_{v \in V} \exp(v \cdot c_i)}. \tag{12}$$

For any node, after calculating the probability of its belonging to all communities, the community corresponding to the maximum probability is selected as the community to which the node belongs:

$$y = Argmax(P(v_j|c_i)) = Argmax\left(\frac{\exp(v_j \cdot c_i)}{\sum_{v \in V} \exp(v \cdot c_i)}\right). \tag{13}$$

Based on the above analysis, we define the community attribution objective function as

$$L_c(s) = \frac{1}{|s|} \sum_{i=1}^{|s|} \sum_{j=i-t}^{i+t} logP(v_j|c_i). \tag{14}$$

When (14) converges, we can obtain the vectorized representation c of the information of the community to which the node belongs, as shown in Figure 2b. The vectorized representa-

tions v_f and v_c of the node's influence attribute information are combined to obtain the final node vector. Each node vector is a 128-dimensional vector, expressed as follows:

$$v_{fc} = v_f \bigoplus v_c. \tag{15}$$

4.2. Dynamic Feature Positioning of Coarse-Grained Community Structure

We analyze the dynamic change characteristics of the community structure according to the increment of modularity and perform coarse-grained division and positioning for the changing nodes and communities. The following definitions are given.

Definition 4 (Cohesion coefficient**).** This is the relative independence of the community. *The larger the value, the more independent the community. The cohesion coefficient formula of community c_i at time t_i is defined as*

$$Coh_{t_i,c_i} = \begin{cases} \frac{|E(c_i)|}{|E(c_i)|+\sum_{i\neq j} I(c_i,c_j)}, & |E(c_i)| > 1 \\ 0, & |E(c_i)| = 0 \end{cases}, \tag{16}$$

where $|E(c_i)|$ is the number of internal edges in community c_i, and $I(c_i,\ c_j)$ is the number of connected edges between communities c_i and c_j.

Definition 5 (Community modularity**).** *The larger the modularity the better the effect of community detection. The modularity formula of community c_i at time t_i is defined as*

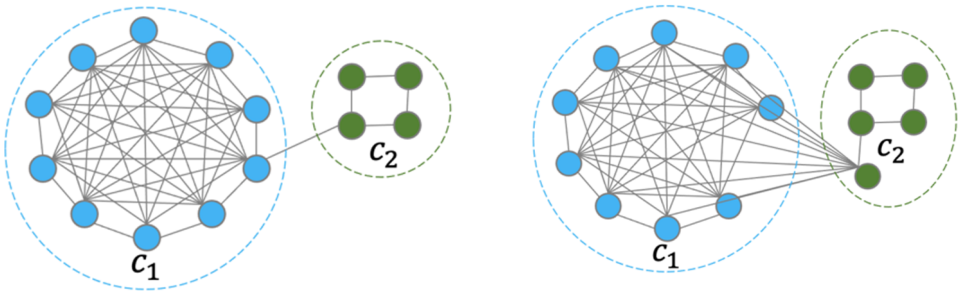
$$Q_{t_i,c_i} = \sum_{C_i} \left[\frac{\sum in}{2m} - \left(\frac{\sum out}{2m} \right)^2 \right] = \sum_{C_i} \left[e_{c_i} - a_{c_i}^2 \right], \tag{17}$$

where $\sum in$ is the sum of the weights of internal edges of community c_i, $\sum out$ is the sum of the weights of external edges connected to community c_i, and m is the sum of the weights of all edges in all communities. The weights of all edges can be regarded as 1.

If the results obtained in Section 4.1 are recorded as the community structure at initial time t_0, then the modularity value of community c_i at time t is $QC_{t_{0,c_i}}$. Considering the two characteristics of the community, the modularity is improved as

$$QC_{t_0,c_i} = QC_{t_{0,\ c_i}} \times Coh_{t_i,c_i}\ . \tag{18}$$

According to the concept of community modularity and the cohesion coefficient, the improved modularity reflects both the closeness of the internal connection of community c_i and the relative independence between it and other communities in the network. The following example illustrates the effect of modularity improvement, as shown in Figure 3a. Community C1 is a complete graph composed of 100 nodes, and community C2 has three nodes. There is an edge connection between the two communities, which is a more reasonable community division method. According to the definition of original modularity, the modularity of division method 1 is 0.00161, and that of division method 2 is slightly greater at 0.00177. However, method 1 is actually more reasonable, which indicates that the original modularity is not suitable for the situation of a large difference in community scale. According to the improved modularity QC, the QC value of method 1 is 0.49980, and that of method 2 is 0.47964, which indicates that the improved modularity can reflect the rationality of community division.



(**a**) (**b**)

Figure 3. Modularity calculation method. (**a**) Community division 1; (**b**) Community division 2.

At time $t_{i-1}(i > 1)$, the modularity of community c_i was recorded as QC_{t_{i-1},c_i}. At time t_i, for any change node, assume that v_i is a new node belonging to community c_i. The modularity $QC_{t_i,c_i}(v_i)$ of community c_i is calculated, and the modular increment $\triangle QC_{t_i,c_i}(v_i)$ caused by the change node v_i to community c_i is calculated as

$$\triangle QC_{t_i,c_i}(v_i) = QC_{t_i,c_i}(v_i) - QC_{t_{i-1},c_i}, \tag{19}$$

where $v_i \in V_{t\_add}$, and V_{t_add} is the set of newly added nodes at time t. At this stage, the condition for node v_i to belong to community c_i is

$$\triangle QC_{t_i,c_i}(t_i) > 0. \tag{20}$$

However, in this way, only coarse-grained division and positioning of changed subsystems and communities can be made, and it cannot be regarded as the final community attribution result. It is also necessary to consider the semantic information of nodes to reduce the error of community division. Therefore, incremental node training is performed to extract the community structure in a fine-grained manner.

4.3. Fine-Grained Extraction of Community Structure

According to the preliminary positioning results of the dynamic community structure change, there are two situations. The number of communities $|C|$ changes or does not change, and appropriate algorithms are adopted for fine-grained feature extraction dynamic community detection.

