Applied Intelligence (2023) 53:18294–18318
https://doi.org/10.1007/s10489-022-04403-5

Local community detection based on influence maximization in dynamic networks

Mohammad Ebrahim Samie[1] · Eileen Behbood[2] · Ali Hamzeh[2]

Accepted: 6 December 2022 / Published online: 27 January 2023

Abstract
Social network analysis (SNA) has opened up different research areas to researchers, such as Community Detection and Influence Maximization. By modeling social networks as graphs, one can detect one's communities or find the most Influential nodes for different applications. Despite extensive research in this area, existing methods have not yet fully met analysts' needs and are still being improved. Researchers have recently begun to apply certain concepts of a research area in social network analysis to improve social network analysis methods in other areas. In this article, we claimed that applying Two-phase Influence Maximization can improve some community detection methods. To prove the claim, we made some changes in one of the current and efficient local community detection methods to improve the way of finding the initial nodes with the new approach to finding the most influential nodes. The results showed a significant improvement. Another problem was applying this method to dynamic networks, which could be time consuming. To solve this problem, proposed a new technique that allows us to find the initial nodes in each snapshot in a new way without carrying time consuming calculations. The experimental results showed that the novel approach and the new method outperformed the previous ones in both static and dynamic social networks.

Keywords Community detection · Dynamic social network · Influence maximization

1 Introduction

The emergence of digital tools and their contribution to the development of social networks have made social networks more important than ever [1]. In other words, Social networks are a set of individuals and organizations interacting with each other [2] Many research areas have been formed in the context of social networks in order to extract useful information from this large source of data [3]. The development of graph theory provides a strong basis for modeling social networks. In the graph model, each node represents an entity, and each edge represents a relationship between them [4]. Social network analysis (SNA) widely uses graph theory and machine learning methods [5]. Community Detection and Influence Maximization are two of these popular research areas. In the past, researchers have presented several methods to solve the problems of each field. In this paper, we try to improve a community detection method with a new hybrid approach using influence maximization concepts. In the following, we will briefly introduce each area.

Community detection One of the most popular research areas in the SNA is to look for existing communities in a social network [6]. Detecting communities in a network is very helpful for its better understanding. A community is said to be a group of nodes in a network that is densely connected. In Newman's definition of community, each community is a set of nodes in which the number of edges between the members is more than the number of edges connecting that set to the rest of the graph [7]. In other words, a community is a group of nodes that have

✉ Mohammad Ebrahim Samie
samie@jahromu.ac.ir

Eileen Behbood
eileen.bhb@gmail.com

Ali Hamzeh
ali@cse.shirazu.ac.ir

1 Department of Computer Engineering and IT, Jahrom University, Khalij Fars Blvd, Jahrom, Fars, Iran

2 Department of Computer Science, Engineering and Information Technology, Shiraz University, Mollasadra Street, Shiraz, Fars, Iran

similarities or have more connections compared to other nodes, and this similarity can include their interest in a specific topic or more interactions among them.

Influence maximization This is another popular area of research in social network analysis. In social networks, some nodes are more impactful than others. The purpose of influence maximization problems is to detect these nodes in a network. As an example, when the distribution of information through a set of these nodes is maximized, more nodes will receive information if these nodes are the initial nodes in the information propagation process [8].

We mainly focus on community detection problems in the following sections. Social networks are divided into two categories in terms of time: Static networks and Dynamic ones. Researchers e.g., Newman, Girvan [9], and Pizzuti [10], initially worked on static networks and provided algorithms to solve their problems. Since 2008, they have focused on dynamic networks in the hope of being able to figure out the structure of communities and their changes over time [1]. Dynamic social networks are defined by the snapshots that are intermittently taken from the network structure over time. The snapshots are also called timestamps. There are two approaches to identify communities in dynamic social networks:

- Independent community mining [11].
- Incremental community mining [12].

From another point of view, researchers use two other approaches to solve these problems; global and local. In the former, they use all the available data at once to solve the problem. On the other hand, they solve the problem locally with the information around each node in the latter. In other words, a set of primary nodes are selected based on which communities can be recognized with local information in local approach. Each of these approaches has advantages and disadvantages, the details of which are given in [1].
In this article, we use the advantages of both local and global approaches to solve community detection problems by presenting a new hybrid approach. In other words, by using the basic concepts of influence maximization, we try to further develop a local-based approach we've previously researched (HLCD) [13] to enable it to identify communities in dynamic social networks. Our contributions in the paper will be threefold:

- To employ an influence maximization concept to determine better initial nodes.
- To develop a community detection algorithm using the initial optimal nodes to improve accuracy.
- To modify the proposed algorithm to work on dynamic social networks while increasing the accuracy and reducing the running time.

We will continue to review the related works of community detection algorithms in Section 2. In Section 3, we will present a sufficient explanation of our method. The results of the proposed method and the comparisons will be presented in Section 4, and the last section presents the conclusion.

2 Related works

Many methods have been proposed to solve the community detection problem. Social networks are divided into two categories in terms of time: static networks and dynamic ones. Initially, researchers worked on static networks and provided algorithms in [9] and [10] for them. Later, they began to investigate dynamic networks in the hope of being able to figure out patterns of graph changes in the community structures over time [1]. They tried to extend and modify the existing algorithms to apply them to dynamic networks. Research on graph communities began in 1970s. Papadopoulos [14] attempted to identify communities in social networks using the partial clustering method. He considered a certain number of clusters and then classified the nodes in the network based on their distance from the center of the clusters. The advantage of this partial method was the simplicity of its implementation. However, it was necessary to determine the number of clusters beforehand, which was not always known. Therefore, the hierarchical clustering algorithm was presented [15]. Hierarchical clustering algorithms consist of two groups, namely agglomerative algorithms and divisive algorithms [16]. These algorithms cut the tree diagram to separate communities. In these methods, it is not necessary to know the number of communities. However, the tree cutting spot affects the accuracy of the detected communities. The current trend in research in community detection methods falls into three categories [3]; methods in which a person could only be in one community such as clustering or Newman algorithm [7], methods of overlapping community detection in which a person can be members of several communities such as [17–19], and methods of local community detection, which we will discuss in the following section. As networks grew and became widespread, exploring the structure of the community by looking at the entire graph and dividing it into several groups became expensive. To date, the community detection algorithms have generally used global information, but with the growth of social networks, it has become more difficult to access global information. Thus, the algorithms which used local information became more widespread. Clauset was the first to introduce the idea of using local information. Although the algorithm was simple and efficient, the size of the communities had to be determined first [20]. Most of the algorithms that locally search for communities start with some initial nodes. There are various ways to find the primary nodes, such as node degree, closeness, eigenvectors, and influence maximization. There are various ways to extend the nodes. Most of these methods have different criteria for expanding the seed set. If adding a node to the community helps to raise the criterion, that node will be added to that community. First, a search is carried on for a series of initial nodes, and then an attempt is made to expand those nodes into communities [3]. Research in this field continues on both static and dynamic networks.

Algorithms for static community detection Wang et al [21] proposed using two strategies to find seed sets called Graclus-conters [22] and Spread Hubs [21]. Their strategy is based on a Kernel distance [22]. Graclus is a high-quality graph partitioning scheme. Using this distance function, they could find the top seed in an existing set of graph vertices. The idea of finding nodes using the spreading of the hubs is to choose a collection of independent nodes of high degree [21].

This method is inspired by the observation of clusters around nodes of high degree in real networks. Wang and his colleagues employed the personalized page rank to expand the seed set [21], while kloumann used the traditional page rank to do so. The main idea of these two methods is to take a random walk on the infomap. Infomap has information such as the direction and weight of all the edges. Another way is to use node centrality. This method puts more emphasis on the hubs and edges connected to them and requires that the most closely related nodes should be in one community [23]. Yakoubi and Kanawati proposed the LICOD algorithm [24]. This algorithm selects a set of leader nodes by comparing them with its neighbors. Leader nodes with a certain percentage of shared neighbors are grouped in the same community. Fagnan proposed a method that randomly chooses a node and then selects its neighbor with the highest degree as the primary node. It then checks the remaining nodes and selects the neighbor with the highest degree as the next primary node [25]. Biswas proposed the ENBC algorithm. This method selects the nodes with the highest degree as the primary node. Checking all the neighbors of each node, the method groups each node whose number of edges with the initial node is greater than a specified value into one community [26]. Jiang and his colleagues used influence maximization to find the initial nodes [27]. They defined a criterion for calculating the level of influence the nodes have on each other. For each node v, they calculated the influence that the initial nodes have over them. Then node v becomes a member of the initial node that has the highest impact on node v. LGIEM [28] uses both local and global information. This algorithm calculates the influence of nodes in the network and selects the top-k nodes as seeds. Some node expansion methods select one initial node and expand it, and then remove the community obtained by extending the initial node from the network and select and expand the remaining nodes that are not yet in any community in the network. This process continues until all nodes are at least in one community. Tabarzad et al. proposed a three-phase method called heuristic local community detection (HLCD) [13]. In this method, the node degree is used to find the initial nodes. Generally, in this algorithm, the highest degree node in the graph is selected as the initial node, which is expanded by several criteria to create a community. The community and the nodes within it are removed afterwards from the network. This process continues until no nodes remain unaccounted for. As mentioned, this algorithm has three phases. The first stage is early community detection; it selects the primary nodes according to their degree and adds other nodes to the initial nodes based on a node-community dependency. By the end of this stage, many early communities are formed. Although these communities are very dense, they are not a real community yet. In the second stage, these communities are merged based on a community autonomy measure to create real communities. After merging early communities to form real communities, there are many overlaps among them. The third stage removes the overlap among the communities
Gue proposed a local community detection algorithm based on internal force between nodes called InfoNode [29]. He uses local degree central nodes and Jaccard coefficient to detect core members of communities as seeds in the network. Then using a fitness function, he expanded the node with the highest degree among the seeds. Afterwards, with a new expansion strategy using similarity and distance among the unassigned nodes, he expanded initial seed set to create communities. Connecting the created communities could be a problem in community detection algorithms. Thus, Leiden [30] algorithm was proposed by Traag to guarantee that the communities are connected. A fast and strong community detection was proposed by Bouyer and Roughani which starts from low degree nodes called LDMS [31]. In LDMS, local community detection begins from low degree nodes by using a modern label assigning in a multilevel diffusion. They also presented. another community detection algorithm based on local balanced label diffusion (LBLD) [32]. This algorithm starts with assigning a score to each node, which presents the importance of that node by using a new local similarity measure. Similarity-Guided Community Merge and Refinement (SimCMR) [33] was proposed by Tunali for large scale community detection based on two novel similarities. In the first stage, by using a mechanism similar to information propagation, the candidates will be generated at a rapid speed. In second stage, small candidates will

merge with larger ones if they have fewer nodes than a calculated threshold. This merge will be done by using similarity between nodes and communities.

Algorithms for dynamic community detection To date most of the proposed algorithms have been developed for static networks, and only a few for dynamic ones. Dynamic networks are networks in which the membership of nodes in communities changes over time. Researchers have paid less attention to dynamic network community detection than to static networks, but research in this area has gained momentum in recent years.

Bansal has classified the community detection methods in dynamic networks into two categories, namely online community discovery and offline community discovery. The former has to do with real-time data changes, and the latter has to do with data in which all changes in network evolution have been defined [34]. Wolf suggested using mathematical and computational rules to analyze dynamic communities based on social interactions occurring on the network [35]. Lin presented the Facet Net algorithm. Facet Net is a unified framework to examine the evolution of dynamic communities. The network structure at any given time includes current snapshot data as well as information from previous snapshots [36]. To find the time dependency in dynamic networks, Pala conducted experiments on two different datasets of the telephone call network and the collaboration network [37]. After constructing a graph containing two snapshots, he used the CPM algorithm. Pala and his colleagues used an automatic correlation function to find the overlap between two snapshots of a community and a static parameter representing the average correlation of different snapshots.

Green et al. showed the dynamic graph as a set of snapshot graphs [38]. The communities of each graph represent the dynamic communities at a particular time. This algorithm starts with a static community detection algorithm. Then dynamic communities are created for each time, and Jaccard similarity [39] is calculated for each snapshot. DiTursi designed a new method for dynamic networks and used Filter-and-verify Framework for community detection in dynamic networks [40].

