Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

Research paper

A Node-Centric Approach for Community Detection in Dynamic Networks

M. Sabzekar [1,], S. Baradaran Nezhad [2], M. Khazaeipoor [2]*

[1]*Department of Computer Engineering, Birjand University of Technology, Birjand, Iran.*

[2]*Department of Computer Engineering, Birjand Branch, Islamic Azad University, Birjand, Iran.*

Article Info

Article History:
Received 25 September 2023
Reviewed 13 October 2023
Revised 03 January 2023
Accepted 15 January 2023

Keywords:
Social networks
Dynamic networks
Community detection
Node influence
Overlapping communities.

*Corresponding Author's Email Address:
sabzekar@birjandut.ac.ir

Abstract

Background and Objectives: Nowadays, social networks are recognized as significant sources of information exchange. Consequently, many organizations have chosen social networks as essential tools for marketing and brand management. Communities are essential structures that can enhance the performance of social networks by grouping nodes and analyzing the information derived from them. This subject becomes more important with the increase in information volume and the complexity of relationships in networks. The goal of community identification is to find subgraphs that are densely connected internally but loosely connected externally.

Methods: While community detection has mostly been studied in static networks in the past, this paper focuses on dynamic networks and the influence of central nodes in forming communities. In the proposed algorithm, the network is captured through multiple snapshots. The initial snapshot calculates the influence of each node. Then, by selecting *k* nodes with higher influence, network communities are formed, and other nodes belong to the community with the most common edges. In the second step, after receiving the next snapshot, communities are updated. Then, *k* nodes with higher influence are selected, and their associated community is created if needed. If the previous community centers are not among the newly selected *k* nodes, the community is dissolved, and the nodes within it belong to other communities.

Results: Based on the results obtained, the proposed algorithm has managed to achieve better results in most cases compared to the compared algorithms, especially in terms of modularity metrics. The reason behind this success could be attributed to the utilization of influential nodes in community formation.

Conclusion: Drawing from the outcomes attained, the suggested algorithm has effectively outperformed the contrasted algorithms in a majority of instances, particularly concerning metrics related to modularity. This accomplishment can potentially be ascribed to the incorporation of influential nodes during the process of community formation.

Introduction

Online social networks have become the most popular interactive medium on the internet due to the possibility of connecting thousands of individuals via the internet [1]. In these networks, users can establish various forms of social relationships such as liking, following, trusting, and more, with each other [2]. These relationships can pave the way for new forms of communication and

sharing emotions and experiences [3], [4]. Nowadays, many organizations view social networks as primary tools for marketing and product management [5]. Each network can be considered as a graph. In this graph, nodes represent individuals, and edges between them represent friendship, interaction, or connection [6]. One of the most important characteristics of networks is the structure of communities within them. Identifying community structure is a significant and challenging topic in social networks, with the aim of finding subgraphs that are internally dense but externally sparsely connected [5].

One of the crucial challenges in community detection is the issue of overlapping communities [7]. The shared membership of some group members is referred to as community overlap in networks [8]. Considering such a concept, each node can belong to multiple groups based on its attributes. When dealing with large-scale data networks, identifying communities with overlaps presents a challenging and computationally complex problem. As a result, many research efforts attempt to minimize the consideration of overlaps [6].

Various studies have been addressed thr community detection problem. However, online social networks possess a dynamic nature, constantly changing and evolving. A dynamic network (DN) can be understood as a network that changes and evolves over time. These changes can be summarized in four types of operations: node creation, node deletion, edge creation, and edge deletion. In the past, community detection was primarily studied for static networks, with the dynamic nature of networks often overlooked. Nowadays, due to the substantial growth in network size and the evolving nature of network structures, researchers have shifted their focus to DNs [9]. Community detection in DNs is essential for crucial applications such as social network analysis [10].

In many social networks, users can express their opinions and feelings about specific products, services, or topics based on their experiences and share them with others. These opinions and feelings are expressed by real users and customers and are observed by their friends or followers. If an opinion is presented by a familiar individual such as a friend or a celebrity, it holds a more significant influence on the user's decision regarding that topic. This phenomenon is considered a new concept in social networks. In social networks, nodes that possess a higher capacity for disseminating information hold more significance and are known as influential nodes or leaders. Identifying influential nodes or leaders in a network can be perceived as ranking nodes in terms of importance within the network [11]. Despite studies indicating the correlation between the behavior of a node and the behavior of its neighboring nodes in the network [12], rarely have the behavioral aspects been incorporated into the problem of community detection.

Alongside the problem of community detection in dynamic social networks, this topic can introduce various challenges such as the speed and formation of communities, dynamics within communities, and the ability to update them, which have not yet received suitable solutions.

In this paper, an attempt has been made to provide an algorithm for community detection in social networks that aims to efficiently identify existing communities in the network based on the influence of nodes on each other and taking into account the possibility of overlapping communities. Furthermore, we strive to address the issue of identifying influential nodes in dynamic networks and enhance current criteria for effectiveness in dynamic networks. Moreover, recognizing that real-world network communities often exhibit overlaps, our aim was to devise a method addressing such overlaps. Our method aims to preserve communities with minimal overlap, allowing their members to participate in multiple overlapping communities. To our knowledge, no prior method exists that combines influential node utilization for community determination in DNs while also accounting for overlapping communities. Furthermore, our proposed method accommodates various changes in dynamic networks, including node and edge additions or removals. The algorithm achieves these objectives through six distinct phases: network snapshot acquisition, influential node identification via local and global information, initialization, community expansion, evaluation and merging of communities, and finally, community updates.

Basic Concepts

A. Dynamic Networks

Social networks are one of the most common types of networks, characterized by a connected structure of entities formed for social interactions [13]. To encode networks and represent their adjacencies, adjacency matrices are also utilized. To express a network in the form of an adjacency matrix, an *n*×*n* matrix is considered, where *n* corresponds to the number of graph vertices [14]:

$$A_{i,j} = \begin{cases} 1 & if\ there\ is\ an\ edge\ between\ i\ and\ j \\ 0 & otherwise \end{cases} \quad (1)$$

It is evident that interactions among members in a network, particularly in social networks, change over time. Networks evolve through the joining or departure of members from a network and the establishment or termination of relationships. This evolution not only alters the fundamental structures of networks but also adjusts the community structures in various snapshots of the network over time [15]. Diverse approaches have treated

social networks as sets of static graphs, each representing entities and their relationships in a momentary snapshot of the network [15], [16]. In Fig. 1, an example of a snapshots of a network is presented.

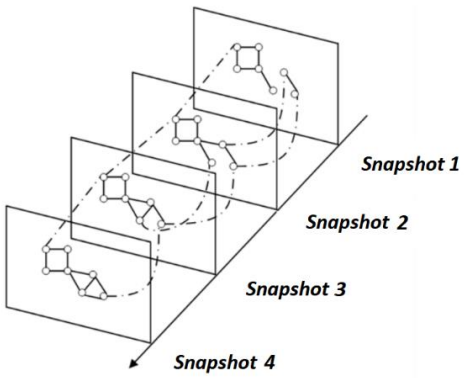

Fig. 1: An example of snapshots in a dynamic network.

B. Community Detection

In general, community detection is an unsupervised learning technique for clustering nodes, considering the network's structure, and is also a key feature that can be used to extract useful information from networks [17]. Internal communications within each community are dense, while external communications between communities are sparse [5]. Each community is a group of network nodes in such a way that the connections between nodes within the group are stronger than their connections with other network nodes [14].

