Future Generation Computer Systems 113 (2020) 41–57

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

LSMD: A fast and robust local community detection starting from low degree nodes in social networks

Asgarali Bouyer *, Hamid Roghani

Department of Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran

ARTICLE INFO

Article history:
Received 27 February 2020
Received in revised form 1 June 2020
Accepted 6 July 2020
Available online 7 July 2020

Keywords:
Community detection
Local similarity
Label diffusion
Low degree nodes
Social networks

ABSTRACT

Community detection is an appropriate approach for discovering and understanding the structure and hidden information in complex networks One of the most critical issues in the community detection problem is the low-time complexity of the algorithm while preserving the accuracy of the algorithm, which is important in large-scale networks such as social networks. Local community detection algorithms try to use local information and provide acceptable results in a reasonable time. This paper proposes a fast and accurate community detection algorithm based on local information for the community's label assigning. In the proposed algorithm, local community detection is started from low degree nodes by label assigning in a multi-level diffusion way, called LSMD algorithm, with significant low time complexity. In the first phase of the LSMD algorithm, at first, the community's label is assigned to the node with degree 1 and its direct neighbor and second level neighbors. In fact, we used this fact that people with a smaller number of neighbors are not likely to be connected to diverse communities. Next, a community label is respectively assigned for the nodes with degrees 2, 3, and so forth. In the second phase, initial communities are merged using a new simple and fast strategy as far as possible to form the final communities. Besides, there is no random nature in the algorithm, as well as the adjustable parameter. Therefore, the obtained results show meaningful stability for the LSMD. Experiments are performed on real-world and synthetic networks to evaluate the performance and accuracy of the proposed algorithm. The results show that the proposed algorithm significantly is more accurate than the other state-of-the-art algorithms. In addition, it substantially is faster than other local algorithms such as LPA, LCCD, NIBLPA, DA, RTLCD, Louvain, G-CN, Infomap, and ECES on large-scale networks.

1. Introduction

A complex network can model many real-world phenomena, Protein-protein networks, Internet networks, information networks, and social networks are examples of complex networks [1]. Analysis of different complex networks can be accomplished with different goals. For example, in social networks, we can analyze the relationships between people by discovering different communities of friends [2,3]. As another instance, analysis of protein-protein networks can discover the function of new proteins [4,5]; or in information networks by analyzing the pages on the web, it is possible to group similar web pages in the same community [6,7]; in online shopping network, by discovering communities different categories of customers are discovered to offer services according to their needs [8].

With the growth in popularity of online social networks such as Facebook, Twitter, WhatsApp, and YouTube, there has been rising attention in social pattern analysis. The massive data in these networks are rapidly produced, which can be represented by graphs. The analysis of these networks helps us to describe the complex systems of the real world [9]. It empowers us to describe the available data, finding groups, influencers, and communities within these networks. One of the most important analyzes in this field is finding communities. Discovering communities help us to analyze the structure of existing communities and the role of their important nodes in more detail [10]. A community is a subgraph of the original graph where the number of connections between nodes within a community is high and the connections between nodes within a community, with nodes in the other communities are low [1]. However, finding a satisfactory community structure in large-scale networks such as social networks is a hard problem due to massive network size and limitation of time complexity. An essential issue in community detection is that the nodes within a community must have the same properties and similarities. The similarity between two nodes measures the relationship of proximity between two nodes [11]. With this in mind, many methods try to discover community structures using local similarity measures or global information and similarities in networks. The global methods need general information about

* Corresponding author.
E-mail addresses: a.bouyer@azaruniv.ac.ir (A. Bouyer), h.roghani@azaruniv.ac.ir (H. Roghani).

https://doi.org/10.1016/j.future.2020.07.011
0167-739X/

all nodes to find meaningful communities, but they are time-consuming and unfeasible for large-scale networks. For instance, the edge-betweenness-based community detection method is a global method with $O(n^3)$ time complexity [12]. On the other hand, local methods use the local similarity such as Jaccard index, Salton index, Sorensen index and so on to find communities with less time complexity [1,13,14]. Since the main operation needed in these local methods, is computing the common neighbors or intersection of two sets with length k, so time complexity of these methods for a graph with n nodes and average degree k is equal to $O(nk^2)$.

The main challenge in community detection of the massive networks is proposing algorithms with less time complexity and acceptable quality in detected communities. Local methods such as label propagation algorithm (LPA) is a well-known local algorithm with $O(m+n)$ time complexity [15]. LPA is the first label-propagation based algorithm with nearly linear time complexity but low quality in community detection. Due to the existence of random behavior in node selection as well as the weakness of the label updating strategy, its results have not been promising due to significant instability in number of detected communities. However, there are many attempts to propose different versions with aim to improve its performance [16–19]. Furthermore, there are other methods for fast community detection in large-scale networks such as G-CN, Louvain, ComTector [20–23]. However, most of them are not adequate for such big networks due to weak performance and inappropriate computational time in massive networks. For example, the LPA algorithm, despite its simple nature can be very time consuming on large-scale networks. Since at each iteration and for each node, the algorithm must check neighbors' label, group them based on label, and selects the most frequent label, therefore the execution time increases due to these operations on large-scale networks.

Therefore, the main impact of this paper is to propose a fast community detection algorithm called LSMD, which is significantly capable for the community detection in massive networks. In fact, the LSMD inheritance its functionality from people's behavior in real communities. Our concentration in this method is on assigning community labels for weak and unimportant nodes (low degree nodes) based on their important connected neighbor. Because in the real world, people with a low number of neighbors are not likely to be connected to different communities. Thus, LSMD is a community detection algorithm based on local similarity measures and multi-level diffusion by starting from low degree nodes. First, nodes are grouped according to their degrees. Then we start with a node from the lowest degree group to assign a community label and then try to diffuse the label of this node up to the second level of neighbors by considering some conditions. In the next step, after detecting the initial communities, some small communities are merged to reach an appropriate size of the community. Given a large-scale sparse graph, the running time of our algorithm is less than O(nk), where k is the average degree of the network. This algorithm is considered the fastest community detection algorithm in comparison with other fast algorithms such as LPA ($O(m+n)$), G-CN ($O(nk)$), Louvain ($O(nlogn)$), Infomap ($O(n(n+m))$), and other compared algorithms. Because, LSMD does not require to find particular nodes such as core, hub, or bridges. In addition, it does not use any time-consuming methods such as ranking and sorting algorithms in discovering communities. Experiments show that even algorithms such as LPA despite having lower time complexity than LSMD, due to their time-consuming operations such as checking all neighbors' label and selecting the most frequent label, has more execution time than LSMD.

This paper is arranged as follows: In Section 2, some related studies are reviewed. In the third section, the proposed algorithm is completely described with details. In Section 4, the experiments and evaluation of the results on real-world and artificial datasets are discussed. Finally, the conclusion and future directions are presented in the last section.

2. Related work

With the rapid development of complex networks, especially social networks, the problem of community detection in these networks has attracted much attention. Thus, presenting algorithms with high accuracy and low computational time is one of the most important and up-to-date challenges in the community detection field. One of the first implemented methods for community detection was the graph partitioning method [1,24,25]. However, finding the best partitioning is an NP-hard problem [1]. In each step of these algorithms, the graph is divided into two parts, such that the size of the groups is nearly equal and use a function to evaluate the quality of the division. The most popular algorithms for this type of method are Spectral bisection [24] and Kernighan–Lin [25]. The low quality and high time-complexity are the main properties in most graph partitioning algorithms, which makes them inapplicable in large-scale networks.

The other types of community detection methods are based on hierarchical clustering. These methods are used in two agglomerative and divisive manner [1,11,22,23,26–28]. Agglomerative hierarchical methods use a similarity index to merge similar nodes into a community, whereas divisive methods try to remove edges from the graph, so the network gets divided into different communities. In [12], Newman proposed a divisive method based on edge betweenness for detecting communities. The edge betweenness criterion is used to identify edges with the high flow so that by removing them, the graph is divided into several communities. In [29], Fortunato et al. proposed a divisive method by removing edges from graph based on their information centrality criterion. In [27] Xi et al. discover communities using an agglomerative hierarchical method based on modularity and similarity criteria. They have used the Louvain algorithm [22] and a similarity index to measure the similarity of nodes. These methods are inapplicable for large-scale networks due to their high computational time.

The other group is called modularity-based methods. These methods try to optimize the gained modularity during several iterations. If assigning a node to a community or merging two initial communities optimizes the modularity, this merging is carried out. In [23] Newman et al. have described the community detection problem as an optimization problem by considering the modularity criterion. In this method, at first, each node is considered as a single community. Then the two nodes are merged in one community if it increases the modularity.

In [22] a modularity-based heuristic method is described, which is commonly known as the Louvain algorithm. At first, Louvain starts with a weighted network and each node is initially considered as an independent community. Then the modularity is calculated between a node and its neighbors. Then, the selected node is assigned to the community if the modularity is increased. These stages are repeated for each node until no more improvement in modularity is achieved (the algorithm is trapped in local maxima). In the second stage of the algorithm, detected communities are merged according to improvement in modularity criterion.

Clauset et al. [30] proposed modularity based hierarchical agglomerative algorithm for detecting communities. They have used a max-heap structure to keep track of the highest modularity gain (ΔQ). In [26] Waltman et al. proposes a method which is optimal both in terms of merging of communities and in terms of moving of nodes between communities. In multi stage improvement, after detecting initial communities, the algorithm checks whether modularity can be increased by moving single nodes or not? Then two communities are merged if the result shows an increase in modularity. [11] and [28] both have similar approaches and they use two stages for detecting communities. In the first step, edges are weighted using a local similarity index. Then edges with weights less than a threshold are eliminated, so that the initial communities are obtained. Then the algorithm merges communities if the modularity value from merging them

is improved. Wu et al. proposed the ImDS method as a density based algorithm [31]. It uses modularity measure for merging nodes into super nodes. In this algorithm dense pairs are found and they are possibly merged according to the ΔQ to reform a super node.

On the other hand, the most important challenge in recent decades is to use local methods for community detection problems. Therefore, another type of community detection approaches, are local similarity-based methods that use structural similarities between nodes and their neighbors. RTLCD is a local community detection based on core detection and community expanding from these cores. It has $O(rmax\{d\,|C|\log|C|\,,|N\,(C)|\,(dlog\,|C|+d^4)\})$ running time. r is the maximum length of the shortest path from the detected community core member to the edge of the detected community, $|C|$ is the size of community neighbors, and d is the mean degree of the network [32]. RTLCD has low performance in most of the time. Local similarity-based methods usually have low time complexity but less accuracy than global methods. For example, label propagation-based methods, local modularity-based methods, some hierarchical clustering algorithms, etc. use local similarities.

Due to the importance of label propagation-based methods, they are defined in a new category, called LPA-based algorithms. Raghavan proposed the first label propagation algorithm, which is called the LPA algorithm [15]. The LPA is one of the fast-local methods with nearly linear time complexity. However, it has limitations, such as having the random selection of initial nodes and selecting random labels in tiebreak states. To solve these limitations, more than 70 different improvements were done on traditional LPA to reach efficient and more accurate algorithms in contrast with traditional LPA. For instance, methods in [16–19,33–37] try to select nodes based on their importance in a specific order to solve the initial selection problem and improving the label updating strategy of nodes.

In [36], the SkipLPA method was proposed to decrease the execution time of the traditional LPA. The main idea behind this algorithm is that some nodes are not necessary to participate in the initialization and label updating steps. In SkipLPA, at the initialization step, only a certain number of nodes have initialized with labels, and the other nodes receive their labels from their connected labeled nodes. In this algorithm, the nodes with degree less than the Δ value are excluded from the label propagation phase. The methods presented in [17–19,33–35] have improved the selecting order of nodes and label updating strategy. These algorithms have used the concept of node importance and the different influence of communities on nodes as well as the strength of the edges between the nodes. They have also used similarity measures to improve the label updating strategy.

In [18], Xing et al. use the concept of K-shell decomposition to determine the importance of nodes in the NIBLPA algorithm. In NIBLPA, nodes are sorted according to their K-shell value and the most important node is used to start the algorithm. When the multiple labels have a maximum number of frequencies, NIBLPA is fallen in tiebreak state. In this case, the label importance is selected based on an effective community. In [19], the concept of nodes importance is used to make a meaningful order for selecting nodes. The Bayesian network is used to find the importance of nodes. LPA-Intimacy algorithm is proposed by Kong et al. [17]. It uses the notion of the influence of nodes and intimacy between nodes. Methods in [18,19] consider the importance of nodes using K-shell or Bayesian network, but LPA-Intimacy uses the intimacy and influence of a node as a criterion for selecting important nodes and assigning an order for selecting and updating their label.

In [35], a new criterion is used to determine the centrality of the nodes by calculating the similarity and local density of the nodes. At first, boundary nodes and the nodes with high centrality value receive their label. At the next, the label propagation step is performed similar to the traditional LPA algorithm. Lin et al. proposed CK-LPA in [34], which is a label propagation based algorithm with finding the kernel nodes of the communities. In CK-LPA, at first, K kernel nodes are determined. Then weights are assigned to the nodes based on their distance from the kernel nodes, and the updating order of nodes is based on these assigned weights. The kernel of communities is the most important nodes, so that, K nodes with highest degree which does not have connections to each other, are selected as the kernel nodes. In the label updating section, a community label with the highest sum of its node's weights, is selected.