From another point of view, there are various methods to detect communities in dynamic networks, which can be divided into two approaches [41]: Independent and Incremental. In the independent approach like [11], each snapshot is considered a single one, and the previous information snapshot is not considered. This approach is suitable for unstable social networks that have experienced abrupt changes at intervals. In the incremental approaches such as [12], each community at a snapshot is identified by considering the information about the previous snapshot(s). For example, Rossetti considers all the previous time steps to detect communities at every snapshot [42]. If there is accurate information about the number of changes between snapshots, it is highly logical to use this approach. For example, Rabbani and Takaffoli presented a new approach based on an incremental approach in which the current snapshot communities have been derived from the previous snapshot [43]. This method assumes there is always a slight change between two consecutive snap- shots, which is not always true. Later, another method was proposed by Samie [41], which first identifies the type of change between two sequential snapshots, and then decides to use either incremental or independent approaches to detect the communities. If the changes between the two consecutive snapshots were gradual, the algorithm adopts an incremental approach, which uses both the current snapshot information and the information from the previous snapshots to detect the communities. If there was an abrupt change, the algorithm adopts the independent approach with only the current. snapshot information to detect communities. Liu et al. proposed an incremental bottom-up method to detect communities from dynamic graphs. This method uses Link Prediction and Information theory to know when the influence of a node has changed to identify the portion of the graph that needs to be recalculated [44]. Local seed expansion methods have often been very successful in finding communities in dynamic networks. In 2016, Laarhoven [45] presented a method that used continuity of transmission and showed that despite the continuity of the optimization, the method resulted in separate communities. Takaffoli employed the evolutionary clustering to follow the changes in any snapshot in the global network. Her algorithm tracks local communities from one snapshot to the next [46]. Wang and Li proposed a novel Dynamic Overlapping Community Evolution Tracking (DOCET) method. This method identifies the overlapping community structure to update the dynamic community structure based on the influence through the network. Then, it tracks community structure evolution based on the variation of core nodes [47] Samie et al. introduced a local evolutionary method called GL-metric [1]. This method is a two-phase algorithm that uses both local and global information. This algorithm first searches for global information and then uses local information to make the organizations more accurate. This algorithm is a continuation of Pizutti's work [12]. In the first phase, it detects communities using global information. Then, using local information, it modifies the communities obtained in the previous phase to identify communities more accurately. Lou proposed the Local Community Detection with the Dynamic Membership Function algorithm. This algorithm has three stages. At the initial stage, a dynamical membership function is designed to detect a local community and add the nodes with the greatest neighborhood intersect rate to the local community. At the middle stage, another dynamical membership function is designed to make the connection of the node in the local community as closely as possible. At the closing stage, the third dynamical membership function is provided, further improving the local community by collecting some nodes that should not be omitted [48]. Another algorithm that uses a multi-objective approach is MDOA [49]. This algorithm uses three optimization objectives: partition density [50], extended modularity [51], and improved mutual information. One of the most recent algorithms in this area is Local Community Detection by the Nearest Nodes with greater centrality. This algorithm starts with an initial node of v; the remaining nodes are added to the local community one by one. For adding u to a local community, u should be the nearest node with greater centrality (NGC) of a node in the local community, or u's NGC is already in the local community; in addition, the fuzzy relation between u and its NGC node is the largest, which is not less than half of the average fuzzy relation of the current local network [52]. The difference between these methods is in their local measures. In general, algorithms of local approach have to answer one or more of the following four questions:

1. What nodes are selected as the initials to achieve a more accurate result in a shorter time?
2. What optimization algorithm should be used to expand the initial nodes and detect communities?
3. Which similarity measure should be employed to recruit nodes for communities?
4. Is the proposed method for static networks applicable to dynamic networks, and if so, how?

As mentioned before, we mainly focus on improving the accuracy of local community detection algorithms, which are applied to static networks, by developing them. Our objective is to make more applicable to dynamic networks. To this end, we use an incremental approach. In the next section, we propose a new approach and the new methods in detail.

3 Proposed methods

Despite extensive studies of community detection in social networks, the proposed methods have not been desirable yet on real social networks, and improvements are still ongoing. Due to the increasing size of social networks, the methods that only use global information are less efficient. In this paper, we will use both global and local information to gain better results. We try to develop one of the newest methods of local approach that we previously worked on (HLCD) and extend it to be applicable to dynamic networks. As mentioned before, the scenario for the methods of the local approach is generally to start from a few nodes that appear to be more closely related to other ones and then to examine their neighbors until all nodes are a member of at least one community. One of the fascinating research areas is proposing methods that select better and more effective initial candidate nodes. In other words, optimal selection of initial nodes must lead to increased accuracy at the priority and then reduced runtime, if possible. As mentioned in the last section, the HLCD algorithm is one of the newest methods of local approach. We present our method in two stages: First stage: We start by improving HLCD [13] algorithm using a new method of finding initial nodes by using the influence maximization concept. At this stage, we achieve our first goal which is increasing the accuracy in detecting communities. Second stage: Developing the method proposed at the first stage to be applicable to dynamic networks

3.1 First stage: local community detection based on influence maximization in static networks

The HLCD algorithm uses the most straightforward centrality measure, degree centrality, to find initial central nodes. Central nodes in HLCD are the nodes with the highest number of edges with their neighbors. Since this measure is not a reliable concept for influence maximization, we will find the most influential 40 nodes in social networks. In other words, we claim that influential nodes are better candidates for initial central nodes because they are more effective than nodes with only a high number of edges. This change will significantly improve the performance of the HLCD [13] algorithm.

3.1.1 Finding more optimal central nodes

As mentioned above, we need to find more optimal primary nodes for early communities to lead to more precise communities. Here we use the two-phase influence maximization [8] algorithm to find the most influential nodes in social networks. In the proposed algorithm, we use Reversely Reachable sets to find the most influential nodes with a reliable guarantee. In order to find influential nodes, one must follow influence throughout the network. Many diffusion algorithms can be used to trace network influence, including the Linear Threshold Model and the Independent Cascade Model [53]. Although these are some of the best propagation algorithms that can be used to estimate network influence, finding a solution for influence maximization under these two models is NP-hard. Thus, we can use a combined method such as TIM (Two-phase Influence Maximization) [8] to estimate the extent of influence in influence maximization problems. Here, we use a Fusion Method to build Reversely Reachable (RR) sets [53]. As shown in Algorithm 1, to build a RR set, let's start with a random node u and add it to the RR set. Then the neighbors of v that have outgoing edges to v are found. In other words, the v's parents are found and put in a set called P. For each node u, the weight of the edge coming from u to v is compared to the probability of being activated to see if node u can activate node e. If the edge weight is higher than the probability of being activated, the node, u, will be added to the RR set, otherwise it will not. This process will continue until there is no other parent. Here, the RR set indicates if propagation starts from node v, what other nodes would be informed. By examining RR sets, we can find out what nodes are in most of the propagation paths. First there is a need to build a large number of RR sets. The set R, contains a large number of RR sets in the algorithm. To find the most influential node, we need to check how many times each node has been repeated in the set R. The node with highest frequency is the most influential node. So far, we have found the first most influential node and want to find more. To do this, we must remove those RR sets in which the first most influential node is a member of the R set and find the next influential node among the remaining RR sets in R. The pseudo-code number 1 and 2 describes the process of building RR sets and finding influential nodes. It is worth mentioning that in the second pseudo-code, we do not specify the number of communities in any way, and the parameter k is the number of influential nodes found in each step of the algorithm which here is equal to 1, rather than the number of communities in a social network. In the proposed method, we find the influential nodes one by one at the first stage, and we continue the process as long as all the nodes belong to at least one community.

```
1: input: weighted graph G(V,E,w)
2: output: Reversely Reachable set R
3: function CREATING RR.SETS (G)
4: R=[ ]
5: num = number of RR sets
6: repeat
7: Candidate set (CS)=[ ]
8: RR_set=[ ]
9: random node v
10: $RR - set \leftarrow v$
11: $CS \leftarrow v$
12: while $CS \neq 0$ do
13:     u=CS[1]
14:     remove u from CS
15:     P=parents(u)
16:     $Cs \leftarrow P$
17:     for all $u \in CS$ do
18:         if $rand(0,1) < wuv$ then
19:             $RR\_set \leftarrow u$
20:             remove u from CS
21:         end if
22:     end for
23: end while
24: $R \leftarrow RR\_set$
25: until size(R)==num
26: return R
27: end function
```

Algorithm 1 Creating RR sets for a network.

```
1: input: Reversely Reachable set R
2: Output: Seed set S
3: function K INFLUENTIALNODES (R.k)
4: S= [ ]
5: k=number of influential nodes
6: count [ ]
7: repeat
8: for all$u \in V$ do
9:     count[u]=number of RR_sets in R that have u
10: end for
11: influential_node=max(count)
12: $S \leftarrow influential\_node$
13: remove RR_set in R that contain influential_node
14: until size(S)==k
15: return S
16: end function
```

Algorithm 2 Finding top-k most influential nodes.

3.1.2 Detecting communities by expanding influential nodes

This Part of the algorithm is related to the HLCD algorithm [13], but we will briefly review the whole algorithm in these three steps for a better understanding.

- First step: The algorithm uses a simple central measure to find the initial nodes and expands them to form the early communities.
- Second step: The algorithm merges neighboring communities.
- Third step: It removes the overlaps between the communities.

As it can be observed in Algorithm 3, to identify the early communities, we first have to find the most influential node from the R set, which contains a large number of RR sets. This node v is considered as the first influential node, and it expands to form an early community. After that, we find the next influential node. The process continues as long as all nodes are assigned to at least one community. It is not necessary to know the exact number of communities to know how many influential nodes we need to find. In this step, we keep finding influential nodes until all social network nodes belong to at least one early community. Then

```
1: input: graph G(V,E,w), Reversely Reachable set R
2: output: early communities EC
3: function EARLY COMMUNITY DETECTING (G,R)
4: Unassignednodes=V
5: EC=[ ]
6: while $Unassignednodes \neq 0$ do
7:     Candidate_Node=k_Influential_Nodes(R,1)
8:     R= remove RR_sets containing Candidate node in R
9:     if $Candidate\_Node \not\subset Unassigned\_nodes$ then
10:        Candidate_Node= k_Influential_Nodes(R,1)
11:        R= remove RR_sets containing CandidateNode in R
12:    end if
13:    early_community= expand Candidate_Node to form early community
14:    $EC \leftarrow early\_community$
15:    Unassigned_nodes= Unassigned_nodes_early_community
16: end while
17: return EC
18: end function
```

Algorithm 3 Early community detection.

by using some measurements, these early communities will merge to form actual communities. Since this step is in the HLCD algorithm, we do not include it in this paper. The rest of this paper refers to community detection. As shown in the next section, even though the expanding and merging scenarios in our proposed method are the same as HLCD, the performance on static social networks is significantly improved by changing the initial nodes finding method.

3.2 Second stage: local community detection based on influence maximization in dynamic networks

The first part of our contribution focused on improving the performance of existing methods for community detection in static networks. The second part is to extend the proposed method so that it also applies to dynamic networks. As mentioned earlier, dynamic networks change over time, which may change the initial RR sets for subsequent snapshots. The first way that comes to mind for dynamic networks is to create new RR sets for each snapshot, which is very time-consuming. According to our research, the graph structure does not change much due to gradual changes in snapshots. It is best to use a criterion that determines when the last RR sets can be used and when it cannot be used. Similarity is one of the best criteria. We can decide whether to use the previous RR sets or update it for a new snapshot by calculating the similarity between two snapshots.

3.2.1 Using deep neural network to predict the similarity of two snapshots

A Deep Neural Network (DNN) is a complex type of Deep Learning which is an Artificial Neural Network (ANN) with multiple layers between the input and output layers. each neural network unit can be represented as a graph node, whereby DNN offers an accurate analysis in problems which involves working with graphs. In this study, we observed that the gradual changes in the two successive snapshots did not always lead to the change of the influential nodes in many cases. Thus, despite the progressive change in two snapshots structures, the RR sets change a little or do not change at all. Thus, the influential nodes do not change in many cases. As a result, the RR sets of the previous snapshot can be used to predict the influential nodes of the current snapshot. Our goal is to find a common criterion between the two snapshots, through which we can identify the amount of similarity between the two networks. We realized that as the network changes over time, so do the RR sets, which sometimes leads to the change of influential nodes. In other words, It can be claimed that the more the network changes; the more the influential nodes change. Therefore, investigating changes in the influential nodes can guide us to the level of differences between the two snapshots and thus give us a measure. The use of influential nodes to compare snapshots has some benefits. In essence, by using influential nodes to compare similarities between snapshot, one is comparing their RR sets without needing to create them for each snapshot. Based on deep neural networks superiorities, we decided to use it to compare the similarity between snapshots. We built a deep neural network that gets two snapshots as inputs and returns their similarity. To do this, we had to provide lots of training samples to train a deep neural network. Briefly, the scenario for this stage is:

1. Training a deep neural network to compare the similarity between two snapshots
2. Compare their similarity with a threshold to decide if the previous snapshots RR sets can be used or the RR sets must be updated.

We will explain how to train a deep neural network, calculate the similarity, and determine the threshold value.

3.2.2 Creating new graphs for training samples

We needed a large number of samples to train the deep neural network at this point. To do so, we used the Launchicinti-Fortunato-Radicchi LFR method, which is very similar to real-life datasets [54]. It is a benchmark for creating artificial networks. In the following section, we will thoroughly explain this database. In the LFR benchmark,

Table 1 Different LFR networks for training deep neural network

Number of nodes in network	number of created graphs
100	50
200	50
400	50
700	50
1000	50

one can define the number of nodes, the maximum number of communities, the lowest and the highest degree of a graph, and many other parameters. The LFR constructs the network according to those parameters. We built 250 artificial neural networks using LFR, and then we created 12500 samples for training. The process of creating training samples for the neural network will be explained afterwards.