With the introduction and prevalence of dynamic networks, the concept of dynamic communities has also emerged [18]. The objective of dynamic community detection is to identify a set of all the existing communities within a dynamic network in a way that the described partitions by it can have overlaps [19].

The problem of community detection in large-scale networks is computationally infeasible and NP-hard. Numerous techniques have been proposed to find optimal communities relatively quickly. Most of these techniques are based on optimizing objective functions, with modularity optimization being one of the most widely used techniques among them. Nevertheless, this problem is still NP-hard [17]. Researchers have developed various techniques that usually start from pre-defined small communities and implement an algorithm that first expands these communities, then identifies newly formed communities (in all cases), and finally uncovers overlaps between communities [20]. To measure and evaluate community quality, different metrics have been proposed. One of the most important metrics is modularity, which is widely used to assess the quality of the network community structure. Communities with high modularity have denser connections among nodes within similar communities but sparser connections with nodes in other communities. Another metric in this field is the normalized mutual information [21].

Numerous algorithms for community detection using different techniques and tools have been developed. However, due to the wide spectrum of networks, a single community detection algorithm cannot perform well and effectively in all types of networks and have good and suitable performance [22].

C. Influential Nodes in Networks

Identifying influential users in social networks has extensive applications in marketing, politics, disease control, and more [13]. Nodes in such networks have the ability to influence their neighboring nodes, and an influenced node can acquire a behavior or attribute from its neighboring nodes. Finding nodes with the highest influence has drawn the attention of social network managers and analysts. Marketing managers might be interested in identifying influential individuals and offering them discounts or free products, hoping that these individuals will encourage their friends to purchase these products [23].

The importance of a node in a network can be assessed using metrics available in graph theory. These metrics rely on the topology of the network. The impact of social networks relates to a user's ability to change the emotions, attitudes, or behaviors of other users in a network. The strength of the link between two nodes in a network depends on the overlap of their neighbors. Influential individuals are significantly associated with more groups compared to ordinary individuals. However, in online social networks, this criterion may not always be applicable for identifying influential users. Various methods for identifying influential nodes have been proposed, with the most common being [24]: degree centrality, centrality, closeness centrality, eigenvector centrality, and leader benefit. Each of these methods identifies influential nodes by examining the nodes and their connections in a certain way.

Literature Review

Numerous studies have been presented that address the problem of community detection using influential nodes. Most of them are proposed to detect communities in the static networks [5], [25], and [26]. Static research methods encompass topological analysis of complex networks, identification of key nodes or community leaders, knowledge-community discovery, and community-structure discovery.

However, real-world networks, particularly prevalent online social networks like Facebook, LinkedIn, and Twitter, are inherently dynamic and continually expanding in size and complexity. Therefore, developing effective and efficient algorithms to detect communities in dynamic networks is a formidable challenge.

An efficient dynamic community detection algorithm should adaptively and incrementally update communities based on changes in the network structure. Redundant computations need to be avoided for computational efficiency. Designing such an algorithm that maintains effectiveness similar to static algorithms solely through historical community structures and incremental changes is challenging. Additionally, there is ongoing uncertainty regarding categorizing incremental changes in dynamic networks and evaluating their influence on community structure updates, which is crucial for an effective and efficient dynamic algorithm.

Recent research has proposed several methods for detecting communities in dynamic networks, broadly categorized into three classes [21]: instant-optimal, temporal trade-off, and incremental approaches. Firstly, instant-optimal methods involve applying static algorithms independently to each network snapshot to detect communities, subsequently matching these communities with those detected in previous snapshots. As an example, the authors in [27] provide a localized modularity optimization approach where only the communities that underwent changes are examined, leaving the rest of the communities untouched. According to the claims of the paper, this algorithm exhibits greater effectiveness compared to similar algorithms.

Secondly, temporal trade-off approaches assume that communities at a certain time are influenced not only by the current network topology but also by past topology or identified partitions. These approaches strike a balance between an optimal solution at the current time and information from the past without considering future changes. As an example of the algorithms in this category, in [15], a method for detecting overlapping community structures is presented. This method considers the task of community detection as a non-negative matrix factorization problem. The proposed approach utilizes a probabilistic model to account for the dynamic nature of community structures and employs a block coordinate descent technique to solve the objective function of the matrix factorization model. This solution introduces a non-negative hidden factor to estimate gradients for faster computation. The results obtained indicate that the proposed method outperforms previous algorithms in terms of well-known evaluation metrics for evolving networks. In another work [28], the adopted approach involves a multi-objective optimization strategy. Initially, the method incorporates the probability fusion technique and employs two distinct approaches, namely neighbor diversity and neighbor crowd. These approaches facilitate the rapid and precise formation of appropriate communities. Also, the utilization of a progression metric enables the authors to identify similarities between the communities formed in two consecutive snapshots.

Finally, cross-time algorithms aim to discover communities that are relevant across the entire network evolution, where communities identified at a given time depend on both past and future network topologies. For example, in [21], a dynamic community detection algorithm based on modularity is introduced. This method aims to identify communities in dynamic networks through the repeated use of static algorithms, but in a more efficient manner. This approach is an adaptive and incremental algorithm designed to maximize incremental modularity during the update of dynamic network community structures. In this paper, the dynamic network is modeled as a sequence of gradual changes, and for each gradual change, an operation was designed to maximize modularity. An influence-based community detection in dynamic networks was proposed in [29] that formulate the problem as a combinatorial optimization problem that aims at partitioning a given social network into disjoint *m* communities. The objective is to maximize the sum of influence propagation of a social network through maximizing it within each community. In another study [30] a community detection method for dynamic networks was represented that is based on tracking of backbones and bridges. They applied the “backbones” to reflect the critical edges of communities and the “bridge” edges to describe the key connections between communities. Table 1 summarizes some of the proposed methods for addressing the problem.

In this section, an attempt has been made to introduce the latest articles in this field. Based on conducted studies, there are numerous challenges that need to be addressed as open research issues. Community detection itself is a computationally expensive and complex problem. The existence of various snapshots of the network necessitates re-computation to update or create communities. Therefore, methods are required to be as cost-effective and computationally efficient as possible. Additionally, for community detection in dynamic networks, it is necessary to compare two momentary images of the past and present. This comparison is often time-consuming and requires moderate to high computations. Having rules or methods to expedite this process can be the key to success in speeding up community detection algorithms in dynamic networks.

Furthermore, influential nodes can serve as a foundation for generating many communities. This aspect has been overlooked in many studies. Moreover, the identification and updating of influential nodes in a network is a subject that has received less attention. Ultimately, many research efforts have disregarded the issue of community overlap. In other words, attempts have been made to develop algorithms for non-overlapping communities. Yet, in today's world, networks, especially social networks, exhibit overlapping communities.