In [10] Berahmand et al. proposed a local algorithm, ECES, based on detecting and expanding the core nodes. A centrality criterion is used, which gives high scores to central nodes with a high degree of embeddedness. At first, the weights between edges are calculated using a similarity index, then the core dominance of the kernel node is calculated to select central nodes. The next step is to expand the central nodes using a membership function. At that time, in the merging phase, final communities are formed. In [38], an algorithm with name LPAM+ is proposed, which uses LPA and modularity for detecting communities. In fact, LPAM+ was proposed as a new version of the LPAM [39] to solve the local optima problem using multi step greedy (MSG) method. The MSG[1] method can increase the modularity and solve the local maxima problem by merging the communities when the algorithm is trapped in the local maxima, and no more modularity is achieved.

In addition, a new non-overlapping label propagation algorithm was presented to improve the efficiency and instability problems of traditional LPA. It uses a semi-local similarity to assign weights to nodes to update their label based on these weights. This method helps to avoid random selection in initialization and tie break state [40]. Eustace et al. proposed a new algorithm based on the nodes' neighborhood and local similarity criterion [41]. In this algorithm, initial communities are first discovered, and then during an iterative step, the local communities are possibly merged to create global communities. First, a random node is selected in the graph. Next, the similarity between that node and its neighbors is computed ($\alpha - close$). If the similarity value is more than α, the selected node and its neighbor are assigned to one local community. α is the average of the neighborhood ratio of nodes in the whole graph and the relation is described with details in [41]. Then in the merge phase, the similarity between communities is computed ($\beta - close$), if the result is more than β, two communities are merged together. β is the average neighborhood ratio (similarity ratio) in all local communities.

Wang et al. proposed the LCCD algorithm for detecting communities [42]. This algorithm initially finds central nodes of communities based on the structural centrality, which uses local density and distances between nodes to calculate it. After identifying central nodes, it expands the local community from inner central nodes to outer nodes. In [43] Li et al. proposed Stepping LPA-S, which is a label propagation-based algorithm. In Stepping LPA-S, labels are propagated using a resource allocation similarity index. This algorithm updates the label of a node according to the most similar label of its neighbors, and then initial communities with high similarity are merged. Liu et al. proposed a divide and agglomerate algorithm (DA) [44]. DA uses local and global information for community detection. Like other methods, it consists of two main steps, one for dividing the network into some small groups and then merging and building the final communities.

3. The proposed method

The proposed algorithm (LSMD) uses local similarity with a multi-level label diffusion algorithm, which is executed for different groups of nodes based on their degrees. The nodes with acceptable similarity at each step are assigned to the same communities. LSMD diffuses a label from periphery nodes of

[1] Multi step greedy.

community to the inside of the community. This algorithm is significantly different from other algorithms. All previously proposed methods try to find core nodes to start propagation from core nodes to other nodes. In these methods, the community's label is propagated from inside to outside of the community and these types of algorithms need to sort nodes based on their importance. However, our proposed algorithm does not use any ranking and sorting operation and therefore it is the fastest community detection method among well-known fast algorithms. In the LSMD method, nodes are placed in different lists based on their degree. For example, the list L1 contains all nodes with degree one. Therefore, after grouping the nodes according to their degrees, the LSMD algorithm starts with the lowest degree list and continues incrementally based on degree. For example, if there is a group of nodes with degree 1, the algorithm starts with that group and after finishing them, it continues the group with degree 2 and so on. In this paper, the selected node in each group is named "*target node*", direct neighbors of the target node are called "*first level neighborhood nodes*", and the neighbors of the target node are called "*second level neighborhood nodes*". The goal is to diffuse the label from the target node to the first and second level neighborhood nodes and in some conditions to the third level neighborhood nodes by evaluating the label diffusion conditions and some necessary local indexes. We use numerical indicators such as "1", "2", "3", etc. as labels of communities in assigning a label to the node. We start community labels from number 1, and each time when an initial target node is selected, it gets the new label. The new label is created as: New_label= current_label+1. Now, this label is assigned to a new community. At the beginning of the LSMD algorithm, all nodes are initialized with label "0" which means none of nodes is assigned to a community. One important feature of the LSMD algorithm is to use the diffusion concept for expanding the community, which means a group of nodes gets their label at once. Needless to mention that in label propagation-based methods all nodes initially get a unique label, and each node is selected to update its label in next iterations in a synchronous or asynchronous way. However, in the LSMD algorithm there is no propagation-based nature, no initial labeling, and no updating strategies. Except target nodes, all other nodes completely are dependent to their target nodes, and they get the label from the desired target node. In other words, the label of each target node is diffused to other nodes in the first, second, and third level of its neighbors if they satisfy the mentioned conditions in flowchart of Fig. 2. That is why LSMD has label diffusion behavior because a group of nodes receives a label at once, and they are not checked in the next steps.

The LSMD consists of three main phases: the label diffusion phase, the processing of overlapping nodes to assign effective community label, and merging the initial communities. The steps of the algorithm are explained in detailed at following sub-sections.

3.1. Label diffusion phase

As mentioned before, nodes are first categorized by their degrees. Initially, the process of community formation begins with a group of nodes with degree 1. The LSMD algorithm must identify the community's label for all nodes with degree 1 before processing the nodes with degree 2. After finishing the nodes with degree 1, it respectively continues processing nodes with degree 2, 3, etc. All nodes are initially labeled with zero ("0"), which means that no node belongs to any community. In addition, in the beginning, each node has a flag with default value "*Undiffused*". Each node that participates in the community detection process, its flag is changed to "*Diffused*" mode. The behavior of the LSMD algorithm for nodes with degree 1 is similar to the nodes with degree 2, 3, etc. However, for nodes with a degree higher than 1, these conditions are slightly different. For this reason, we separately explain the algorithm for nodes with degree one and nodes with a degree higher than 1. Needless to mention that, before all, we need to calculate the edge-clustering coefficient for each edge using Eq. (1) and the average edge-clustering coefficient using Eq. (2).

$$ECC_{i,x} = \frac{|N_i \cap N_x|}{|N_i| \times |N_x|} \tag{1}$$

where $ECC_{i,x}$ is the edge-clustering coefficient for the edge $e_{(i,x)}$, and N_i and N_x are the neighbors of node i and node x, respectively.

Therefore, the average edge-clustering coefficient is computed using Eq. (2):

$$ECC_i = \frac{\sum_{x=1}^{m} ECC_{i,x}}{m} \tag{2}$$

where m is the number of neighbors for node i and ECC_i is the average edge-clustering coefficient computed for node i.

3.1.1. The nodes with degree one

In the first step, the LSMD algorithm selects an unlabeled node with degree 1 as target node v, which its direct neighbor has a higher degree than other degree 1 nodes neighbor's degree. If its direct neighbor does not have any community's label (its label is equal to zero), both of them receive a new label and the flag of target node v is changed to "*Diffused*" mode. The new label is assigned to the target node and its first level neighbor as their community label. For example, suppose that a target node is initially selected from degree 1 nodes at the beginning step of the algorithm. The new label is created with *c=1*, and it is assigned to the target node and its first, second, and third level of neighbors, which satisfy required conditions, obtain the label $c =$ "1". After finishing label diffusion by this target node, LSMD selects another initial target nodes and the new label is set to "2. However, if the neighbor of target node previously had a label (any possible label value with higher than "0"), it is assigned to the target node and its flag is also changed to "*Diffused*" mode. In the second step, first level node diffuses its label to its direct neighbor nodes (the second level neighbors of the target node). For this purpose, the average edge-clustering coefficient for this node (ECC_i) is calculated using Eq. (2). Therefore, all neighbors of first level node which edge-clustering coefficient between them and first level node is higher than the ECC_i ($ECC_{i,x} \geq ECC_i$), will receive this label. In the third step, a node from second level nodes with the highest sum of $ECC_{i,x}$ is considered as a diffuser node. In fact, this ith node can also diffuse the community label to the third level nodes. As a conclusion for this step, the target node is assigned with a label (numerical value) then all first level, second level, and third level nodes which satisfy the mentioned conditions, are all assigned with the same label as target node, and after that, they are not checked any condition for assigning a label to these nodes in next steps.

At all steps of the algorithm, when a node performs the label diffusion to its neighbors, it changes its flag to "*Diffused*" mode in order to avoid of reselection as a diffuser in the next steps. It is possible that a node has received a label in earlier steps from its neighbors, but has not diffused its label yet. In this case, these types of nodes possibly diffuse their label to the direct and indirect neighbors in the second and third level if they are selected as a diffuser in any step of the algorithm. After performing all steps for a node v with degree 1, its diffusion process is terminated, and the other unlabeled degree 1 node is selected as a new initial target node to repeat the same steps. Since new target node is selected, the new label is computed for it. Due to the diffusion nature of LSMD, a node w may previously have received a community label, and another target node in next time may try to assign a different label to node w. In this case, the new label is assigned for this node and it is saved in an array of overlapping labels for this node w. After finishing the diffusion steps, the most appropriate community label from overlapping labels is selected by Eq. (4). It is necessary to mention that the LSMD algorithm

is not an overlapping community detection method. The pseudo-code of label assigning and diffusion for nodes with degree 1 is shown in Algorithm 1 as follows.

Algorithm 1. Label assigning and diffusion for nodes with degree 1

Input: Nodes grouped by their degrees
Output: All degree 1, labeled nodes with other several labeled nodes

1. Group nodes based on their degree.
2. Initialize each node with label equal to "0"
3. Label_counter ←1
4. Repeat step 4, 5 for each unlabeled node with degree 1
   - Select a target node which has not diffused label before and its neighbor has the highest degree
   If Label (target node == 0)
   Check first level node's label
   If Label (first level node ==0)
   – Label (target node) ← Label_counter
   – diffuse label to the first level neighbor
   Else if Label (first level node ≠ **0**)
   – Label (target node) ← Label (first level node)

   - Diffuse label to second level neighbors according to condition $\boldsymbol{ECC_{i,x} \geq ECC_i}$
5. Select a node from second level nodes with the highest sum of $\boldsymbol{ECC_{i,x}}$
   If its Flag = "Undiffused" then
   - Diffuse its label to the third level nodes if $\boldsymbol{ECC_{i,x} \geq ECC_i}$
   If any other node with degree 1 remains
   repeat step 2
   Else terminate algorithm
6. End

In Algorithm 1, Label_counter shows the recent new label for assigning to the current community. After finishing the diffusion process for nodes with degree 1, the label diffusion is started for the nodes with degree 2 and higher. In fact, the processing of these nodes is similar to degree 1 nodes, but there is a little difference in selecting and diffusing the label. Among these nodes, the LSMD algorithm must select a node with degree 2, which sum of its neighbors' degree is higher than other nodes. This selection method is also used for nodes with degree 3, and so on.

3.1.2. The nodes with degree two and higher

Before starting this phase, the sum of the degree of neighbors, S_i, for a node i is computed by Eq. (3):

$$S_i = \sum_{w \in N(i)} |N_w| \tag{3}$$

where $|N_w|$ is the degree of node w, and $N(i)$ is neighbors of node i. In the beginning, between the nodes with degree 2, the node which both of its neighbors have the same label is selected as the target node v. In this case, because the selected target node has label equal to "0", it receives the label of its neighbors and diffuses it to the second and third level nodes; related steps are explained below. However, for a node i, if $ECC_i = 0$, this node only receives label from its neighbors, but it cannot diffuse label to other nodes at this step.

To determine the target node's label, the existing labels around it must be checked. If all the neighbors around the target node have labels, the label diffusion is not performed. However, if some neighbors of target node v have community labels, *Connectedness* $(v)_{C_i}$ is considered as a measure to select the most appropriate community label for the target node. This measure must be computed for each community C_i around the target node v, using Eq. (4) which presents amount of connectivity and effectiveness of the community C_i on target node v. If there still some nodes without any community label (these nodes have label equal to "0"), they are temporarily imagined as an individual community

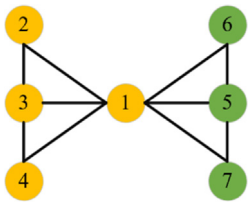

Fig. 1. A sample network for comparison Jaccard's Index, Intimacy and Influence measures.

with *"unknown or zero"* label to evaluate their connectedness.

$$Connectedness\ (v)_{C_i} = \sum_{w=1}^{|\forall w \in C_i, w \in N_v|} \deg w + |N_w \cap N_z| \ where\ \forall w, z \in C_i,\ w,\ z \in N_v \tag{4}$$

In Eq. (4), w and z are the neighbors of target node v which they belong to the community C_i, N_v is the set of neighbors of target node v, and N_w and N_z respectively are the set of neighbors of node w and z.

For all communities around the target node, Eq. (4) is calculated. The higher *Connectedness* value reveals the higher connectivity of the target node with the desired community. If multiple nodes have equal *Connectedness* value, the label of a node with a higher degree is assigned to the target node v.