As seen in Table 1, by changing the LFR generators parameters, we made sure each network is different from the others, and we built various network models with a different number of nodes. Our goal was to compare networks with the same number of nodes to build a training sample. We had to find the influential nodes for each snapshot and compare them to another snapshot with an equal number of nodes. In other words, we wanted to find the relationship between the similarity of two snapshots and their influential nodes.

3.2.3 Mapping graph into vector

So far we have represented the networks with graphs and its adjacency matrix for the calculations, but we have to change the graphs representation and display the adjacency matrix as a one-dimensional vector. To do so, we used a Convolutional auto-encoder to convert the graph to a vector. These vectors are the training samples for the deep neural network (Algorithm 4).

3.2.4 Creating training samples

To construct training samples, we compared the networks with the same number of nodes. First, for each LFR network, we calculated the RR sets and then used them to obtain the influential nodes of each network. We then compared the networks influential nodes one by one and assigned their similarity as the label of each two networks. If we did this for each pair of networks with the equal node number in all of the created LFR networks, we would eventually have 12500 training samples, a fair number of training samples for a deep neural network. Check Algorithm 5 and Fig. 1 for more details.

3.2.5 Training neural network to compare similarities between snapshots

At this point, we used the created training samples to train feed-forward perceptron network.Using a deep neural network helped us to have a tool that can quickly estimate any two given snapshots similarity without having to calculate the RR sets. Training this neural network is time-consuming, but neural networks have an outstanding benefit. We can train neural networks just once and then save the weights of the model and then, we can use it to predict the similarity of any two snapshots. It is worth noting that we have used Dense and Dropout lavers in this network. The dropout layer reduces part of the input data and thus prevents overfitting.

3.2.6 Updating reversely reachable sets

As mentioned earlier, we trained a deep neural network to compare the similarity of each two snapshots. This allows us to easily determine whether to use the previous RR set for the current snapshot, or to update it. In case the latter is true, it is advisable to update the RR sets purposefully, based on the type of change, to consume less time. we should consider four modes:

1. A new edge is added,
2. A new node is added,
3. An edge is removed,
4. A node is removed.

```
1: input: graph G(V,E,w)
2: Output: vectorized_network
3: function VECTORIZING.NETWORK(graph)
4: network_model=training a convolutional auto-encoder network
5: latent_jayer=network_model.predict(graph).get_layer_out(latent_layer)
6: vectorized_network=latent_layer
7: return vectorized_network
8: end function
```

Algorithm 4 Vectorizing a network.

```
1:  Input: network_1,network_2
2:  Output: training_sample
3:  function CREATING TRAINING_SAMPLE(network_1,network_2)
4:  RR_set_1=Creating_RR_sets(network_1)
5:  RR_set_2=Creating_RR_sets(network_2)
6:  seed_set_1=k_influential_Nodes(RR_set1,1)
7:  seed_set_2=k_influential_Nodes(RR_set2,1)
8:  similarity=compare(seed_set_1,seed_set_2)
9:  vectorized_network_1=vectorising_network(network_1)
10: vectorized_network_2=vectorising_network(network_2)
11: training_sample=(rectorized_network_1,vectorized_network_2,similarity)
12: return training_sample
13: end function
```

Algorithm 5 Creating training samples.

Since adding a new node adds a new edge, and removing the node removes the edge, we could only consider adding and removing edges.

* New edge

If there is a new edge (u - v) in the snapshot, weights of incoming edges of v will change. As a result, all the RR sets in which the node is, are no longer trustworthy and need to be updated.

In this case, we have to remove all nodes after v and recreate the RR set from there.

* Removed edge

If an edge (u - c) no longer exists in the snapshot, weights of all incoming edges of e will change. Now we should change all the RR sets in which both u and e exists. The critical key is to change RR sets that have c after the node u. After finding all the RR sets that have the necessary condition, we have to remove all the nodes after u and recreate the RR set from that point.

3.2.7 The final proposed method

In local community detection algorithms, initial nodes are selected, and then the nodes expand to form communities. We used an influence maximization concept to find the seed set. We then developed this algorithm to apply to dynamic networks as well. To prevent the calculation of the RR set in each snapshot, which is very time-consuming, we trained a deep neural network to predict the similarity of two consecutive snapshots. If the similarity is higher than a threshold value, we use the RR sets of the previous snapshot for the new one, and if it is less than the threshold value, we update the RR sets of the previous snapshot for the new snapshot and use it. Although training neural networks may be time-consuming, we can train the neural network only once and save the neural network model weights and use it repeatedly for every new dataset easily. Algorithm 6 provides all the details of our final method. Before we

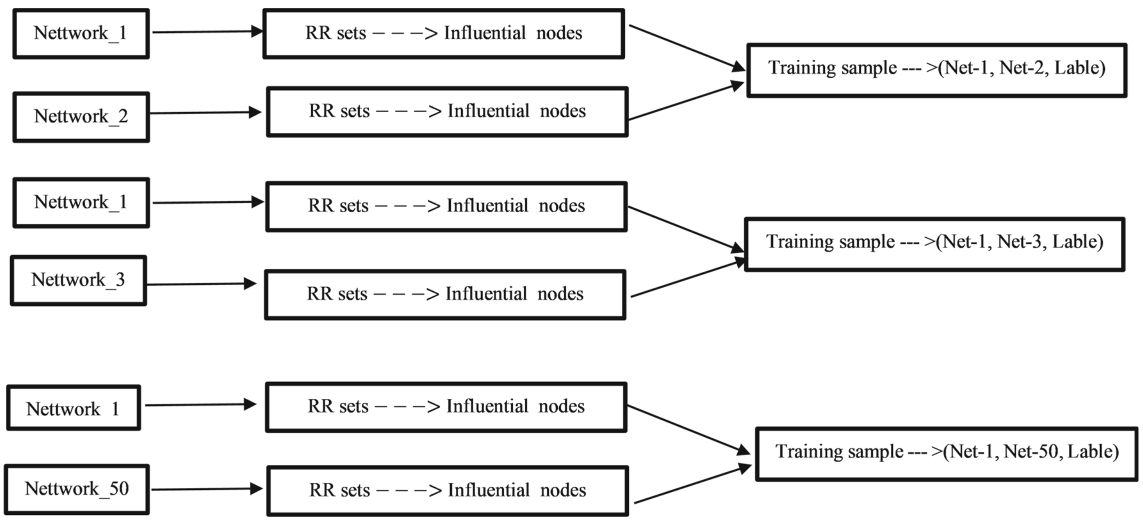
Nettwork_1
RR sets − − −> Influential nodes
Nettwork_2
RR sets − − −> Influential nodes
Training sample --- >(Net-1, Net-2, Lable)
Nettwork_1
RR sets − − −> Influential nodes
Nettwork_3
RR sets − − −> Influential nodes
Training sample --- >(Net-1, Net-3, Lable)
Nettwork 1
RR sets − − −> Influential nodes
Nettwork_50
RR sets − − −> Influential nodes
Training sample --- >(Net-1, Net-50, Lable)

Fig. 1 Creating training sample for deep neural network

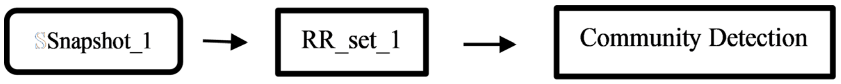

Fig. 2 First snapshot's community detection

carry on, the question arises about the optimal value of the numerical threshold that can determine the similarity for us. In other words, how similar two snapshots need to be in order not to update RR sets. It is worth mentioning that we came up with the best threshold by experiments. In our study, we worked with many networks. We created the RR sets for each network and got their influential nodes and then we compared them two by two which led to over 15000 comparisons. We were looking for the highest possible amount. At first, we started by choosing θ = 10%. We only updated RR sets for the social networks the similarities of which to other ones were less than 10%. This means we did not update social networks with more than 10% similarities. After this, we continued with community detection. Many snapshot RR sets that were not updated gave poor results. Thus, we changed θ to 15. We did so until θ = 80%. In θ = 80% or θ = 85% or θ = 90%, we did not see any difference in the results. This meant that not updating social networks with more than 80% similarity was a good choice. We chose θ = 80% as the optimal threshold value and used this for all the experimental datasets and got the acceptable results.

The following is an example of the proposed algorithm. Suppose we have a network with five snapshots.

1. At first, The neural network model and the neural network weights must be loaded.
2. The first snapshot is treated as a static network. For this snapshot, we compute the RR sets and detect communities (Fig. 2).
3. Then it is time to detect communities in the second snapshot. At first, we map the first and second snapshots to a one-dimensional network using a feed-forward perceptron network, and then we should find the similarity of the first and second snapshots using the deep neural network we loaded earlier (Fig. 3).

   Here we see that the similarity of these two snapshots is 100%, so we can use the first snapshots RR sets to identify communities in the second snapshot. Even though the similarity between snapshot 1 and snapshot 2 is more than theta, it does not necessarily mean the graph structures are the same. There may be a difference in the number of edges or nodes. It would be best if you remembered that the trained neural network predicts this similarity. Therefor, we still need to do community detection.
4. It is time for the third snapshots community detection (Fig. 4). Since we have the RR sets of the first snapshot, we compare the third snapshots RR set with the first one.

   Now, we see that the similarity of the first snapshot and the third one is less than the defined threshold, so we should update the RR sets to attain the third snapshots RR sets. We then detect the communities using the updated RR sets.
5. For the fourth snapshot (Fig. 5), we have the RR sets of the first and third snapshots. So we have to compare the fourth snapshot with both of them.

   We can see that the fourth snapshot is more similar to the third snapshot than the first one. As a result, we use the RR sets of the third snapshot to detect the communities.

   In this example, by using the deep neural network to compare the similarity of two snapshots, the RR sets were created just once and needed to be updated once, instead of creating them four times. As can be seen, the use of deep neural networks significantly reduced the computational cost. This method will be handy for real and large social networks.

4 Experimental results

In this section, we compare the performance of the proposed method with some other well-known algorithms. It is recommended to use synthetic benchmark datasets or real-life networks with the ground-truth label to evaluate community detection algorithms. We ran our codes 30 different times on MATLAB environment. The PC in use had a core i7 processor and 8GB RAM.

4.1 Evaluation measure

There are several measures for comparing the results of community detection algorithms with the ground-truth. One of these efficiency measures is Normalized Mutual Information (NMI) [55]. This measure uses the ground-truth to determine the accuracy of the community detection algorithms. NMI is one of the best-known tests for comparing the similarity of two networks. For the two

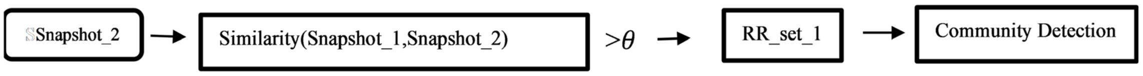

Fig. 3 Second snapshots community detection

```
1: input: similarity_model_weights.h5, similarity model_architecture.json, Graph(V,E,t)
2: output: communities
3: function FINAL_COMMUNITY_DETECTION(Graph(V,E,t)), similarity_model_weights h5, similarity_model_architercture.json
4: load similarity_model_architeture.jason
5: load similarity_model_weights.h5
6: vectorized_snupshot_1 = vectorising_network(snapshot_1)
7: RR_set_1= Creating_RR_sets(snapshot_1)
8: communities = community_detection(snapshot_1, RR_sct_1)
9: for all snapshots in graph do
10:     vectorized_current_snapshot=vectorising_network(current_snapshot)
11:     similarity= compare ( vectorized_current_snapshot, [all snapshots that have RR_sets ])
12:     best RR_set_similarity=Max(similarity)
13:     if $best\_RR\_set\_similarity < threshshold$ then
14:         new_RR_set= update best_RR_set
15:         communities=community_detection(current_snapshot, new_RR_set)
16:     else
17:         communities= community_detection(current_snapshot,best_RR_set)
18:     end if
19: end for
20: return Communities
21: end function
```

Algorithm 6 Final community detection.

partitions: A = A1,..., Aa) and B = B1,..., Bb) of a social network partitioned in communities and C is the confusion matrix with element C_{ij}, the number of nodes of the community A_i, ∈ A which are in the community B_i ∈ B too. The normalized mutual information NMI (A, B) is:

$$NMI(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_b} C_{ij}\log\left(\frac{C_{ij}N}{C_iC_j}\right)}{\sum_{i=1}^{c_A} C_i\log\left(\frac{C_i}{N}\right)+\sum_{j=1}^{c_B} C_j\log\left(\frac{C_j}{N}\right)} \tag{1}$$

In the above equation, C_A and C_B are the community numbers in the partition A and B. C_i and C_j are the sum of the elements of C in row i and column i, and N is the number of nodes. If A and B are the same, NMI (A, B) = 1. If they are entirely different, NMI (A, B) = 0. Another measure to compare community detection algorithms is Q modularity [51]. Modularity is widely used as a measure for how functional a clustering is. Higher modularity means nodes within the cluster have dense connections between themselves but sparse connections with other nodes. In the equation below, m is the number of edges in a network. K_i shows the degree of node i, and A_{ij} is the network's adjacency matrix. s represents the membership of each node.