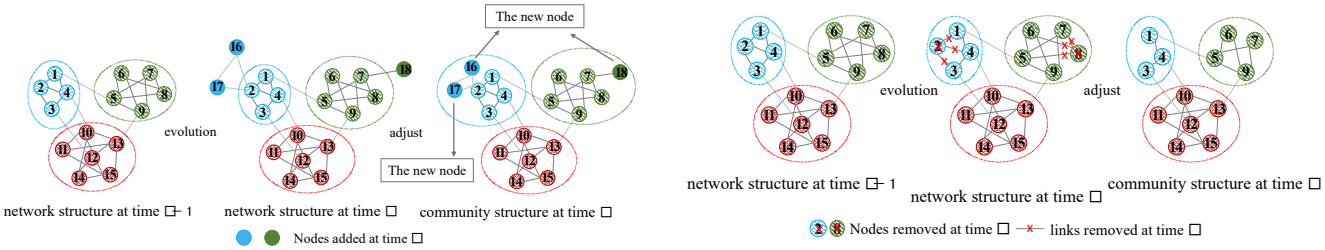
When a complex network is dynamically evolving, there are few new nodes, which can all belong to existing communities, or the deletion of nodes does not affect the structure of other parts of the network. These circumstances do not change the number of communities. We refer to this as the case of a constant number of communities.

New node, as shown in Figure 4a, compared with time $t - 1$; nodes 16, node 17 and 18 are deleted in network t. The three newly added nodes do not affect the network structure of other parts of the network. In this case, we propose training only the newly added nodes: In t moment, the newly added node $v \in V$, V for sampling, the node representation detection based on influence information is carried out in the random walk corpus S with newly added nodes. After the training, the vector representation of the original node is unchanged, and the vector representation of the new node and probability of the new node belonging to the community are obtained. This algorithm is described as Algorithm 1.

Algorithm 1 Dynamic community detection when new nodes are added and the number of communities remains unchanged

Input: Current time network diagram $G^t = (V_t, E_t)$; newly added node set V_{t_add}; last time node vector V_{t-1}; last time community vector C_{t-1};
Output: Current time node vector V_t; current community vector C_t; current community ownership $C_{v_i,t}$.

1. Initialize new node vector v_i, $v_i \in V_{t\_add}$
2. Freezing the original node vector v_j and community vector c_k, $v_j \in V_{t\_add}$, $c_k \in C_{t-1}$
3. $S = InfluenceSamplePath(G^t, v_i)$
4. FOR each iter = 1: L do
5. FOR each $v_i \epsilon s$ do (s ∈ S)
6. $L(s) = \frac{1}{|s|}\sum \log P(v_i|c_k)$
7. $v_i = v_i - \alpha * \frac{\partial L}{\partial v_i}, v_i \in V_{t\_add}$
8. END FOR
9. END FOR
10. $C_{v_i,\,t} = DetecCommunity(V_{t\_add} + V_{t-1}, C_{t-1})$



(**a**) (**b**)

Figure 4. Schematic diagram of dynamic network evolution when $|C|$ does not change. (**a**) $|C|$ increases after adding nodes; (**b**) $|C|$ decreases after adding nodes.

Among them, the third line is to find the random walk corpus S_{t_add} of the newly added nodes of the community at time t by the method proposed in Section 4.1; the time complexity is the number n of nodes in the network; then, the time complexity is $O(n \cdot l)$, where l represents the length of the random walk sequence. Lines 5–8 use the node embedding method proposed in Section 4.1 to find the node representation of the new node. The time complexity is $O(\log|V_t|)$. Therefore, the time complexity of node embedding according to the new node is $O(n \cdot l + \log|V_t|)$.

Node deletion: As shown in Figure 4b, compared with time $t-1$, nodes 2 and 8 are deleted in network t, and edges connected with these two nodes are also deleted, with no effect on other structures in the network. For this case, we propose an incremental training method of node vectors, sampling the changed network structure, using a new random walk corpus S_t to replace part of the changed corpus, and performing node representation detection based on influence information on S_t. After training, the vector representation of all nodes in the affected node set V_{t_inf} at time t is obtained. This algorithm is described as Algorithm 2.

Among them, line 4 is to find the random walk corpus S_{t_add} of the community at time t by the method proposed in Section 4.1; the time complexity is $O(n \cdot l)$. Lines 6–9 use the incremental training node vector through the node embedding method proposed in Section 4.1, and the time complexity is $O(n \log|V_t|)$. Therefore, the time complexity of the incremental training node vector is used when nodes are deleted and the number of communities unchanged is $O(n \cdot l + n \log|V_t|)$.

In the dynamic evolution of complex networks, the number of changing nodes is large, which has an impact on the topology of the network and causes its community structure change. It can be divided into four categories: new nodes increase the number of communities (Figure 5a), new nodes reduce the number of communities (Figure 5b),

deleted nodes reduce the number of communities (Figure 5c), and deleted nodes increase the number of communities (Figure 5d).

Algorithm 2 Dynamic community detection when nodes are deleted and the number of communities remains unchanged

Input: Current time network diagram $G^t = (V_t, E_t)$; deleted node set V_{t_del}; set of nodes affected by deleted nodes V_{t_inf}; last time community vector C_{t-1};
Output: Current time node vector V_t; current community vector C_t; current community ownership $C_{v_i,\,t}$.

1. Initialize new node vector $v_i,\ v_i \epsilon V_{t\_inf}$
2. Freezing the original node vector v_j and community vector $c_k, v_j \in V_{t\_inf},\ c_k \in C_{t-1}$
3. $S = InfluenceSamplePath(G^t,\ v_i)$
4. FOR each iter = 1: L do
5. FOR each $v_i \in s$ do (s ∈ S)
6. $L(s) = \frac{1}{|s|}\sum \log P(v_i|c_k)$
7. $v_i = v_i - \alpha * \frac{\partial L}{\partial v_i}, v_i \in V_{t\_inf}$
8. END FOR
9. END FOR
10. $C_{v_i,\,t} = DetecCommunity\left(V_{t\_inf}, C_{t-1}\right)$

