Detecting Community Structure in Signed and Unsigned Social Networks by Using Weighted Label Propagation

Bagher Zarei[a], Mohammad Reza Meybodi[b], and Behrooz Masoumi[a]

[a]Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, 3419915195, Iran
[b]Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, 1591634311, Iran
[b]corresponding author: mmeybodi@aut.ac.ir

Abstract

Detecting community structure is one of the most important problems in analyzing complex networks such as technological, informational, biological, and social networks and has great importance in understanding the operation and organization of the networks. One of the significant properties of social networks is the communication intensity between the users, which has not received much attention so far. Most of the proposed methods for detecting community structure in social networks have just considered communications between users. In this paper, using MinHash and label propagation, an algorithm called Weighted Label Propagation Algorithm (WLPA) has been proposed to detect community structure in signed and unsigned social networks. WLPA takes into account the intensity of communications in addition to the communications. In WLPA, first, the similarity of all adjacent nodes is estimated by using MinHash. Then, each edge is assigned a weight equal to the estimated similarity of its end nodes. The weights assigned to the edges somehow indicate the intensity of communication between users. Finally, the community structure of the network is determined through the weighted label propagation. Experiments on the benchmark networks indicate that WLPA is efficient and effective for detecting community structure in both signed and unsigned social networks.

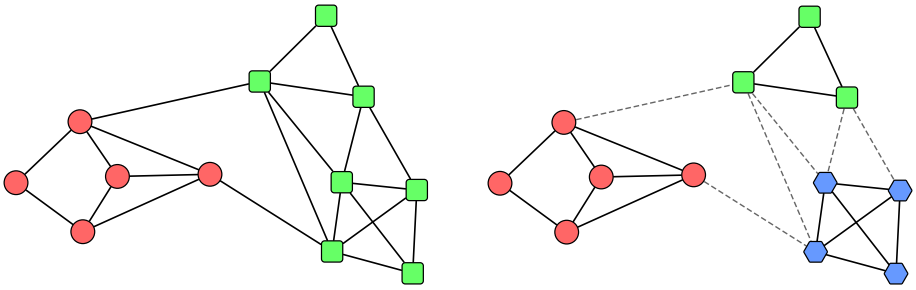
Keywords: social networks; social networks analysis; community structure detection; Jaccard similarity; MinHash; label propagation

A social network is a social structure of social actors such as individuals, groups, and organizations that are linked together based on social relationships such as friendship, co-operation, and common interests. In an unsigned social network, relationships can only be positive like friendship, whereas in a signed social network, relationships can be both positive like friendship and negative like hostility. Social network analysis (SNA) is the process of acquiring knowledge about social structures through the use of networks and graph theory. Such knowledge is useful in many domains, such as marketing, management, epidemiology, homeland security, and psychology. Over the past decades, community structure detection has emerged as an essential task in the area of SNA. It provides insights into the underlying structure and potential functions of the social networks. The purpose of community structure detection is to identify groups of nodes such that the density of links within the groups is high and between the groups is low. In this paper, by using MinHash and label propagation, an algorithm called WLPA has been proposed for detecting community structure in signed and unsigned social networks. Experiments on real-world and synthetic benchmark networks indicate that WLPA outperforms state-of-the-art algorithms GenLouvain, CPMap, MEAs-SN, and SN-MOGA for signed networks and two well-known algorithms LPA and SLPA for unsigned networks.

1. Introduction

A social network can be defined as a graph $G = (V, E)$, where V is the set of n vertices or nodes, and $E \subseteq V \times V$ is the set of m edges connecting pairs of vertices. For example, in a human social network, each vertex denotes an individual, and each edge denotes a relation or a link between two individuals. In the field of social science, the networks that include only positive links are called *unsigned social networks*, and the networks with both positive and negative links are called *signed social networks*. Positive links in signed social networks indicate positive relations such as friendship, collaboration, common interests, trust, and enhancing, whereas negative links indicate negative relations such as hostility, dislike, different interests, distrust, and repressing. For example, the technology news website Slashdot[1] allows its users to tag other users as friends and foes, as well as the product website Epinions[2] lets users trust and distrust each other [1].

Social network analysis (SNA) is the process of acquiring knowledge about social structures through the use of networks and graph theory. One of the most important problems in SNA is to detect the community structure. Detecting communities has great importance in understanding the operation and organization of the networks [2, 3]. In unsigned networks, community structure is defined as the group of nodes which have dense connections within groups and sparse connections between groups [2, 3] (see Figure 1(a)). In signed networks, communities are defined by the density of links and the signs of links. That is, within communities, the links should be positive and dense, and between communities, the links should be negative or positive and sparse (see Figure 1(b)). However, this problem is by no means straightforward since it is natural to have some negative links within groups and, at the same time, some positive links between groups. Also, nodes connected by positive links do not belong to the same community, either. Thus, more robust community partitions should properly disregard and retain some positive and negative links to identify more natural communities [4].



(a) (b)

Figure 1. (a) Community structure in an unsigned sample network and (b) community structure in a signed sample network. Positive edges are shown with solid lines, and negative edges are shown with dashed lines. Different communities are shown with different shapes and colors.

Due to the importance of community structure as a topological property of social networks, methods that can detect communities from social networks are hardly needed. The existing algorithms to solve the problem of community structure detection in social networks are divided into two categories: *local algorithms* and *global algorithms*. The time complexity of global algorithms is relatively high. On the other hand, the local algorithms are time efficient, but the communities found by these algorithms are not desirable in terms of the evaluation criteria. Also,

[1] https://slashdot.org/
[2] http://epinions.com/

most of the proposed methods for detecting community structure in social networks have just considered the communications between the users and have not considered the intensity of communications, which is one of the significant properties of social networks. In this paper, using MinHash and label propagation, an algorithm called Weighted Label Propagation Algorithm (WLPA) has been proposed to detect community structure in signed and unsigned social networks. WLPA takes into account the intensity of communications in addition to the communications. In WLPA, first, the similarity of all adjacent nodes is estimated by using MinHash. Then, each edge is assigned a weight equal to the estimated similarity of its end nodes. The weights assigned to the edges somehow indicate the intensity of communication between users. Finally, the community structure of the network is determined through the weighted label propagation. Experiments on the benchmark networks indicate that WLPA is efficient and effective for detecting community structure in both signed and unsigned social networks. The main contributions of this paper are summarized as follows:

- a single algorithm has been proposed for detecting the community structure in both signed and unsigned social networks. The proposed algorithm requires no prior knowledge about the community structure, such as the number and size of the communities, and identifies the communities of a network in linear time. Furthermore, in the proposed algorithm, each node only requires information about its neighbors. That is, the proposed algorithm is a local one.
- Jaccard similarity has been modified for specifying the similarity between two nodes in the graph of a social network. Also, a single equation has been introduced for specifying the similarity between two nodes in both signed and unsigned social networks.
- MinHash has been used for estimating the similarity between two nodes in the graph of a social network. This significantly reduces the execution time of the proposed algorithm.

The rest of this paper is organized as follows: In Section 2, some of the related works have been explained. The proposed algorithm has been presented in Section 3. The experimental results and discussion have been given in Section 4. Finally, Section 5 includes the conclusion of the paper.

2. Related works

In this section, some algorithms for detecting community structure in unsigned and signed social networks are explained. According to the frequency and variety of the proposed algorithms, only some well-known algorithms have been described.

A) Algorithms for detecting community structure in unsigned networks

In [5], a divisive hierarchical algorithm called GN has been presented, which uses *edge betweenness* as a criterion for identifying the boundary of communities. The main idea of this algorithm is that inter-community edges have more betweenness than intra-community edges. So, by repetitive calculation and eliminating the edges having the highest betweenness values, the network will be split into separate connected components. The algorithm will be continued until all the network edges are eliminated. The algorithm has time complexity $O(m^2n)$ for a network with n nodes and m edges. The high time complexity of the algorithm will limit its use for the large networks. Similarly, in [6], the *edge clustering coefficient* has been used as a criterion for identifying the boundary of communities instead of edge betweenness. The main idea of this algorithm is that inter-community

edges have less clustering coefficient than intra-community edges. The algorithm has time complexity $O(\frac{m^4}{n^2})$.