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{K_iK_j}{2m}\right]\frac{S_iS_j+1}{2} \tag{2}$$

4.2 Datasets

To evaluate our method, we have used a variety of datasets, including synthetic and real-life datasets, as following.

4.2.1 Static datasets

- **LFR benchmark** This artificial dataset is provided by **Launchicinti-Fortunato-Radicchi** (LFR) [54].
  This dataset has various parameters that we can control network structure by changing their values.
  N: Number of nodes in the network
  Average_degree : the mean degree of each node
  Mox_degree : Maximum node degree in the entire network

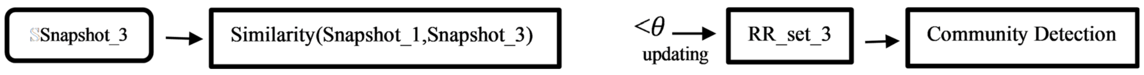

Fig. 4 Third snapshot's community detection

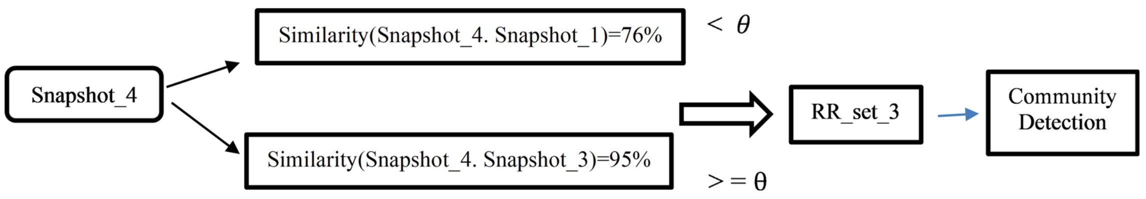
Snapshot_4
Similarity(Snapshot_4. Snapshot_1)=76%
Similarity(Snapshot_4. Snapshot_3)=95%
< θ
> = θ
RR_set_3
Community Detection

Fig. 5 Last snapshot's community detection

Min_community : The minimum number of communities in the network
Max_community : The maximum number of communities in the network
Mu: Specifies the amount of network complexity. The value of this parameter is in the range of (0, 1). The more complex a network becomes, the harder it is to detect communities(Tables 2, 3 and 4).

LFR_1

These parameters led to the creation of a network with 128 nodes, 341 edges, and 15 communities

LFR_2

With these parameters, a network of 1000 nodes, 7680 edges, and 30 communities was made

LFR_3

This network has 10,000 nodes, 150,740 edges, and 152 communities.

Considering that modern networks have large number of nodes, we tried to assess the performance of the proposed method by creating three LFR synthetic dataset with 50,000, 75,000 and 100,000 nodes. The parameters of these datasets are in Table 5.

- **Zachary karate dataset**

This dataset has 34 nodes representing individuals who were members of a karate club in the United States [56]. The edges also indicate the friendship between these people. After the conflict between the coach and the manager of the club, the club was divided into two groups. The first group built a new club with the coach, and the other group stayed with the club manager. This dataset is used as a benchmark in the field of clustering and community detection.

Overall, the network consists of 34 nodes, 78 edges, and 2 communities

- **Static football dataset**

This network includes the college football tournament schedule [7]. The nodes represent the teams that participated in the tournament. We considered the football network for the year 2000.
It has 115 nodes, 616 edges, and 12 communities.

4.2.2 Dynamic datasets

- **SYNFIX**

The Synfix dataset is an artificial dataset designed by Girvan-Newman [7]. This software generates dynamic networks that are divided into a number of communities in the T snapshots. In order to investigate the proposed method, we used two dynamic Synfix networks. The networks have 128 nodes, 4 communities, and 10 snapshots, and each node has an average of 16 edges. The mixing parameter is 3 and 5 in the first and second networks.

In each snapshot, 3 nodes randomly exit each community to join one of the other three communities(Table 6).

- **SYNVAR**

The Synvar synthetic dataset is created by making changes to the Synfix dataset [7]. In the dataset, the parameters are set such that the number of communities in each snapshot changes.

Table 2 Parameters of first LFR benchmark

I	N	128
II	Average_degree	150
III	Max_degree	15
IV	Min_community	4
V	Max_community	16
VI	mu	0.1

Table 3 Parameters of second LFR

I	N	1000
II	Average_degree	15
III	Max_degree	50
IV	Min_community	20
V	Max_community	50
VI	mu	0.1

Table 4 Parameters of third LFR benchmark

I	N	10000
II	Average_degree	30
III	Max_degree	100
IV	Min_community	40
V	Max_community	100
VI	mu	0.1

The first snapshot contains 393 nodes and 5 communities; 4 communities have 64 nodes, and one community has 137 nodes.

- **Dynamic football dataset**

  In order to evaluate the dynamic approach, we used the football games of 2005-2007, which were used in [1]. These networks have 120 nodes and 12 communities.

- **Dynamic facebook dataset**

  This dataset consists of friends lists from Facebook as circles [57]. The dataset includes profiles as node features, ego networks, and circles. The ids of each user have been replaced with a new value to stay anonymous. The dataset consists of 4039 nodes and 88234 edges in 10 snapshots.

4.3 Evaluation algorithms

In this section, we review the algorithms that are to be compared with our proposed method. The first method presented in this article is a community detection algorithm for static networks, and the second method is the algorithm for community detection in dynamic social networks. We first compare the result of the first with some community detection algorithms in static networks and then compare the second method with some community detection algorithms in dynamic networks. It is worth mentioning that the comparison was made with methods whose results were in their reference or the source code was available to the authors. It is possible that in some cases the results of a few methods is shown with "-" which means that the results was not in the references or the source code was not available.

Table 5 Specifications of produced data

Name	N	maxK	K	mu	minC	maxC
N50	50,000	100	30	0.1	40	1000
N75	75,000	100	30	0.1	40	1000
N100	100,000	100	30	0.1	40	1000

Table 6 Parameters of Synfix datasets

	Mixing parameter	Average degree	Snapshot(T)
Synfix_1	3	16	10
Synfix_2	5	16	10

4.3.1 Community detection algorithms in static networks

- The first algorithm to compare is the [13]. This algorithm uses top nodes with the highest degrees as the initial nodes and expands communities around them
- The second is the one proposed by Yakoubi and Kanawati called LICOD [24]. This algorithm selects a set of leader nodes by comparing them with its neighbors. Leader nodes with a certain percentage of shared neighbors are grouped in the same community.
- Fagnan proposes the third selected algorithm [25]. This method randomly selects a node and then selects its neighbor with the highest degree as the primary node. It checks the remaining nodes, and the next-highest neighbor becomes the next primary node.
- ENBC algorithm was presented by Biswas [26]. This method selects the nodes with the highest degree as the primary node. It checks all its neighbors for each node, and each node whose number of edges with the initial node is higher than a specified value falls into one community.
- Another method that we are going to compare with our method is the DOCET algorithm [47]. This algorithm first detects the initial overlapping community structure. Then, it incrementally updates the dynamic community structure.
- We also considered the method Tinghuai Ma proposed. He proposed LGIEM method which first finds influential nodes and then expands them to form final communities [28].
- InfoNode is another algorithm which is used in this field. This algorithm uses internal force between nodes. This algorithm first detects core nodes by using node degree and Jaccard coefficient. Then by using a fitness function top k nodes are extended with internal force between nodes create communities [29].
- A fast and robust local community detection starting from low degree nodes in social networks (LSMD) [31]. This algorithm starts with assigning a score for each node which presents importance of that node by using a new local similarity measure.

- A fast community detection algorithm based on local balanced label diffusion (LBLD) [32]. This algorithm starts with assigning a score for each node which presents importance of that node by using a new local similarity measure.
- Traag proposed Leiden algorithm, guaranteeing well-connected communities [30]. In this algorithm, communities are guaranteed to be connected by using a fast local approach. In Leiden, all subsets of all communities are locally assigned with an optimal partitioning.
- Similarity-Guided Community Merge and Refinement (SimCMR) [33] was proposed by Tunali for large scale community detection based on two novel similarities. In SimCMR, candidates will be generated at high pace by using a mechanism similar to information propagation. Then these candidates will be merged with larger ones if they have fewer nodes than a certain threshold.

4.3.2 Community detection algorithms in dynamic networks

The algorithm proposed for dynamic networks will be compared to four algorithms.

- The first algorithm is the DOCET method [47], which was explained in the previous section.
- Lin proposes an algorithm called FacetNet [36]. FacetNet is a unified framework for examining the evolution of dynamic communities. The structure of a network at any given time includes current snapshot information and the information of the previous snapshot.
- The third algorithm is proposed by Pizzuti called DYNMOGA [12]. This algorithm uses the genetic algorithm approach to detect communities in dynamic networks.
- GL-metric [1] proposed by Samie et al. This method is a two-phase algorithm. The GL-metric detects community in a global approach and then tune them in a local approach. In other words, they proposed a global-local method for community detection in dynamic social networks.
- The last method is the (MDOA) [49], which was proposed by Wan et al. This method is based on MOEA/D. This algorithm decomposes the multi-objective Community detection problem into a number of single objective optimization sub-problems.

4.4 Compared results

This section will compare the results of our static and dynamic community detection algorithm with other community detection methods.

4.4.1 Comparing community detection algorithms in static networks

All of these methods detect the communities in static networks using the local approach. In other words, they first select the seed set by a specific criterion and then extend it to form the communities.

As can be seen in Table 7 and Fig. 6, our proposed method has increased the accuracy of the algorithm HLCD and has achieved better results. In cases where the NMI value is equal to one, it means that according to the formula used, the algorithm displays a perfect performance and the detection has been done correctly.

Considering that modern networks have a large number of nodes, we compared our proposed method with three large scale social networks with 50,000, 75,000, and 100,000 nodes. Since some of the methods mentioned above couldn’t show any results on large networks, we compared results of our proposed method with new methods which are compatible with large networks (Tables 8 and 9).

4.4.2 Comparing community detection algorithms in dynamic networks

The metrics used in this paper are NMI [50] and modularity Q [51]. Modularity shows how good the detected communities are, and NMI evaluates the community detection performance [47]. As mentioned before, in the proposed method, we first estimate the similarity of each two consecutive snapshots using the deep neural network and then decide whether we can use the RR sets obtained in the previous snapshot or we should update them. We compared NMI and modularity Q of the proposed method to FACETNET [36], DYNMOGA [12], GLmetric [1], DOCET [47], and MDOA [49] algorithms on different dynamic datasets.

In the SYNFIX dataset with mixing parameter = 3 (Figs. 7, 8 and Tables 10 and 11), Reversely Reachable sets were created for the first snapshot and were updated in the third and the seventh snapshots. In other words, by only building the RR sets once and updating it twice, we were able to detect communities of the 10 snapshots of the SYNFIX dataset.

In the SYNFIX dataset with mixing parameter = 5 (Figs. 9, 10 and Tables 12 and 13), Reversely Reachable sets were made for the first snapshot and updated for the 6th and 8th snapshots. As a result, by only building the RR sets once and updating them twice, we managed to detect the communities of 10 snapshots.

In the SYNVAR dataset with mixing parameter = 3 (Figs. 11, 12 and Tables 14 and 15), we created the Reversely Reachable sets for the first snapshot and updated them in the 4th, 5th, and 9th snapshots. As a result, by only building the RR sets once and updating them three times, we were able to detect the communities of 10 snapshots.