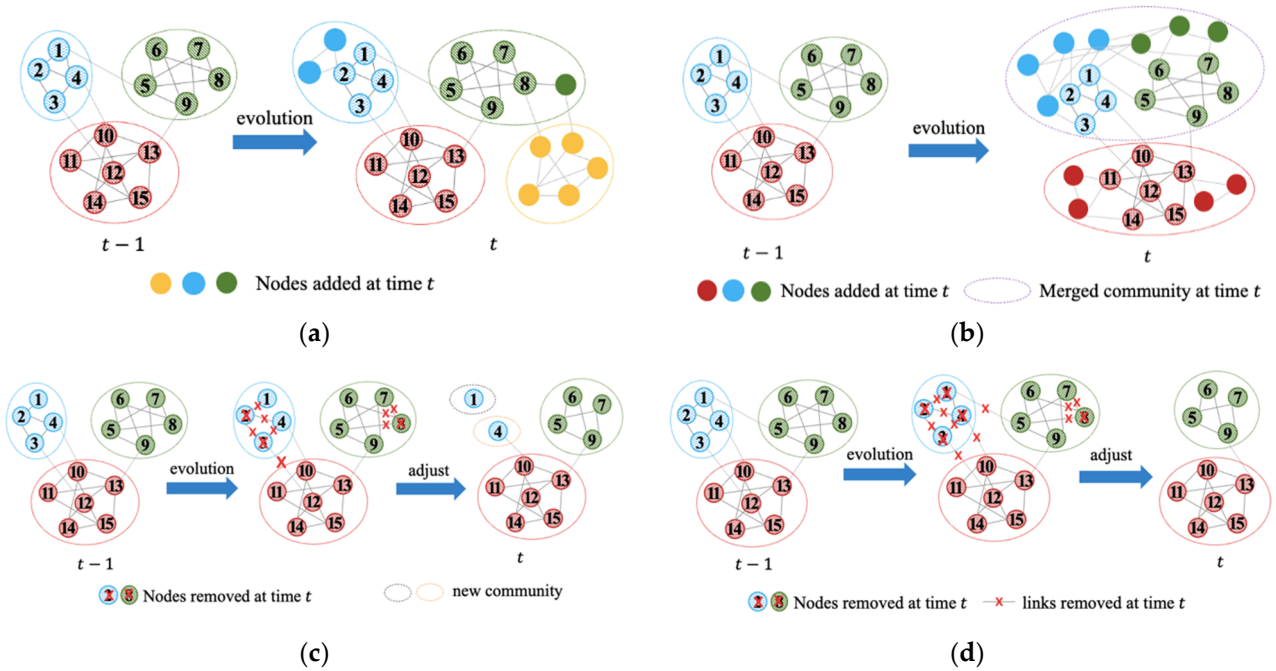


Figure 5. Schematic diagram of dynamic network evolution when |C| changes. (**a**) |C| increases after adding nodes; (**b**) |C| decreases after adding nodes; (**c**) |C| increases after deleting nodes; (**d**) |C| decreases after deleting nodes.

In view of this situation, it is necessary to recalculate the community belonging probabilities of all nodes in the network. We propose incremental training nodes and community vector representation. Not only the changing node set V_{t_add} is sampled, but the whole network at time t. The random walk corpus s is generated again, and the node representation check based on influence information is performed on the new corpus s. After training, the vector representation of all nodes in the whole network at time t is obtained, including the vector representation of changing nodes and communities, and the probability of all nodes belonging to the community. Based on this, the community belonging of all nodes at time t is adjusted. Taking the case of increasing the number of communities by adding new nodes as an example, Algorithm 3 shows the details.

Algorithm 3 Dynamic community detection when new nodes are added and $|C|$ changes

Input: Current time network diagram $G^t = (V_t,\ E_t)$; newly added node set V_{t_add}; set of nodes affected by newly added nodes V_{t_inf}; last moment node vector V_{t-1}; last moment community vector C_{t-1};
Output: Current time node vector V_t; current community vector C_t; current community ownership $C_{v_i,\ t}$.

1. Initialize new node vector $v_i,\ v_i \epsilon V_{t\_inf}$
2. Freezing the original node vector v_j and community vector $c_k, v_j\ \in V_{t\_inf},\ c_k\ \in C_{t-1}$
3. Initialize C_t with the original community vector and C_{t-1}. If a community is newly added, randomly initialize the new community vector.
4. $S = InfluenceSamplePath(G^t,\ v_i), v_i\ \epsilon V_{t\_add}, V_{t\_inf}$
5. FOR each iter = 1: L do
6. FOR each $v_i\ \in\ s$ do (s ∈ S)
7. $L(s) = \frac{1}{|s|}\sum \log P(v_i|c_k)$
8. $v_i = v_i - \alpha * \frac{\partial L}{\partial v_i}, v_i \in\ V_{t\_inf}$
9. $c_k = c_k - \alpha * \frac{\partial L}{\partial c_k}, c_k\ \in\ C_t$
10. END FOR
11. END FOR
12. $C_{v_i,\ t} = DetecCommunity\left(V_{t\_inf}, C_{t-1}\right)$

Among them, line 4 is to find the random walk corpus S_{t_add} of the community at time t by the method proposed in Section 4.1, then the time complexity is $O(n \cdot l)$. Lines 7–9 use the method proposed in Section 4.1 to indicate that the node vector and community vector are incrementally trained when the node is deleted and the number of communities changes; the time complexity is $O(n \log |\ V_t|)$. Therefore, the time complexity of the dynamic community detection algorithm for removing nodes is $O(n \cdot l + n \log |\ V_t|)$.

4.4. Algorithm Summary

We summarize the proposed two-step algorithm for dynamic community detection: (1) Locate the changing community structure according to the increment of modularity. (2) According to the change in the number of communities, corresponding training methods are adopted to divide the fine-grained dynamic communities. The procedures are shown in Algorithms 4 and 5.

In Algorithm 4, lines 1–3 are to find the modularity QC_{t-1,c_i} of the community in the last moment. If the time complexity is the number of communities in the network K, then the time complexity is $O(k)$. Lines 5–13 are to find the community change at the current time. For each node, we judge whether Q_{t,c_i} will increase when it is divided into c_i, and the time complexity is $O(nk)$. Therefore, the time complexity of the coarse-grained positioning community change algorithm is $O(nk)$.

In Algorithm 5, line 1 computes the change in two adjacent moments. Lines 3 and 4 indicate that there are new nodes in the network, but the number of communities remains unchanged. Line 5 is the resampling of newly added nodes. The time complexity is $O(n{\cdot}l)$, where l is the length of the random walk sequence. Lines 6 and 7 are the training of new nodes and communities, with time complexity $O(\log|V|)$. Line 8 indicates that there are new nodes in the network, but the number of communities changes. Line 9 represents the resampling of all nodes at the current time, with time complexity $O(n{\cdot}l)$. Lines 10 and 11 represent the training of all nodes and communities at the current time, with time complexity $O(\log|V|)$. Lines 13–17 indicate that an incremental training node vector is used when nodes are deleted and the number of communities is unchanged, with time complexity $O(n\log|V|)$. Lines 18–22 indicate that an incremental training node vector and community vector are used when nodes are deleted and the number of communities is changed, with time complexity $O(n\log|V|)$. Therefore, the time complexity of the dynamic community detection algorithm is $O(n\log|V| + nl)$.