On the other hand, if none of the neighbors of the target node have a label, the target node v acquires a new label. Once a label for target node v is specified, this label is possibly diffused to other neighbors. At first, the influence of the target node over its neighbors is computed using Eq. (5). If the influence of target node v over neighbor node w is equal or higher than the average influence of target node on its neighbors, then the node w (a first level neighbor) receives the target node's label. In addition, the common neighbors of target node v and node w receive this label too. Moreover, if node w previously has a different label from the current one, this label is also assigned to it as an overlapping community label.

$$Influence_{v,w} = \frac{(|N_v \cap N_w| + 2).\ |N_v|}{|N_w| + 1} \tag{5}$$

Here, N_v and N_w are the set of neighbors of node v and node w, respectively. In fact, Eq. (5) is a modified version of Jaccard's index [45] and Intimacy measure [17]. However, from the social point of view, our Influence measure is more realistic than the two mentioned criteria. In real networks, two connected nodes have different influences on each other, and this concept is different from nodes similarity. For example, consider the network of Fig. 1 with 7 nodes and 10 edges. In Fig. 1, the Jaccard's index (1,5) = Jaccard's index (5,1) = 0.4, Intimacy (1,5)= 1.33 and Intimacy (5,1)= 1, $Influence_{1,5} = 3.5$ and $Influence_{5,1} = 1.14$. It reveals that the influence value of node 1 over node 5 is more prominent in our proposed Influence measure Eq. (5).

As mentioned above, in order to assign a label to the first level and second level nodes the target node diffuses its label to the all first level neighbors and some second level neighbors based on Eq. (5). The first level neighbors which have currently received the label of the target node also must diffuse the obtained label to the second level neighbors. For this purpose, if these recently labeled nodes of first level node have common neighbors with each other, all common neighbors in the second level directly receive the target node's label. If some of the common neighbors previously have received a label and it is different from the current label, then this new label is also assigned as an overlapping label for them.

Moreover a node z with flag = "*undiffused*" and highest edge-clustering coefficient is selected between the second level nodes of target node to diffuse the target node's label to its neighbors y (in third level) if it passes one of two conditions: (1) If the

influence value among node z and its neighbor is equal to or greater than the average influence value node z, then the label is diffused to the related neighbor node in third level. (2) If the $ECC_{z,y} \geq ECC_z$ then the neighbor node y will receive the label from z. If all of the neighbors of a current node have labels, then label diffusion does not occur, and the algorithm is continued with another node.

After repeating these steps for all nodes with label equal to "0", the diffusion steps are completed. Then for some nodes which have received more than one label, the LSMD algorithm must select the most appropriate label between the overlapping labels. In the following section, this phase is described in detail.

As a result, the aim of the diffusion step is summarized as follows. At first, the label of the target node must be specified. There happen two case for assigning label to the target node. The first case is to generate new label, and the second case is to check label of neighbors of the target node and assign the most suitable label to the target node if its neighbors have label. After that, label of the target node is diffused to the first, second, and third level of neighbor nodes based on some conditions.

3.2. Choosing the most appropriate community label from overlapping labels

During the label diffusion phase, a node may receive different labels from several communities. Since the proposed LSMD algorithm is a non-overlapping community detection algorithm, only one label is possible to be assigned to each node. Therefore, it is necessary to select the best-fitting label as an individual label for a node with various labels. In addition, the existence of different communities around some nodes can be both due to the overlapping nature of some nodes or because of the nature of the algorithm that makes several initial communities in previous steps, whereas most of these initial communities may be merged later. To select the best label for a node v, the $Connectedness(v)_{C_i}$ is separately computed for each label (community) around node v using Eq. (4). Finally, the highest computed *Connectedness* value is considered as the best label for node v. After finishing these steps, initial communities are formed. In order to get more dense and accurate communities, we utilize merge step.

3.3. Merge phase

Through previous steps, initial communities were discovered, but some of these initial communities should be merged to form final communities. Therefore, the merge phase is classically performed to merge some weakly connected communities. Unlike the other methods that use the modularity criterion for merging initial communities, in this paper, a new simple approach is used to merge initial communities.

At this point, the number of nodes in each community is determined first. The largest community is then ignored to avoid its effect on small communities. For the remaining communities, the average size of communities is calculated. Communities with size less than average size are considered as small communities. Then for a selected small community, Eq. (6) is computed for each of its nodes as follows.

$$Candidate\_node\_score(v) = |N_v| + \sum_{w=1}^{|N_v|} Influence_{v,w} \tag{6}$$

In Eq. (6), N_v is v's neighbors and $|N_v|$ is the degree of node v and $\sum_{w=1}^{|N_v|} Influence_{v,w}$ is calculated from Eq. (5) and represents the sum of the influence of node v on all of its neighbors. The higher Candidate_node_score value for a node, the best representative node for its community. After selecting the representative node, the Candidate_node_score for all of the selected candidate node's neighbors is also calculated.

A neighbor node whose *candidate_node_score* value is higher than the value of the other neighbors and higher than the representative node's is selected. Then, if selected neighbor's degree is higher than degree of the representative node and also has a different label from the current label of the representative node, then representative node will receive its neighbor's label. Then the new received label by the representative node is also applied to all the nodes in its own community. By changing the label of representative node in a community, the label of the entire nodes in this community also needs to be changed to the new selected label. Therefore, the two related communities merge and create a new one. All of these steps is shown in Fig. 2.

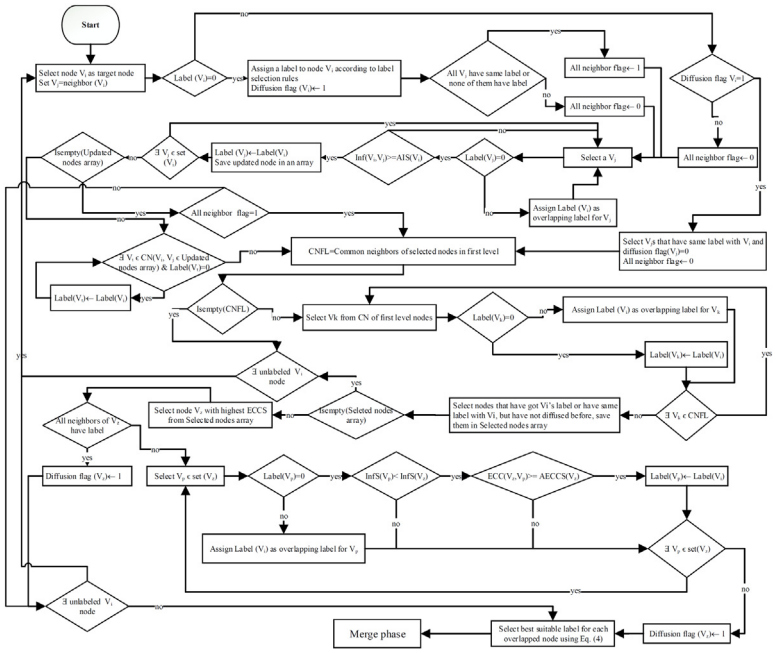

Fig. 2. The flowchart of proposed LSMD method (N(Vi) = neighbor (Vi), Inf = Influence, AIS = Average influence of node Vi, ECC = Edge clustering coefficient, CN = Common neighbors, ECCS = edge clustering coefficient, InfS = sum of influences, AECCS = Average edge clustering coefficient, CNFL = Common neighbors of selected nodes in the first level).

Table 1
Zachary's Karate club nodes grouped by their degrees.

Group	Nodes
Degree 1	12
Degree 2	10 13 15 16 17 18 19 21 22 23 27
Degree 3	5 11 20 25 26 29
Degree 4	6 7 8 28 30 31
Degree 5	9 14 24
Degree 6	4 32
Degree 9	2
Degree 10	3
Degree 12	33
Degree 16	1
Degree 17	34

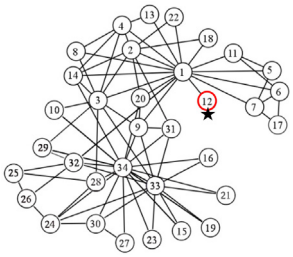

Fig. 3. The target node selection from degree 1 nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.4. Example

To understand more details about LSMD algorithm, a step-by-step example on Zachary's Karate club dataset [46] is described. The Zachary's Karate Club consists of 34 nodes,78 edges, and 2 communities. Each step of the algorithm is visualized in different figures.

According to the description of the proposed LSMD algorithm, it starts with the lowest degree. Table 1 shows the grouping of the nodes based on their degree. The LSMD method starts from the lowest group, the node with degree equal to 1. In Zachary's Karate club dataset, the only node with degree 1 is the node 12, which is connected to node 1. Both nodes 12 and 1 initially do not have labels. Therefore, at the first step, node 12 is considered as the target node, which is shown in Fig. 3. It receives a new label (red-colored label) and then diffuses its label to node 1, in the first level. Fig. 4 shows this step. The blue arrow indicates the diffusion direction, and the black colored star shows that the node has diffused its label and should not be selected in the next steps, again (flag = diffused).

The current community label should be diffused from first level node to its neighbors in second level. For this purpose, the average of edge-clustering coefficient (ECC_i) for node 1 must be calculated. The $ECC_1 = 0.0349$, thus the nodes 2, 4, 5, 8, 11, and 14 receive the label of node 1 because their edge-clustering coefficient with node 1 is higher than 0.0349. However, the nodes 18, 22, and 13 have a lower edge-clustering coefficient than 0.0349, and they do not receive label from node 1 in this stage. The edge-clustering coefficient between node 1 and its neighbors is shown in Table 2. Fig. 5 shows the new status of the network after diffusing label from node 1 to some proper neighbors. After label diffusion, the flag of node 1 is changed to "Diffused" mode.

After finishing label diffusion by node 1, in the next step, the best neighbor of node 1 (best second level neighbor of the target node 12) may also diffuse the label to its neighbors. Therefore, the best neighbor of node 1 is node 4, because it has the highest edge-clustering coefficient sum. The sum of the edge-clustering coefficient for each second level node is shown in Table 3.

The node 4 with highest edge-clustering coefficient is selected to diffuse the label to third level neighbors of the target node 12. Nodes 1, 2, 3, 8, 13, and 14 are neighbors of node 4 that only nodes 3 and 13 do not have labels. Hence, the diffusion

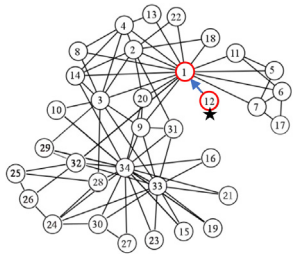

Fig. 4. The first step of label assigning to target node.

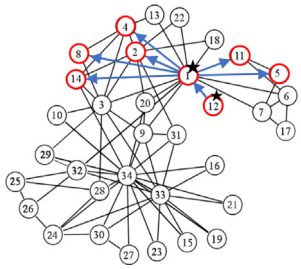

Fig. 5. Label diffusion from first level neighbor to second level neighbors.

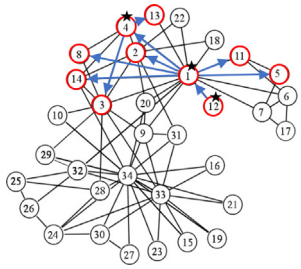

Fig. 6. Label diffusion from second level neighbor (node 4) to third level neighbors.

Table 2
Edge-clustering coefficient between node 1 and its neighbors.

Node	Edge-Clustering coefficient
2	0.048611
3	0.03125
4	0.052083
5	0.041667
6	0.03125
7	0.03125
8	0.046875
9	0.0125
11	0.041667
12	0
13	0.03125
14	0.0375
18	0.03125
20	0.020833
22	0.03125
32	0

Table 3
Second level nodes with their sum of edge-clustering coefficient. (SECC indicates the sum of edge-clustering coefficient)

Node	4	2	8	14	5	11
SECC	**0.501**	0.465	0.330	0.264	0.236	0.236

step occurs to the neighbors of node 4, which does not have any label (nodes 3 and 13). Fig. 6 demonstrates the new status of the network after diffusing label from node 4 to its neighbors. After that, the flag of node 4 is changed to "*Diffused*" mode.

Based on the aforementioned steps, the diffusion process from target node 1 is terminated, and the other node must be selected to continue the algorithm. There is no other degree 1 node. Hence, the LSMD selects a node with degree 2. Among the nodes with degree 2, LSMD prefers to start with a node with the highest S_i

Table 4
Nodes with degree 2 and the sum of their neighbors' degree (S_i).

N	15	16	19	21	23	10	18	22	13	27	17
S_i	29	29	29	29	29	27	25	25	22	21	8

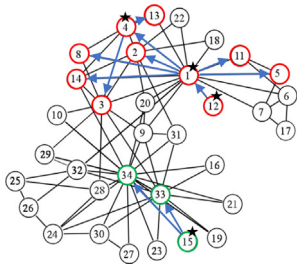

Fig. 7. Identifying node 15 as the new target node among nodes with degree 2.

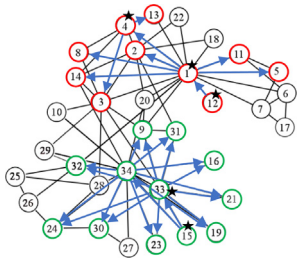

Fig. 8. Label diffusion from first level neighbors of node 15 (nodes 33 and 34) to the second level of neighbors.

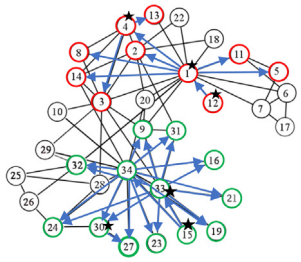

Fig. 9. Label diffusion from second level neighbor node of node 15 (node 30) to the third level neighbors.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

value. According to Eq. (3), LSMD starts with node 15 which has the highest S_i. Table 4 lists the nodes of degree 2 based on S_i value.