Table 1: Overview some of the reviewed studies

Method	Description	Strengths/Weaknesses
LGIEM [5]	A community detection algorithm for static networks utilizing influential nodes to find communities.	Strengths: • Using a suitable criterion to reduce overlap between communities Weaknesses: • time-consuming and computationally expensive
LPA_NI [25]	Detection of communities based on label propagation Conducting label propagation operations based on node importance and influence.	Strengths: • Utilizing node importance for label propagation can identify more appropriate communities. • The influence of nodes on each other is well modeled. • The detected communities have good quality. Weaknesses: • Calculating node importance and influence increases computational load.
NANI [26]	Utilizing group influence for identifying communities Using various metrics to determine the influence of each node on others Utilizing a method similar to hierarchical agglomerative clustering for community detection	Strengths: • High simplicity of the proposed algorithm using a wide range of metrics to determine influence and compare node similarities for community creation. Weaknesses: • No specific corrective mechanism for refining clusters and finding optimal clusters overlap.
DynaMo [21]	Proposing a community detection algorithm for dynamic networks Considering six categories of changes in communities and designing strategies for each	Strengths: • Appropriate speed for introducing changes in communities Weaknesses: • Repetitive computations and operations
PODCD [15]	Using a probabilistic method for identifying overlapping communities	Strengths: • Taking overlap into account in communities. Weaknesses: • High computational load in large networks.
D-Louvain [27]	Based on the modularity optimization algorithm (Louvain algorithm), which is one of the strongest algorithms in this field.	Strengths: • Examining changing communities instead of all communities, leading to reduced computations and increased algorithm speed. Weaknesses: • The algorithm is stochastic, resulting in unstable results. Unable to cope with overlap in communities.
MOCCD [28]	Based on characteristics fusion of dynamic social networks	Strengths: • Utilize multi-objective optimization that fuses the characteristics of dynamic network communities. • Fast convergence and high accuracy.
Sandwich [29]	formulate the problem as a combinatorial optimization problem based on sandwich approximation framework.	Strengths: • influence maximization • develop a lower bound and an upper bound of the objective function.
BBTA [30]	Introducing two concepts, backbones and bridges for edges.	Strengths: • novel incremental algorithm to detect dynamic communities based on the network change rate and changes on the backbones or bridges. Weaknesses: • Stability and robustness

The Proposed Method

The proposed method consists of six main phases, and in the following, we will explain each phase. Fig. 2 illustrates the overall phases of the proposed method. The details of each phase will be discussed at the following of the section.

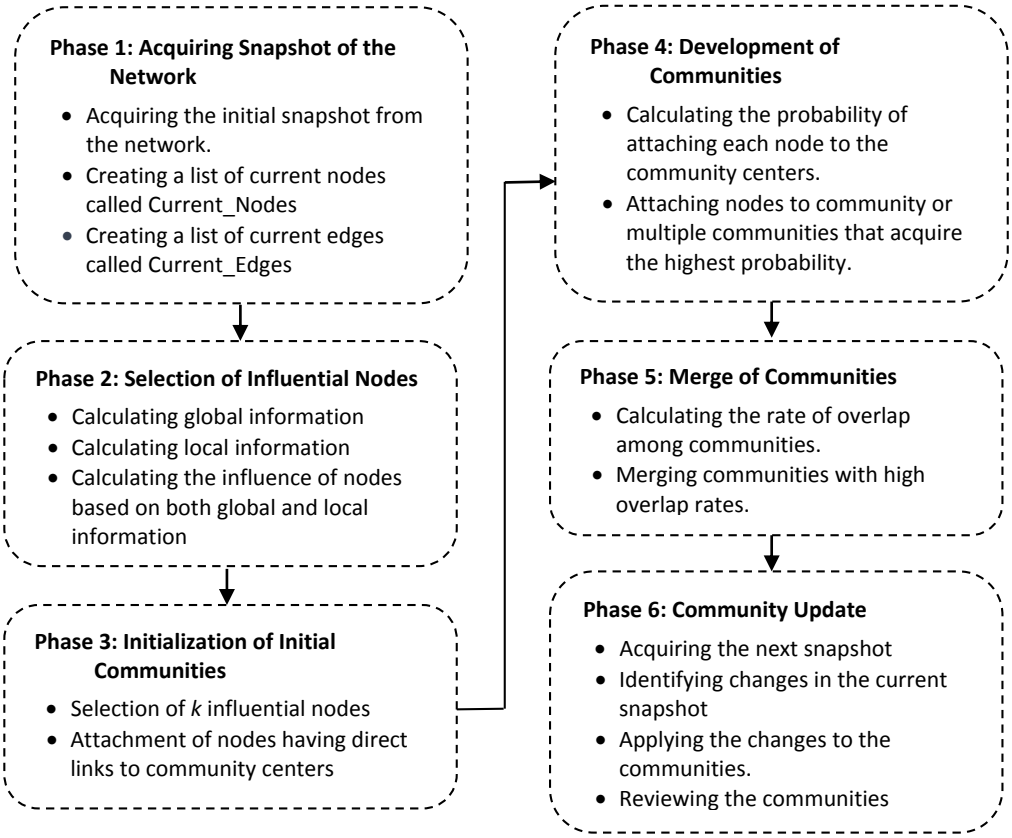

Fig. 2: Phases of the proposed method.

A. Phase 1: Acquiring Snapshot of the Network

In this phase, multiple snapshot images of the network are taken at different time intervals. Each image represents a set of nodes and the links between them, divided based on the time interval. Each image is independently given to the algorithm for community identification. The first image of the network is used for community detection, and the subsequent images are used to update the communities. Additionally, the nodes and edges present in the current network snapshot are extracted and encoded. For this purpose, two sets will be available: one containing the nodes and the other containing the edges. It is assumed that each node has a unique identifier, and the network can identify a node using this identifier. Therefore, based on this identifier, the sets of nodes and edges are generated.

The set of nodes in the current network snapshot is stored in an array called *Current_Nodes*. To store the edges, a matrix of size $m \times 2$ named *Current_Edges* is used, where m represents the number of edges, and each row contains the identifiers of the nodes forming the edge.

B. Phase 2: Selection of Influential Nodes

In this phase, following the approach in [5], three steps are taken to find influential nodes based on their local and global information. In the first step, to identify the global information of a node, the k-shell network decomposition algorithm is employed. Various metrics exist for calculating node importance in the network, but only node degree and clustering coefficient can indicate local network information. The k-shell is a connected subgraph of the maximum possible size in graph G where each vertex has a degree of at least k. The k-shell value for node i denoted as Ks(i) indicates that node i belongs to shell k but not to any other (k+1) shell. The k-shell decomposition method is often used to identify core nodes and peripheral nodes of the network. It starts by removing all nodes with only one link until no nodes remain and assigns them to shell 1. Likewise, it recursively removes all nodes with degree 2 or less and creates shell 2. This process continues until all nodes of the network are assigned to a single shell. Shells with higher indices are located in the core or center of the network. The k-shell decomposition method can be efficiently implemented with a linear time complexity of O(m), where m represents the number of edges in the network. An example of the k-shell algorithm's operation is depicted in Fig. 3.

In the second step, both global and local information of each node is computed. Global information indicates the node's status within the entire network. A node with high centrality has a higher *k*-shell value. The global

information of a node *i*, denoted as GI_i, indicates the dependency strength of other nodes in the network on node *i*. In other words, GI_i is calculated based on the average shells of the neighboring nodes. Thus, for a node like node *i*, it is computed according to (2).

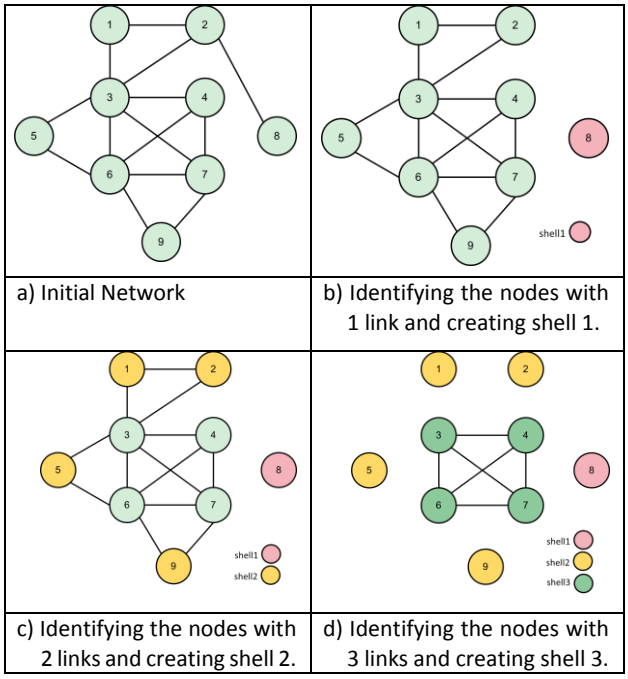

Fig. 3: An example of the k-shell decomposition algorithm's.