Algorithm 4 Locate the changing community structure

Input: The last moment network graph $G^{t-1} = (V_{t-1},\ E_{t-1})$; current network graph $G^t = (V_t,\ E_t)$;
Output: Current community modularity QC_{t_0,c_i} ; current increment of modularity $\Delta QC_{t_0,c_i}$; changes in community structure Δn_c.

FOR each community $c_i \in C_{t-1}$:
    compute modularity QC_{t-1,c_i} based on Equations (17)–(19)
END FOR
FOR each mode $v_i \in V_{t\_new}$:
    FOR each community $c_i \in C_t$:
        compute modularity QC_{t-1,c_i} based on Equations (17)–(19)
        $\Delta QC_{v_i,c_i}\ (t_i) = QC_{t_i,c_i}\ (v_i) - QC_{t_{i-1},c_i}$
        IF $\Delta QC_{t_0,c_i}\ (t_i) > 0$
            $v \in c_i$
        ELSE
            $v \in \not\ c_i$
        END IF
    END FOR
END FOR

Algorithm 5 Fine-grained division of dynamic community

Input: The last moment network graph $G^{t-1} = (V_{t-1},\ E_{t-1})$; current time network diagram $G^t = (V_t,\ E_t)$; last time node vector V_{t-1}; last time community vector C_{t-1};
Output: Current time node vector V_t; current community vector C_t; current community ownership $C_{v_i,\ t}$.

$\Delta V = V - V_0$
$\Delta n = |V| - |V_0|$
IF $\Delta n > 0$:
    IF $\Delta n_c = 0$:
        $walk = randomWalk(v_i),\ v_i \in V_{t\_add}$
        $\Delta V = TrainNodeVec(\Delta V,\ V_0,\ C_0)$
        $C_1 = DetactCommunity(\Delta V + V_0,\ C_0)$
    ELSE IF $\Delta n_c \neq 0$:
        $walk = randomWalk(v_i),\ v_i \in V_t$
            $V_3, C_3 = FinetuneNodeCommunity(V, V_0, C_0)$
        $C_3 = DetectCommunity(V_3, C_3)$
    END IF
ELSE IF $\Delta n < 0$:
    IF $\Delta n_c = 0$:
        $walk = randomWalk(v_i),\ v_i \in V_{t\_del}$
        $V_2 = FinetuneNodeVec(V, V_0, C_0)$
        $C_2 = DetectCommunity(V_2, C_0)$
    ELSE IF $\Delta n_c \neq 0$:
        $walk = randomWalk(v_i),\ v_i \in V_t$
            $V_3, C_3 = FinetuneNodeCommunity(V, V_0, C_0)$
        $C_3 = DetectCommunity(V_3, C_3)$
    END IF
END IF

5. Experimental Evaluation

The platform environment of the experiment in this paper is briefly introduced as follows: Hardware Configuration: CPU Processor: Intel(R) Core(TM) i5-8300H CPU @ 2.30 GHz 2.30 GHz, internal storage: 8.00 GB, hard disk: 512 GB. Software Configuration: Operating System: Windows 10, 64-bit operating system. Development Tools: Jet Brains Py Charm Community Edition 2019.2, Visual Studio Code Python; Version: Python 3.7.3 Anaconda; Version: Anaconda Command line client (version 1.7.2).

5.1. Datasets

(1) Cora: The Cora dataset consists of machine learning papers. The network contains 2708 nodes and 5429 edges, forming 7 categories.
(2) Wiki: A wiki is a collaborative encyclopedia written by its users and is widely used in social network analysis. The network includes 2405 nodes and 15,985 edges, forming 19 categories. Obviously, the social network relationship in this dataset is relatively dense.
(3) BlogCatalog: The dataset of BlogCatalog is extracted from a directory of blogs and bloggers managed by BlogCatalog, which forms a social network between bloggers. The network includes 10,312 nodes and 333,983 sides. Topics published by bloggers are tag values, forming 39 categories.

5.2. Baseline and Evaluation Metrics

This study selects three dynamic community detection algorithms, GraphScope, FaceNet, and QCA, as contrast algorithms [29–31].

(1) The GraphScope [29] algorithm merges network snapshots of different time segments into new network snapshots, executes community detection algorithms on new network snapshots, and judges whether a new time segment has significant changes in network structure, i.e., it is a change point. If so, we start a new time segment and independently execute the community detection algorithm; if not, we merge the current time snapshot into the current time segment, and the merged network snapshot contains the change information in the network evolution process of the current time segment.
(2) FaceNet [30] adopts the classic two-step strategy, i.e., community detection is carried out in a single time segment, and the current community detection result is adjusted according to the community detection results of the previous time. The algorithm uses the extremum optimization method to adjust the current community detection results, which transforms the community detection problem into one of minimizing the loss function, and the algorithm can detect overlapping communities.
(3) QCA [31] is a classical adaptive incremental detection method of network dynamic changes. It determines the changes in community structure in the current network according to the changes between the network structure at the previous and current times.

In the experiment, NMI and QC modularity values are used as evaluation indexes.

To verify the effectiveness of the incremental training node method based on modularity to detect communities, the normalized mutual information NMI is used as the evaluation standard for the accuracy of community division:

$$NMI(x,y) = 1 - \frac{1}{2}[H(x|y)_{norm} + H(y|x)_{norm}], \tag{21}$$

where x and y, respectively, represent the real community detection result and that obtained according to the algorithm, the real community detection results are provided by the dataset, and the value range of NMI is [0, 1]. The closer NMI is to 1, the higher the accuracy of the community detection model.

In addition to comparative experiments, we conducted experiments on the influence of different modularity thresholds on the NMI value to verify the accuracy of the proposed modularity-based incremental dynamic community detection model.

5.3. Parameter Setting

For the node representation method, we test the effect of different parameter choices on different algorithms, The parameters in the network presentation learning algorithm are described in Table 1. In order to evaluate the impact of parameters on the result of embedding, we will adopt the Macro-F1 fraction as the measurement function of the parameters in this section.

Table 1. Description of parameters in the network presentation learning algorithm.