Table 7 The NMI results of community detection methods on static networks

	Proposed method	Docet	LGIEM	InfoNode	HLCD	Local_T	LICOD	ENBC
Karate	1	0.90	1	0.99	0.83	0.83	0.67	0.83
Football	0.95	0.89	0.66	0.65	0.90	0.86	0.90	0.85
LFR_1	1	1	–	–	1	0.99	0.99	0.99
LFR_2	1	0.99	0.68	–	1	0.77	0.99	0.99
LFR_3	1	1	–	–	1	0.82	–	0.99

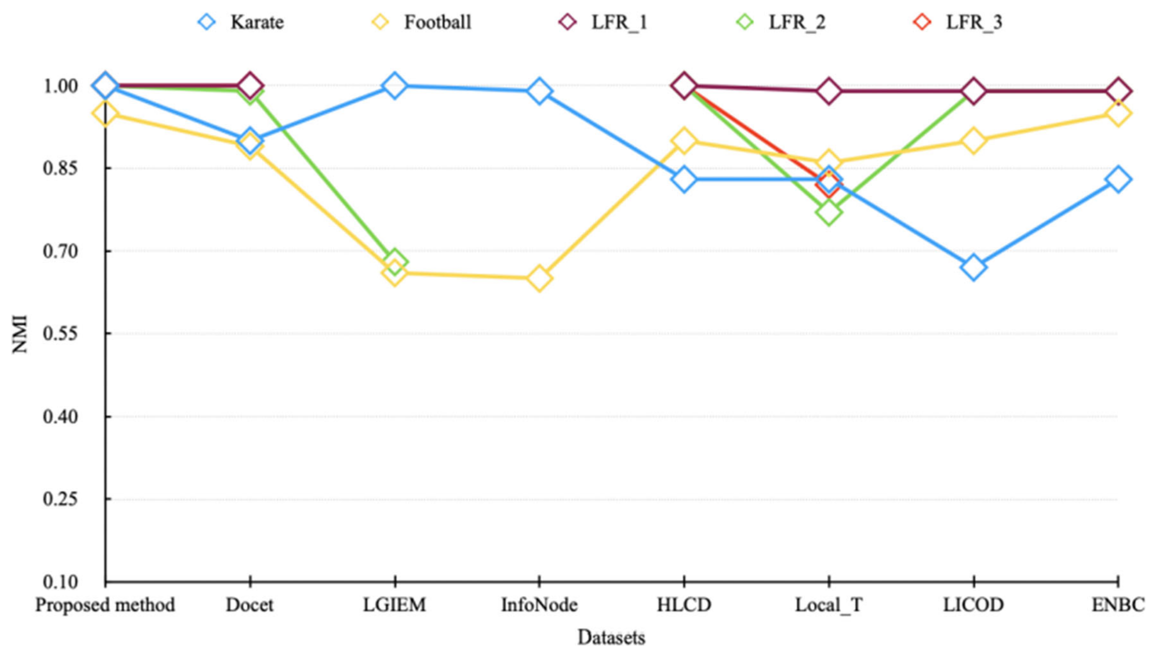

Fig. 6 Plot of NMI results of community detection methods in static networks

Table 8 NMI values of methods

	NMI				
	Proposed method	simCMR	Leiden	LBLD	LDMD
N50	0.614752391	0.164057393	0.461887358	0.001173144	0.203948
N75	0.608532753	0.193076005	0.496224825	0.00027362	0.2424775
N100	0.627348912	0.203729461	0.51240647	0.000487791	0.2581904

Table 9 Modularity values of methods

	Q				
	Proposed method	simCMR	Leiden	LBLD	LDMD
N50	0.695238	0.0	0.20151	0.0017651	0.616377
N75	0.713624	0.0	0.214752	0.0	0.625672
N100	0.708257	0.0	0.207781	0.0	0.600935

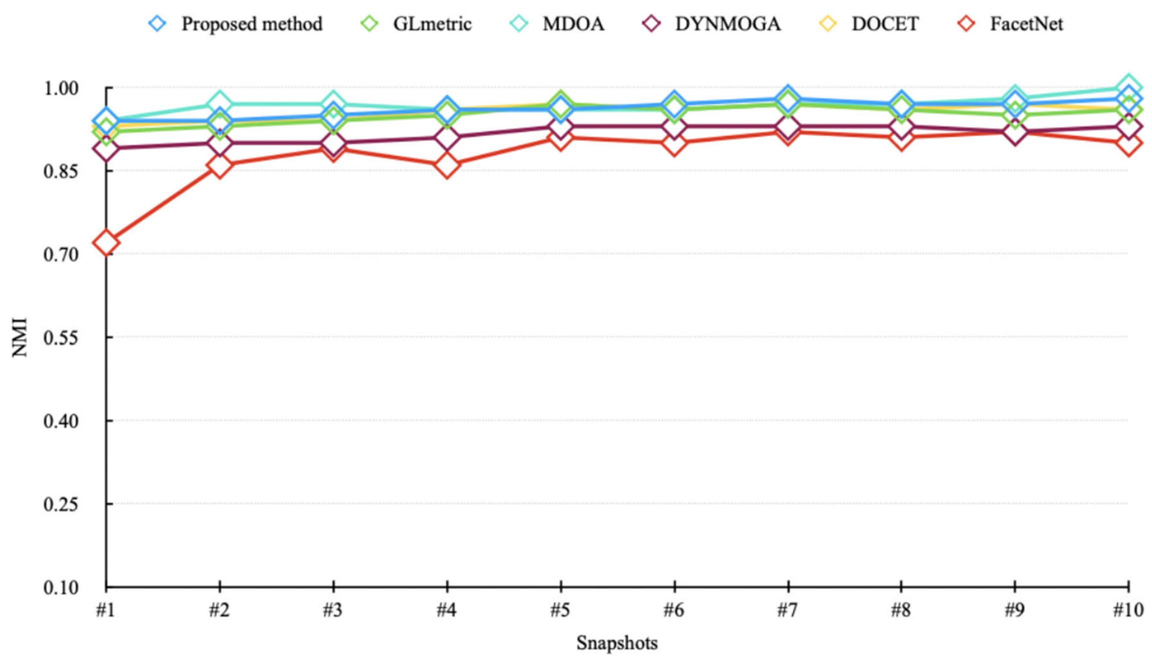

Fig. 7 Plot of NMI results on dynamic SYNFIX with the mixing parameter = 3 dataset

In the SYNVAR dataset with mixing parameter = 5 (Figs. 13, 14 and Tables 16 and 17), Reversely Reachable sets were made for the first snapshot and updated in the 4th, 5th and 8th snapshots. As a result, by only building the RR sets once and updating them three times, we were able to detect the communities of 10 snapshots.

We also applied the proposed method to some famous synthetic dynamic datasets named: Birth/Death, Expand, Hide and Merge/Split. These include 1000 nodes in which specific changes of communities over time are modeled in 5 snapshots (Tables 18 and 19). The name of these datasets is based on the events that are modeled in them. The NMI – Q– values of results are in the following tables.

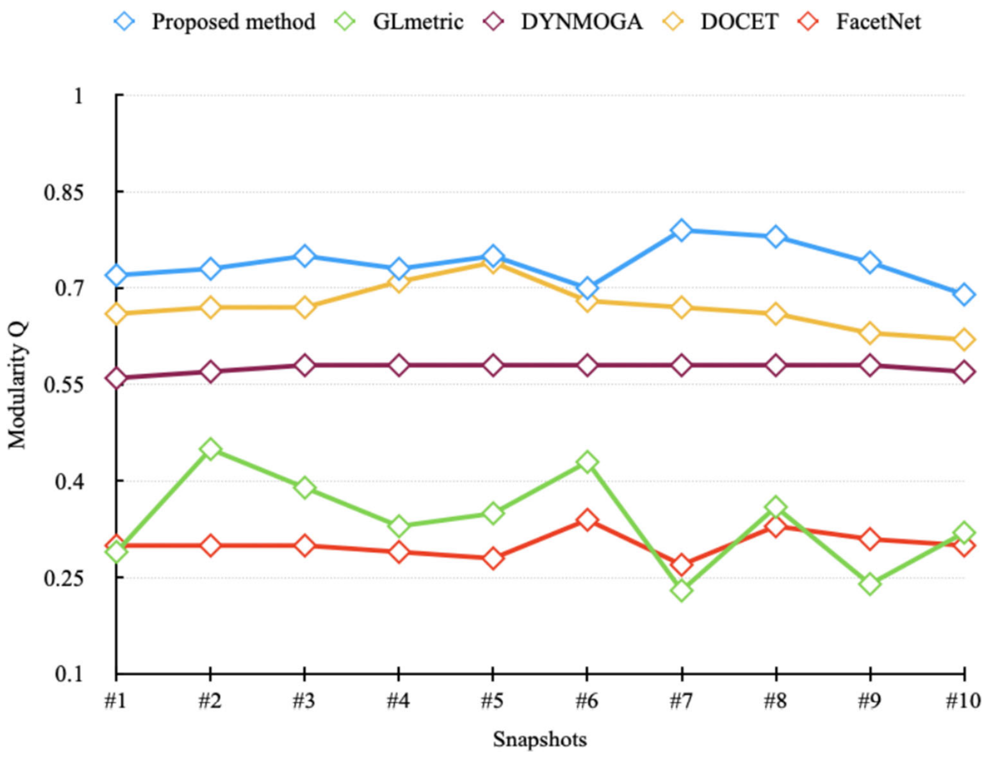

Fig. 8 Plot of Q at different time steps of the SYNFIX network with the mixing parameter = 3

Table 10 NMI at different time steps of the SYNFIX network with the mixing parameter = 3

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Proposed Method	0.94	0.97	0.97	0.96	0.96	0.97	0.98	0.97	0.98	1
GLmetric	0.92	0.93	0.94	0.95	0.97	0.96	0.97	0.96	0.95	0.96
MDOA	0.94	0.97	0.97	0.96	0.96	0.96	0.97	0.97	0.98	1
DYNMOGA	0.89	0.90	0.90	0.91	0.93	0.93	0.93	0.93	0.92	0.93
DOCET	0.83	0.94	0.94	0.96	0.97	0.96	0.97	0.96	0.97	0.96
FacetNet	0.72	0.86	0.89	0.86	0.91	0.90	0.92	0.91	0.92	0.90

Table 11 Q at different time steps of the SYNFIX network with the mixing parameter = 3

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Proposed Method	0.72	0.73	0.75	0.73	0.75	0.70	0.79	0.78	0.74	0.69
GLmetric	0.29	0.45	0.39	0.33	0.35	0.43	0.23	0.36	0.24	0.32
DYNMOGA	0.56	0.57	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.57
DOCET	0.66	0.67	0.67	0.71	0.74	0.68	0.67	0.66	0.63	0.62
FacetNet	0.30	0.30	0.30	0.29	0.28	0.34	0.27	0.33	0.31	0.30

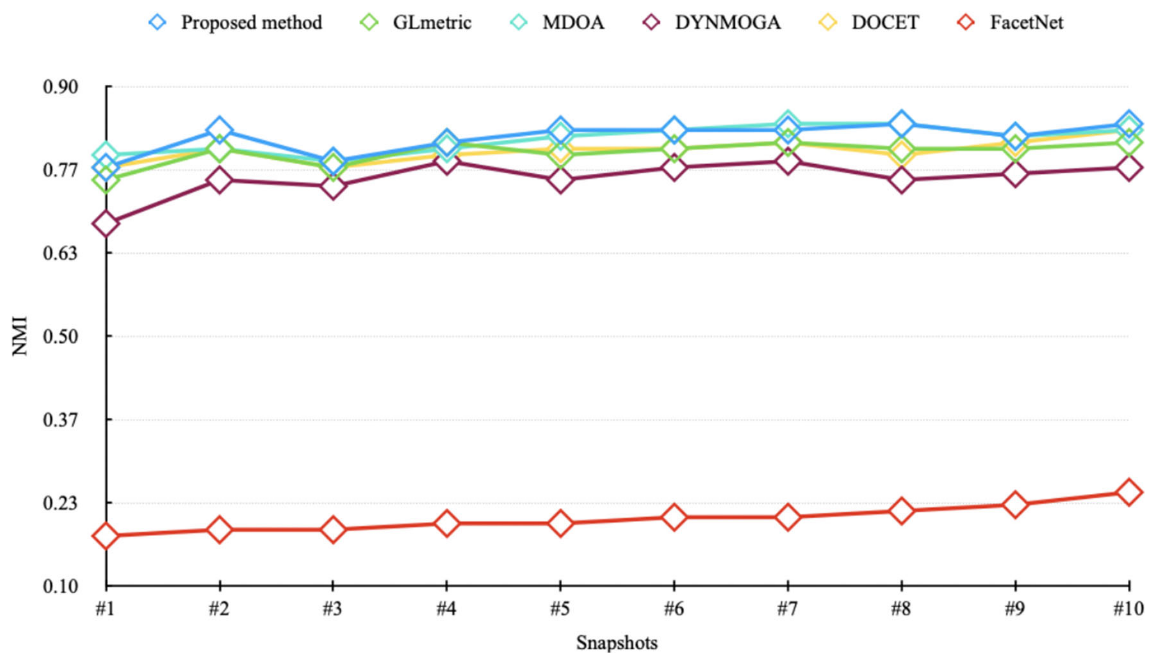

Fig. 9 Plot of NMI results on dynamic SYNFIX with the mixing parameter = 5 dataset