In this equation, *NumShell* represents the number of layers created by *k*-shell decomposition. Moreover, *neighbor(i,j)* is the number of neighbors of node *i* that belong to layer *j* of the *k*-shell decomposition.

$$GI_i = \frac{\sum_{j \in \text{NumShell}} |neighbor(i,j)| \times j}{\text{NumShell}} \quad (2)$$

After obtaining the global information of nodes, the measurement of local information follows. For measuring local information, the number of neighbors of each node is utilized. Thus, based on (2), the value of LI_i, representing the local information of node *i*, is derived.

$$LI_i = |neighbor(i)| \quad (3)$$

In the final step, to calculate the node's influence in the network, global and local information are combined according to (4). In this equation, α and β are coefficients for global and local information, respectively.

$$influence(i) = \alpha(GI_i) + \beta(LI_i) \quad (4)$$

Nodes with higher influence are considered as community centers. The pseudocode corresponding to the algorithm of this phase is provided in Algorithm 1.

C. Phase Three: Initialization of Initial Communities

Based on Phase Two, the list of influential nodes is obtained and sorted in descending order according to their influence level. In the next step, *k* nodes with higher influence are chosen as the cluster centers. Among the remaining nodes in the network, those having a direct link to cluster center nodes form the basis of communities. If two cluster centers have a direct link to each other, this link is disregarded, and these two centers will not be part of each other's communities.

Algorithm 1: Calculating nodes' influences

Input: Graph
Output: find influence of each node

1. initialize V = all nodes in G
2. **for** *i* = 1: *n*
3. | compute k-shell by k-shell decomposition algorithm
4. **end for**
5. compute k-shell for each node
6. calculate the number of neighbors of each node
7. **for** each node like *i*:
8. | calculate GI(*i*) by formula (2)
9. | calculate LI (*i*) by formula (3)
10. | calculate influence(*i*) by formula (4)
11. **end for**

D. Phase Four: Development of Communities

In this phase, nodes that are not yet part of any community are integrated into existing communities. To achieve this, attention is given to the nodes' neighbors. Essentially, a node joins a cluster where the majority of its neighbors are already affiliated. If the number of neighbors is the same for multiple clusters, the node becomes a member of all those clusters. The degree of association with a community is determined using (5):

$$dependency(i) = \max_{j \in ClusterHeads} (|cluster(neighbor\ (i),j)|), \quad (5)$$

where, $|clulouvainster(neighbor\ (i),j)|$ represents the count of neighbors of node *i* belonging to cluster *j*. If a node has neighbors to which no cluster has been assigned yet, those neighbors are disregarded in the aforementioned equation. Algorithm 2 outlines the steps related to community creation and expansion.

In line 1 of Algorithm (2), nodes are sorted based on their influence levels. In line 2, k nodes with higher influence are selected as seed nodes and stored in a list named "HeadCluster," responsible for maintaining the community centers. To better manage node processing for community expansion, a list called "Candidate" is created in line 3. This list is responsible for keeping track of nodes that have at least one neighbor belonging to a community. This ensures that nodes chosen for community extension are those which are guaranteed to have at least one neighboring node with an assigned

cluster. Otherwise, the node will remain without a community affiliation. Consequently, in line 3, all neighbors of the community centers are added to the Candidate list. Additionally, another list named *"assign_Nodes"* is created in line 4, tasked with holding nodes that have joined at least one community.

Algorithm 2: Algorithm for community creation and development

Input: Get Graph snapshot
Output: Output: Communities

```
1. Sort nodes in a descending order
2. HeadCluster ← select k top nodes
3. Candidate ← Neighbors(HeadCluster )
4. assign_Nodes←assign_Nodes ∪ HeadCluster
5. for j in Candidate
6.     flag ← false
7.     for HeadCluster like h
8.         if j is neighbors h
9.             Communities (h) ← j
10.            flag ← true
11.        end if
12.    end for
13.    if (flag == false)
14.        maxCom ← calculate dependency(j)
15.        for each community in maxCom
16.            Communities (community) ← j
17.        end for
18.    end if
19.    for each node in Neighbars(j ) like t
20.        if t ∉ assign_Nodes
21.            Candidate ← Candidate ∪ t
22.        end if
23.    end for
24.    assign_Nodes ← assign_Nodes ∪ j
25. end for
26. while (there is node in Graph that don't exist in
        assign_Nodes like j)
27.    neighbor ← Neighbars(j ) ∪ j
28.    HeadCluster ← maxinflunce (neighbor )
29.    for each node in neighbor like t
30.        Communities (maxinfluence(neighbor)) ← t
31.        assign_Nodes ← assign_Nodes ∪ t
31.    end for
32. end while
```

Continuing, all nodes selected as seed nodes are added to this list to prevent them from reappearing during the clustering process (line 5). In line 6, a variable named "flag" is initialized to identify cases where a node is a neighbor of a community center. Line 7 iterates through each community center to check whether node j is a neighbor or not. If node j is a neighbor of a community center, it is integrated into that community, and the flag is set to true (lines 7-12). If node j is not a neighbor of any community center, its association with communities is calculated using (5), and it is added to the possible communities (lines 13-18). Subsequently, all neighbors of node j that do not belong to any community are added to the Candidate list, and node *j* is also added to the *assign_Nodes* list to exclude it from the community creation process (lines 19-24). This process continues until the Candidate list contains at least one member (lines 26-30). Finally, lines 26-30 address cases of community dispersion. If nodes without a community exist, the most influential node among them and its neighbors are selected as the community center, and other nodes join this cluster as members.

E. Phase Five: Evaluation and Merge of Communities

In the process of selecting influential nodes, k nodes with high influence and their neighbors form initial communities. Therefore, if a node is connected to multiple community centers, it can be assigned to different communities. During the community expansion process, if a node of equal priority belongs to multiple communities, it is assigned to all of them. Consequently, many nodes exhibit overlapping membership. However, excessive overlapping between communities can increase algorithm complexity. Thus, controlling community overlap is essential for optimizing community structure. To control overlap, the overlap rate is employed, calculated based on (6):

$$\delta = \frac{|C_i \cap C_j|}{\min\{|C_i|,|C_j|\}}, \tag{6}$$

where C_i and C_j are two clusters or communities and $|C_i|$ and $|C_j|$ denote the number of their nodes. As δ increases, more nodes share membership in overlapping communities, making two communities more susceptible to merging. Nevertheless, in real-world scenarios, although two communities might have high overlap, practical considerations such as the community's subject matter might prevent their merger. Therefore, this paper also introduces another criterion called "community fitness". Community fitness indicates the significance of a community in the network and is calculated according to (7):

$$\begin{aligned} fitness(C_i) &= \frac{density_i + unique_i}{2}, \\ density_i &= \frac{|C_i|}{totalNodes}, \\ unique_i &= \frac{unique\_member_i}{member_i} \end{aligned} \tag{7}$$

Here, $density_i$ represents the density of the community which is calculates as the result of dividing the number of community members by the total number of nodes in the network. Furthermore, $unique_i$ shows the ratio of all nodes belonging to the same community i to the total nodes present in the community. Consequently,

communities that can achieve a fitness value exceeding a certain threshold persist, while others merge with different communities. In other words, for the community fitness criterion, lower values increase the likelihood of cluster merging.