Parameter	Paraphrase
m	The number of random walks per node
l	The length of a random walk sequence
d	Dimension of feature
k	Community size
p	Control parameters
q	Control parameters
β	Weight parameter
γ	Weight parameter

Assume that for more than a classification problem, a total of three categories, respectively for 1, 2, 3,

TP(True positive): Points of the same category are divided into the same community;
TN(True negative): Different types of nodes are divided into different communities;
FP(False positive): Nodes of different categories are divided into the same community;
FN(False negative): Nodes of the same category are divided into different communities;
TP_i: TP of category i
TN_i: TN of category i
FP_i: FP of category i
FN_i: FN of category i

Macro-precision is the average value of all classification accuracy:

$$precision_{macro} = \frac{1}{3}(precision_1 + precision_2 + precision_3) \tag{22}$$

Macro-recall is the average of all categories of recall:

$$recall_{macro} = \frac{1}{3}(recall_1 + recall_2 + recall_3) \tag{23}$$

Finally, the calculation formula of Macro-F1:

$$F1_{macro} = 2 \times \frac{recall_{macro} \times precision_{macro}}{recall_{macro} + precision_{macro}} \tag{24}$$

It can be found that the larger the value of Macro-F1, the better the performance of the algorithm.

In the experiment of how the parameters affect the performance of the algorithm, the following conclusions are drawn through the analysis of the experimental results.

When the characteristic dimension of the representation vector reaches around 100, the performance of the model tends to saturate, which means that increasing the value of dimension d will not affect the algorithm at all.

The number of random walks of each node m and the length of random walk sequence l also affect the performance of the algorithm. When the number of random walks of each node $m < 16$, with the increase of m, the performance of the algorithm can be improved. When $m \geq 16$, the algorithm tends toward saturation.

The number of random walks of each node m and the length of random walk sequence l also affect the performance of the algorithm. When the number of random walks of each node $m < 16$, with the increase of m, the performance of the algorithm can be improved. When $m \geq 16$, the algorithm tends toward saturation.

The performance of the proposed algorithm will improve as the control parameters p and q decrease, Where parameters p and q are balance parameters that determine whether depth-first DFS or breadth-first DFS is adopted, the parameter p is the hyperparameter of the control returned to the original node, and the parameter q is the hyperparameter of the

control node traversed outward. As shown in Figure 6, the algorithm performs well when the parameters are set as $p = 0.25$ and $q = 0.25$.

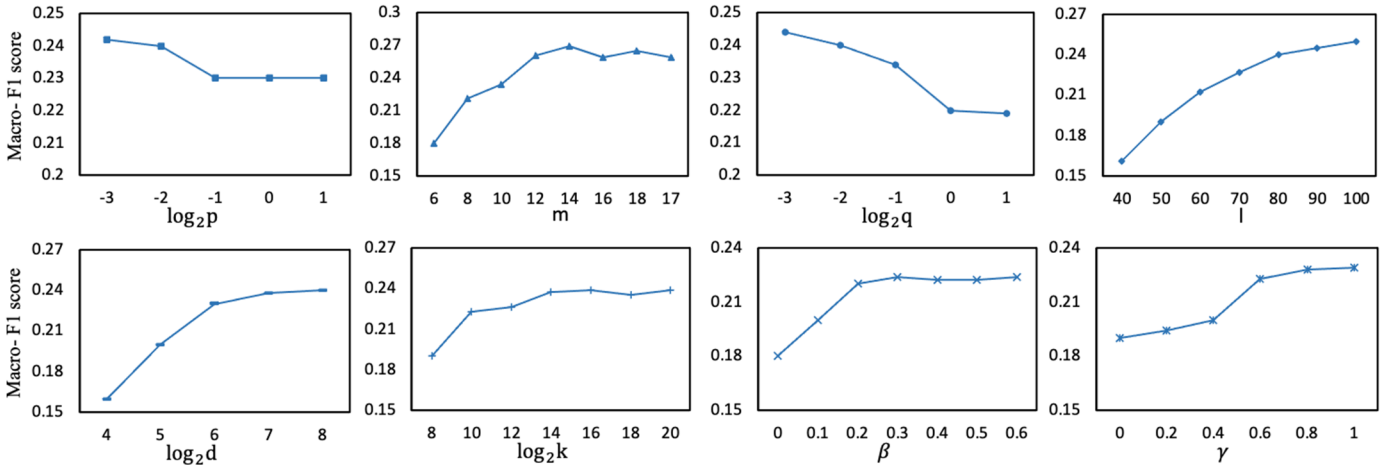


Figure 6. Parameter influence analysis.

As the parameters β and γ increase, the performance of the proposed algorithm will improve; the parameter β is used to control the weight of node degree of influence factor centricity super parameters, and the parameter γ influence factor is used to control the node of the cluster coefficient of the weight parameters. Moreover, $\beta + \gamma = 1$, and the node influence considered in this paper are only affected by two factors: degree centrality and agglomeration coefficient. According to Figure 6, we found that when the parameter setting of β = 0.2, γ = 0.8, the performance of the algorithm performance is better, and tends to be stable.

This experiment sets the node embedding dimension of all algorithms to 128. To ensure a fair comparison, the parameters of the benchmark algorithm were adjusted to the best. For the DeepWalk and Node2vec algorithms, we set the window size w to 10 and random walk length to 80. For the LINE algorithm, we set the negative ratio to 5. For GraphScope, FaceNet, and QCA, all parameters were set to the best 27–29 to ensure the best performance.

5.4. Experimental Results and Analysis

In the experiment of dynamic community detection, the two datasets of Cora and Wiki were preprocessed. Each dataset was set to 10 moments. The first five moments of the Cora dataset contained two communities, and the sixth to tenth added one community each time. The first five moments of the Wiki dataset contained 15 communities, and the sixth to tenth added one community each time. To simulate the dynamic evolution characteristics of real networks, in the first five moments, 100 nodes were randomly selected from the previous snapshot to join other communities. In this experiment, the model performance was verified on the artificial dynamic network dataset composed of such network snapshots.

As shown in Figure 7a, the overall running times of the algorithms were shorter because the Cora dataset contains fewer community clusters. After the fifth moment, the running times of the four algorithms increased rapidly; from 1 to 5 moments; the four moments of the algorithm were relatively stable, because the number of communities did not change during this period. Additionally, the running time of FaceNet was higher than that of other algorithms, because its two-step calculation method has strong interdependence between adjacent moments. The overall running time of the algorithm in this paper was lower than that of the other algorithms, which shows the effectiveness of its incremental training method.