In [7], a greedy agglomerative hierarchical algorithm called NM has been presented. In this algorithm, each node is considered as a single community, first. Then, the pair of communities caused to the maximum increase in modularity will repeatedly be merged. The algorithm is being continued until all the nodes will be put in one community. The algorithm has time complexity $O((m+n)n)$. In the fast version of this algorithm called CNM [8], the efficient data structures (balanced binary tree and max heap) have been used for saving and retrieving the required information for updating modularity. The time complexity of the algorithm is $O(md\log n)$ in which d is the depth of dendrogram.

In [9], a greedy algorithm based on optimizing the modularity called Louvain has been presented. In this algorithm, at first, each network node is assigned to a separate community. So, in the initial partition, the number of communities will equal the number of nodes. Then, the following phases will be repeated frequently. In the first phase, for each node i the amount of increase in the modularity will be calculated by transferring it to the community of its neighbors. The node i will be transferred to the neighbor community, causing the maximum increase in the modularity. The node i will remain in its community if transferring it to its neighbors' community does not increase the modularity. This process is applied repeatedly for all the nodes in sequence until no improvement will be obtained in the modularity. In the second phase, the algorithm will make a meta-network whose nodes are found communities in the first phase, and an edge is placed between two nodes with a weight of w. w is the sum of the weight of the edges that placed between the two communities corresponding to the two nodes. After finishing the second phase, the first one can be applied to the new network. These two phases will be repeated until no changes have to be yielded in the communities or maximum modularity is obtained. The algorithm is very fast. Computer simulation in the large case modular networks shows that its time complexity in the sparse networks is linear. The time complexity of the algorithm is $O(\gamma n)$ where γ is the number of repetitions for convergence of the algorithm.

In [10], for evaluation of a partition quality, the *community score* has been proposed, and the genetic algorithm (GA-net) has been used for optimizing it. In this algorithm, the locus-based adjacency representation [11] has been used. The main advantage of this representation is that the number of communities is determined automatically in the decoding process. Furthermore, in this algorithm, some special variation operators (combination and mutation) have been used. These operators decreased the search space by producing safe solutions so that the algorithm convergence will be improved. In a safe solution, the i^{th} gene can have the j value if there is an edge between the nodes i and j.

In [12], the multi-objective genetic algorithm called MOGA-net has been used to detect community structure. This algorithm simultaneously optimizes two objective functions called *community score* and *community fitness*. By maximizing the community score and community fitness, the number of edges inside the communities will be maximized and the number of edges among the communities will be minimized, respectively. Both objective functions have one setting parameter which controls the size of communities. Larger amounts for these parameters cause to finding communities with small sizes. The same as the GA-net algorithm, the locus-based adjacency representation and special variation operators, which produce safe solutions, have been used in this algorithm. The main advantage of the multi-objective approaches is that they do not only generate

one solution, but they generate a set of solutions. Each solution corresponds to a different compromise between the objective functions, and hence, they can divide the network into diverse partitions with a different number of communities. This enables the analysis of different community structures at different levels of the hierarchy.

In [13], the genetic algorithm (GA) and the object migrating automata (OMA) has been integrated as a single framework, called GAOMA-net, to solve the problem of community structure detection in complex networks. Unlike classic GAs, in the GAOMA-net algorithm, binary coding is not used for the chromosomes. In the GAOMA-net algorithm, each chromosome is denoted by a learning automaton of object migrating type so that each of the chromosome genes is assigned to one of the object migrating automaton actions, and it is located at the certain depth of that action. The GAOMA-net algorithm, by combining GA with the OMA, largely overcomes the problem of premature convergence. Also, in the GAOMA-net algorithm, a portion of the initial population has been generated using the MEM-net heuristic. This accelerates the convergence of the algorithm. The MEM-net heuristic is also introduced in [13].

In [14], a chaotic memetic algorithm called CMA has been proposed and used to detect the community structure in complex networks. In the CMA algorithm, the combination of the genetic algorithm (global search) and a dedicated local search has been used to search the solution space. In the global search process, by using genetic operators, new solutions are generated. In the local search process, good quality solutions are discovered by searching around the newly generated solutions. Also, to improve the convergence speed and efficiency, in both global search and local search processes, chaotic numbers have been used instead of random numbers. The population diversity is preserved by using chaotic numbers, and it prevents the algorithm from falling into local optimum.

Most presented algorithms need a global view from the network to detect the community structure in complex networks. Such algorithms are not scalable for networks with millions of nodes. In return, the algorithms based on label propagation only need local information. It was indicated that these algorithms are very efficient. In [15], a method called LPA has been suggested, which uses the label propagation idea for the detection of communities. In this algorithm, first of all, a different label is assigned to each node. Then, all the nodes will repeatedly update their labels in a random order in accordance with the majority of their neighbors. Updating the labels of the nodes can be done synchronously or asynchronously. In synchronous updating, the label of a node in instant t, is determined by the label of its neighbors in instant $t-1$. In asynchronous updating, the label of a node in instant t, is determined by the label of some of its neighbors in instant $t-1$ (the neighbors that their label in instant t has not been updated yet) and by the label of some of its neighbors in instant t (the neighbors that their label in instant t has been updated). The algorithm will terminate when each node has the label of the majority of their neighbors. At the end of the algorithm, the connected nodes with the same label form a community. The time complexity of the algorithm is $O(n+Tm)$, where T is the number of repetitions for convergence of the algorithm. This algorithm only uses the network structure for guiding its process and has some advantages such as no need for any extra input parameter and optimization objective function, simple implementation, and fast execution.

In [16], an algorithm called SLPA has been presented, which has extended the LPA algorithm for the detection of overlapping communities. This algorithm imitates human's communication behaviors. In this algorithm, each speaker node will select a label among the available labels in its memory based on the speaker rule and propagate it to the neighboring nodes, and each listener

node adopts a label from the received labels based on the listener rule and save it in its memory. Label propagation and label adopting will be continued for all the nodes until the maximum number of pre-determined repetitions is reached. Finally, by post-processing, the probability of available labels in the memory of each node will be calculated, and the labels with the probability less than a specified threshold will be eliminated from the memory. After post-processing, the connected nodes with the same label form a community. If a node includes several labels, it will belong to more than a community. By processing the available labels in the memory of a given node, the probability of belonging to different communities will be determined for that node. This algorithm will be turned into the LPA algorithm by setting the size of node memories to 1. The time complexity of this algorithm is $O(Tm)$ where T is the maximum number of repetitions and is the only input parameter of the algorithm.

B) Algorithms for detecting community structure in signed networks

Doreian and Mrvar [17] have suggested one of the first partitioning approaches for the structural balance. This method divides the nodes into k clusters randomly and tries to optimize the *frustration* function by transferring the nodes to the neighbor clusters. In this method, after dividing the network into k clusters, two neighbor clusters are selected randomly, then either one node of the first cluster will be transferred to the second one or two nodes will be replaced between these two clusters. If the new partition has less *frustration* than the previous partition, the new partition will be accepted. The process of selecting two neighboring clusters and transferring the node between them will continue until reaching the termination condition. The main problem of this method is that it does not consider the density of edges, which is one of the main used properties in the community structure detection methods. Moreover, the number of clusters will be given to the algorithm as an input parameter.

In [18], a multi-objective evolutionary method called MEAs-SN has been suggested for finding communities in the signed networks. This algorithm uses MOEA/D [19] for optimizing two conflicting objective functions f_{pos-in} and $f_{neg-out}$ simultaneously. By maximization of f_{pos-in}, the sum of positive similarities inside the communities and by maximization of $f_{neg-out}$, the sum of negative similarities among the communities will be maximized. The authors have been extended the similarity of Huang et al. [20] between two neighbor nodes to the signed networks based on the social balance theory. Moreover, they presented a direct and indirect combined representation that the algorithm can switch between different representations during the evolution process and can use the advantages of both representations. In the suggested combined representation, each solution has two components. The first component, $A(P) = \{\pi_1, \pi_2, \dots, \pi_n\}$, is a permutation from 1 to n (number of nodes) and the second one, $A(C) = \{c_1, c_2, \dots, c_n\}$, is a vector with n elements in which c_i indicates the cluster label that node i belongs to. In this algorithm, the $A(P)$ vectors (as many as the population members) have been generated randomly, and then by using the designed decoder (heuristic search), they will turn into $A(C)$ vectors. The $A(C)$ vectors are considered as the members of initial population. Then, the cycle of the evolutionary algorithm will be started, and the evolutionary operators will be performed on the $A(C)$ vectors. In this algorithm, the initial $A(P)$ vectors can produce better population through the decoder and evolutionary operators on the $A(C)$ vectors can be done time-efficient. So, the algorithm can benefit from the advantages of both $A(P)$ (indirect representation) and $A(C)$ (direct representation). The time complexity of the algorithm is $O(pn^2) + O(gpn)$.