To merge two communities, the node with the highest influence is selected as the new community center. All members of both clusters, including the old community center, are considered members of the new merged community. Algorithm 3 outlines the procedure for merging communities.

Algorithm 3: Algorithm for Community Merge

Input: communities
Output: final communities

for each community like C_i
  for each community like $C_j \neq C_i$
    calculate δ according to (6) and (7)
    if (δ > threshold$_\delta$ and fitness < threshold$_{fitness}$)
      $C_{new} \leftarrow max(seed_{influence,i}, seed_{influence,j})$
      $C_{new} \leftarrow \{C_i \cup C_j\}$
      $community \leftarrow community - C_i - C_j$
      $community \leftarrow C_{new}$
    end if
  end for
end for

F. Phase Six: Community Update

This phase consists of four steps, including: obtaining a new snapshot and identifying changes, incorporating changes into communities, reviewing communities, determining communities for orphan nodes, and evaluating and merging communities. Algorithm 4 illustrates the steps involved in performing this task. Assume that the designated nodes in the previous network are placed in the list *Current_Node* and the corresponding edges are in the list *Current_Edge*.

In the first step, a new snapshot of the network is acquired, and based on the existing communities, the communities are updated. To update communities, considering t as the current time, changes occurring in the network at time t compared to time *t*-1 (the previous network) must be determined. Network changes may involve adding or removing a node to/from the network or adding/removing an edge to/from the network. Set operators, including union and intersection, are utilized to identify network changes.

In the second step, following the identification of changes, the changes are applied to the network and the existing community structure from the previous network snapshot. The network can provide five types of changes, including adding or removing a node, or adding/removing an edge. In each of these scenarios, the influence levels of nodes and node membership in communities might undergo changes, necessitating decisions regarding them.

After implementing changes in the previous step, a reevaluation of the communities is carried out. During this stage, all communities that have undergone at least one change are reviewed. In this phase, the community centers and the nodes present within them are examined. If necessary, a community might be reassigned to a different node. Furthermore, if a community cannot meet the threshold for influence, it is dissolved. In this case, the dependency value (dependency(i)) is calculated for the members of the dissolved community, and decisions are made concerning orphan nodes.

In the final step, the degree of overlap and the fitness of nodes within communities are evaluated and determined. If necessary, communities are merged. In the next section, a detailed discussion will be presented regarding the effectiveness of the proposed method.

Algorithm 4: Algorithm for Network Change Detection

Input:
New_{Graph} //Get new Graph snapshot
$Current_{\mathrm{Node}}$ // Nodes in pervious network
$Current_{Edge}$ // Edges in pervious network
Output: $Current_{\mathrm{Node}}$, $Current_{Edge}$

Step 1: Initialization
1. $old_{\mathrm{Node}} \leftarrow Current_{\mathrm{Node}}$, $old_{Edge} \leftarrow Current_{Edge}$
2. get New_{Graph} and create New_{Node} , New_{Edge}

Step 2: find node changes
3. $NodeList \leftarrow old_{\mathrm{Node}} \cap New_{Node}$
4. $Omited_{Node} = old_{\mathrm{Node}} - NodeList$
5. $Added_{Node} = New_{\mathrm{Node}} - NodeList$
6. $Current_{\mathrm{Node}} = Current_{\mathrm{Node}} - Omited_{Node}$
7. $Current_{\mathrm{Node}} = Current_{\mathrm{Node}} + Added_{Node}$

Step 3: find edge intersect
8. $EdgeList \leftarrow old_{Edge} \cap New_{Edge}$
9. $Omited_{Edge} = old_{Edge} - EdgeList$
10. $Added_{Edge} = New_{Edge} - EdgeList$
11. $Current_{Edge} = Current_{Edge} - Omited_{Edge}$
12. $Current_{Edge} = Current_{Edge} + Added_{Edge}$

Step 4: Calculate influence
13. **for** each Node that adds or changes
14. Calculate influence according (Algorithm 1).
15. **end for**

Results and Discussion

In this section, we focus on the implementation and evaluation of the proposed algorithm (DIC). For implementation, the Python programming language is utilized. To execute the algorithm, relevant datasets are loaded. The datasets employed in this study are presented in Table 2.

To assess the performance of the proposed algorithm, we compare it with five recent methods introduced in studies DynaMo [21], D-Louvain [27], BBTA [30], DPC-DLP [31], ECD [32], IncNSA [33].

Table 2: Datasets used for the experiment

Dataset	#Nodes	#Edges	#Snapshots
Cit-Hep Ph	34546	421578	11
sx-mathoverflow	24818	506550	8
CollegeMsg	1899	59835	7

The evaluation metrics employed in this article consist of Newman's modularity, modularity with split penalty, and modularity density. The Newman's modularity (Q), applicable to undirected and unweighted networks, is defined as the difference ratio between the actual and expected number of edges within a community, as shown in (8).

$$Q = \sum_{c_i \in C} \left[\frac{\left|E_{c_i}^{in}\right|}{|E|} - \left(\frac{2\left|E_{c_i}^{in}\right| + \left|E_{c_i}^{out}\right|}{2|E|} \right)^2 \right], \tag{8}$$

where C represents the set of all communities, c_i denotes the i-th community, $\left|E_{c_i}^{in}\right|$ is the number of edges within the community, $\left|E_{c_i}^{out}\right|$ is the number of edges outside the community, and |E| is the total number of edges in the network. A higher value of this metric indicates the suitability of the communities. It's worth noting that due to the complexity of social networks' interactions and the possibility of links between any two edges, a value around 0.5 is considered appropriate for modularity. Therefore, if the value of this metric is around 0.5, it signifies the appropriateness of the community structure in a social network. Table 3 to 5 display the results of this metric for the compared methods across various datasets.

Table 3: Comparison of results using the Newman's modularity metric for the cit-Hep Ph dataset

TS*	DIC	D-Louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	**0.655**	0.628	0.534	0.538	0.566	0.575	0.631
2	**0.088**	0.083	0.021	0.087	0.0722	0.059	0.078
3	0.040	0.036	0.034	0.034	0.0508	**0.061**	0.049
4	**0.022**	0.017	0.013	0.015	0.012	0.016	**0.025**
5	0.012	0.005	0.010	0.010	0.0018	**0.015**	0.013
6	0.011	0.005	0.010	0.011	0.0006	0.010	**0.012**
7	**0.199**	0.111	0.174	0.123	0.195	0.118	0.163
8	**0.218**	0.201	0.215	0.211	0.213	0.209	0.200
9	**0.374**	0.223	0.116	0.238	0.263	0.292	0.374
10	**0.405**	0.103	0.157	0.365	0.280	0.351	0.295
11	**0.412**	0.225	0.261	0.303	0.255	0.252	0.359

* TS: Time stamp

As observed, according to the results obtained from Table 3 to 5, the proposed method has shown better performance compared to the other compared methods in most time intervals for each dataset.