Since there are several nodes with the highest S_i, LSMD can start with each of them. In this example, we start with node 15. Node 15 does not have any label, so it is vital to check its neighbors' label (first level neighbors). Nodes 33 and 34 are both neighbors of node 15 and do not have labels. Therefore, node 15 is considered as the target node v and receives a new label (green label). Then it diffuses this green label to its neighbors based on the node's influence ($Influence_{v,w}$) measure. The influence of node 15 on its neighbor is calculated via Eq. (5). The average influence of node 15 on its neighbors is equal to 0.3974. The influence of node 15 on node 33 is 0.4615, and node 34 is 0.330. Hence, node 33 receives the label of node 15, because 0.4615 is higher than the average value (0.3974). In addition, the common neighbors of node 15 and node 33 also get this label. Therefore, this label is also assigned to the node 34. The updated network after selecting node 15 as a new target node is shown in Fig. 7.

After diffusing new label to first level neighbors, they can also diffuse this label to the second level neighbors. It is initially checked whether all the first level nodes have common neighbors or not? The common neighbors of two nodes 33 and 34 are 9, 15, 16, 19, 21, 23, 24, 30, 31, and 32. Among these nodes, some nodes which have an "unknown" label (do not have label) receive the current target node's label, and nodes which have a label different from the current label, store it as an overlapping community label.

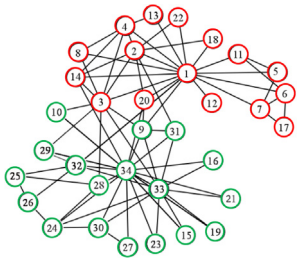

Fig. 10. The result of community detection by LSMD algorithm on Karate network.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 5
Second level nodes with their sum of edge-clustering coefficient (target node is node 15). (SECC indicates the sum of edge-clustering coefficient)

N	9	16	19	21	23	24	30	31	32
SECC	0.22	0.07	0.07	0.07	0.07	0.21	**0.31**	0.17	0.2

The updated network is shown in Fig. 8. After this step, the flag of node 33 is changed to "*diffused*" mode. From second level neighbors, node 30 is selected for label diffusion, because it has a higher edge-clustering coefficient sum to diffuse its label to third level neighbors. Table 5 summarizes the sum of edge-clustering coefficients of each node in second level.

Among the neighbors of node 30, only node 27 is not labeled. Because the total influence of node 27 is less than the total influence of node 30 (total influence of node 27 is 1.533 and total influence of node 30 is 9.008) and since the edge-clustering coefficient between node 30 and 27 (0.1250) is higher than the average edge-clustering coefficient of node 30 (0.0776), consequently node 27 takes label of node 30. Now, the diffusion process of node 30 is terminated (flag = "*diffused*") and the other target node must be determined (Fig. 9). The new target node is node 16. Next, we only show the selected target node and the diffused label to its first, second, and third level neighbors in Table 6.

Now, the diffusion process for all nodes is finished. At the next, among overlapping labels for some nodes such as nodes 5, 6, 14, 20, and 31, only one label must be selected. Therefore, based on Eq. (4) the best label for nodes is determined as follows: nodes 5 = Red, 6 = Red, 14 = Red, 20 = Red, and 31 = Green.

After selecting the appropriate label from the overlapping labels, the merge phase is executed. In this example, there are three communities, including red, green, and purple. The green community has 18 members, the red community has 14 members, and the purple community has 2 members. The green community is excluded because it is the largest. The average size of communities is computed based on the remaining communities. Therefore, the average size is (14+2)/2 = 8. Consequently, only the purple community's size is less than the average value. So, it is selected as a small community. This community has two nodes 7 and 17. Then, node 7 is selected as the representative node for the purple community because it has the best *candidate_node_score* (15.14). At the next, among neighbors of node 7, because node 1 in the red community has a better *candidate_node_score* than the node 7, and it has a higher degree than node 7, for this reason the label of node 7 is changed to the red label. Therefore, since node 7 gets the red community's label, node 17 has to get the same Red community label. The node 17 is therefore merged with the Red color community. The final result of the LSMD algorithm is shown in Fig. 10.

3.5. Time complexity

The proposed algorithm consists of three main sections including the label diffusion step (detecting initial communities), selecting the most appropriate label between overlapping labels and merging phase.

At first, we group nodes based on their degrees. For this purpose, a head vector with a length of maximum degree is used.

Table 6
Selected target nodes and their diffusion process on the example dataset.

Target node	Diffused label to the FLN[a]	Selected node from SLN[a] to diffuse	Diffused label to the SLN[a]	Diffused label to the TLN[a]	Flag = "diffused"
16(Green label)	All have label	9	All have label	No	16, 9
19(Green label)	All have label	24	All have label	28	19, 24
21(Green label)	All have label	32	All have label	25,26,29	21, 32
23(Green label)	All have label	31	All have label	No	23, 31
10(Green label)	All have label	No	All have label	No	10
18(Red label)	All have label	8	All have label	No	18, 8
22(Red label)	All have label	3	All have label	No	22, 3
13(Red label)	All have label	No	All have label	No	13
27(Green label)	All have label	33	All have label	No	27, 33
17(Purple label)	6,7 (Purple label)	No	All have label	No	17
20(Red label)	All have label	No	All have label	No	20
The rest nodes	All have label	No	All have label	No	The rest nodes

[a]FLN: first level neighbors, SLN: second level neighbors, TLN: third level neighbors.

Table 7
Properties of 17 real-world datasets used for tests.

Dataset	N	M	CN	K
Zachary's Karate Club [46]	34	78	2	4.59
Dolphins [47]	62	159	2	5.13
U.S Political books [48]	105	441	3	8.4
Football [49]	115	613	12	10.66
DBLP [50]	317080	1049866	13477	6.62
Amazon [50]	334863	925872	75149	5.53
YouTube [50]	1134890	2987624	8385	5.26
Email [51]	1133	5451	–	10.62
Net-science [52]	1589	2742	–	3.75
Power Grid [53]	4941	6594	–	2.66
CA-GRQC [54]	5242	14490	–	5.53
Collaboration [55]	8361	15751	–	3.77
CA-HEPTH [54]	9877	25985	–	5.26
PGP [56]	10680	24316	–	4.55
Condmat-2003 [23]	31163	120029	–	7.7
Email-Enron [57]	36692	183831	–	10.02
Condmat-2005 [58]	40421	175691	–	8.7

Each cell of vector is referenced to a list of nodes. For example, a node with degree k is directly inserted to the end of k^{th} list. The complexity of this stage is $O(n)$. When a new node is added to the network, it can be easily grouped in the network.

The LSMD algorithm computes the edge-clustering coefficient and node influence for some nodes in the steps of the algorithm. Eqs. (1) and (5) are computed for insignificant number of nodes (n') during the steps of algorithm where n'<< n in graph. If k is the average degree of network, the computational time of edge-clustering coefficient and node influence measure is $O\left(n'k\right)$ where $n' \ll n$ and $O\left(n'k\right) < O\left(n\right)$ because it is used to compute for some nodes in some conditions.

On the other hand, we have p nodes with degree 1 that each of them has one neighbor. The initial label assignment and diffusion is started with p' nodes of p where $p' \ll p$. Actually, the two nodes with degree 1 and with distance ≤ 3 distinctly get the same label and also some other nodes with degree 1 may get that label in the next steps. Therefore, only a small number of nodes with degree 1 can start the label assignment and diffusion process. The first level node has k neighbors which it should check them and diffuse label to its neighbors. In addition, the best node v in second level diffuses label to its k neighbors in the third level. So total time complexity for performing all operations for p' nodes with degree 1, is: $O\left(p'(k+k)\right)$. Therefore, the time complexity of this step is $O\left(p'k\right)$ where $O\left(p'k\right) < O\left(pk\right) < O(n)$. In addition, suppose we have q nodes with degree two and higher. Only q' nodes of q are selected to diffuse label to its neighbors where $q' \ll q$. In Eq. (4), for some nodes in q' nodes, we compute intersection of nodes in each community which have edges with target node, which time complexity for each node is $O\left(k\right)$. Therefore, the diffusion process from these q' nodes to their neighbors is $O(q'k)$. The total time complexity for diffusion process form p' and q' nodes is $\left(p'k + q'k\right) < O\left(n\right)$ because label checking and labeling diffusion are not completely duplicated due to the use of flags for each node. We consider $O(n)$ for these steps for easy expression.

In choosing the label from overlapping community labels, we have d overlapping nodes that each of them has received c community labels, and in each community, l nodes are connected with the selected overlapping node. So, the time complexity is $O(dcl) \ll O(n)$.

The time complexity of the merge phase consists of two main steps. At first, the size of the community is calculated. The time complexity for finding the size of communities is equal to $O(n)$ because we should check all of n nodes. In the next step, for each small community with n_i members we calculate Eq. (6). Then we should select a candidate node between these n_i nodes and check its k neighbors. The Time complexity for merging step is $O\left(\frac{p}{2} + \frac{n''k}{2}\right) < O(n)$ where p is the number of nodes in all communities except the largest community; and $n'' = \sum_{i=1,\ldots,c} n_i$ is the number of nodes in small communities. Accordingly, the overall time complexity for the LSMD algorithm is: $T\left(n\right) = O\left(n + n'k + dcl + \frac{p}{2} + \frac{n''k}{2}\right) \leq O(nk) \cong O(nk)$. Consequently, the proposed algorithm is meaningfully the fastest community detection algorithm among compared linear algorithms.

4. Experiments

The experimental results are performed on several standard real-world and synthetic networks. The description of real-world networks is shown in Table 7. The proposed algorithm (LSMD) is compared with basic and recently presented algorithms such as the CNM [12,30], LPA [15], NIBLPA [18], LPA-Intimacy [17], Infomap [59], LCCD [42], ECES [10], G-CN [21], DA [44], Louvain [22], RTLCD [32] and Stepping LPA-S [43]. All the experiments were performed on a desktop computer with core-i5 (3.40 GHz) processor and 12GB memory in MATLAB (2017) and Windows10 OS.

4.1. Real-world datasets

To evaluate the performance of the proposed LSMD algorithm, 17 real-world networks, including small, medium, and large-scale datasets are used. The smallest dataset is the Zachary's Karate Club dataset with 34 nodes, 78 edges, and 2 communities. The largest dataset is the YouTube dataset with 1,134,890 nodes and 2987624 edges. The detailed information about datasets are shown in Table 7. The columns N, M, CN, and K of Table 7 represent the number of nodes, edges, number of communities, and the average degree of network, respectively. The seven datasets are ground truth datasets with predefined number of communities.

Table 8
Parameters of LFR datasets.

Parameters	Description
N	Number of nodes
Min K	Minimum degree of nodes
Max K	Maximum degree of nodes
μ	Mixing parameter
Min c	Number of nodes within the smallest community
Max c	Number of nodes within the biggest community
γ	Node degree distribution exponent
β	Community size distribution exponent

Table 9
Properties of generated LFR synthetic networks.

Network	N	Min K	Max K	Min C	Max C	γ	β	μ
LFR1	5000	20	50	10	50	2	1	0.1-0.8
LFR2	5000	20	50	20	100	2	1	0.1-0.8
LFR3	10000	20	50	10	50	2	1	0.1-0.8
LFR4	10000	20	50	20	100	2	1	0.1-0.8
LFR5	20000	20	50	10	50	2	1	0.1-0.8
LFR6	20000	20	50	20	100	2	1	0.1-0.8
LFR7	50000	20	50	10	50	2	1	0.1-0.8
LFR8	50000	20	50	20	100	2	1	0.1-0.8

4.2. Synthetic networks

Besides the real-world networks, we have used 8 types of configurations of LFR[2] synthetic datasets that each of them is created with eight different parameters of μ. Synthetic datasets are created with the LFR benchmark [60]. LFR benchmark is one of the popular artificial benchmarks for creating synthetic networks. One of the important parameters of LFR is the mixing parameter μ which indicates the probability of the nodes being connected to other nodes among communities. Therefore, the variation of mixing parameter (μ) can cause differences in connecting the structure and topology of networks. If μ is close to 0, it means that the communities' structure is distinct, and when it is close to 1 the network has similar behavior like random network, and communities' structure is indistinct accordingly. The parameters for LFR synthetic datasets are shown in Table 8, and the properties of the generated datasets are shown in Table 9.

4.3. Evaluation parameters and measures

In order to evaluate the performance of algorithms on real-world and LFR synthetic datasets, normalized mutual information (NMI), F-measure (F-score), and Modularity metrics are used. For ground truth datasets, the NMI and F-measure are used, and for other datasets, the modularity measure is used to evaluate the obtained results. It should be noticed that the NMI and F-measure show the real accuracy in obtained results, but the modularity measure is not necessarily the most accurate measure.

NMI is a measure for evaluating the similarity between the results of the communities detected by the algorithm and the actual communities, with values between 0 and 1. The closer this value is to 1, the more accurate the algorithm is [61]. The NMI is obtained via Eq. (7).

$$\mathrm{NMI}\,(\mathrm{A},\mathrm{B}) = \frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B} N_{ij}\log\left(\frac{N_{ij}N}{N_i N_j}\right)}{\sum_{i=1}^{C_A} N_i \log\left(\frac{N_i}{N}\right) + \sum_{j=1}^{C_B} N_j \log\left(\frac{N_j}{N}\right)} \tag{7}$$

In Eq. (7), NMI (A, B) shows the similarity of partitions of real communities and discovered communities based on the information theory. *A* represents the real communities, and *B* represents the communities discovered by the algorithm. C_A and C_B show the real number of communities and the number of detected communities, respectively. *N* is a confusion matrix where the rows denote real communities, and columns denote the discovered communities. N_{ij} is the number of common nodes between the real community *i* in set *A* with the discovered community *j* in set *B*. N_i is the sum of the row *i* in the matrix N_{ij} and N_j is the sum of the column *j* in the matrix N_{ij}.