In [21], a method called SN-MOGA has been suggested, which combines the genetic algorithm and a local refinement strategy for the detection of communities in the signed networks. This algorithm uses NSGA-II [22] for optimizing two objective functions *frustration* and *signed modularity* simultaneously. Modularity maximization allows some network divisions to be detected that are distant from the random divisions; however, the frustration minimization ensures that as much as possible, the number of negative connections inside the communities and the number of positive connections among them be low. This algorithm evolves a population of candidate solutions to obtain the best compromise between modularity and frustration. At the end of the evolution process, a solution with the minimum frustration or maximum signed modularity is selected from the Pareto front, and a local search strategy is done on it for improving the signed modularity. In the local search, the nodes with a positive relation with other community nodes will be transferred to the neighboring communities if the signed modularity increases. The time complexity of the algorithm is $O\big((gp\log p)\times(n\log n+m)\big)$ where g is the number of generations and p is the population size.

In [23], two evolutionary algorithms called EA-SN and CSA-SN and two Memetic algorithms called EA_{HC}-SN and CSA_{HC}-SN have been presented. Both later cases are different from the first ones because they involve hill climbing. Hill climbing has been used as the local search strategy for two later cases. The string encoding has been applied for these four algorithms. That is, each node is assigned a community label to which it belongs. Moreover, every four algorithms use the *extended modularity* and the *extended modularity density* for the signed networks as the objective function. The experiments on different networks indicate that Memetic approaches act better than evolutionary ones.

According to the literature, it can be said that most of the proposed algorithms for unsigned networks and especially signed networks are single-objective or multi-objective evolutionary algorithms. Evolutionary algorithms try to find communities that optimize one or more objective functions by exploring the problem state space and not making much use of the network structural information. Therefore, it is necessary to present new methods that make maximum use of network structural information. In this paper, a new method is proposed, which is based on weighted label propagation. The proposed method is not an evolutionary algorithm and finds communities by weighted label propagation. The proposed method does not directly optimize any objective function. However, it uses label propagation (network structural information) to find good communities in terms of community structure quality criteria. The use of network structural information has been the main motivation for presenting the proposed method. In the next section, details of the proposed method are given.

3. WLPA algorithm

In this section, by using MinHash and label propagation, an algorithm called WLPA has been proposed for detecting community structure in signed and unsigned social networks. WLPA includes the following two general phases: 1) assigning weight to the network edges, and 2) determining the community structure. In phase 1, first, the similarity of all adjacent nodes is estimated by using MinHash. Then, each edge is assigned a weight equal to the estimated similarity of its end nodes. In phase 2, the community structure of the network is determined through the weighted label propagation. In sub-section 3-1, we first explain how to determine the Jaccard similarity of two adjacent nodes in unsigned and signed social networks. Also, a single equation is introduced to

determine the Jaccard similarity of two adjacent nodes in both unsigned and signed social networks. Then, we explain how to estimate the Jaccard similarity using MinHash. The reason for estimating the Jaccard similarity is that the exact calculation of it for all the adjacent nodes in large scale networks is time-consuming. Afterward, in sub-section 3-2, the details of WLPA are explained. Finally, in sub-section 3-3, the time complexity of WLPA is analyzed.

3-1. Jaccard similarity and MinHash

A) Jaccard similarity for unsigned networks

Jaccard similarity [24] is used for determining the amount of similarity between two sets and is calculated by equation 1.

$$Sim_{jaccard}(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

To determine the similarity between two nodes u and v in an unsigned social network using Jaccard similarity, set A is considered as the set of neighbors of node u and set B as the set of neighbors of node v. So, we have:

$$Sim_{jaccard}^{unsigned}(u,v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{2}$$

In the above equation, in calculating the similarity between two nodes u and v, the existence/non-existence of an edge between them has no impact, whereas the existence/non-existence of an edge between these two nodes must have some impacts in their similarity. So, in this paper, the Jaccard similarity has been modified in order to resolve this issue. In the modified Jaccard similarity, the node has also been considered as a member of its neighboring set.

$$Sim_{modified\_jaccard}^{unsigned}(u,v) = \frac{|\{N(u),u\} \cap \{N(v),v\}|}{|\{N(u),u\} \cup \{N(v),v\}|} \tag{3}$$

For example, consider the networks of Figures 2(a) and 2(b). In the network of Figure 2(a), there is no edge between two nodes c and d, and in the network of Figure 2(b), there is an edge between them. Since there is an edge between nodes c and d in the network of Figure 2(b), it is expected that the Jaccard similarity between them is more than the network of Figure 2(a). According to the equation 2, Jaccard similarity between two nodes c and d in the network of Figure 2(a) is 0.5, and in the network of Figure 2(b) is 0.3333, but this is not as expected. This problem has been eliminated in the modified Jaccard similarity. According to the equation 3, the modified Jaccard similarity between two nodes c and d in the network of Figure 2(a) is 0.3333, and in the network of Figure 2(b) is 0.6667, and this is as expected.

B) Extension of the Jaccard similarity for signed networks

In the signed networks, the sign of edges must be considered to calculate the similarity between two nodes. In these networks, the set of neighbors of a node will be divided into two separate positive and negative ones. The set of positive neighbors of a node includes the nodes connected to it with a positive edge, and the set of negative neighbors of a node includes the nodes connected to it with a negative edge. It is obvious that in calculating the similarity between two nodes, the intersection of their neighboring set with the same signs (positive with positive and negative with negative) has a positive effect and the intersection of their neighboring set with the different signs

(positive with negative and negative with positive) has a negative effect on the similarity of them. So, the Jaccard similarity between two nodes u and v in a signed network will be as the equation 8.

$$N^+(x) = \{\{y|(x,y) \in E^+\} \cup x\} \tag{4}$$

$$N^-(x) = \{y|(x,y) \in E^-\} \tag{5}$$

$$S^+(u,v) = |N^+(u) \cap N^+(v)| + |N^-(u) \cap N^-(v)| \tag{6}$$

$$S^-(u,v) = |N^+(u) \cap N^-(v)| + |N^-(u) \cap N^+(v)| \tag{7}$$

$$Sim^{signed}_{modified\_jaccard}(u,v) = \frac{S^+(u,v) - S^-(u,v)}{|\{N(u),u\} \cup \{N(v),v\}|} \tag{8}$$

In the above equations, $N^+(x)$ is the set of positive neighbors of node x, $N^-(x)$ is the set of negative neighbors of node x, $S^+(u,v)$ is the number of the common same sign neighbors of two nodes u and v and $S^-(u,v)$ is the number of the common different sign neighbors of two nodes u and v. Supposing that the signs of all the network edges are positive (the network is unsigned), the equation 8 will be reduced to equation 3. So, equation 8 can be used to calculate the similarity between two nodes in both signed and unsigned networks.

C) Estimating the Jaccard similarity by using MinHash

Calculating the Jaccard similarity between two sets A and B with n members has time complexity $O(n \log n)$. The base of the logarithm is 2. To calculate the similarity between two nodes in a social network, the similarity between their neighbors must be calculated. Supposing that the number of neighbors of a node is d_{avg} on average, the time complexity of calculating the Jaccard similarity between two nodes will be $O(d_{avg} \log d_{avg})$. Therefore, calculating the Jaccard similarity between all adjacent nodes will be $O(|E| d_{avg} \log d_{avg})$. $|E|$ is the number of edges or the number of node pairs that are connected. This time is too much for the networks in which the number of edges and/or the average of node degrees is high. To decrease the required time for calculating the Jaccard similarity between two nodes using equation 8, we can estimate it by using MinHash [25].