Table 4: Comparison of results using the Newman's modularity metric for the CollegeMsg dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	**0.380**	0.329	0.331	0.278	0.322	0.354	0.348
2	0.146	0.149	0.122	0.129	0.122	0.108	**0.151**
3	0.494	0.477	0.431	0.378	0.443	0.359	**0.510**
4	**0.253**	0.244	0.232	0.245	0.202	0.210	0.211
5	**0.137**	0.117	0.110	0.104	0.111	0.108	0.110
6	0.167	0.150	0.181	0.171	**0.195**	0.154	0.181
7	**0.288**	0.206	0.209	0.238	0.183	0.239	0.213

* TS: Time stamp

Table 5: Comparison of results using the Newman's modularity metric for the sx-mathoverflow dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	**0.042**	0.214	0.190	0.209	0.216	0.151	0.041
2	0.000	0.068	0.001	0.001	0.007	0.057	**0.074**
3	0.000	0.218	0.188	0.213	0.176	0.160	**0.221**
4	0.001	0.004	0.032	0.084	0.020	**0.114**	0.046
5	**0.100**	0.054	0.069	0.049	0.008	0.062	0.074
6	**0.118**	0.096	0.086	0.106	0.106	0.104	0.110
7	**0.124**	0.071	0.107	0.111	0.114	0.061	0.101
8	**0.110**	0.016	0.074	0.108	0.064	0.102	0.079

* TS: Time stamp

Modularity with split penalty (Q_s) is another evaluation criterion used to assess the quality of community structures. It is calculated based on (9). In this equation, Q represents the modularity measure, and SP is the number of shared edges between communities, calculated according to (10).

$$Q_s = Q - \mathcal{SP} \tag{9}$$

$$\mathcal{SP} = \sum_{c_i \in C} \left[\sum_{\substack{c_j \in C \\ c_j \neq c_i}} \frac{\left|E_{c_i,c_j}\right|}{2|E|} \right] \tag{10}$$

In (10), $\left|E_{c_i,c_j}\right|$ represents the number of edges that connect community c_i to c_j. In this criterion as well, a higher value indicates a more suitable community structure. The results obtained for this criterion, separated by dataset, are presented in Table 6 to 8.

According to the results obtained from Table 6 to 8, for most datasets and time intervals, the proposed method based on the modularity with penalty division criterion has performed better.

Both Newman's modularity (Q) and modularity with split penalty (Qs) are independent of the number of nodes within communities. Modularity density (Q_{ds}) examines the nodes' density and their compactness within communities. For undirected networks, modularity density (Q_{ds}) is defined by (11).

Table 6: Comparison of results using Qs criterion for the cit-Hep Ph dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	**0.196**	0.120	0.136	0.195	0.145	0.130	0.156
2	0.433	0.437	0.328	0.412	0.403	0.456	**0.555**
3	0.4663	**0.4695**	0.446	0.426	0.424	0.427	0.410
4	**0.488**	0.485	0.433	0.384	0.384	0.383	0.419
5	0.489	**0.495**	0.406	0.388	0.417	0.474	0.436
6	**0.490**	0.485	0.452	0.450	0.459	0.462	0.450
7	0.143	0.104	**0.176**	0.101	0.173	0.134	0.160
8	0.492	0.498	0.389	0.422	0.391	0.395	**0.515**
9	0.491	0.486	0.410	0.446	0.446	0.465	**0.532**
10	**0.491**	0.486	0.457	0.446	0.397	0.398	0.364
11	**0.153**	0.015	0.087	0.131	0.161	0.096	0.140

* TS: Time stamp

Table 7: Comparison of results using Qs criterion for the CollegeMsg dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	**0.499**	0.491	0.143	0.399	0.296	0.399	0.368
2	**0.500**	0.500	0.493	0.490	0.410	0.415	0.432
3	**0.499**	0.399	0.408	0.430	0.433	0.484	0.360
4	**0.498**	0.499	0.397	0.324	0.313	0.424	0.460
5	0.040	**0.048**	0.162	0.034	0.027	0.011	0.042
6	**0.074**	0.054	0.040	0.0339	0.041	0.038	0.055
7	**0.068**	0.058	0.141	0.136	0.141	0.132	0.059

* TS: Time stamp

Table 8: Comparison of results using Qs criterion for the sx-mathoverflow dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	0.5660	0.1829	0.2727	0.2722	0.3167	0.3268	**0.6245**
2	0.0002	0.0002	0.1279	0.0417	0.1514	0.1212	**0.1540**
3	0.0005	**0.0008**	0.0001	0.0004	0.0005	0.0005	0.0006
4	0.0018	**0.0031**	0.0013	0.0013	0.0014	0.0012	0.0025
5	**0.1750**	0.1321	0.1067	0.1590	0.1644	0.1600	0.1236
6	**0.2080**	0.0101	0.0155	0.1747	0.1495	0.1090	0.1608
7	**0.1577**	0.0146	0.1404	0.1393	0.1341	0.0188	0.1554
8	**0.0109**	0.0104	0.0107	0.0023	0.0102	0.0104	0.0105

* TS: Time stamp

$$Q_{ds} = \sum_{c_i \in C} \left[\frac{|E_{c_i}^{in}|}{2|E|} d_{c_j} - \left(\frac{2|E_{c_i}^{in}| + |E_{c_i}^{out}|}{2|E|} \right)^2 - \sum_{\substack{c_j \in C \\ c_j \neq c_i}} \frac{|E_{c_i,c_j}|}{2|E|} d_{c_i,c_j} \right], \quad (11)$$

where d_{ci} represents the internal density of cluster c_i, and $d_{ci,cj}$ represents the between-community density between communities c_i and c_j, calculated using (12).

$$d_{c_i} = \frac{2|E_{c_i}^{in}|}{|c_i|(|c_i|-1)}, \quad d_{c_i,c_j} = \frac{|E_{c_i,c_j}|}{|c_i||c_j|} \quad (12)$$

Similar to the previous criteria, in this criterion as well, a higher value indicates a more suitable community structure. The results obtained for this criterion, separated by dataset, are presented in Table 9 to 11.

Table 9: Comparison of results using Qds criterion for the cit-Hep Ph dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	**0.513**	0.474	0.399	0.479	0.433	0.472	0.480
2	0.070	0.064	0.1229	0.140	0.116	**0.197**	0.165
3	0.095	0.030	0.122	0.060	0.116	**0.128**	**0.120**
4	**0.121**	0.015	0.066	0.108	0.119	0.118	0.110
5	0.111	0.005	0.138	0.059	**0.166**	0.063	0.160
6	0.128	0.001	0.123	0.126	0.072	0.058	**0.148**
7	**0.257**	0.181	0.229	0.150	0.231	0.239	0.249
8	**0.167**	0.100	0.169	0.141	0.153	0.146	0.127
9	**0.208**	0.101	0.143	0.043	0.126	0.070	0.183
10	**0.210**	0.103	0.127	0.100	0.045	0.131	0.115
11	**0.223**	0.133	0.138	0.216	0.192	0.169	0.186

* TS: Time stamp

Table 10: Comparison of results obtained using Qds criterion for the CollegeMsg dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	0.200	**0.219**	0.186	0.170	0.166	0.171	0.180
2	**0.003**	0.001	0.001	0.000	0.000	0.000	0.003
3	**0.002**	0.001	0.001	0.000	0.001	0.001	0.001
4	**0.001**	0.000	0.000	0.000	0.000	0.000	0.000
5	0.339	**0.373**	0.190	0.243	0.269	0.195	0.345
6	**0.208**	0.129	0.120	0.125	0.136	0.222	0.208
7	**0.286**	0.206	0.204	0.162	0.237	0.284	0.274

* TS: Time stamp

Table 11: Comparison of results obtained using Qds criterion for the sx-mathoverflow dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	**0.566**	0.182	0.272	0.272	0.316	0.326	0.410
2	0.000	0.000	0.127	0.041	0.151	0.121	**0.160**
3	0.001	**0.001**	0.000	0.00	0.000	0.000	0.001
4	0.002	**0.003**	0.001	0.001	0.001	0.001	0.002
5	0.075	**0.132**	0.106	0.059	0.064	0.060	0.124
6	**0.208**	0.0001	0.195	0.074	0.049	0.009	0.197
7	**0.157**	0.014	0.140	0.109	0.034	0.018	0.136
8	0.010	0.016	0.001	0.003	0.010	0.004	**0.016**

* TS: Time stamp

Considering the obtained results, it can be observed that the proposed method has demonstrated better performance compared to the compared methods on the introduced datasets, as well as with various evaluation criteria.