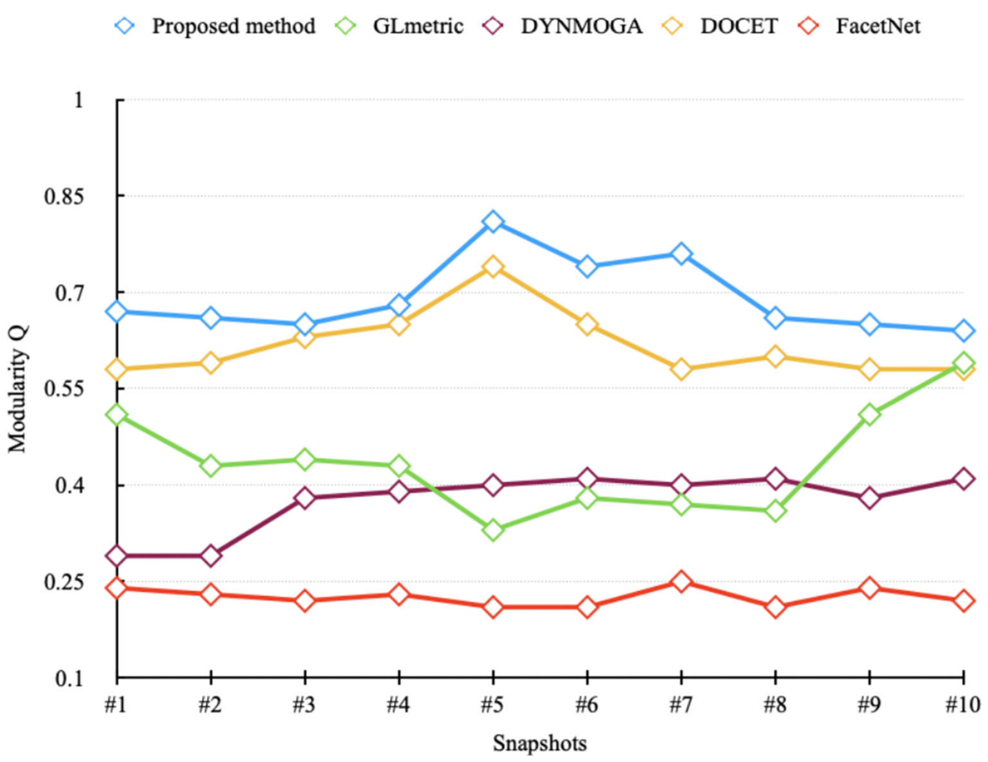

Fig. 10 Plot of Q at different time steps of the SYNFIX network with the mixing parameter = 5

Table 12 NMI at different time steps of the SYNFIX network with the mixing parameter = 5

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Proposed Method	0.79	0.83	0.78	0.81	0.83	0.83	0.84	0.84	0.82	0.84
GLmetric	0.75	0.80	0.77	0.81	0.82	0.80	0.81	0.80	0.80	0.81
MDOA	0.79	0.80	0.78	0.80	0.82	0.83	0.84	0.84	0.82	0.83
DYNMOGA	0.68	0.75	0.74	0.78	0.75	0.77	0.78	0.75	0.76	0.77
DOCET	0.77	0.80	0.77	0.76	0.80	0.80	0.81	0.79	0.81	0.83
FacetNet	0.18	0.19	0.19	0.20	0.20	0.21	0.21	0.22	0.23	0.25

Table 13 Q at different time steps of the SYNFIX network with the mixing parameter = 5

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Proposed Method	0.67	0.66	0.65	0.68	0.81	0.74	0.76	0.66	0.65	0.64
GLmetric	0.51	0.43	0.44	0.43	0.33	0.38	0.37	0.36	0.51	0.59
DYNMOGA	0.29	0.29	0.38	0.39	0.40	0.41	0.40	0.41	0.38	0.41
DOCET	0.58	0.59	0.63	0.65	0.74	0.65	0.58	0.60	0.58	0.58
FacetNet	0.24	0.23	0.22	0.23	0.21	0.21	0.27	0.21	0.24	0.22

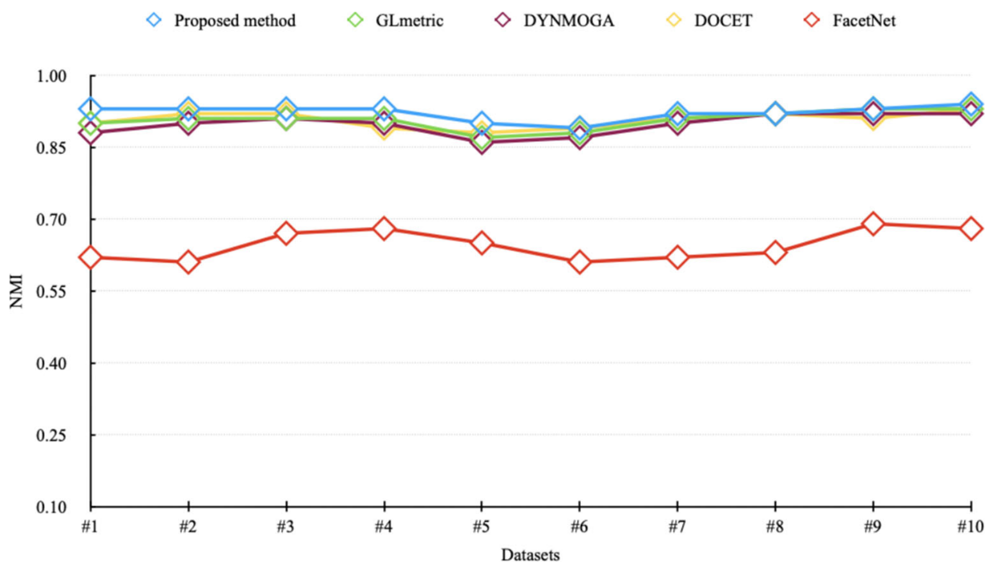

Fig. 11 Plot of NMI at different time steps of the SYNVAR network with the mixing parameter = 3

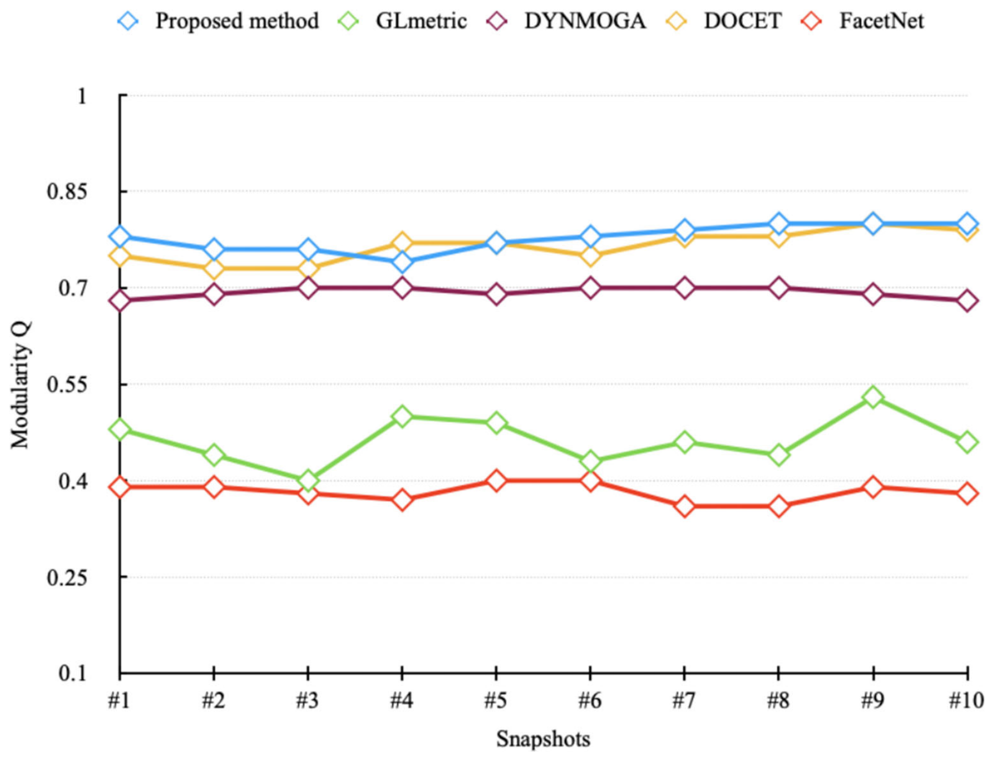

Fig. 12 Plot of Q at different time steps of the SYNVAR network with the mixing parameter= 3

Table 14 NMI at different time steps of the SYNVAR network with the mixing parameter = 3

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Proposed Method	0.93	0.93	0.93	0.93	0.90	0.89	0.92	0.92	0.93	0.94
GLmetric	0.90	0.91	0.91	0.91	0.87	0.88	0.91	0.92	0.93	0.93
DYNMOGA	0.88	0.90	0.91	0.90	0.96	0.87	0.90	0.92	0.92	0.92
DOCET	0.90	0.92	0.92	0.89	0.88	0.89	0.91	0.92	0.91	0.93
FacetNet	0.63	0.61	0.67	0.68	0.65	0.61	0.62	0.63	0.69	0.68

Table 15 Q at different time steps of the SYNVAR network with the mixing parameter = 3

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Proposed Method	0.78	0.76	0.76	0.74	0.77	0.78	0.79	0.70	0.80	0.80
GLmetric	0.48	0.44	0.40	0.50	0.49	0.43	0.46	0.44	0.53	0.46
DYNMOGA	0.68	0.69	0.70	0.70	0.69	0.70	0.70	0.70	0.69	0.68
DOCET	0.75	0.73	0.73	0.77	0.77	0.75	0.78	0.78	0.80	0.79
FacetNet	0.39	0.39	0.38	0.37	0.40	0.40	0.36	0.36	0.39	0.38

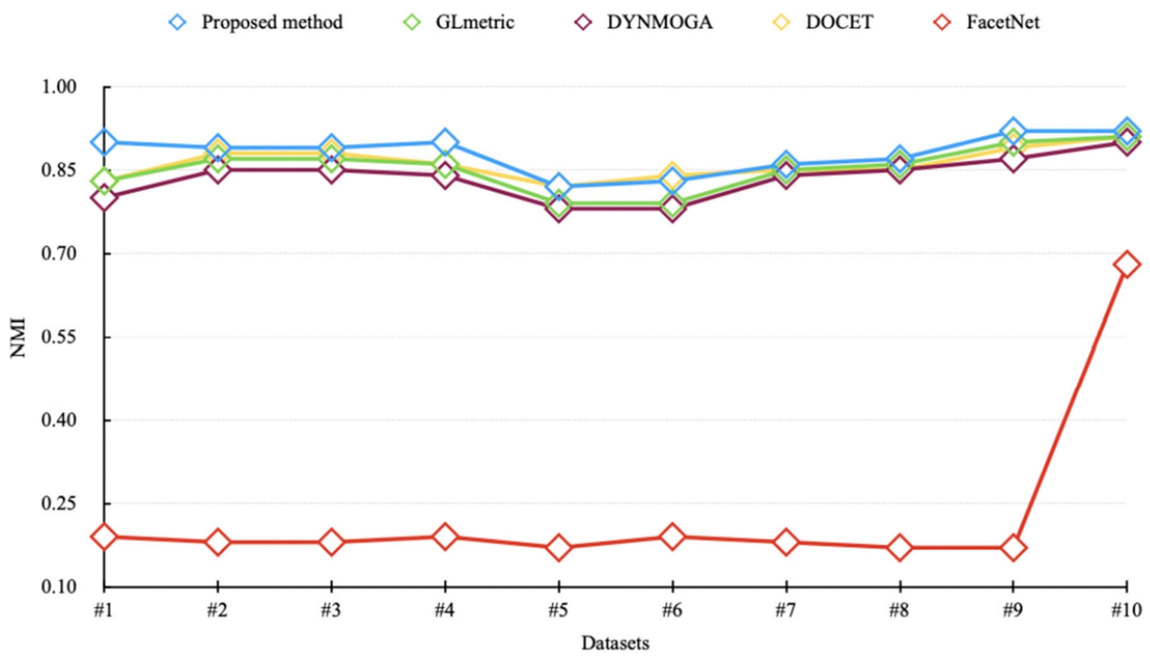

Fig. 13 Plot of NMI at different time steps of the SYNVAR network with the mixing parameter = 5

Fig. 14 Plot of Q at different time steps of the SYNVAR network with the mixing parameter = 5