On the other hand, modularity is a measure for evaluating the density of a community's inner edges relative to the community's outer edges (edges between communities). Modularity measures the quality of discovered communities by comparing the number of edges within a community with the expected value for a random network of the same size and degree [30]. The modularity value for a network is between −1 and 1. The closer this value to 1, the better the quality of partitioning is. Modularity is calculated via Eq. (8).

$$\mathrm{Q} = \frac{1}{2\mathrm{m}}\sum_{\mathrm{i,j}\epsilon\mathrm{V}}\left(\mathrm{A_{ij}} - \frac{\mathrm{d_i d_j}}{2\mathrm{m}}\right) \times \delta(\mathrm{c_i},\ \mathrm{c_j}) \tag{8}$$

In Eq. (8), Q shows the total modularity of a network. A higher value of Q for discovered communities in a network shows the ability of algorithm in finding the dense and more connected nodes within each community. *m* shows number of edges in network, and *A* is the adjacency matrix of the network. If there is an edge between *i* and *j*, then $A_{ij} = 1$, otherwise it is equal to zero. $\mathrm{d_i}$ and $\mathrm{d_j}$ show the degree of nodes *i* and *j*, respectively. c_i and c_j are labels of nodes *i* and *j*, respectively. $\delta\left(\mathrm{c_i},\ \mathrm{c_j}\right)$ is Kronecker delta and if two nodes *i* and *j* both are in the same community, then $\delta\left(\mathrm{c_i},\ \mathrm{c_j}\right) = 1$, otherwise it is equal to zero. Eq. (8) attempts to show how connected are nodes in a community by checking the edges between nodes that share the same label.

The third measure is F-measure, which shows the accuracy of the obtained results on a dataset. It is defined as the harmonic mean of precision and recall [62–64]. Precision, recall, and F-measure are used in classification problems. F-measure is a balance between precision and recall, and it has a value between 0 and 1. If the method's precision and recall are both equal to 1, then the F-measure will have its highest value. F-measure is calculated via Eq. (11). Eqs. (9) and (10) define the precision and recall measures, respectively. Precision and recall both have a result between 0 and 1.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$F - measure = 2 \times \frac{precision \times recall}{precision + recall} \tag{11}$$

In Eqs. (9) and (10), if we imagine being member of a community as positive, TP indicates true positive results. By true positive, we mean the positive results (positive class) that the algorithm has correctly predicted them. In community detection TP shows the correctly identified members of a community. FP indicates false positive results. FP is the result that the algorithm incorrectly has predicted the positive class. Nodes which are members of a selected community, but the algorithm has not predicted them as member of the selected community (incorrectly identified), and FN shows the false negative results. By false negative we mean the result that algorithm incorrectly predicts the negative class (incorrectly rejected). It means that nodes that are not member of a selected community, but are predicted by algorithm as members of the selected community.

4.4. Experimental results on real-world datasets

The performance of our proposed algorithm (LSMD) compared to other algorithms is evaluated on 17 real-world and 8 artificial networks. The obtained results are evaluated based on NMI, F-measure, and modularity metrics and are represented in Tables 10 to 13. In these tables, Q and CN indicate the modularity value and number of communities, found by the algorithms. In addition, it is necessary to mention that the modularity measure

[2] Lancichinetti–Fortunato–Radicchi.

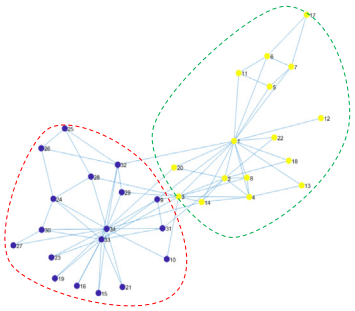

Fig. 11. Detected two communities by LSMD in Karate dataset with NMI = 1.

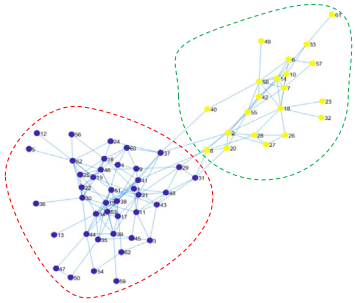

Fig. 12. Detected two communities by LSMD in Dolphins dataset with NMI = 1.

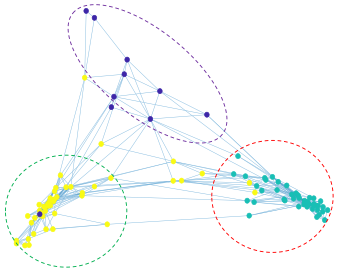

Fig. 13. Detected three communities by LSMD in Polbooks dataset with NMI = 0.59.

Fig. 14. Discovered 12 communities by LSMD in Football dataset with NMI = 0.93.

is not an exact measure to evaluate the performance of the algorithms, but that is the only good, widely used, and relatively reliable measure in networks with unknown communities. For example, in the Zachary Karate club dataset the label of community for each node is clear. Therefore, in the best case, its NMI or F-measure is equal to 1, and modularity is equal to 0.371. However, some algorithms have the wrong detection, which reduces the NMI or F-measure to 0.89 whereas modularity is increased to 0.401. Consequently, in ground truth networks, NMI and F-measure are the best measures for evaluation.

At first, the evaluations are performed on ground truth datasets using NMI measure to show the accuracy of algorithms. Among the 17 real-world datasets, seven of them are ground truth datasets which are grouped in small and large networks in Tables 10 and 11, respectively. Small networks consist of Karate, Dolphins, Polbooks, and Football datasets, and large networks are as follows: DBLP, Amazon, and YouTube datasets. According to the descriptions in [65], in DBLP, Amazon, and YouTube datasets, communities with less than three nodes are removed from the ground truth file, and all the nodes are not included in the ground truth file. NMI results for Stepping LPA-S are taken from [43], and NMI results for DA algorithm are taken from Ref. [44], and all are represented in Table 10. To compare our proposed algorithm's NMI in DBLP, Amazon and YouTube datasets with CNM, its NMI results are taken from reference [21]. The NMI for LPA and Infomap on the YouTube dataset is taken from [21].

Table 10 shows the results of different algorithms evaluated with NMI measure on small networks. It is revealed that in small datasets the proposed LSMD algorithm has the best NMI results compared to other algorithms. The obtained results show that the LSMD has identified the communities completely true and without any wrong detection in Karate and Dolphins datasets. Besides this, in the Dolphins dataset, second best NMI is Stepping LPA-S's NMI with NMI = 0.88. It should be noted that between the other 12 compared methods, none of them has NMI = 1, except the LSMD method. In the Polbooks dataset, LSMD also has the highest NMI value between all other algorithms, and Stepping LPA-S has the second best NMI. In the Football dataset, LSMD and DA have the best NMI values, and the second best NMI belongs to Stepping LPA-S algorithm.

In Table 11, the NMI results of algorithms are evaluated on large-scale networks. In the DBLP dataset, the highest NMI belongs to LPA-Intimacy. The LSMD algorithm has obtained the highest NMI values in Amazon and YouTube datasets. Our proposed method was performed in 10 different runs, and in all runs, its result was the same. Moreover, some methods such as CNM, NIBLPA, and LPA-Intimacy are not executed on YouTube datasets after 2 days, and their NMI value is "unknown". Among the 7 ground truth datasets, the LSMD in 6 datasets has obtained the best NMI results, and it obviously shows the significant accuracy of LSMD in both small and large datasets. The detected communities in Karate, Dolphins, Polbooks, and Football datasets are shown in Figs. 11 to 14.

4.5. F-measure evaluation on real-world datasets with ground truth

F-measure is a more accurate criterion than NMI in evaluating the accuracy of detected communities. Since F-measure is the harmonic mean of precision and recall, the obtained results for F-measure can show the ability of algorithm in specifying correct and incorrect results. In classification problems, one method could be able to find true results, but we should consider how this method was successful in finding the incorrect results too. For example, in the classification of sick and healthy people, precision indicates that among predicted sick people how many are actually sick, and recall indicates that from all sick people, how many of them we have correctly predicted. We use this concept to measure how algorithms were accurate and successful in finding true members of communities they have detected. Since F-measure is calculated for each class or community, in order to compare algorithms with each other, we calculate the average F-measure for each algorithm on a specific dataset for 10 different runs. Results of F-measure are listed in Table 12 for seven ground truth datasets.

The LSMD and LPA-Intimacy have F-measure equal to 1 in Karate dataset, and it means both of them have precision and recall equal to 1. In the Dolphins dataset, only LSMD has F-measure equal to 1, and it has detected all the communities correctly. The second-best F-measure belongs to RTLCD on the Dolphins dataset, but it has a significant difference with LSMD. In the Polbooks dataset, LSMD outperforms other algorithms, and RTLCD has the second-best F-measure. LSMD has obtained

Table 10
NMI results obtained from experiments on small real-world networks with ground truth. (Highest values on each row are bolded.)

Dataset	CNM	Infomap	LCCD	ECES (s=1)	LPA	NIBLPA	LPA-Intimacy	Stepping LPA-S	RTLCD	Louvain	DA	G-CN	LSMD
Karate	0.69	0.7	0.71	0.63	0.68	0.58	**1**	0.92	**1**	0.59	**1**	0.83	**1**
Dolphins	0.55	0.54	0.53	0.49	0.53	0.50	0.63	0.88	0.45	0.52	0.51	0.54	**1**
Polbooks	0.53	0.50	0.52	0.50	0.52	0.53	0.54	0.57	0.48	0.45	0.50	0.53	**0.59**
Football	0.75	0. 919	0.88	0.75	0.81	0.72	0.86	0.92	0.51	0.89	**0.93**	0.87	**0.93**

Table 11
NMI results obtained from experiments on large-scale real-world datasets with ground truth. (Highest values on each row are bolded.)

Dataset	CNM	RTLCD	Infomap	LPA	NIBLPA	LPA-Intimacy	Louvain	ECES (s=1)	G-CN	LSMD
DBLP	0.16	0.41	0.47	0.49 ± 4	0.46	**0.61**	0.13	0.36	0.51	0.50
Amazon	0.11	0.72	0.51	0.53 ± 5	0.60	0.63	0.11	0.58	0.59	**0.715**
YouTube	–	–	0.13	0.07 ± 1	–	–	0.06	–	0.1	**0.19**

Table 12
Average F-measure results obtained from experiments on real-world datasets with ground truth. (Highest values on each column are bolded.)

Dataset	CNM	RTLCD	Infomap	ECES (s=1)	LPA	NIBLPA	LPA-Intimacy	Louvain	G-CN	LSMD
Karate	0.54	1	0.59	0.55	0.58	0.39	**1**	0.5	0.95	**1**
Dolphins	0.41	0.73	0.33	0.39	0.29	0.39	0.43	0.3	0.29	**1**
Polbooks	0.52	0.66	0.43	0.51	0.55	0.53	0.61	0.46	0.38	**0.7**
Football	0.38	0.60	0.81	0.63	0.6	0.47	0.68	0.74	0.76	**0.91**
DBLP	0.23	0.37	0.34	0.33	0.33	0.40	0.45	0.34	0.36	**0.49**
Amazon	0.28	0.74	0.39	0.37	0.66	0.66	0.71	0.51	0.69	**0.82**

Table 13
Experimental results on real-world datasets based on modularity measure.