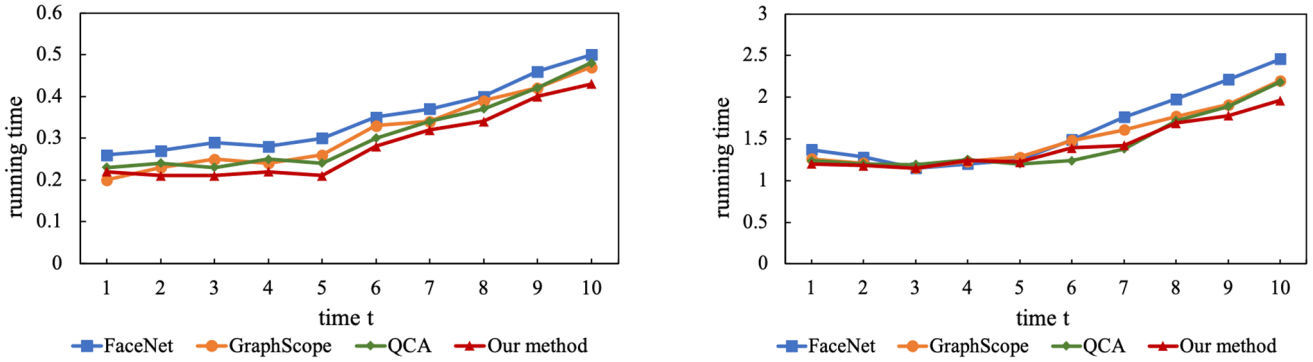


Figure 7. Running time of the four algorithms on the two datasets after manual processing. (**a**) Cora; (**b**) Wiki.

As shown in Figure 7b, from 0 to 5, the network size and number of communities did not change; the running times of the four algorithms showed little difference, and they were relatively stable. On the whole, the running time of FaceNet was longer than that of the other algorithms. This is because FaceNet uses a two-step calculation method, i.e., community detection at a single moment is required, and it constantly adjusts. Starting from the fifth moment, a new community was added at each moment. The algorithm in this study must perform coarse-grained positioning changes based on the modularity increment in the first step of the community structure, resulting in an increase in running time, but in the second step, the incremental training vector method greatly reduces the time complexity, which can explain why the algorithm is superior to the others in terms of running time.

(1) QC module degree value and NMI value

As shown in Figure 8a,b, on the Cora dataset, the overall QC modularity value of the proposed algorithm was better than that of the other algorithms. At the ninth moment, the QC modularity value of GraphScope increased significantly, while the QC modularity value of FaceNet and QCA showed a downward trend from the fifth to tenth moment, which indicates that the performance of the two algorithms was not ideal. The NMI values of the four algorithms showed a downward trend over time because of the increase in the modularity value caused by the dynamic network division community, which led to the decrease of NMI. Moreover, the NMI value of the proposed algorithm was better than the benchmark, which shows the effectiveness of using the improved modularity.

The results of the proposed algorithm, as seen in Figure 8c,d, are better than those of the other algorithms. The NMI values of the proposed model and FaceNet algorithm decreased over time. This is because some complex evolution events appear in the dynamic complex network at a certain time, which makes the community structure in the network not obvious. From the experimental results, we can see that the QC value of the GraphScope algorithm was relatively stable. This is because it is based on the improvement of a static community detection algorithm, and it is executed in each time segment, so there is no cumulative error.

(2) Parameter sensitivity

In the first stage, i.e., when the community structure is located according to the modular increment, the new node v_i belongs to community c_i. The condition of this change is that $\triangle QC_{t_i,c_i}(t_i) > 0$. In the experiment, to improve the accuracy of the partition, a modular increment threshold ε was set. When $\triangle QC_{v_i,c_i}(t_i) > \varepsilon$, V belonged to community C.

As shown in Table 2, NMI decreased with the increase in the threshold ε. When $\varepsilon = 0.1$, the performance of the algorithm was optimal. According to the same threshold value, the NMI values at different times showed a downward trend, and in some moments, there

was a sudden drop. This is because the algorithm had to be executed under the premise of a fixed number of communities from 1 to 5; a new community was added each time from 6 to 10, resulting in the frequent execution of the incremental dynamic community detection algorithm, and NMI showed a downward trend.