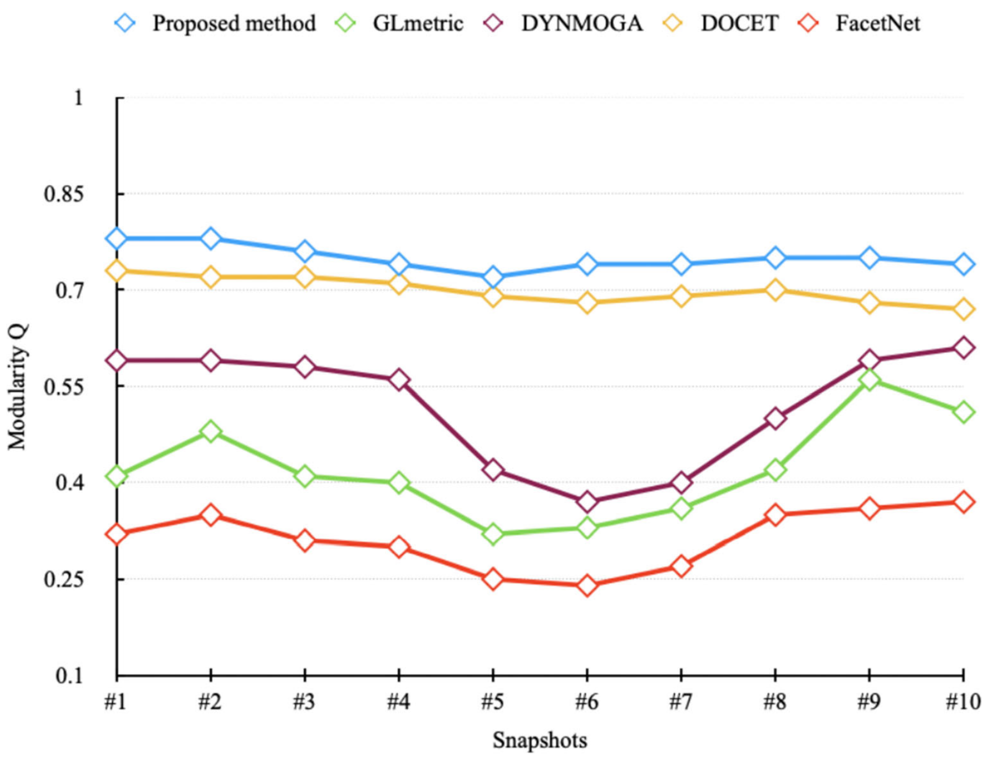

Table 16 NMI at different time steps of the SYNVAR network with the mixing parameter = 5

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Proposed Method	0.90	0.89	0.89	0.90	0.82	0.83	0.86	0.87	0.92	0.92
GLmetric	0.83	0.87	0.87	0.86	0.79	0.79	0.85	0.86	0.90	0.91
DYNMOGA	0.80	0.85	0.85	0.84	0.78	0.78	0.84	0.85	0.87	0.90
DOCET	0.83	0.88	0.88	0.86	0.82	0.84	0.85	0.85	0.89	0.91
FacetNet	0.19	0.18	0.18	0.19	0.17	0.19	0.18	0.17	0.37	0.18

Table 17 Q at different time steps of the SYNVAR network with the mixing parameter = 5

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Proposed Method	0.78	0.78	0.74	0.72	0.74	0.74	0.75	0.75	0.75	0.74
GLmetric	0.41	0.48	0.41	0.40	0.32	0.33	0.36	0.42	0.56	0.51
DYNMOGA	0.59	0.59	0.58	0.56	0.42	0.37	0.40	0.50	0.59	0.61
DOCET	0.73	0.72	0.72	0.71	0.69	0.68	0.69	0.70	0.68	0.67
FacetNet	0.32	0.35	0.1831	0.30	0.25	0.24	0.27	0.35	0.36	0.37

Table 18 the NMI values of the proposed method on some synthetic dynamic datasets

	#1	#2	#3	#4	#5
Birth/Death	0.55388	0.60942	0.644059	0.665857	0.678459
Expand	0.648238	0.682482	0.685829	0.684937	0.881563
Hide	0.666853	0.601622	0.598658	0.603874	0.615580
Merge/Split	0.595148	0.582653	0.569135	0.524036	0.603720

Table 19 the Q values of the proposed method on some synthetic dynamic datasets

	#1	#2	#3	#4	#5
Birth/Death	0.814393	0.821583	0.826620	0.816604	0.810907
Expand	0.921583	0.913534	0.919675	0.878913	0.920352
Hide	0.967412	0.962886	0.954216	0.955331	0.950586
Merge/Split	0.9191543	0.951727	0.951823	0.936478	0.919791

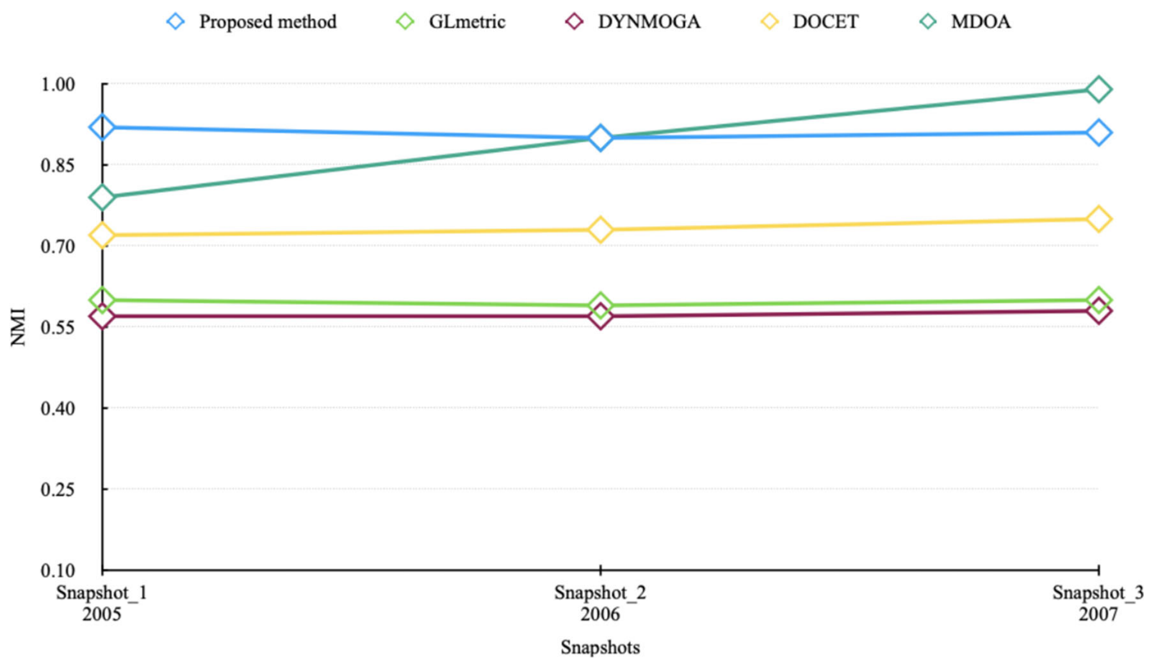

Fig. 15 Plot of NMI results on dynamic football dataset

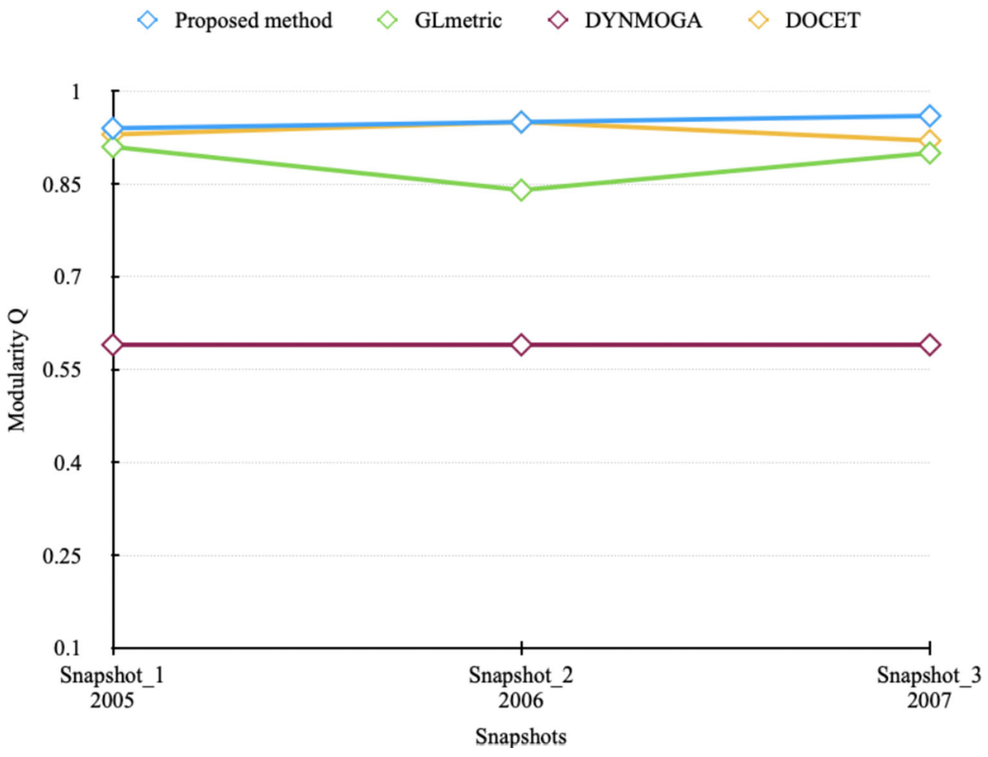

Fig. 16 Plot of Q at different time steps of the dynamic football data set

In the dynamic football dataset (Figs. 15, 16 and Tables 20 and 21), instead of creating the Reversely Reachable sets three times for each of the three snapshots, we only created it once for the first snapshot. We then compared the similarity of the first snapshot with the other two snapshots using the deep neural network. Since their similarity was higher than the threshold, we did not need to update new RR sets for other snapshots, and the first snapshot's RR sets were used to detect the communities in second and third snapshots.

In the Facebook dataset (Figs. 17, 18 and Tables 22 and 23), Reversely Reachable sets were made for the first snapshot and updated for the 4h snapshot. As a result, by only building the RR sets once and updating them one time, we were able to detect communities of 6 snapshots.

4.4.3 Computational complexity analysis

In this section, we analyze the computational complexity of the proposed method. In the first phase of the proposed method, we used the technique proposed in reference [8] to solve the concern of Computational complexity. The HLCD algorithm is also implemented in three stages, according to the author's analysis; the complexity of the first stage, $O(Kn^2)$ where k is the maximum degree of the graph and n is the number of nodes. In the second step, the maximum complexity is $O(c^2)$, where c is the number of detected communities. In the third step of the HLCD, the maximum complexity is $O(nc)$, where n is the number of nodes and c is the number of communities. In the development of the algorithm to the dynamic network, only the training phase of the Convolutional Auto Encoder network takes a little time, and after the first training, the network compares the

Table 20 NMI at different time steps of the dynamic football dataset

	Snapshot_1 2005	Snapshot_2 2006	Snapshot_3 2007
Proposed method	0.92	0.90	0.91
GLmetric	0.60	0.59	0.60
DYNMOGA	0.57	0.57	0.58
DOCET	0.72	0.73	0.75
FacetNet	0.79	0.90	0.99

Table 21 Q at different time steps of the dynamic football dataset

	Snapshot_1 2005	Snapshot_2 2006	Snapshot_3 2007
Proposed method	0.9	0.95	0.96
GLmetric	0.91	0.84	0.90
DYNMOGA	0.59	0.59	0.59
DOCET	0.93	0.95	0.92