	CNM		LPA		NIBLPA		LPA-Intimacy		Infomap		Louvain		G-CN		LCCD		LSMD	
Datasets	Q	CN	Q	CN	Q	CN	Q	CN	Q	CN	Q	CN	Q	CN	Q	CN	Q	CN
Karate	0.38	3	0.38	3	0.40	5	0.3715	2	0.4	3	0.41	4	0.3718	2	0.41	2	0.3715	2
Dolphins	0.49	4	0.46	5	0.43	5	0.5	4	0.52	5	0.51	5	0.51	5	0.52	4	0.3787	2
Polbooks	0.50	4	0.45	2	0.55	4	0.48	3	0.52	5	0.52	5	0.51	6	0.52	2	0.446	3
Football	0.54	6	0.54	9	0.5	8	0.60	10	0.58	12	0.60	10	0.57	12	0.60	12	0.58	12
DBLP	0.72	3301	0.60	30384	0.61	30986	0.70	12097	0.71	16921	0.81	225	0.70	22056	0.61	7903	0.65	17280
Amazon	0.87	6086	0.64	27959	0.67	44036	0.74	22549	0.75	17319	0.90	286	0.73	29484	0.22	9984	0.68	34304
YouTube	–	–	–	–	–	–	–	–	–	–	–	6026	–	40256	0.54	11306	0.42	9636
Condmat-2003	0.66	1916	0.53	3814	0.50	3725	0.69	2692	0.61	2540	0.72	98	0.61	3387	0.61	1909	0.57	3502
Email-Enron	0.51	1605	0.38	2112	0.12	2131	0.17	1423	0.52	2605	0.55	1463	0.11	1731	0.41	2311	0.39	1221
Email	0.48	17	0.34	77	0.40	149	0.35	500	0.52	60	0.54	15	0.36	624	0.56	58	0.42	142
Collaboration	0.72	193	0.69	1991	0.68	2389	0.70	1769	0.72	1083	0.86	638	0.70	1876	0.73	270	0.72	1421
PGP	0.85	190	0.77	1176	0.73	1882	0.76	633	0.75	964	0.87	101	0.79	911	0.74	532	0.59	643
Net-science	0.90	276	0.91	336	0.87	366	0.88	307	0.90	294	0.92	276	0.88	317	0.92	261	0.946	275
Power Grid	0.89	41	0.74	774	0.61	1356	0.71	523	0.79	496	0.84	57	0.64	678	0.86	238	0.793	446
CA-GRQC	0.76	419	0.70	793	0.72	951	0.74	609	0.75	719	0.76	398	0.72	781	0.72	904	0.771	456
CA-HEPTH	0.71	551	0.61	1255	0.60	1642	0.68	941	0.64	1110	0.70	452	0.60	1344	0.49	996	0.63	586
Condmat-2005	0.63	2253	0.50	4503	0.23	3523	0.58	2955	0.63	2966	0.71	1077	0.62	3906	0.63	2730	0.44	3952

F-measure = 0.7 in Polbooks dataset. In the Football dataset, LSMD also has a higher F-measure result than other algorithms, and LPA-Intimacy has the second-best F-measure. It proves that LSMD is more effective in finding true members in the Football dataset than other methods. LSMD with F-measure result 0.49 in the DBLP dataset has achieved the highest result between other methods. After LSMD, LPA-Intimacy has the second-highest result. In the Amazon dataset, LSMD has obtained F-measure = 0.82, which is in the first rank among compared algorithms. The analysis of Tables 10 to 12 reveals that LSMD has the best F-measure in all datasets with ground-truth among the compared algorithms whereas some of these algorithms have higher modularity than LSMD, while they have low NMI and F-measure results in comparison with LSMD. The results obviously show that LSMD finds the true communities, true community numbers, and more accurate results for ground-truth datasets.

4.6. Evaluation of modularity measure (Q) on real-world datasets

As we mentioned above, the best measures for evaluation are the NMI and F-measure in ground truth datasets. Because, as it is shown in Table 13, while discovered communities are 100% correct, the modularity necessarily is not highest. For example, the real modularity for Karate and Dolphins datasets respectively, is 0.371 and 0.378 in case of 100% correct community detection (CN = 2). However, the obtained modularity for some methods with the wrong detection is higher than real values. However, the only acceptable criterion for datasets without ground truth is the modularity criterion. The higher modularity value, the better quality of the detected communities. According to Table 13, in the Karate dataset, the highest modularity value belongs to LCCD and Louvain because of some wrong detections. For example, Louvain's F-measure is 0.5 with 4 detected communities. The best result belongs to the LSMD method with F-measure = 1 and modularity Q = 0.371. In the Dolphins dataset, the LCCD and Infomap have the highest modularity whereas their modularity is far from real modularity, which is equal to Q = 0.378 with F-measure = 1. In addition, the real number of communities in the Dolphins dataset is CN = 2, whereas LCCD finds 4 communities, and Infomap finds 5 communities. LSMD has the lowest modularity among all the algorithms, but it is equal to the real modularity without any wrong detection with F-measure = 1, NMI = 1, and CN = 2.

Needless to mention that modularity has the resolution limit problem. For example, in the Dolphins dataset, Infomap and LCCD have detected 5 and 4 communities, respectively with modularity

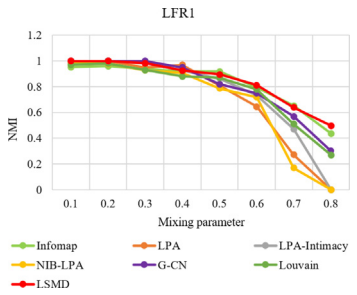

Fig. 15. NMI results obtained by Infomap, LPA, LPA-Intimacy, NIBLPA, G-CN, Louvain and LSMD in LFR1 dataset.

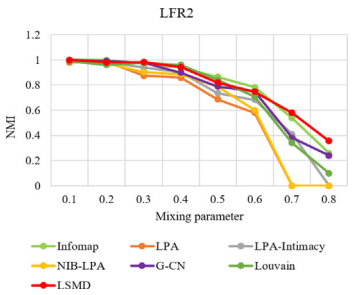

Fig. 16. NMI results obtained by Infomap, LPA, LPA-Intimacy, NIBLPA, G-CN, Louvain and LSMD in LFR2 dataset.

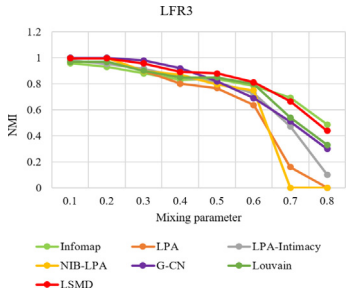

Fig. 17. NMI results obtained by Infomap, LPA, LPA-Intimacy, NIBLPA, G-CN, Louvain and LSMD in LFR3 dataset.

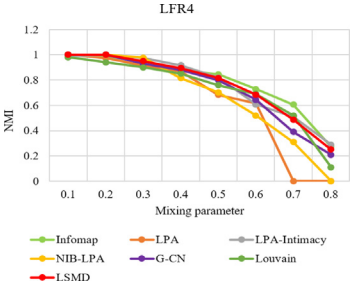

Fig. 18. NMI results obtained by Infomap, LPA, LPA-Intimacy, NIBLPA, G-CN, Louvain and LSMD in LFR4 dataset.

equal to 0.52. These algorithms have detected some small dense communities to obtain higher modularity, but it is the wrong detection. In the Polbooks dataset, NIBLPA with Q = 0.55 has the highest modularity value. However, the real modularity based on ground truth is Q = 0.4149 with CN = 3. LSMD has the closest modularity Q = 0.446 to real value with CN = 3, best F-measure, and better NMI than other algorithms.

In the Football dataset, the LCCD algorithm has the highest modularity (Q = 0.60) with 12 communities and NMI = 0.88. The real modularity value for Football is Q = 0.554. LSMD has also detected 12 communities with obtained modularity Q = 0.58, which is much closer to real modularity value than LCCD, Louvain and LPA-Intimacy algorithms. In addition, as mentioned in Tables 10 and 12, LSMD's NMI value and F-measure is higher than all compared algorithms. In large scale Amazon and YouTube datasets, our proposed LSMD algorithm has the best accuracy due to having the highest NMI and F-measure, and it has the closest modularity values to real modularity.

However, despite LSMD's modularity in Amazon and YouTube datasets, it has better performance than other algorithms in Amazon and YouTube datasets due to the highest F-measure and NMI values and closest CN number to real communities' number. Although Louvain has the highest modularity value Q = 0.9 with CN = 286 in Amazon dataset, but its NMI and F-measure values are lower than LSMD's. This result shows that modularity is substantially an untrusted measure for evaluations because Louvain has the weakest accuracy among all compared methods with the highest modularity values.

On the other hand, LSMD is also evaluated on several datasets without ground truth based on modularity measure. LSMD only has the highest modularity values on Netscience and CA-GRQC datasets. In most of these datasets, Louvain, CNM, and Infomap have higher modularity than other methods. However, the analysis of 7 ground truth datasets shows that none of them do not have better NMI and F-measure results than LSMD, and also their obtained CN values significantly is different with real CN values of ground truth. Therefore, we cannot discuss about actual modularity values. In general, LSMD tries to utilize the structure, similarities and relationships among nodes to get accurate results and modularity near to the real modularity of the network.

4.7. LFR synthetic datasets

In addition to the real-world datasets, in this section, we have performed experiments on LFR synthetic datasets based on NMI measure. For this experiment, eight LFR networks are created with different configurations with μ varying from 0.1 to 0.8. Figs. 15 through 22 show the NMI results comparison for 7 algorithms in 8 types of synthetic datasets.

For small values of mixing parameter (e.g., $0.1 \leq \mu \leq 0.5$), the network has a high resolution, and communities' structure is clear, and all compared algorithms have near performance in this case. But the main difference between the robustness and stability of the algorithms lies in $0.6 \leq \mu \leq 0.8$ because as the value of μ increases, connections between communities increase too, and detecting the real structure of communities becomes difficult. For example, some algorithms have NMI value equal to zero as the mixing parameter value increases.

In Fig. 15, when $0.1 \leq \mu \leq 0.4$, LSMD, LPA-Intimacy, and G-CN have nearly equal NMI results and higher than others, and Louvain and NIBLPA both have lower results than others. When $0.5 \leq \mu \leq 0.8$, the network's fuzziness gradually increases. In this range, some algorithms lose their accuracy, and only stable and more robust algorithms can maintain their results as far as possible. LSMD and Infomap have better

NMIs than other algorithms in this range. LSMD also can maintain its accuracy when $\mu = 0.8$ and can obtain better results than other methods. When $0.6 < \mu \leq 0.8$, methods such as LPA and NIBLPA, which have weak robustness and have random nature, lose their accuracy, and are not successful to discover communities when μ is equal to 0.8.

In Fig. 16, all of the considered methods have similar NMI values in the range of $0.1 \leq \mu \leq 0.2$. But when $\mu \geq 0.3$, algorithms are divided into 2 groups. LSMD and Infomap are the first group that are able to obtain higher results than others and can maintain their accuracy for high values of μ and the second group contains LPA, NIBLPA, G-CN, LPA-Intimacy, and Louvain which have lower results than LSMD and are more sensitive to μ. As it is obvious, their NMI extremely decrease for high values of μ, and even they obtain NMI equal to zero when μ is equal to 0.8

In Fig. 17, for $0.1 \leq \mu \leq 0.4$, LSMD and G-CN have the highest NMI results, whereas LPA and Infomap nearly have lower results than others. But due to weak robustness in LPA, NIBLPA,

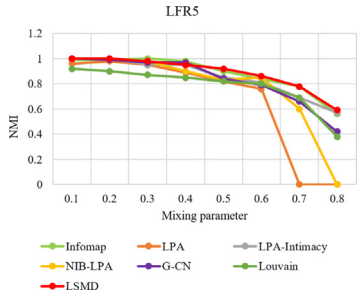

Fig. 19. NMI results obtained by Infomap, LPA, LPA-Intimacy, NIBLPA, G-CN, Louvain and LSMD in LFR5 dataset.

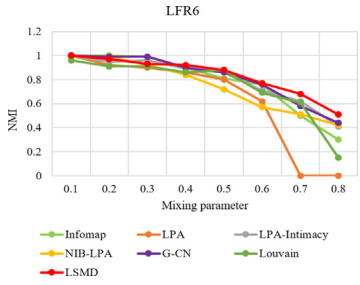

Fig. 20. NMI results obtained by Infomap, LPA, LPA-Intimacy, NIBLPA, G-CN, Louvain and LSMD in LFR6 dataset.

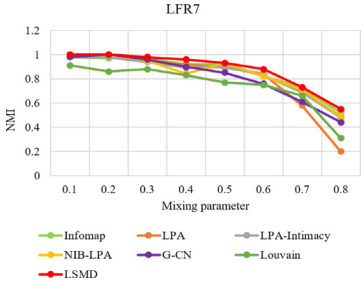

Fig. 21. NMI results obtained by Infomap, LPA, LPA-Intimacy, NIBLPA, G-CN, Louvain and LSMD in LFR7 dataset.

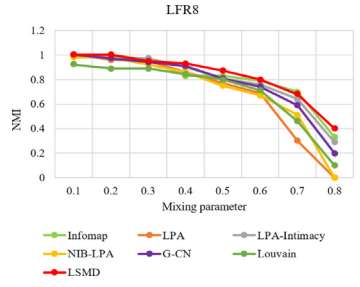

Fig. 22. NMI results obtained by Infomap, LPA, LPA-Intimacy, NIBLPA, G-CN, Louvain and LSMD in LFR8 dataset.

LPA-Intimacy, and G-CN algorithms, again they are not able to maintain their performance, and after 0.5, they have considerable decreases in NMI while a robust algorithm such as LSMD can obtain higher results. Also, Infomap shows good performance by obtaining high NMI results when $\mu \geq 0.5$.

In Fig. 18, we can say that except the Louvain method, other algorithms nearly have the same behavior when $0.1 \leq \mu < 0.5$ because their NMI nearly decreases in even pace. LPA, NIBLPA, and G-CN are algorithms that lose their accuracy earlier than others, but LSMD, Infomap, and LPA-Intimacy perform better than other algorithms.

In Figs. 19 to 22, as the network size increases, the real difference between LSMD and other methods becomes more apparent. Finding acceptable results on big datasets, require to have accurate measures and robustness in designing of the algorithm. LSMD makes good use of its accurate steps to obtain acceptable results even on big LFR datasets. Generally, in these 4 LFR datasets, during Figs. 19 to 22 LSMD outperforms other methods and has obtained the highest NMI results on most of the cases. Even when $\mu \geq 0.6$, LSMD has successfully obtained the highest results too. Other important point to be considered is LSMD's robustness that it has been able to have good results continuously when $0.1 \leq \mu < 0.5$, and it is only the LSMD, which has the lowest decrease in NMI for different values of μ. These results obviously show that if there exists randomness or weak robustness in an algorithm, it can cause to obtain weak results on big and fuzzy networks. LPA, NIBLPA, LPA-Intimacy, G-CN and Louvain are algorithms with more decrease in their NMI during the experiments on LFR datasets.