Finally, it is interesting to compare the execution time of different methods. Table 12 to 14 summarize the results for different datasets.

Table 12: Comparison of execution time (seconds) for the cit-Hep Ph dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	**0.06**	0.07	1.3	2.5	4.4	3.2	1.3
2	**3.6**	4.4	6.6	8.7	9.1	6.2	4.6
3	**17.1**	59.8	63.3	63.1	63.0	64.9	46.7
4	**51.9**	91.6	93.6	93.5	96.7	94.2	68.7
5	**95.7**	126.5	129.1	128.6	130.8	129.1	140.3
6	**146.6**	182.6	183.9	187.1	185.8	185.9	168.1
7	250.2	**246.6**	248.0	250.7	251.0	248.1	269.3
8	**145.6**	181.5	184.6	184.4	186.3	183.5	191.9

* TS: Time stamp

Table 13: Comparison of execution time (seconds) for the CollegeMsg dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	0.8	**0.3**	0.6	1.3	0.6	1.6	0.4
2	**18.3**	19.6	20.4	21.0	20.8	11.1	16.4
3	**37.6**	40.9	41.5	42.1	42.1	41.6	40.6
4	**13.6**	14.4	14.8	14.8	15.5	18.4	16.5
5	3.3	**3.1**	4.0	3.6	3.8	4.8	3.6
6	1.4	**1.1**	2.1	1.7	1.5	2.6	2.0
7	0.7	**0.2**	1.6	0.7	0.8	1.5	0.6

* TS: Time stamp

Table 14: Comparison of execution time (seconds) for the sx-mathoverflow dataset

TS*	DIC	D-louvain	DynaMo	IncNSA	ECD	DPC-DLP	BBTA
1	**14.0**	21.9	22.7	23.0	22.9	23.4	20.6
2	**300.8**	460.5	472.7	519.9	541.2	491.8	521.6
3	**997.6**	1515.3	1291.2	1371.3	1487.2	1871.8	1418.6
4	**1198.7**	1804.8	1336.0	1425.1	1745.3	1409.8	1245.7
5	**1048.2**	1573.9	1741.5	1575.0	1322.8	1612.8	1510.0
6	**1002.1**	1503.9	1505.0	1504.8	1504.6	1505.3	1469.1
7	**986.0**	1479.4	1480.1	1480.2	1479.8	1480.9	1341.1
8	**789.4**	1184.1	923.2	1012.5	964.2	811.1	794.0

* TS: Time stamp

According to the obtained results, the proposed method has been reported lower execution time in comparison with other methods.

Conclusion and Future Works

In this paper, we have presented a method for detecting communities in dynamic networks based on influential nodes. Considering that communities and groups formed in a network in the real world can overlap, another goal of this paper is to provide a method to address community overlaps. Through this method, an attempt is made to preserve communities that have low overlap and allow their members to belong to both overlapping communities. To the best of our knowledge, a method for detecting communities in dynamic networks that simultaneously utilizes influential nodes to determine communities while considering overlapping communities has not been presented so far. Furthermore, the proposed method supports all possible changes in dynamic networks, including the addition and removal of nodes as well as the addition and removal of edges in dynamic networks. The proposed algorithm accomplishes the desired objectives through six phases: obtaining a snapshot of the network, selecting influential nodes based on local and global information, initialization, community expansion, evaluation and merging of communities, and finally updating communities. To evaluate the performance of the proposed method, we compared it with five recently proposed methods using three modularity metrics.

Based on the results obtained, the proposed algorithm has managed to achieve better results in most cases compared to the compared algorithms, especially in terms of modularity metrics and execution time. The reason behind this success could be attributed to the utilization of influential nodes in community formation. In the proposed algorithm, two metrics, namely node degree and k-shell decomposition, are used to determine influential nodes, both of which focus on node degree. As a result, the algorithm aims to shape the community around nodes that have the most connections to other nodes. On the other hand, the proposed method attempts to form a community around nodes that have the potential to establish a community, meaning they have high influence and lack connections with current communities. Ultimately, communities capable of merging based on the degree of congruence and overlap rate are merged with each other. Consequently, the proposed method retains only those communities that, on the one hand, have the potential to form a community and, on the other hand, cannot merge with other communities. In contrast, in other methods, attempts are made to use baseline modularity as a criterion for deciding whether to merge two communities or form a new one. Moreover, no clear and coherent idea exists regarding which nodes should be recognized as the center of a community. Only if the algorithm determines that separating one or more nodes would improve the

baseline modularity, will the operation of forming a new community take place. Furthermore, in the evaluation and community merging phase, the algorithm addresses inter-community edges and merges communities with overlapping members as much as possible. Therefore, in terms of modularity metrics, the proposed algorithm has achieved more favorable results compared to other methods.

Despite all the advantages of the proposed method, there are still challenges that are posed as open research issues. Firstly, we should emphasize that rapid identification of a change in the network and its application to communities in real-time might be closer to real-world applications. Thus, it is interesting that extend our method to detect the communities in real-time. Secondly, in this paper, we used two metrics, k-shell and node degree, for identifying influential nodes. The k-shell metric itself has high computational and time complexity. Therefore, considering other metrics and examining their results in the created communities could be another subject in this field.

Author Contributions

S. Baradaran Nejad implemented the methods and evaluated their performance. M. Sabzekar wrote the paper, coordinated the study and contributed to the analysis of the results. M. Khazaeipoor edited the manuscript. All authors revised and discussed the results and approved the final manuscript.

Acknowledgment

This work is completely self-supporting, thereby no any financial agency's role is available.

Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

Abbreviations

LI	local information
DN	Dynamic Network
SN	Social Network

References

[1] B. Yang, D. Liu, J. Liu, "Discovering communities from social networks: Methodologies and applications," Handb. Soc. Netw. Technol. Appl.: 331–346, 2010.

[2] S. Souravlas, S. D. Anastasiadou, T. Economides, S. Katsavounis, "Probabilistic community detection in social networks," IEEE Access, 11: 25629–25641, 2023.

[3] A. Abbasi, H. Chen, A. Salem, "Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums," ACM Trans. Inf. Syst., 26(3): 1–34, 2008.

[4] R. K. Bakshi, N. Kaur, R. Kaur, G. Kaur, "Opinion mining and sentiment analysis," in Proc. 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 16: 452-455, 2016.

[5] T. Ma, Q. Liu, J. Cao, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, "LGIEM: Global and local node influence-based community detection," Futur. Gener. Comput. Syst., 105: 533–546, 2020.

[6] N. Chen, Y. Liu, J. Cheng, Q. Liu, "A novel parallel community detection scheme based on label propagation," World Wide Web, 21: 1377–1398, 2018.

[7] Y. Niu, D. Kong, L. Liu, R. Wen, J. Xiao, "Overlapping community detection with adaptive density peaks clustering and iterative partition strategy," Expert Syst. Appl., 213: 119213, 2023.