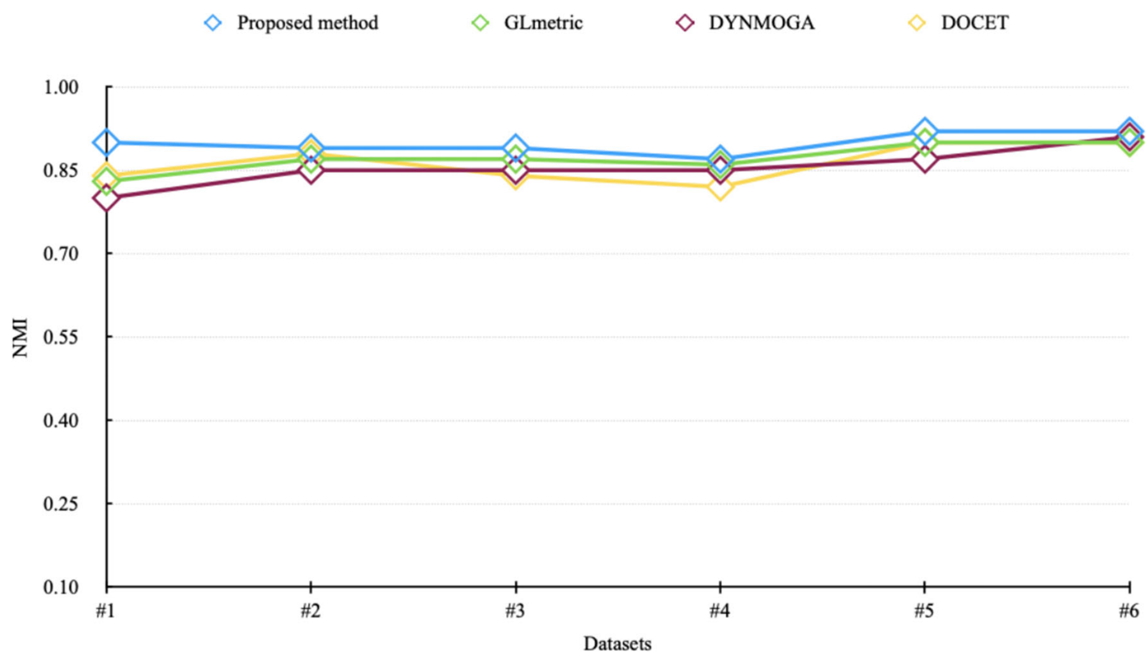

Fig. 17 Plot of NMI at different time steps of the dynamic facebook network

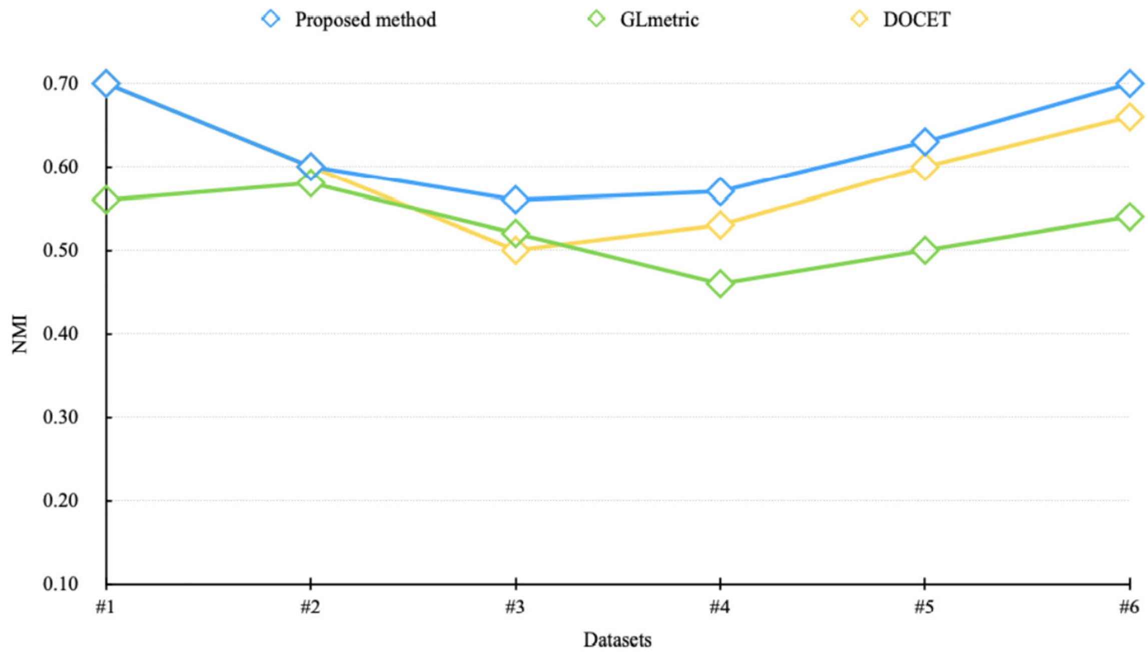

Fig. 18 Plot of Q at different time steps of the dynamic facebook data set

Table 22 The NMI results on dynamic facebook network

	#1	#2	#3	#4	#5	#6
Proposed Method	0.90	0.89	0.89	0.87	0.92	0.92
GLmetric	0.83	0.87	0.87	0.86	0.90	0.91
DYNMOGA	0.80	0.85	0.85	0.85	0.87	0.90
FacetNet	0.19	0.18	0.18	0.17	0.17	0.18

Table 23 Q at different time steps of the facebook dynamic network

	#1	#2	#3	#4	#5	#6
Proposed Method	0.70	0.60	0.56	0.57	0.63	0.70
GLmetric	0.56	0.58	0.52	0.46	0.50	0.54
DOCET	0.70	0.60	0.50	0.53	0.60	0.66

two snapshots in a fraction of a second, which is not a time-consuming process. It can be seen that there is no concern in the computational complexity with the techniques used in this method.

5 Conclusion

The main purpose of this article was to provide a new method to more accurately solve the problems of community detection in social networks. We used the concept of influential nodes from the influence maximization area to find the optimal initial nodes, which are a critical phase in the algorithms of the local approach in the community detection area. We tried to prove that the proposed approach was efficient in algorithms of the local approach to detect communities, and the experimental results confirmed this in both static and dynamic networks. From another perspective, most community detection algorithms only use either global or local information to detect communities, but in the proposed method, we used both local and global information, taking advantage of both. Yet, extending this approach to dynamic networks was time-consuming. We tackled this problem by introducing a new method, a deep neural network that was designed to calculate the similarity of two snapshots, which enabled us to decide on the RR sets and Initial nodes of each snapshot.

We can define a new cost function for the neural network for future works. We can also use other influence maximization algorithms or even other concepts of other areas in social network analysis and compare them with our method. We can use this method as an extension for other community detection algorithms that use seed set to detect communities more accurately.

References

1. Samie ME, Hamzeh A (2017) Community detection in dynamic social networks: a local evolutionary approach. J Inf Sci 43(5):615–634
2. Aggarwal CC (2011) An introduction to social network data analytics. In: Social network data analytics. Springer, pp 1–15
3. Wang C, Tang W, Sun B, Fang J, Wang Y (2015) Review on community detection algorithms in social networks. In: 2015 IEEE international conference on progress in informatics and computing (PIC). IEEE, pp 551–555
4. Kaiser M (2011) A tutorial in connectome analysis: topological and spatial features of brain networks. Neuroimage 57(3):892–907
5. Choudhury D, Paul A (2013) Community detection in social networks: an overview. Int J Res Eng Technol 2:6–13
6. Papadopoulos S, Kompatsiaris Y, Vakali A, Spyridonos P (2012) Community detection in social media. Data mining and knowledge discovery 24(3):515–554
7. Girvan M, Newman ME (2002) Community structure in social and biological networks. Proceedings of the National Academy of sciences 99(12):7821–7826
8. Tang Y, Xiao X, Shi Y (2014) Influence maximization: Near-optimal time complexity meets practical efficiency. In: Proceedings of the 2014 ACM SIGMOD international conference on management of data, pp 75–86
9. Newman ME, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113
10. Pizzuti C (2008) Ga-net: a genetic algorithm for community detection in social networks. In: International conference on parallel problem solving from nature. Springer, pp 1081–1090
11. Takaffoli M, Fagnan J, Sangi F, Zaïane OR (2011) Tracking changes in dynamic information networks. In: 2011 International conference on computational aspects of social networks (CASoN). IEEE, pp 94–101
12. Folino F, Pizzuti C (2013) An evolutionary multiobjective approach for community discovery in dynamic networks. IEEE Trans Knowl Data Eng 26(8):1838–1852
13. Tabarzad MA, Hamzeh A (2017) A heuristic local community detection method (hlcd). Appl Intell 46(1):62–78
14. Papadopoulos S, Vakali A, Kompatsiaris Y (2011) Community detection in collaborative tagging systems. In: Community-built databases. Springer, pp 107–131
15. Hastie T, Tibshirani R, Friedman J (2009) Overview of supervised learning. the elements of statistical learning
16. Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 76(3):036106
17. Wu Y-J, Huang H, Hao Z-F, Chen F (2012) Local community detection using link similarity. J Comput Sci Technol 27(6):1261–1268
18. Gregory S (2010) Finding overlapping communities in networks by label propagation. New J Phys 12(10):103018
19. Derenyi I, Palla G, Vicsek T (2005) Clique percolation in random networks. Phys Rev Lett 94(16):160202
20. Clauset A (2005) Finding local community structure in networks. Phys Rev E 72(2):026132
21. Whang JJ, Gleich DF, Dhillon IS (2013) Overlapping community detection using seed set expansion. In: Proceedings of the 22nd ACM international conference on information & knowledge management, pp 2099–2108
22. Dhillon IS, Guan Y, Kulis B (2007) Weighted graph cuts without eigenvectors a multilevel approach. IEEE Trans Pattern Anal Mach Intell 29(11):1944–1957
23. Kloumann IM, Kleinberg JM (2014) Community membership identification from small seed sets. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1366–1375

24. Yakoubi Z, Kanawati R (2014) A leader-driven algorithm for community detection in complex networks. Vietnam J Comput Sci 1:241–256
25. Fagnan J, Zaïane O, Barbosa D (2014) Using triads to identify local community structure in social networks. In: 2014 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM 2014), IEEE, pp 108–112
26. Biswas A, Biswas B (2015) Investigating community structure in perspective of ego network. Expert Syst Appl 42(20):6913–6934
27. Jiang F, Jin S, Wu Y, Xu J (2014) A uniform framework for community detection via influence maximization in social networks. In: 2014 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM 2014). IEEE, pp 27–32
28. Ma T, Liu Q, Cao J, Tian Y, Al-Dhelaan A, Al-Rodhaan M (2020) Lgiem: global and local node influence based community detection. Futur Gener Comput Syst 105:533–546
29. Guo K, He L, Chen Y, Guo W, Zheng J (2020) A local community detection algorithm based on internal force between nodes. Appl Intell 50(2):328–340
30. Traag VA, Waltman L, Van Eck NJ (2019) From louvain to leiden: guaranteeing well-connected communities. Sci Rep 9(1):1–12
31. Bouyer A, Roghani H (2020) Lsmd: a fast and robust local community detection starting from low degree nodes in social networks. Futur Gener Comput Syst 113:41–57
32. Roghani H, Bouyer A (2022) A fast local balanced label diffusion algorithm for community detection in social networks. IEEE Transactions on Knowledge and Data Engineering
33. Tunali V (2021) Large-scale network community detection using similarity-guided merge and refinement. IEEE Access 9:78 538–78 552
34. Bansal S, Bhowmick S, Paymal P (2011) Fast community detection for dynamic complex networks. In: Complex networks. Springer, pp 196–207
35. Berger-Wolf TY, Saia J (2006) A framework for analysis of dynamic social networks. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, pp 523–528
36. Lin Y-R, Chi Y, Zhu S, Sundaram H, Tseng BL (2008) Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In: Proceedings of the 17th international conference on world wide web, pp 685–694
37. Palla G, Barabási A-L, Vicsek T (2007) Quantifying social group evolution. Nature 446(7136):664–667
38. Greene D, Doyle D, Cunningham P (2010) Tracking the evolution of communities in dynamic social networks. In: 2010 international conference on advances in social networks analysis and mining. IEEE, pp 176–183
39. Jaccard P (1912) The distribution of the flora in the alpine zone. 1. New Phytol 11(2):37–50
40. DiTursi DJ, Ghosh G, Bogdanov P (2017) Local community detection in dynamic networks. In: 2017 IEEE international conference on data mining (ICDM). IEEE, pp 847–852
41. Samie ME, Hamzeh A (2018) Change-aware community detection approach for dynamic social networks. Appl Intell 48(1):78–96
42. Rossetti G (2017) graph benchmark handling community dynamics. Journal of Complex Networks 5(6):893–912
43. Takaffoli M, Rabbany R, Zaïane OR (2013) Incremental local community identification in dynamic social networks. In: 2013 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM 2013). IEEE, pp 90–94
44. Liu W, Suzumura T, Chen L, Hu G (2017) A generalized incremental bottom-up community detection framework for highly dynamic graphs. In: 2017 IEEE international conference on big data (Big Data). IEEE, pp 3342–3351
45. Van Laarhoven T, Marchiori E (2016) Local network community detection with continuous optimization of conductance and weighted kernel k-means. The Journal of Machine Learning Research 17(1):5148–5175
46. Takaffoli M, Sangi F, Fagnan J, Zaiane O (2011) Modec—modeling and detecting evolutions of communities. Proceedings of the International AAAI Conference on Web and Social Media 5(1):626–629
47. Wang Z, Li Z, Yuan G, Sun Y, Rui X, Xiang X (2018) Tracking the evolution of overlapping communities in dynamic social networks. Knowl-Based Syst 157:81–97
48. Luo W, Zhang D, Jiang H, Ni L, Hu Y (2018) Local community detection with the dynamic membership function. IEEE Trans Fuzzy Syst 26(5):3136–3150
49. Wan X, Zuo X, Song F (2020) Solving dynamic overlapping community detection problem by a multiobjective evolutionary algorithm based on decomposition. Swarm Evol Comput 54:100668
50. Ahn Y-Y, Bagrow JP, Lehmann S (2010) Link communities reveal multiscale complexity in networks. nature 466(7307):761–764
51. Shen H, Cheng X, Cai K, Hu M-B (2009) Detect overlapping and hierarchical community structure in networks. Physica A 388(8):1706–1712
52. Luo W, Lu N, Ni L, Zhu W, Ding W (2020) Local community detection by the nearest nodes with greater centrality. Inform Sci 517:377–392
53. Yang Y, Wang Z, Pei J, Chen E (2017) Tracking influential individuals in dynamic networks. IEEE Trans Knowl Data Eng 29(11):2615–2628
54. Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. Phys Rev E 78(4):046110
55. Danon L, Diaz-Guilera A, Duch J, Arenas A (2005) Comparing community structure identification. J Stat Mech: Theory Exp 2005(09):P09008
56. Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33(4):452–473
57. Leskovec J, Mcauley J (2012) Learning to discover social circles in ego networks. Advances in neural information processing systems, vol 25