4.8. Evaluation of execution time

One of the most important evaluation measures in community detection is the execution time. Besides having high accuracy in detecting communities, it is expected to find communities in a reasonable time. For this purpose, most of the recently presented algorithms try to discover communities with low time complexity. Therefore, this section shows the running times for our proposed LSMD algorithm and other famous algorithms such as LPA, Louvain, Infomap, NIBLPA, LPA-Intimacy, and G-CN on medium and large-scale datasets with more than 9000 nodes. Some algorithms, such as RTLCD, DA, LCCD, and ECES are not considered in this evaluation due to their considerable time complexity.

LPA is one of the fastest algorithms with $O(n + m)$ time complexity. Louvain, NIBLPA, and LPA-Intimacy have time complexity $O(n \log n)$. We have listed the time complexity of all mentioned algorithms in this paper in Table 15. Table 14 shows running time of examined algorithms on 12 datasets. The obtained results show the average execution time of 10 different runs. One of the advantages of the LSMD method is its stability feature. In all executions of LSMD, it provided stable results with similar execution time. We will talk about stability of algorithms in details in Section 4.10. Since LPA, NIBLPA, and LPA-Intimacy need several iterations to converge, we have executed them with maximum 10 iterations and a termination condition that if the algorithm has same results in two sequential iterations, then it is terminated. In large-scale datasets because 10 iterations are time consuming, so in DBLP and Amazon we executed LPA, NIBLPA and LPA-Intimacy for maximum 5 iterations. Since one iteration of LPA, NIBLPA and LPA-Intimacy is not finished in 24 h, they are not evaluated on YouTube dataset. On YouTube dataset, Infomap was not able to finish execution in 24 h, too. We have also used 4 large LFR datasets to test the execution time of algorithms. LFR5 and LFR6 are datasets with 20000 nodes and averagely 393 092 edges. LFR7 and LFR8 are datasets with 50 000 nodes and averagely 1 million edges. The mentioned times for algorithms on LFR datasets are the average of the execution times for eight different values of μ. On some datasets, algorithms such as LPA, NIBLPA and LPA-Intimacy converged on 5 or 6 iterations and on some datasets, they did not converge even in 10 iterations. All the algorithms were performed on a desktop computer with core-i5 (3.40 GHz) processor and 12 GB memory in MATLAB (2017) and Windows10 OS.

As shown in Table 14, in all the mentioned datasets, LSMD significantly has the lowest execution time among other algorithms. In CA-HEPTH and PGP datasets, LSMD with 6 s and G-CN with 9 s, have the fastest execution times, respectively. In LFR5 and LFR6, again LSMD has the best execution time, and G-CN has the second-best execution time, but G-CN is two times slower than the LSMD algorithm. We can see that there is a significant difference between Louvain, Infomap, LPA, NIBLPA and LPA-Intimacy execution times with LSMD in LFR5 and LFR6. In Condmat −2003, Email-Enron and Condmat −2005, LSMD has the fastest running time among the other methods. For example, in Condmat −2003, Infomap is about 700% slower than LSMD. In all datasets the

Table 14
Execution times on networks with more than 9000 nodes (All times are in seconds). (Lowest values on each row are bolded.)

Dataset	LPA	NIBLPA	LPA-Intimacy	Louvain	Infomap	G-CN	LSMD
CA-HEPTH	21	23	27	14	28	9	**6**
PGP	23	24	26	15	18	8	**5**
LFR5	166	154	179	75	154	60	**30**
LFR6	190	210	190	70	197	62	**29**
Condmat-2003	180	204	230	66	170	33	**23**
Email-Enron	341	285	223	98	440	77	**37**
Condmat-2005	345	298	385	91	300	73	**38**
LFR7	850	970	1086	260	380	140	**83**
LFR8	1053	1076	1135	257	391	146	**92**
DBLP	20610	22146	21237	1068	13305	340	**182**
Amazon	22736	23090	20120	843	9376	365	**161**
YouTube	71029	N/A	N/A	2959	N/A	13174	**609**

Table 15
Comparison of time complexity of different community detection algorithms.

Algorithm	Time complexity
CNM [30]	$O(n \log 2n)$
LPA [66]	$O(m+n)$
NIBLPA [18]	$O(nlogn)$
LPA-Intimacy	$O(nlogn)$
G-CN	$O(nk)$
Infomap [67]	$O(n(n+m))$
Louvain [68]	$O(nlogn)$
LCCD [42]	$O(n^2)$
ECES [10]	$O(nlogn)$
Stepping LPA-S [43]	$O(n^2)$
RTLCD [32]	$O(rmax\{d\lvert C\rvert \log\lvert C\rvert , \lvert N(C)\rvert (dlog\lvert C\rvert + d^4)\})$
DA [44]	$O(n^3)$
LSMD	$O(nk)$

second fast algorithm is G-CN, but there are also significant differences with LSMD in execution times. Nevertheless, Louvain as a fast algorithm is not able to execute these datasets as fast as LSMD does, and it is averagely 500% slower than LSMD. In LFR7 and LFR8, after LSMD, G-CN has the second-best execution time.

In addition, there is a considerable difference in execution times, especially in DBLP, Amazon, and YouTube datasets. LSMD executes DBLP in 182 s, while LPA executes it in 20610 s in five iterations. In other words, LPA's average execution time for each iteration is 4122 s, which is nearly about 1 h just for one iteration. After LSMD, G-CN has the lowest execution time on DBLP with 340 seconds. In the Amazon dataset, LSMD is finished in 161 s, while LPA-based methods averagely execute one iteration of Amazon in 3900 s. Furthermore, LSMD is 20 times faster than Infomap and 5 times faster than Louvain in DBLP and Amazon datasets. In the YouTube dataset, LPA is not completely executed in 48 h, while LSMD in 609 s executes it and gives stable results. After LSMD, Louvain has the second-best result on YouTube dataset, and G-CN has the third-best execution time in the YouTube dataset, and it is about 20 times slower than LSMD. NIBLPA, LPA-Intimacy, and Infomap were not able to finish execution on YouTube dataset in reasonable time. Therefore, we did not continue their execution after 48 h. However, we have recorded the execution of one iteration of LPA in Table 14 which is about 20 h.

As mentioned before, the main advantage of proposed LSMD algorithm is its significant fast execution time. The second advantage is its stability, which is discussed in Section 4.10. Among the compared algorithms, LSMD has the best stability in obtaining the same results in different executions. The third advantage of LSMD is its reliable accuracy in 7 ground truth datasets.

4.9. Evaluation of time complexity

In this section, we summarize the time complexity of all algorithms mentioned in this paper in order to give a better perspective on the time complexity of LSMD in comparison with other examined algorithms. Table 15 lists the time complexity of 13 algorithms. n indicates the number of nodes in a network, and m indicates the number of edges. For RTLCD algorithm, r is the maximum length of the shortest path from the detected community core member to the edge of the detected community, $|C|$ is the size of community neighbors and d is mean degree of network

As it is apparent from Table 15, LPA has the lowest time complexity, and second-lowest time complexity belongs to LSMD. An important point to be considered is that LPA despite of its simple design, can be very slow on large datasets. The first reason is because of its iterative nature. It needs several iterations to obtain final results, which can be time consuming, but in contrast, LSMD does not need any iteration. The second reason is the label selection strategy of LPA. In each iteration of LPA, each node checks the label of all neighbors, then selects the most frequent one. This operation may seem simple, but when the network is large, grouping neighbors based on their label and selecting maximum label can slow down the algorithm, but LSMD uses a diffusion concept to reduce redundant checks, and only each node is checked once. Because of that, LSMD has better performance than LPA and other mentioned algorithms.

4.10. Evaluation of the stability

As mentioned before, stability is one of the main advantages of LSMD. To compare LSMD's stability with other algorithms, we designed an experiment on real-world datasets. In this experiment, each algorithm is executed 5 times to obtain values of NMI, modularity (Q), and number of communities (CN) on each execution. In addition, LPA, NIBLPA, and LPA-Intimacy were executed with 10 iterations for each running. Table 16 shows the comparison between obtained results for 5 different executions. Due to the space limitations on paper, we just mention the minimum and maximum obtained results for each dataset. Furthermore, the Δ measure is used to show the difference of minimum and maximum values. For example, minimum NMI shows the lowest NMI on ground truth datasets. The maximum result shows the maximum result of NMI in ground truth datasets.

As it is apparent from Table 16, only LSMD has stable results in all of the executions. Its minimum and maximum results are the same, and there are no changes in results on different executions. In ground truth datasets, LSMD provides stable NMI and stable number of communities in each of its executions. LPA has the most instability in its results. LPA provided different results for each running. For example, in the Karate dataset, the lowest NMI is 0.22 for the LPA algorithm, and its highest NMI is 0.79. In other datasets, as it is obvious, LPA has discovered different number of communities. NIBLPA and G-CN also are instable, but their instability is lower than LPA. After LPA, NIBLPA and G-CN have the most instability in their results. LPA-Intimacy due to its accurate measures, has better stability than LPA, NIBLPA, and G-CN. Louvain also has considerable instabilities in its obtained results, especially in large-scale datasets such as Condmat −2003 and Condmat −2005. After LSMD, Infomap has the best stability and robustness in comparison with other methods. Although Infomap due to its random walk nature has some instabilities, but according to its map function, it can control the randomness nature.

5. Conclusion

This paper proposes a multi-level label diffusion algorithm based on local similarity measures (LSMD) with low time complexity. LSMD is a fast and robust local community detection starting from low degree nodes to discover communities without dependency to core or hub nodes. It starts label diffusion from any periphery nodes to inside communities. In the first phase, the first community's label is assigned to a node with the lowest degree (e.g., degree 1 nodes) and its direct neighbor and some second and third level neighbors. Next, a community label is

Table 16
Comparison of stability of algorithms on obtaining results on 5 different executions on real-world datasets. The value of Δ shows the difference value of minimum and maximum obtained result after 5 different execution.

Dataset		LPA			NIBLPA			LPA-Intimacy			G-CN			Louvain			Infomap			LSMD		
		Min	Max	Δ	Min	Max	Δ	Min	Max	Δ	Min	Max	Δ	Min	Max	Δ	Min	Max	Δ	Min	Max	Δ
Karate	NMI	0.22	0.79	0.57	0.27	0.58	0.31	1	1	0	0.83	0.83	0	0.58	0.61	0.03	0.69	0.7	0.01	1	1	**0**
	Q	0.13	0.4	0.27	0.11	0.4	0.29	0.37	0.37	0	0.37	0.37	0	0.41	0.43	0.02	0.4	0.41	0.01	0.37	0.37	**0**
	CN	2	3	1	3	5	2	2	2	0	2	2	0	4	5	1	3	4	1	2	2	**0**
Dolphins	NMI	0.48	0.55	0.07	0.48	0.5	0.02	0.63	0.63	0	0.51	0.54	0.03	0.48	0.53	0.05	0.53	0.54	0.01	1	1	**0**
	Q	0.49	0.51	0.02	0.43	0.44	0.01	0.5	0.5	0	0.51	0.52	0.01	0.45	0.5	0.05	0.52	0.527	0.007	0.37	0.37	**0**
	CN	4	5	1	5	5	0	4	4	0	4	5	1	5	6	1	5	6	1	2	2	**0**
Polbooks	NMI	0.5	0.53	0.03	0.53	0.53	0	0.54	0.54	0	0.51	0.53	0.02	0.45	0.47	0.02	0.5	0.51	0.01	0.59	0.59	**0**
	Q	0.48	0.49	0.01	0.55	0.55	0	0.48	0.48	0	0.51	0.52	0.01	0.49	0.51	0.02	0.522	0.525	0.003	0.44	0.44	**0**
	CN	3	3	0	0.4	0.4	0	3	3	0	4	6	2	5	5	0	5	6	1	3	3	**0**
Football	NMI	0.79	0.81	0.02	0.7	0.72	0.02	0.86	0.86	0	0.85	0.87	0.02	0.86	0.89	0.03	0.9	0.91	0.01	0.93	0.93	**0**
	Q	0.53	0.55	0.02	0.5	0.51	0.01	0.6	0.6	0	0.58	0.58	0	0.55	0.59	0.04	0.58	0.6	0.02	0.58	0.58	**0**
	CN	9	10	1	8	8	0	10	10	0	12	13	1	9	10	1	12	12	0	12	12	**0**
Netscience	Q	0.89	0.91	0.02	0.87	0.88	0.01	0.87	0.88	0.01	0.87	0.89	0.02	0.91	0.92	0.01	0.9	0.91	0.01	0.94	0.94	**0**
	CN	328	337	9	361	379	18	301	307	6	312	328	16	271	278	7	277	294	17	275	275	**0**
PGP	Q	0.73	0.78	0.05	0.71	0.75	0.04	0.73	0.76	0.03	0.77	0.79	0.02	0.86	0.87	0.01	0.715	0.75	0.035	0.59	0.59	0
	CN	990	1165	175	1836	1869	33	642	671	29	857	893	36	92	118	26	908	961	53	643	643	0
CA-HEPTH	Q	0.59	0.63	0.04	0.59	0.61	0.02	0.65	0.68	0.03	0.59	0.6	0.01	0.7	0.72	0.02	0.62	0.64	0.02	0.63	0.63	0
	CN	1151	1270	119	1559	1724	165	921	944	23	1220	1318	98	459	477	18	1105	1119	14	586	586	0
CA-GRQC	Q	0.69	0.73	0.04	0.7	0.72	0.02	0.73	75	74.27	0.7	0.72	0.02	0.75	0.76	0.01	0.74	0.75	0.01	0.77	0.77	0
	CN	745	768	23	963	988	25	598	605	7	749	789	40	385	398	13	715	722	7	456	456	0
Email-Enron	Q	0.33	0.38	0.05	0.1	0.12	0.02	0.14	0.17	0.03	0.09	0.11	0.02	0.52	0.55	0.03	0.52	0.53	0.01	0.39	0.39	0
	CN	2025	2161	136	2174	2459	285	1423	1473	50	1753	1892	139	1451	1483	32	2608	2617	9	1221	1221	0
Condmat-2003	Q	0.51	0.54	0.03	0.49	0.51	0.02	0.65	0.69	0.04	0.59	0.61	0.02	0.71	0.73	0.02	0.61	0.633	0.023	0.57	0.57	0
	CN	3513	3722	209	3586	3675	89	2668	2691	23	3399	3413	14	91	116	25	2540	3222	682	3502	3502	0
Condmat-2005	Q	0.5	0.54	0.04	0.2	0.23	0.03	0.56	0.58	0.02	0.6	0.63	0.03	0.71	0.73	0.02	0.6	0.63	0.03	0.44	0.44	0
	CN	4156	4268	112	2169	2368	199	2923	2950	27	3341	3827	486	1056	1108	52	2957	3266	309	3952	3952	0