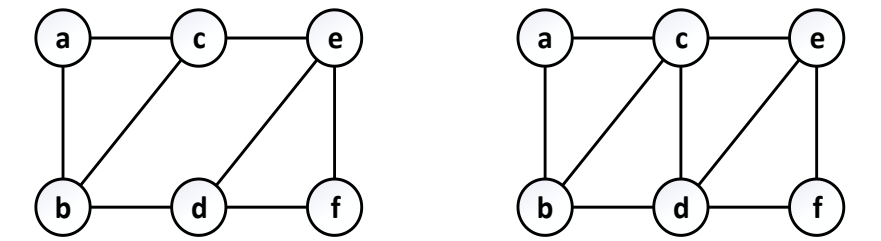


Figure 2. (a) Comparison of Jaccard similarity and modified Jaccard similarity between two non-adjacent nodes c and d in an unsigned sample network and (b) Comparison of Jaccard similarity and modified Jaccard similarity between two adjacent nodes c and d in an unsigned sample network.

Suppose that we want to estimate the Jaccard similarity between two sets A and B with n members by using MinHash. First, by using a hash function like h, each member of the sets A and B is hashed. Then, a member with the minimum hash value is found from each set. Suppose that $h_{min}(A)$ is equal to a member of set A with the minimum hash value and $h_{min}(B)$ is equal to a member of set B with the minimum hash value. In other words:

$$h_{min}(A) = \underset{x \in A}{arg\min}\, h(x)$$
$$h_{min}(B) = \underset{x \in B}{arg\min}\, h(x) \quad (9)$$

$I[h_{min}(A) = h_{min}(B)]$ is a simple estimator for the Jaccard similarity, which

$$I[x] = \begin{cases} 1, & x\ is\ true \\ 0, & otherwise \end{cases} \quad (10)$$

$h_{min}(A)$ is equal to $h_{min}(B)$ if and only if $h_{min}(A \cup B) \in \{A \cap B\}$. So, we have:

$$\begin{aligned} P[h_{min}(A) = h_{min}(B)] &= P[h_{min}(A \cup B) \in \{A \cap B\}] \\ &= \frac{|A \cap B|}{|A \cup B|} \\ &= Sim_{jaccard}(A, B) \end{aligned} \quad (11)$$

The estimator presented above has much variance and its value will always be 0 or 1. To decrease the variance, we can use k hash functions $h^1, h^2, \dots, h^k$ instead of a single hash function. So, we have:

$$Sim_{jaccard}(A, B) = \frac{1}{k}\sum_{i=1}^{k} I\left[h^i_{min}(A) = h^i_{min}(B)\right] \quad (12)$$

Estimating the Jaccard similarity between two sets A and B with n members by using equation 12 has time complexity $O(nk)$ and error $O\left(1/\sqrt{k}\right)$. For example, to have an error of less than or equal to 0.1, k must be set to 100. In the proposed method, it is necessary to calculate the Jaccard similarity between all adjacent nodes. By using equation 12, estimating the Jaccard similarity between all adjacent nodes will have time complexity $O(|E|k)$. So, if $k < d_{avg} \log_2 d_{avg}$, estimating the Jaccard similarity between all adjacent nodes will take less time than the exact calculating of it.

3-2. WLPA phases

Due to the advantages of the standard label propagation algorithm [15], the major deficiency of this algorithm has not been considered so far. Its major deficiency is that when a node wants to adopt a label, it considers the same significance (weight) for the label of all neighboring nodes. This cannot be true in social networks because all the friends/hostiles of an individual in a social network do not have the same significance, and the intensity of friendship/hostility is different for different friends/hostiles. In other words, the influence of different friends/hostiles of an individual on him/her is not the same. Inspired by this point, in WLPA, by using MinHash, a weight is assigned to the label of the neighboring nodes of a given node, and the label of the node is updated based on the weighted label of its neighboring nodes. In the following, the details of WLPA are explained. WLPA includes two general phases. In the first phase, a weight is assigned to each edge of the

network, and in the second phase, the community structure of the network is determined through the weighted label propagation. Each phase is explained in detail below.

Phase 1: in this phase, at first, the similarity of all adjacent nodes is estimated by using MinHash (equation 12). The reason for estimating the Jaccard similarity is that the exact calculation of it for all the adjacent nodes in large scale networks is time-consuming (see sub-section 3-1 for more details). Then, each edge is assigned a weight equal to the estimated similarity of its end nodes. The weights assigned to the edges somehow indicate the intensity of communication between users. For example, consider the network of Figure 3(a). The corresponding weighted network is shown in Figure 3(b). The weight of each edge is equal to the Jaccard similarity (estimated Jaccard similarity) of its end nodes.

Phase 2-Step1: in this step, the label of each node is initialized by its identifier (ID). The label of a node indicates the community identifier to which the node belongs.

Phase 2-Step 2: in this step, the label of all nodes is updated based on the weighted label of their neighboring nodes according to a local rule. In WLPA, majority voting has been used as a local rule. For example, consider the weighted network of Figure 4. The current label of the nodes is written inside the circles, and the name of the nodes is written next to the circles. Assume that the number on the edges indicates the Jaccard similarity of the end nodes, which are estimated by MinHash (equation 12). By using majority voting, the middle node n_i tries to update its label to that of the neighboring nodes, which overall has the most positive similarity and least negative similarity with them. Although the middle node n_i has the most positive similarity with node $n3$, it does not adopt the node $n3$ label, i.e., "a". Because in this case, its label will be equal to the node $n1$ label, which has much negative similarity with it. According to the updating rule (majority voting), the label of the middle node n_i will be equal to "b".

Phase 2-Step 3: step 2 of phase 2 is repeated until the label of all the nodes is converged. At the end, connected nodes with the same label form a community.

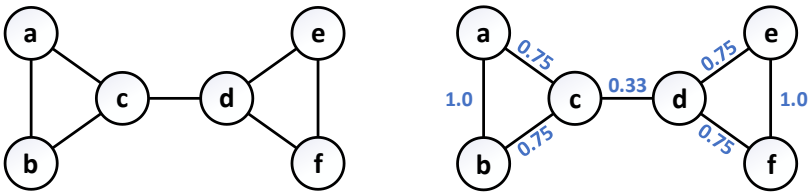
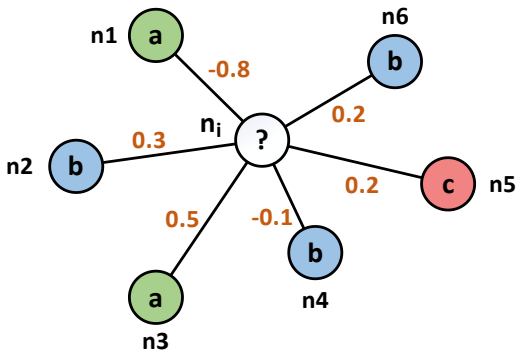


Figure 3. (a) A sample network and (b) corresponding weighted network.



$$label(n_i) = Majority\_Voting((-0.8 + 0.5)a, (0.3 - 0.1 + 0.2)b, 0.2c)$$
$$= Majority\_Voting(-0.3\text{a}, 0.4\text{b}, 0.2\text{c}) = b$$

Figure 4. Updating the label of a node in WLPA.

WLPA Algorithm

Input: Social Network Graph $G = (V, E)$
Output: Community Structure com

Phase 1:
    estimate the similarity of all adjacent nodes by using MinHash (equation 12)
    assign a weight to each edge equal to the estimated similarity of its end nodes

Phase 2:
    foreach node n in the network
        Label(n) = the identifier (ID) of the node n
    end

    repeat
        forall nodes in the network asynchronously
            Label(n_i) = **Majority_Voting**(weighted label of the neighboring nodes to n_i)
        end
    until the label of all the nodes is converged

com = connected nodes with the same label form a community
return com

Figure 5. Pseudo-code of the WLPA algorithm.

The pseudo-code of WLPA for detecting community structure in signed and unsigned social networks has been given in Figure 5.

3-3. The time complexity of WLPA

In this sub-section, the time complexity of WLPA has been analyzed. In WLPA, first, the similarity of all adjacent nodes is estimated by using MinHash. Estimating the similarity of all adjacent nodes has time complexity $O(|E|k)$. Then, the label of all the nodes is initialized. This requires $O(|V|)$ time. Afterward, in a repeat loop, the label of all the nodes is updated asynchronously until the label of all the nodes is converged. Since updating the label of a node such as n needs $O(d_n)$ time, updating the label of all the nodes needs $O\left(\sum_{n=1}^{|V|} d_n\right) = O(|E|)$ time. d_n is the degree of the node n. Supposing that the maximum number of repetitions is equal to the constant number T, the repeat loop of updating the label of nodes needs $O(T|E|)$ time. So, the total time complexity of WLPA will be equal to $O(|V| + (k + T)|E|) \cong O(|V| + |E|)$, which is linear in terms of the number of nodes and edges.