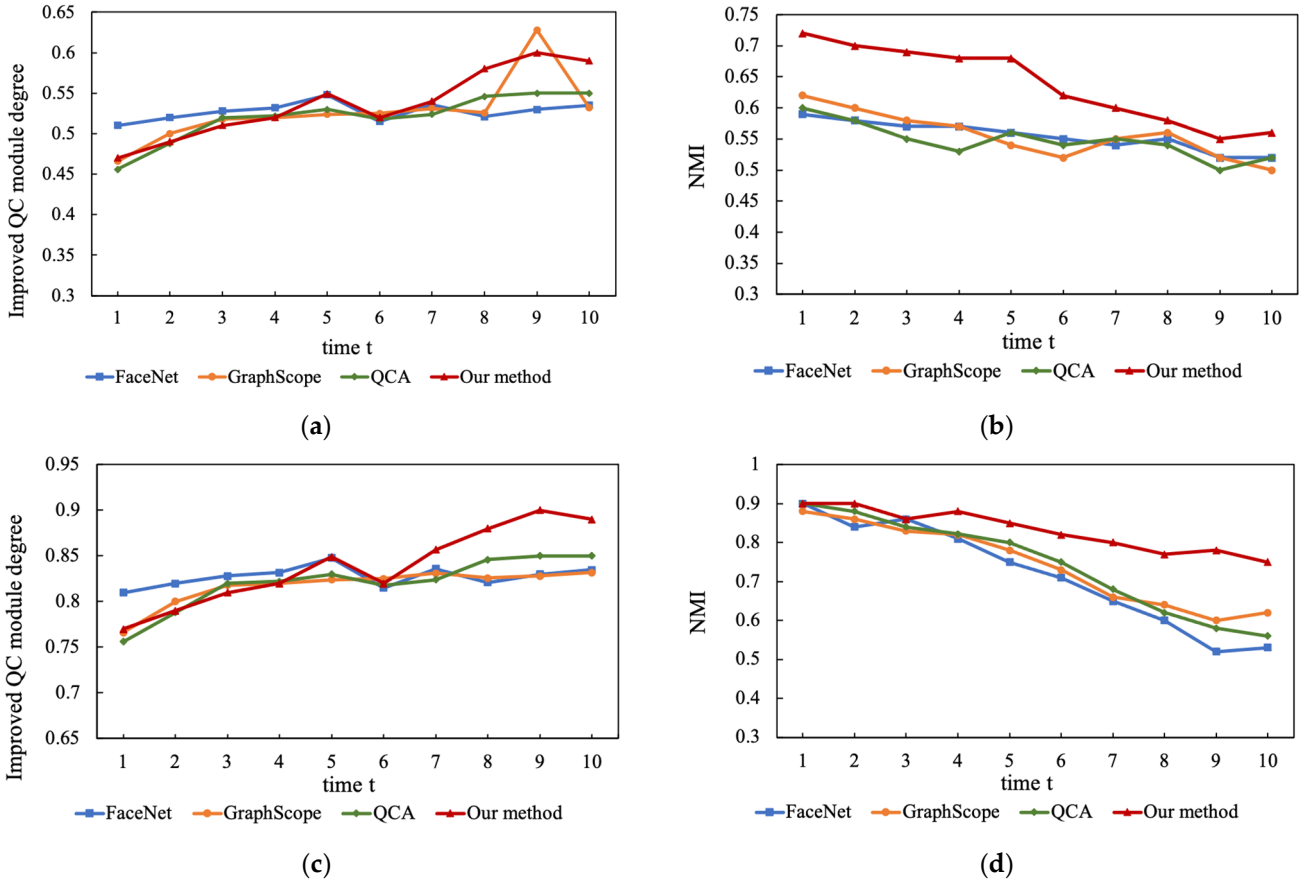


Figure 8. Two evaluation indexes of four algorithms on two datasets after manual processing. (**a**) QC modularity on Cora dataset; (**b**) NMI values on Cora dataset; (**c**) QC modularity on Wiki dataset; (**d**) NMI values on Wiki dataset.

Table 2. Influence of Different ε Influence on NMI value of our algorithm.

t	ε = 0.1	ε = 0.3	ε = 0.5
1	0.975	0.950	0.824
2	0.974	0.910	0.732
3	0.980	0.936	0.698
4	0.967	0.911	0.735
5	0.919	0.873	0.703
6	0.986	0.857	0.688
7	0.329	0.296	0.238
8	0.289	0.246	0.187
9	0.204	0.176	0.129
10	0.159	0.166	0.085
AVG	0.6782	0.6321	0.5019

As shown in Figure 9, the NMI values at time points 1–5 under different values of ε showed little difference, and the performance was stable, because the number of communities in the network remained unchanged during this period. After the fifth time, NMI showed a downward trend due to the addition of new communities. However,

owing to the coarse-grained positioning division and fine-grained incremental training adjustment, the final community detection results were still relatively true, and avoided the error accumulation of community detection results at each time slot as much as possible, so the fluctuation range of the NMI value with the change in ε is small, which proves the superiority of our model.

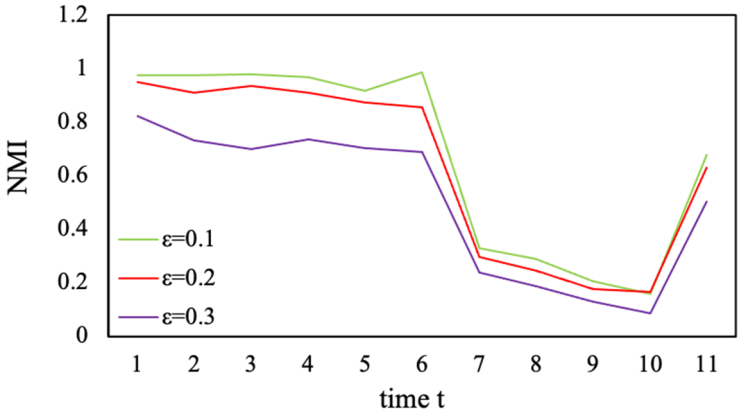


Figure 9. Influence of different ε values on NMI value of our algorithm.

6. Discussion

We proposed an incremental dynamic community detection algorithm based on node embedding and modularity. In terms of node embedding, the node's own influence attribute is added to the consideration of the random walk direction, and a new objective function is defined, so that the node vector obtained from the model training retains the network topology and node attribute information at the same time, which improves the quality of the node vector representation. By combining the cohesion coefficient and ordinary modularity, the modularity is redefined, and based on the preliminary positioning results of the community structure change. In this case, the corresponding node training algorithm was proposed to calculate the probability that the node belongs to the community, and realize the detection of the dynamic community at the current moment. Experiments showed that compared with the benchmark algorithms, the time complexity and accuracy of this algorithm were improved. The model proposed in this paper can reduce the errors of each detection and improve the accuracy of the final community detection results. However, when analyzing the dynamic evolution state of complex networks, there may be insufficient consideration of the problem, which requires further improvement.

7. Conclusions

As the complexity of human relationships increases, the complexity and scale of social network structures are also changing, and the research on the community structure of complex networks has become particularly important; and because of the development of deep learning technology, more and more attention has been paid to the community detection method based on graph neural networks. However, it is also imminent that the quality of the corpus is insufficient and the semantic information on the node vector learned is not rich, thus leading to inaccurate results of community detection. In the process of dynamic community detection, the elimination of each result error, improvement of the accuracy of the final detection results, and restoration of the community structure in the real network are other challenges faced by the existing research work.

Recently, some important node applications in various fields have been developed such as spatialtemporal graph neural networks based on node attention, information transmission simulation of Internet of things communication nodes under a collision-free probability equation, and spatialtemporal graph neural networks based on node attention. In order to solve the above two challenges, this study firstly summarizes and analyzes the existing community detection algorithms based on graph neural networks

and points out the shortcomings of the existing algorithms. This study improves the existing research work and proposes a dynamic community detection model based on node embedding representation. The improved modularity is used to extract and partition the dynamic community structure. The improved modularity is used to extract and partition the dynamic community structure. The model uses dynamic community detection of new nodes and dynamic community detection of removing old nodes to fine-grained adjust the partitioning results of the previous step. Finally, the dynamic community detection situation at the current moment is obtained.

The following research will be conducted in the future. (1) More than one type of attribute information of a node in the network will be studied. This study uses influence information, and more data information can be comprehensively considered to characterize node attributes. (2) In reality, there are many dynamic changes in the complex network structure. For example, when nodes are not added or deleted, the community structure changes, which is not considered in this article.