[8] A. Reihanian, M. R. Feizi-Derakhshi, H. S. Aghdasi, "An enhanced multi-objective biogeography-based optimization for overlapping community detection in social networks with node attributes," Inf. Sci. (Ny)., 622: 903-929, 2023.

[9] P. Agarwal, R. Verma, A. Agarwal, T. Chakraborty, "DyPerm: Maximizing permanence for dynamic community detection," in Proc. Pacific-Asia conference on knowledge discovery and data mining: 437-449, 2018.

[10] X. Zeng, W. Wang, C. Chen, G. G. Yen, "A consensus community-based particle swarm optimization for dynamic community detection," IEEE Trans. Cybern., 50(6): 2502–2513, 2020.

[11] S. Ahajjam, M. El Haddad, H. Badir, "A new scalable leader-community detection approach for community detection in social networks," Soc. Networks, 54: 41-49, 2018.

[12] K. Dasgupta et al., "Social ties and their relevance to churn in mobile telecom networks," in Proc. 11th international conference on Extending database technology: Advances in database technology: 668–677, 2008

[13] M. A. Al-Garadi et al., "Analysis of online social network connections for identification of influential users: Survey and open research issues," ACM Comput. Surv., 51(1): 1-37, 2018.

[14] Y. Zhao, "A survey on theoretical advances of community detection in networks," Wiley Interdiscip. Rev. Comput. Stat., 9(5): 1403, 2017.

[15] S. Bahadori, H. Zare, P. Moradi, "PODCD: Probabilistic overlapping dynamic community detection," Expert Syst. Appl., 174: 114650, 2021.

[16] N. Chen, B. Hu, Y. Rui, "Dynamic network community detection with coherent neighborhood propinquity," IEEE Access, 8: 27915-27926, 2020.

[17] B. S. Khan, M. A. Niazi, "Network community detection: A review and visual survey," arXiv Prepr. arXiv1708.00977, 2017.

[18] R. Cazabet, G. Rossetti, F. Amblard, "Dynamic community detection," In: Alhajj, R., Rokne, J. (eds) Encyclopedia of Social Network Analysis and Mining. Springer, New York, NY., 2017.

[19] G. Rossetti, R. Cazabet, "Community discovery in dynamic networks: a survey," ACM Comput. Surv., 51(2): 1-37, 2018.

[20] S. Souravlas, S. Anastasiadou, S. Katsavounis, "A survey on the recent advances of deep community detection," Appl. Sci., 11(16): 7179, 2021.

[21] D. Zhuang, J. M. Chang, M. Li, "DynaMo: Dynamic community detection by incrementally maximizing modularity," IEEE Trans. Knowl. Data Eng., 33(5): 1934-1945, 2019.

[22] M. A. Javed, M. S. Younis, S. Latif, J. Qadir, A. Baig, "Community detection in networks: A multidisciplinary review," J. Netw. Comput. Appl., 108: 87-111, 2018.

[23] J. Scripps, "Discovering influential nodes in social networks through community finding.," in Proc. WEBIST: 403-412, 2013.

[24] J. Dai et al., "Identifying influential nodes in complex networks based on local neighbor contribution," IEEE Access, 7: 131719-131731, 2019.

[25] X. K. Zhang, J. Ren, C. Song, J. Jia, Q. Zhang, "Label propagation algorithm for community detection based on node importance and label influence," Phys. Lett. A, 381(33): 2691-2698, 2017.

[26] C. Li et al., "NANI: an efficient community detection algorithm based on nested aggregation of node influence," in Proc. the ACM Turing 50th Celebration Conference-China: 1-10, 2017.

[27] M. Cordeiro, R. P. Sarmento, J. Gama, "Dynamic community detection in evolving networks using locality modularity optimization," Soc. Netw. Anal. Min., 6: 1-20, 2016.

[28] W. Li, X. Zhou, C. Yang, Y. Fan, Z. Wang, Y. Liu, "Multi-objective optimization algorithm based on characteristics fusion of dynamic social networks for community discovery," Information Fusion, 79: 110–123, 2022.

[29] Q. Ni, J. Guo, W. Wu H. Wang, "Influence-Based community partition with sandwich method for social networks," IEEE Trans. Comput. Social Syst., 10: 2: 819-830, 2023.

[30] H. Long, X. Li, X. Liu, X. et al., "BBTA: Detecting communities incrementally from dynamic networks based on tracking of backbones and bridges," Applied Intelligence, 53:1084–1100, 2023.

[31] S. A. Seyedi, A. Lotfi, P. Moradi, N. N. Qader, "Dynamic graph-based label propagation for density peaks clustering," Expert Syst. Appl., 115: 314–328, 2019.

[32] F. Liu, J. Wu, C. Zhou, J. Yang, "Evolutionary community detection in dynamic social networks," in Proc. 2019 International Joint Conference on Neural Networks (IJCNN): 1–7, 2019.

[33] X. Su, J. Cheng, H. Yang, M. Leng, W. Zhang, X. Chen, "IncNSA: Detecting communities incrementally from time-evolving networks based on node similarity," Int. J. Mod. Phys. C, 31(7): 2050094, 2020.

Biographies

Mostafa Sabzekar received the B.S. degree in computer engineering from Kharzami University of Tehran, Iran, in 2007, and the M.S. and Ph.D. degrees in Computer Engineering from Ferdowsi University of Mashhad, Iran, in 2009, and 2017, respectively. He is currently an assistant professor at Birjand University of Technology, Birjand, Iran. His research focuses mainly on machine learning, evolutionary computation, and Bioinformatics.

- Email: Sabzekar@birjandut.ac.ir
- ORCID: 0000-0002-6886-1240
- Web of Science Researcher ID: NA
- Scopus Author ID: 35796344600
- Homepage: https://cv.birjandut.ac.ir/sabzekar/en

Shima Baradaran Nezhad was born in Birjand, Iran. She received the B.Sc. degree in Computer Engineering, from University of Birjand, Iran, in 2007. She received the M.Sc. dgree in Computer Engineeringtion from Birjand Branch, Islamic Azad University, Birjand, Iran, in 2022. She is currently an employee of the General Department of Natural Resources and Watershed Management of South Khorasan, Iran. Her research interests include Pattern Recognition and Algorithms.

- Email: sh.baradaran@frw.ir
- ORCID: NA
- Web of Science Researcher ID: NA
- Scopus Author ID: 35796344600
- Homepage: NA

Mahdi Khazaiepoor recived the B. Sc. Degree in Software Engineering from Islamic Azad university of mashhad, iran, in 2003 and M.Sc. degree in Software Engineering from Islamic Azad University Science and Research Branch, Tehran, Iran, in 2008, and Ph. D. degree in Software Engineering- Development of software systems from Islamic Azad University kerman branch, kerman, Iran. He is He is currently an assistant professor and the director of the computer department of the Islamic Azad University, Birjand branch, Iran. His research interest includes Data Mining, Software Effort Estimation and swarm intelligence algorithms.

- Email: mkhazaeipoor@iaubir.ac.ir
- ORCID: NA
- Web of Science Researcher ID: NA
- Scopus Author ID: 35796344600
- Homepage: NA

How to cite this paper:
M. Sabzekar, S. Baradaran Nezhad, M. Khazaeipoor, "A node-centric approach for community detection in dynamic networks," J. Electr. Comput. Eng. Innovations, 12(2): 305-318, 2024.

DOI: 10.22061/jecei.2024.10202.687

URL: https://jecei.sru.ac.ir/article_2043.html