4. Experimental results and discussion

In this section, the efficiency of WLPA has been evaluated by both signed and unsigned benchmark networks and has been compared with state-of-the-art algorithms GenLouvain [26, 27], CPMap [28], MEAs-SN [18], SN-MOGA [21], LPA [15], and SLPA [16]. The algorithms GenLouvain, CPMap, MEAs-SN, and SN-MOGA are for detecting community structure in the signed networks, and the algorithms LPA and SLPA are for detecting community structure in the unsigned networks. All the algorithms have been run on a PC with a quad-core 4GHz processor and 24GB of RAM. For each algorithm,

using preliminary experiments, the best value for the parameters of that algorithm is determined and used in the main experiments. It should be noted that most of these values are the same as the values reported in the article of relevant algorithms. In the algorithm GenLouvain, nodes consideration order and nodes movement type are set to "fixed order" and "best move", respectively. In the algorithm CPMap, the accuracy, resolution interval, probability of teleportation in map equation and number of threads for parallel computations are set to 0.002, [0.001, 0.05], 0.1 and 4, respectively. In the algorithm MEAs-SN, the population size, crossover rate, mutation rate, the neighborhood size, the tightness power and maximum number of generations are set to 100, 0.8, 0.2, 21, 0.5 and 100, respectively. In the algorithm SN-MOGA, the population size, crossover rate, mutation rate and maximum number of generations are set to 100, 0.8, 0.6 and 100, respectively. In the algorithm SLPA, the version v1 of the speaker-listener rule is used and the number of repetitions is set to 25. In WLPA, the maximum number of repetitions and the number of hash functions, k, are set to 100 and 50, respectively. Considering $k = 50$, the relation $k < d_{avg} \log_2 d_{avg}$ is fulfilled for all the used benchmark networks. Therefore, estimating the Jaccard similarity between all adjacent nodes will take less time than the exact calculating of it.

4-1. Evaluation criteria

To evaluate the efficiency of WLPA and comparing it with other algorithms, two criteria *modularity* (Q) and *normalized mutual information* (NMI) have been used. The modularity criterion compares the density of edges inside the communities with the expected number in the null model. NMI is an external criterion and is used when the ground-truth community structure is available. The criterion NMI is used for estimating the similarity of a community structure with the ground-truth community structure.

The modularity criterion for unsigned networks is defined as [29]:

$$Q = \frac{1}{2m} \sum_{v,w \in V} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(C_v, C_w) \tag{13}$$

where A is the adjacency matrix of the network, m is the number of the network edges, k_v is the degree of node v and C_v is the community of node v. The Kronecker delta function $\delta(C_v, C_w)$ is 1 if the nodes v and w are in the same community, and 0 otherwise. The criterion Q lies in the interval $[-0.5,1]$ and the greater value indicates that the density of edges inside the communities is more than the expected number in the null model. The modularity criterion for signed networks is defined as [30]:

$$Q_{signed} = \frac{1}{2m^+ + 2m^-} \sum_{v,w \in V} \left[A_{vw} - \left(\frac{k_v^+ k_w^+}{2m^+} - \frac{k_v^- k_w^-}{2m^-} \right) \right] \delta(C_v, C_w) \tag{14}$$

where m^+ is the number of the positive edges, m^- is the number of the negative edges, k_v^+ is the positive degree of node v and k_v^- is the negative degree of node v.

The criterion NMI for two partitions (two community structure) A and B is defined as [31]:

$$NMI(A,B) = \frac{-2 \sum_{i=1}^{c_A} \sum_{j=1}^{c_B} C_{ij} \log \left(\frac{n C_{ij}}{C_{i.} C_{.j}} \right)}{\sum_{i=1}^{c_A} C_{i.} \log \left(\frac{C_{i.}}{n} \right) + \sum_{j=1}^{c_B} C_{.j} \log \left(\frac{C_{.j}}{n} \right)} \tag{15}$$

where C is a c_A by c_B matrix, which is called the *confusion matrix*. The element C_{ij} of the confusion matrix is the number of nodes in the community i of partition A that are also in the community j of partition B. In the above equation, n is the number of the network nodes, c_A (c_B) is the number of the communities in partition $A(B)$ and $C_{i.}(C_{.j})$ is the sum of the elements in row i (column j). The criterion NMI lies in the interval $[0,1]$ and the greater value indicates that two partitions A and B have more similarity to each other. If $A = B$, $NMI(A,B) = 1$, if A and B are completely different, $NMI(A,B) = 0$.

4-2. Benchmark networks

Real-world and synthetic benchmark networks have been used to evaluate the efficiency of the WLPA. To generate unsigned synthetic benchmark networks, the $LFR(n,k,maxk,t_1,t_2,minc,maxc,\mu)$ [32] generator has been used. In LFR benchmark networks, both the node degrees and community sizes follow the power-law distribution. In LFR generator, n is the number of nodes; k and $maxk$ are the average and maximum degree of nodes, respectively; t_1 and t_2 are the minus exponent of degree and community size distribution, respectively; $minc$ and $maxc$ are the minimum and maximum size of communities, respectively. The mixing parameter μ is used for controlling the community structure in the network and indicates a fraction of the edges which a node has to other community nodes. So, by increasing this parameter, the community structure in the network becomes unclear. To generate signed synthetic benchmark networks the extended LFR for the signed networks [18, 33] is used. In the extended LFR for the signed networks, in addition to the above parameters, two other parameters P^- and P^+ are also considered. The parameter P^- indicates a fraction of negative edges inside the communities and parameter P^+ indicates a fraction of positive edges between the communities. Ideally, negative edges should be between communities and positive edges should be inside communities. So, these two parameters are used for creating noise in the community structure. Similar to the mixing parameter μ, by increasing these two parameters, the community structure in the network becomes unclear. In Tables 1 and 2, general information of the benchmark networks used to evaluate the WLPA are given.

Table 1. General information of the synthetic benchmark networks.

Name*	n	K	maxk	t_1	t_2	minc	maxc	μ, P^-**, and P^+**
LFR_x1	1000	10	20	2	1	20	100	{0.0,0.1,0.2,0.3,0.4,0.5}
LFR_x2	1000	15	30	2	1	20	100	
LFR_x3	1000	20	50	2	1	20	100	
LFR_x4	1000	40	100	2	1	20	100	
LFR_x5	5000	15	40	2	1	20	200	
LFR_x6	5000	25	60	2	1	20	200	
LFR_x7	5000	40	100	2	1	20	200	
LFR_x8	5000	80	200	2	1	20	200	
LFR_x9	10000	15	40	2	1	20	400	
LFR_x10	10000	25	60	2	1	20	400	
LFR_x11	10000	40	100	2	1	20	400	
LFR_x12	10000	80	200	2	1	20	400	

* for signed benchmark networks, the letter "x" is replaced by "s" and for unsigned benchmark networks by "u".
** Parameters P^- and P^+ only exist in signed benchmark networks.

Table 2. General information of the real-world benchmark networks.

Name	Type	Description	Node	Edge
Karate	Unsigned	Zachary karate club network	34	78
Dolphin	Unsigned	Bottlenose dolphins' network	62	159
Football	Unsigned	American football network	115	613
Supreme Court	Signed	U.S. supreme court justices' network	9	36
Slovene	Signed	Slovene parliamentary party network	10	45
Gahuku-Gamma	Signed	Gahuku-Gama subtribes network	16	58

4-3. Evaluation by signed benchmark networks

In this section, the efficiency of WLPA has been evaluated on the signed benchmark networks. First, comprehensive results from the operation of the WLPA have been presented. Then, the efficiency of WLPA on the synthetic benchmark networks has been compared with some of the existing algorithms. At last, the efficiency of WLPA has been investigated on the real-world benchmark networks.