Author Contributions: Conceptualization, B.Z., Y.M. and L.Z.; methodology, B.Z. and Y.M.; software, B.Z.; validation, B.Z. and Y.M.; formal analysis, Y.M. and L.Z.; investigation, B.Z. and Y.Z.; resources, Y.Z. and M.L. (Meizi Li); data curation, B.Z.; writing—original draft preparation, B.Z. and L.Z.; writing—review and editing, Y.M. and Q.Z.; visualization, B.Z., Y.M. and Q.Z.; supervision, M.L. (Meizi Li); project administration, B.Z. and M.L. (Maozhen Li); funding acquisition, B.Z. and M.L. (Mizi Li). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (61802258, 61572326), Natural Science Foundation of Shanghai (18ZR1428300).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bahadori, S.; Zare, H.; Moradi, P. PODCD: Probabilistic Overlapping Dynamic Community Detection. *Expert Syst. Appl.* **2021**, *174*, 114650. [CrossRef]
2. Yu, L.; Li, P.; Zhang, J.; Kurths, J. Dynamic Community Discovery via Common Subspace Projection. *New J. Phys.* **2021**, *23*, 033029. [CrossRef]
3. Manning, C.D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.J.; McClosky, D. The Stanford CoreNLP natural language processing toolkit. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, MD, USA, 22–27 June 2014.
4. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* **2018**, *13*, 55–75. [CrossRef]
5. Zhang, W.; Jing, Y. DeepINN: Identifying Influential Nodes Based on Deep Learning Method. In Proceedings of the 11th International Conference on Computer Engineering and Networks, Hechi, China, 21–25 October 2021; Springer: Singapore, 2022.
6. Munikoti, S.; Laya, D.; Balasubramaniam, N. Scalable graph neural network-based framework for identifying critical nodes and links in complex networks. *Neurocomputing* **2022**, *468*, 211–221. [CrossRef]
7. Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; Tan, T. Session-based recommendation with graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33.
8. Sun, J.; Zhang, W.; Zhang, Q.; Xu, Z. Graph neural network encoding for community detection in attribute networks. *IEEE Trans. Cybern.* **2021**, *52*, 7791–7804. [CrossRef]
9. Kakisim, A.G. Enhancing attributed network embedding via enriched attribute representations. *Appl. Intell.* **2022**, *52*, 1566–1580. [CrossRef]
10. Lai, D.; Wang, S.; Chong, Z.; Wu, W.; Nardini, C. Task-oriented attributed network embedding by multi-view features. *Knowl. Based Syst.* **2021**, *232*, 107448. [CrossRef]
11. Yuan, W.; Shi, C.; Guan, D. Dynamic network embedding via multiple sequence learning. *Neural Comput. Appl.* **2022**, *34*, 3843–3855. [CrossRef]
12. Shen, X.; Dai, Q.; Mao, S.; Chung, F.-L.; Choi, K.-S. Network together: Node classification via cross-network deep network embedding. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 1935–1948. [CrossRef]
13. Li, M.; Lu, S.; Zhang, L.; Zhang, Y.; Zhang, B. A community detection method for social network based on community embedding. *IEEE Trans. Comput. Soc. Syst.* **2021**, *8*, 308–318. [CrossRef]
14. Zhou, J.; Liu, L.; Wei, W. Network representation learning: From preprocessing, feature extraction to node embedding. *ACM Comput. Surv.* **2022**, *55*, 1–35. [CrossRef]
15. Škrlj, B.; Kralj, J.; Konc, J.; Robnik-Šikonja, M.; Lavrač, N. Deep node ranking for neuro-symbolic structural node embedding and classification. *Int. J. Intell. Syst.* **2022**, *37*, 914–943. [CrossRef]

16. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 3111–3119.
17. Perozzi, B.; Rami, A.-R.; Steven, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014.
18. Grover, A.; Jure, L. Node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016.
19. Li, H.; Zhang, R.; Zhao, Z.; Liu, X. LPA-MNI: An improved label propagation algorithm based on modularity and node importance for community detection. *Entropy* **2021**, *23*, 497. [CrossRef] [PubMed]
20. Guerrero, M.; Gil, C.; Montoya, F.G.; Alcayde, A.; Baños, R. Multi-objective evolutionary algorithms to find community structures in large networks. *Mathematics* **2020**, *8*, 2048. [CrossRef]
21. Lu, X.; Kuzmin, K.; Chen, M.; Szymanski, B.K. Adaptive modularity maximization via edge weighting scheme. *Inf. Sci.* **2018**, *424*, 55–68. [CrossRef]
22. Guo, K.; Huang, X.; Wu, L.; Chen, Y. Local community detection algorithm based on local modularity density. *Appl. Intell.* **2022**, *52*, 1238–1253. [CrossRef]
23. Cheng, Q.; Liu, Z.; Huang, J.; Cheng, G. Community detection in hypernetwork via density-ordered tree partition. *Appl. Math. Comput.* **2016**, *276*, 384–393. [CrossRef]
24. Zhao, Z.; Li, C.; Zhang, X.; Chiclana, F.; Viedma, E.H. An incremental method to detect communities in dynamic evolving social networks. *Knowl. Based Syst.* **2019**, *163*, 404–415. [CrossRef]
25. Rossetti, G.; Pappalardo, L.; Pedreschi, D.; Giannotti, F. Tiles: An online algorithm for community discovery in dynamic social networks. *Mach. Learn.* **2017**, *106*, 1213–1241. [CrossRef]
26. Shang, J.; Liu, L.; Li, X.; Xie, F.; Wu, C. Targeted revision: A learning-based approach for incremental community detection in dynamic networks. *Phys. A Stat. Mech. Its Appl.* **2016**, *443*, 70–85. [CrossRef]
27. Xu, Z.; Rui, X.; He, J.; Wang, Z.; Hadzibeganovic, T. Superspreaders and superblockers based community evolution tracking in dynamic social networks. *Knowl. Based Syst.* **2020**, *192*, 105377. [CrossRef]
28. Wu, Z.; Ming, Z. An incremental community detection method for social tagging systems using locality-sensitive hashing. *Neural Netw.* **2014**, *58*, 14–28. [CrossRef] [PubMed]
29. Sun, J.; Faloutsos, C.; Papadimitriou, S.; Yu, P.S. Graphscope: Parameter-free mining of large time-evolving graphs. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, CA, USA, 12–15 August 2007.
30. Lin, Y.-R.; Chi, Y.; Zhu, S.; Sundaram, H.; Tseng, B.L. Facetnet: A framework for analyzing communities and their evolutions in dynamic networks. In Proceedings of the 17th International Conference on World Wide Web, Beijing, China, 21–25 April 2008.
31. Nguyen, N.P.; Dinh, T.N.; Xuan, Y.; Thai, M.T. Adaptive algorithms for detecting community structure in dynamic social networks. In Proceedings of the IEEE INFOCOM. IEEE, Shanghai, China, 10–15 April 2011.