respectively assigned for the nodes with degrees 2, 3, etc. In fact, we used this belief that people with a smaller number of neighbors are not likely to be connected to diverse communities. After finishing this step, initial communities are discovered. Then, during a fast merge phase, some closely connected communities are merged as far as possible to form the final communities. Needless to mention that, in LSMD, more than 90% of initial communities are final communities and are not participated in the merge phase. When a new algorithm is proposed, it is normally needed to test in real-world with known node groups derived from some metadata. Correspondingly, large-scale synthetic benchmark networks such as LFR networks with predefined community labels is necessary for testing the proposed algorithm. In this paper, experiments were performed on real-world small and large-scale networks and synthetic networks. The experimental results proved that the proposed LSMD algorithm has better performance than earlier state-of-the-art algorithms. The obtained results show that LSMD has three main advantages, including: a) the significantly fastest community detection algorithm in comparison with compared algorithms, b) the best stable algorithm among compared algorithms and c) the best accuracy based on F-measure and NMI metric in all artificial networks, all large-scale real-world ground truth datasets, and three small real-world networks. The other advantage of the proposed algorithm is to avoid random nature in all steps of the algorithm. Finally, the last advantage of the proposed algorithm is that it does not use any adjustable parameters in the steps of algorithms.

Future researches may improve the accuracy of the algorithm by presenting newer local similarity in the label diffusion process. It can also be developed to use in overlapping community detection.

CRediT authorship contribution statement

Asgarali Bouyer: Conceptualization, New innovation proposing and evaluation, Methodology, Software, Editing. **Hamid Roghani:** Writing - original draft, Programming, Advisor, Writing - reviewing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3–5) (2010) 75–174.
[2] L. Freeman, The development of social network analysis, Study Sociol. Sci. 1 (2004) 687.
[3] J. Moody, D.R. White, Structural cohesion and embeddedness: A hierarchical concept of social groups, Am. Sociol. Rev. (2003) 103–127.
[4] J. Chen, B. Yuan, Detecting functional modules in the yeast protein–protein interaction network, Bioinformatics 22 (18) (2006) 2283–2290.
[5] A.W. Rives, T. Galitski, Modular organization of cellular networks, Proc. Natl. Acad. Sci. 100 (3) (2003) 1128–1133.
[6] Y. Dourisboure, F. Geraci, M. Pellegrini, Extraction and classification of dense communities in the web, in: Proceedings of the 16th International Conference on World Wide Web, 2007, pp. 461–470.
[7] G.W. Flake, S. Lawrence, C.L. Giles, F.M. Coetzee, Self-organization and identification of web communities, Computer 35 (3) (2002) 66–70.
[8] P.K. Reddy, M. Kitsuregawa, P. Sreekanth, S.S. Rao, A graph based approach to extract a neighborhood customer community for collaborative filtering, in: International Workshop on Databases in Networked Information Systems, Springer, 2002.
[9] L. Freeman, The development of social network analysis, Study Sociol. Sci. 1 (2004).
[10] K. Berahmand, A. Bouyer, M. Vasighi, Community detection in complex networks by detecting and expanding core nodes through extended local similarity of nodes, IEEE Trans. Comput. Soc. Syst. 5 (4) (2018) 1021–1033.
[11] B. Saoud, A. Moussaoui, Node similarity and modularity for finding communities in networks, Physica A 492 (2018) 1958–1966.
[12] M.E. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.
[13] J. Chen, O. Zaïane, R. Goebel, Local community identification in social networks, in: Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in, IEEE, 2009, pp. 237–242.
[14] P. Zhang, D. Qiu, A. Zeng, J. Xiao, A comprehensive comparison of network similarities for link prediction and spurious link elimination, Physica A 500 (2018) 97–105.

[15] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (3) (2007) 036106.
[16] C. Gui, R. Zhang, Z. Zhao, J. Wei, R. Hu, LPA-CBD an improved label propagation algorithm based on community belonging degree for community detection, Internat. J. Modern Phys. C 29 (02) (2018) 1850011.
[17] H. Kong, Q. Kang, C. Liu, W. Li, H. He, Y. Kang, An improved label propagation algorithm based on node intimacy for community detection in networks, Internat. J. Modern Phys. B 32 (25) (2018) 1850279.
[18] Y. Xing, F. Meng, Y. Zhou, M. Zhu, M. Shi, G. Sun, A node influence based label propagation algorithm for community detection in networks, Sci. World J. 2014 (2014) 9.
[19] 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) (2017) 2691–2698.
[20] X. Wang, G. Chen, H. Lu, A very fast algorithm for detecting community structures in complex networks, Physica A 384 (2) (2007) 667–674, http://dx.doi.org/10.1016/j.physa.2007.05.013.
[21] M. Tasgin, H.O. Bingol, Community detection using boundary nodes in complex networks, Physica A 513 (2019) 315–324.
[22] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 2008 (10) (2008) P10008.
[23] M.E. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (6) (2004) 066133.
[24] E.R. Barnes, An algorithm for partitioning the nodes of a graph, SIAM J. Algebraic Discrete Methods 3 (4) (1982) 541–550.
[25] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell Syst. Techn. J. 49 (2) (1970) 291–307.
[26] L. Waltman, N.J. Van Eck, A smart local moving algorithm for large-scale modularity-based community detection, Eur. Phys. J. B 86 (11) (2013) 471.
[27] J. Xi, W. Zhan, Z. Wang, Hierarchical community detection algorithm based on node similarity, Int. J. Database Theory Appl. 9 (6) (2016) 209–218.
[28] F.D. Zarandi, M.K. Rafsanjani, Community detection in complex networks using structural similarity, Physica A 503 (2018) 882–891.
[29] S. Fortunato, V. Latora, M. Marchiori, Method to find community structures based on information centrality, Phys. Rev. E 70 (5) (2004) 056104.
[30] A. Clauset, M.E. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (6) (2004) 066111.
[31] J. Wu, Y. Hou, Y. Jiao, Y. Li, X. Li, L. Jiao, Density shrinking algorithm for community detection with path based similarity, Physica A 433 (2015) 218–228.
[32] X. Ding, J. Zhang, J. Yang, A robust two-stage algorithm for local community detection, Knowl.-Based Syst. 152 (2018) 188–199, http://dx.doi.org/10.1016/j.knosys.2018.04.018.
[33] K. Berahm, A. Bouyer, LP-LPA: A link influence-based label propagation algorithm for discovering community structures in networks, Internat. J. Modern Phys. B 32 (06) (2018) 1850062.
[34] Z. Lin, X. Zheng, N. Xin, D. Chen, CK-LPA: Efficient community detection algorithm based on label propagation with community kernel, Physica A 416 (2014) 386–399.
[35] H. Sun, et al., CenLP: A centrality-based label propagation algorithm for community detection in networks, Physica A 436 (2015) 767–780.
[36] S.B. Thakare, A.W. Kiwelekar, SkipLPA: An efficient label propagation algorithm for community detection in sparse network, in: Proceedings of the 9th Annual ACM India Conference, ACM, 2016, pp. 97–106.
[37] X.-K. Zhang, S. Fei, C. Song, X. Tian, Y.-Y. Ao, Label propagation algorithm based on local cycles for community detection, Internat. J. Modern Phys. B 29 (05) (2015) 1550029.
[38] X. Liu, T. Murata, Advanced modularity-specialized label propagation algorithm for detecting communities in networks, Physica A 389 (7) (2010) 1493–1500.
[39] M.J. Barber, J.W. Clark, Detecting network communities by propagating labels under constraints, Phys. Rev. E 80 (2) (2009) 026129.
[40] K. Berahm, A. Bouyer, A link-based similarity for improving community detection based on label propagation algorithm, J. Syst. Sci. Complexity 32 (3) (2019) 737–758, http://dx.doi.org/10.1007/s11424-018-7270-1.
[41] J. Eustace, X. Wang, Y. Cui, Community detection using local neighborhood in complex networks, Physica A 436 (2015) 665–677.
[42] X. Wang, G. Liu, J. Li, J.P. Nees, Locating structural centers: A density-based clustering method for community detection, PLoS One 12 (1) (2017) e0169355.
[43] W. Li, C. Huang, M. Wang, X. Chen, Stepping community detection algorithm based on label propagation and similarity, Physica A 472 (2017) 145–155.
[44] Z. Liu, Y. Ma, A divide and agglomerate algorithm for community detection in social networks, Inform. Sci. 482 (2019) 321–333.
[45] P. Jaccard, Étude comparative de la distribution florale dans une portion des Alpes et des Jura, Bull. Soc. Vaudoise Sci. Nat. 37 (1901) 547–579.
[46] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (4) (1977) 452–473.
[47] D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, S.M. Dawson, The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations, Behav. Ecol. Sociobiol. 54 (4) (2003) 396–405.
[48] M.E. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. 103 (23) (2006) 8577–8582.
[49] M. Girvan, M.E. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (12) (2002) 7821–7826.
[50] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowl. Inf. Syst. 42 (1) (2015) 181–213.
[51] R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt, A. Arenas, Self-similar community structure in a network of human interactions, Phys. Rev. E 68 (6) (2003) 065103.
[52] M.E. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74 (3) (2006) 036104.
[53] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440.
[54] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: Densification and shrinking diameters, ACM Trans. Knowl. Discov. Data (TKDD) 1 (1) (2007) 2.
[55] J. Duch, A. Arenas, Community detection in complex networks using extremal optimization, Phys. Rev. E 72 (2) (2005) 027104.
[56] M. Boguná, R. Pastor-Satorras, A. Díaz-Guilera, A. Arenas, Models of social networks based on social distance attachment, Phys. Rev. E 70 (5) (2004) 056122.
[57] J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: densification laws, shrinking diameters and possible explanations, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, ACM, 2005, pp. 177–187.
[58] M.E. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. 98 (2) (2001) 404–409.
[59] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. 105 (4) (2008) 1118–1123.
[60] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008) 046110.
[61] L. Danon, A. Diaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, J. Stat. Mech. Theory Exp. 2005 (09) (2005) P09008.
[62] N. Chinchor, MUC-4 evaluation metrics, in: Proceedings of the 4th Conference on Message Understanding, Association for Computational Linguistics, 1992, pp. 22–29.
[63] Y. Sasaki, The truth of the F-measure, Teach. Tutor Mater. 1 (5) (2007) 1–5.
[64] K.M. Ting, Precision and recall, in: C. Sammut, G.I. Webb (Eds.), Encyclopedia of Machine Learning, Springer US, Boston, MA, 2010, p. 781.
[65] S.S.N.A. Project, https://snap.stanford.edu, (Accessed).
[66] S.E. Garza, S.E. Schaeffer, Community detection with the label propagation algorithm: a survey, Physica A (2019) 122058.
[67] Z. Zhao, S. Zheng, C. Li, J. Sun, L. Chang, F. Chiclana, A comparative study on community detection methods in complex networks, J. Intell. Fuzzy Systems 35 (1) (2018) 1077–1086.
[68] Z. Yang, R. Algesheimer, C.J. Tessone, A comparative analysis of community detection algorithms on artificial networks, Sci. Rep. 6 (2016) 30750.

Asgarali Bouyer is an Associate Professor in the Faculty of Computer engineering and information technology at Azarbaijan Shahid Madani University. He received the Ph.D degree in Computer Science from Universiti Teknologi Malaysia (UTM) in 2011. His current research interests are in complex network, social media mining and data clustering, and resource management systems in Cloud/ Grid computing.

Hamid Roghani is the M.Sc. student at the Department of Computer engineering at Azarbaijan Shahid Madani University. He is M.Sc. student in Information engineering. His current research interests are in Online Education, social networks and social media analysis.