a) Operation of the WLPA on the synthetic benchmark networks: in Figures 6 and 7, the average Q and NMI obtained from the WLPA for different combinations of μ, P^- and P^+ on the signed synthetic benchmark networks LFR_s7 have been presented. As you see, by increasing mixing parameter μ because the community structure becomes unclear, the efficiency of WLPA has been decreased in terms of both Q and NMI criteria. Furthermore, WLPA has more sensitivity to the parameter P^- than parameter P^+. That is, by increasing P^+ (for fixed values of μ and P^-), the algorithm efficiency does not change much, whereas by increasing P^- (for fixed values of μ and P^+), the algorithm efficiency decreases significantly. The reason that the algorithm efficiency decreases significantly by increasing P^- is that by increasing this parameter, the number of negative edges inside the communities increases. So, the nodes may assign a very low weight to the label of their groupmate nodes and adopt a different community identifier than their groupmate nodes. The reason that the algorithm efficiency does not decrease so much by increasing P^+ is that by increasing this parameter, the number of positive edges between the communities increases. Since the number of positive edges between the communities is usually less than the number of positive edges inside them, they have no significant effect on the chosen label of a given node. It should be noted that the WLPA performs similarly on the networks LFR_s1, LFR_s2, LFR_s3, LFR_s4, LFR_s5, LFR_s6, LFR_s8, LFR_s9, LFR_s10, LFR_s11, and LFR_s12. So, the operation of the WLPA on these networks is not presented.

b) Comparing WLPA with the existing algorithms: in this section, WLPA has been compared with GenLouvain, CPMap, MEAs-SN, and SN-MOGA. In Figures 8 and 9, the average Q and NMI obtained from the different algorithms for different combinations of μ, P^- and P^+ on the signed synthetic benchmark networks LFR_s7 have been presented, respectively. As you see, by increasing mixing parameter μ because the community structure becomes unclear and also by increasing P^- because of increasing noise inside the communities, the efficiency of all the algorithms has been decreased. Furthermore, all the algorithms have very low sensitivity against increasing P^+ (noise between the communities) so that by increasing P^+, their efficiency does not decrease much. For all combinations of $\mu \in [0,1]$, $P^- \in [0,0.2]$ and $P^+ \in [0,1]$, the efficiency of WLPA is approximately equal to the algorithms GenLouvain and CPMap. By increasing P^- ($P^- \geq 0.3$), WLPA outperforms all the algorithms. It should be noted that by increasing noise inside the communities ($P^- > 0.3$), the efficiency of the algorithm GenLouvain is decreased sharply in terms of the NMI criterion. Also, for all combinations of μ, P^- and P^+ except for some cases, algorithm SN-MOGA has the worst

efficiency. In short, it can be said that WLPA performs better than all the algorithms in terms of both Q and NMI criteria.

In Table 3, the average Q and NMI obtained from the different algorithms on the signed synthetic benchmark networks have been reported. Each data is an average on the obtained Q/NMI by an algorithm for all the combinations of μ, P^- and P^+. The largest obtained Q/NMI for each network is boldfaced. As you see, WLPA has found the community structure with the largest Q_{signed} on all the networks except for LFR_s1, LFR_s3, LFR_s9, and LFR_s11. In LFR_s1, LFR_s3, LFR_s9, and LFR_s11, the GenLouvain algorithm found the community structure with the largest Q_{signed}. WLPA has also found the community structure with the largest NMI on all the networks except for LFR_s1 and LFR_s9. In LFR_s1 and LFR_s9, the CMPap algorithm found the community structure with the largest NMI. Values close to 1 for the NMI criterion indicate that WLPA has been able to find a community structure close to the ground-truth community structure. Note that the SN-MOGA algorithm is not applicable in large scale networks LFR_s8, LFR_s9, LFR_s10, LFR_s11, and LFR_s12 due to its very high time complexity.

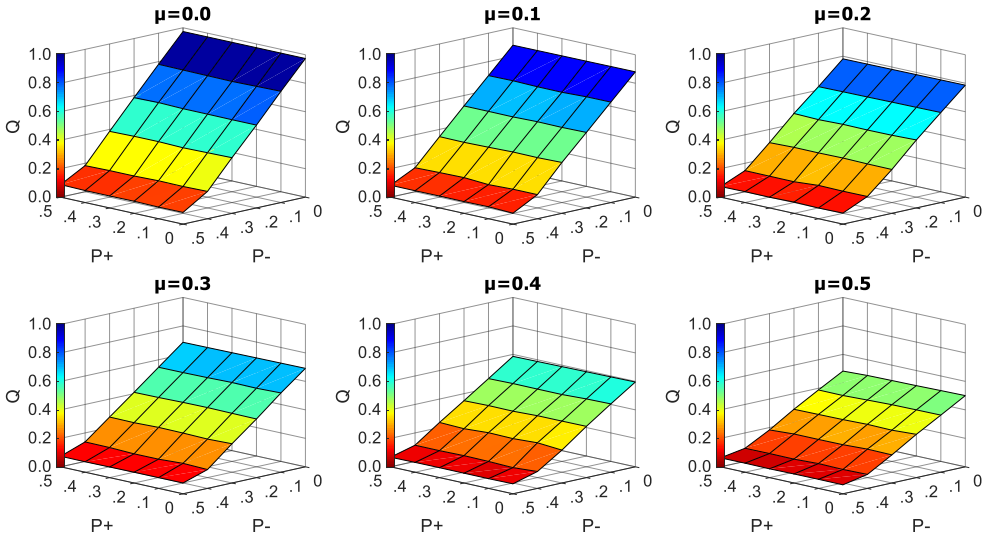


Figure 6. Average Q obtained from the WLPA for different combinations of μ, P^- and P^+ on the LFR_s7 networks.

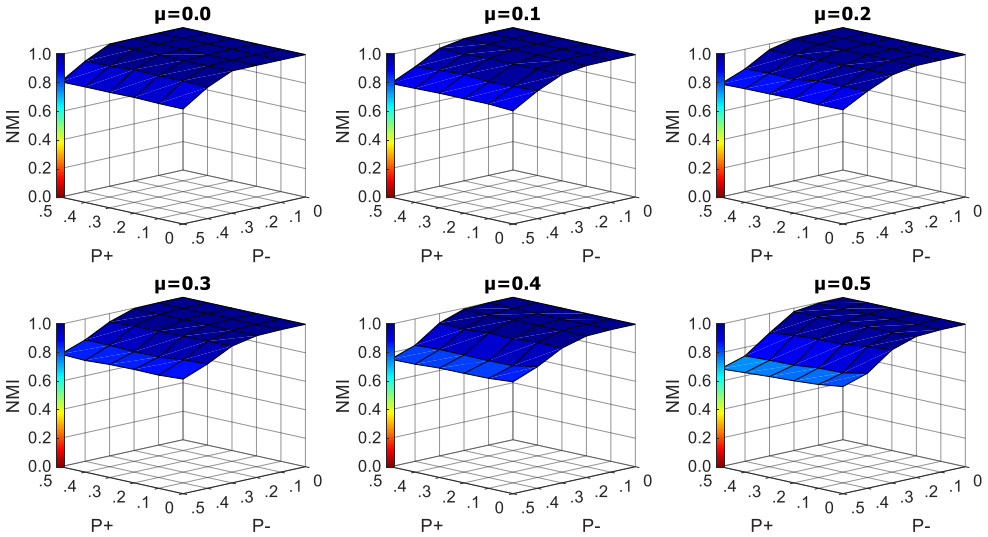


Figure 7. Average NMI obtained from the WLPA for different combinations of μ, P^- and P^+ on the LFR_s7 networks.

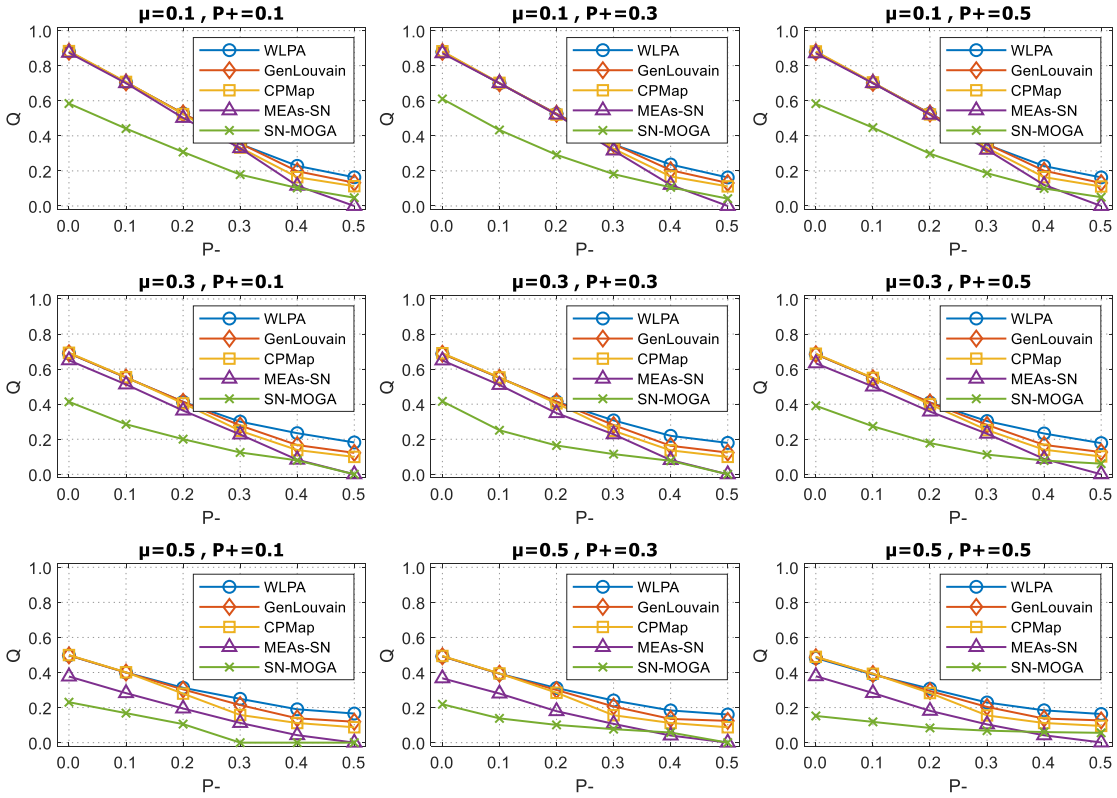


Figure 8. Average Q obtained from the different algorithms for different combinations of μ, P^- and P^+ on the LFR_s7 networks.

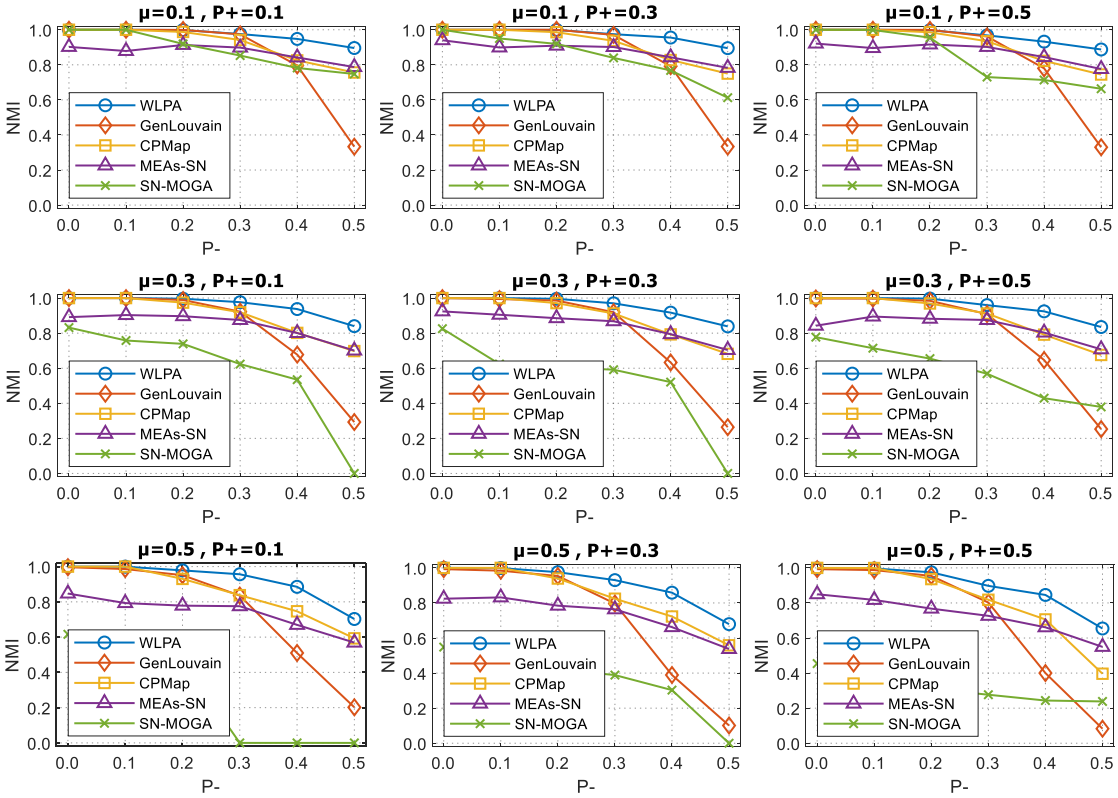


Figure 9. Average NMI obtained from the different algorithms for different combinations of μ, P^- and P^+ on the LFR_s7 networks.

In Table 4, the average execution time of different algorithms on the signed synthetic benchmark networks has been reported. The shortest obtained execution time for each network is boldfaced. As you see, in small networks (networks with 1000 nodes) GenLouvain and large networks (networks with 5000 and 10000 nodes) CPMap have less execution time. Although the execution time of WLPA in all networks is longer than the algorithms GenLouvain and CPMap, its execution time increases linearly with the network size. As you see, it only takes 4.38s for networks with 1000 nodes, 19.66s for networks with 5000 nodes, and 40.72s for networks with 10000 nodes, on overage, which are acceptable.

Table 3. Average Q and NMI obtained from the different algorithms on the signed synthetic networks.

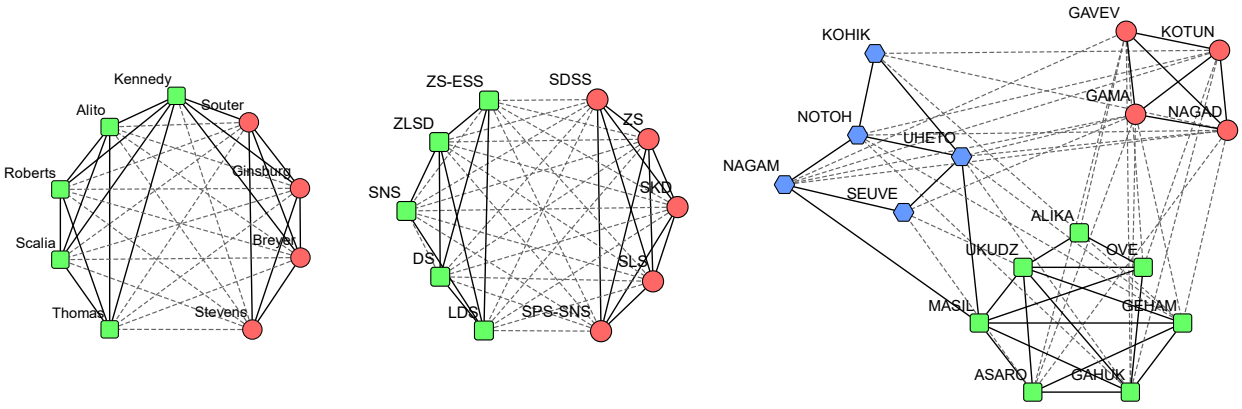
Network	Q_{signed}					NMI				
	WLPA	Gen-Louvain	CPMap	MEAs-SN	SN-MOGA*	WLPA	Gen-Louvain	CPMap	MEAs-SN	SN-MOGA*
LFR_s1	0.3769	**0.4355**	0.4051	0.1499	0.2351	0.7512	0.6824	**0.7954**	0.6377	0.6450
LFR_s2	**0.4161**	0.4131	0.3905	0.2077	0.2392	**0.8396**	0.7340	0.8358	0.6935	0.6542
LFR_s3	0.3906	**0.4038**	0.3873	0.2866	0.2533	**0.8807**	0.7592	0.8665	0.7646	0.7012
LFR_s4	**0.3848**	0.3722	0.3737	0.3588	0.2427	**0.9395**	0.8045	0.9096	0.8290	0.6741
LFR_s5	**0.4228**	0.4223	0.4003	0.1845	0.1671	**0.8692**	0.7364	0.8686	0.7368	0.6117
LFR_s6	**0.4420**	0.4059	0.3830	0.2619	0.1785	**0.9246**	0.7789	0.8900	0.7989	0.6193
LFR_s7	**0.3974**	0.3947	0.3798	0.3405	0.1932	**0.9439**	0.8078	0.9130	0.8520	0.5814
LFR_s8	**0.3930**	0.3834	0.3773	0.3782	-	**0.9731**	0.8404	0.9408	0.8711	-
LFR_s9	0.4020	**0.4219**	0.4025	0.1350	-	0.8286	0.7325	**0.8618**	0.6927	-
LFR_s10	**0.4343**	0.4059	0.3856	0.1947	-	**0.9028**	0.7768	0.8863	0.7552	-
LFR_s11	0.3951	**0.3953**	0.3720	0.2848	-	**0.9293**	0.8068	0.8989	0.8216	-
LFR_s12	**0.3946**	0.3851	0.3718	0.3742	-	**0.9687**	0.8410	0.9240	0.8727	-

*This algorithm is not applicable in large scale networks LFR_s8, LFR_s9, LFR_s10, LFR_s11, and LFR_s12 due to its very high time complexity.

Table 4. The average execution time of different algorithms on the signed synthetic networks (k denotes kilo).

Network Info.			Execution Time (Second)				
Network	Nodes	Edges	WLPA	Gen-Louvain	CPMap	MEAs-SN	SN-MOGA*
LFR_s1	1000	5k	4.58	**0.67**	1.52	11.80	65.24
LFR_s2	1000	7.5k	4.70	**0.41**	1.25	9.85	92.85
LFR_s3	1000	10k	4.53	**0.36**	1.26	10.76	146.82
LFR_s4	1000	20k	3.71	**0.28**	1.25	12.74	257.28
LFR_s5	5000	37.5k	22.82	12.45	**2.56**	140.32	1606.48
LFR_s6	5000	62.5k	26.12	11.78	**2.60**	123.45	2095.72
LFR_s7	5000	100k	19.47	7.94	**2.74**	139.50	8563.31
LFR_s8	5000	200k	10.25	4.80	**3.47**	181.59	-
LFR_s9	10000	75k	49.07	69.99	**4.08**	572.25	-
LFR_s10	10000	125k	64.98	49.42	**4.74**	453.32	-
LFR_s11	10000	200k	22.00	38.83	**5.26**	440.25	-
LFR_s12	10000	400k	26.81	22.25	**6.66**	552.39	-

*This algorithm is not applicable in large scale networks LFR_s8, LFR_s9, LFR_s10, LFR_s11, and LFR_s12 due to its very high time complexity.



(a) (b) (c)

Figure 10. Community structure found by the WLPA on the signed real-world networks. Positive edges are shown with solid lines, and negative edges are shown with dashed lines. Different communities are shown with different shapes and colors. (a) U.S. supreme court judges network, (b) Slovene parliamentary parties network, and (c) Gahuku-Gama subtribes network.

c) Operation of the WLPA on the real-world benchmark networks: in this section, the operation of the WLPA has been investigated on three signed real-world benchmark networks. The first network is the network of the U.S. supreme court judges, which describes the voting behavior of nine judges in the supreme court of the United States during 2006-2007. Positive edges mean that a judge supports another, and negative edges have contradictory meaning. The second network is the network of the Slovene parliamentary parties, which represents the relationships between ten parties of the Slovenian parliamentary in 1942. Positive edges mean that the parliamentary activities of two parties are the same, whereas negative edges mean that their activities are different. The third network is the network of the Gahuku-Gama subtribes, which reflects political alliances and oppositions among 16 Gahuku-Gama subtribes, which have been distributed in a particular area and are involved in warfare with each other. Positive and negative edges represent the political arrangements with positive and negative ties, respectively. Figure 10 illustrates the community structure found for these networks by the WLPA. In all three networks, the community structure found by the WLPA is as same as the ground-truth community structure.

4-4. Evaluation by unsigned benchmark networks

In this section, the efficiency of WLPA has been evaluated on the unsigned benchmark networks. First, the efficiency of WLPA on the synthetic benchmark networks has been compared with some of the existing algorithms. Then, its efficiency has been investigated on the real-world benchmark networks.

a) Comparing WLPA with the existing algorithms: in this section, WLPA has been compared with LPA and SLPA. In Table 5, the average Q, NMI, and execution time (ET) obtained from the different algorithms on the unsigned synthetic benchmark networks have been reported. Each data is an average on the obtained Q/NMI/ET by an algorithm for all the values of mixing parameter μ. The largest obtained Q/NMI/ET for each network is boldfaced. As you see, WLPA has found the community structure with the largest Q on all the networks except for LFR_u1 and LFR_u11. In LFR_u1 and LFR_u11, the LPA algorithm found the community structure with the largest Q. WLPA has also found the community structure with the largest NMI on all the networks except for LFR_u5 and LFR_u9. In LFR_u5 and LFR_u9, the LPA algorithm found the community structure with the

largest NMI. In the networks LFR_u4, LFR_u7, LFR_u8, LFR-u10, and LFR_u12, WLPA has found the ground-truth community structure ($NMI = 1$). In the other networks except for LFR-u1, the community structure found is very close to the ground-truth community structure ($NMI \simeq 1$). Furthermore, WLPA has the longest and LPA has the shortest execution time in all benchmark networks.

b) Operation of the WLPA on the real-world benchmark networks: in this section, the operation of the WLPA has been investigated on three unsigned real-world benchmark networks. The first network is the network of the Zakary karate club, which shows the relationships between 34 members of a karate club. Because of the dispute between the club administrator (node 34) and the instructor (node 1), the club members were divided into two groups. The second network is the network of the bottlenose dolphins, which shows the frequent associations between 62 bottlenose dolphins in a community living off Doubtful Sound, New Zealand. The third network shows the American football games between Division IA colleges during the regular season in fall 2000. This network includes 115 nodes. Figure 11 illustrates the community structure found for these networks by the WLPA. In the Karate network, the community structure found by the WLPA is as same as the ground-truth community structure. In the Dolphins network, the orange dashed line shows the node divisions in the ground-truth community structure. As you can see, in the community structure found by the WLPA, one of the ground-truth communities has been divided into two smaller communities. Based on the network topological structure, this division is meaningful. In the Football network, in the community structure found by the WLPA, all the nodes are in the correct communities except for a few nodes. In the Dolphins and Football networks, the modularity of the community structure found by the WLPA is more than the modularity of the ground-truth community structure. The modularity of the community structure found by the WLPA for these two networks is 0.4886 and 0.6010, respectively, whereas the modularity of the ground-truth community structure for these two networks is 0.3735 and 0.5540, respectively.

Table 5. Average Q, NMI, and execution time obtained from the different algorithms on the unsigned synthetic networks.

Network	Q			NMI			Execution Time (Second)		
	WLPA	**LPA**	**SLPA**	**WLPA**	**LPA**	**SLPA**	**WLPA**	**LPA**	**SLPA**
LFR_u1	0.5982	**0.6038**	0.5697	**0.8414**	0.8074	0.7535	5.55	**0.45**	0.75
LFR_u2	**0.6927**	0.6265	0.6162	**0.9684**	0.8537	0.8403	4.62	**0.15**	0.54
LFR_u3	**0.6836**	0.6683	0.6188	**0.9948**	0.9522	0.8525	2.71	**0.16**	0.59
LFR_u4	**0.6753**	0.6568	0.6146	**1.0000**	0.9544	0.8566	2.20	**0.21**	0.82
LFR_u5	**0.7301**	0.7190	0.6810	0.9781	**0.9807**	0.9121	22.22	**0.46**	1.82
LFR_u6	**0.7487**	0.7269	0.7020	**0.9995**	0.9990	0.9428	15.78	**0.62**	2.23
LFR_u7	**0.7256**	0.7254	0.7240	**1.0000**	0.9991	0.9890	15.57	**0.87**	2.92
LFR_u8	**0.7205**	0.7203	0.7199	**1.0000**	0.9984	0.9866	9.37	**1.58**	5.48
LFR_u9	**0.7217**	0.7187	0.6639	0.9537	**0.9755**	0.8876	65.05	**0.99**	4.22
LFR_u10	**0.7515**	0.7288	0.7076	**1.0000**	0.9991	0.9529	38.25	**1.31**	5.06
LFR_u11	0.7280	**0.7282**	0.7264	**0.9999**	0.9996	0.9882	27.39	**1.80**	6.95
LFR_u12	**0.7262**	0.7261	0.7217	**1.0000**	0.9991	0.9828	27.00	**3.13**	12.93

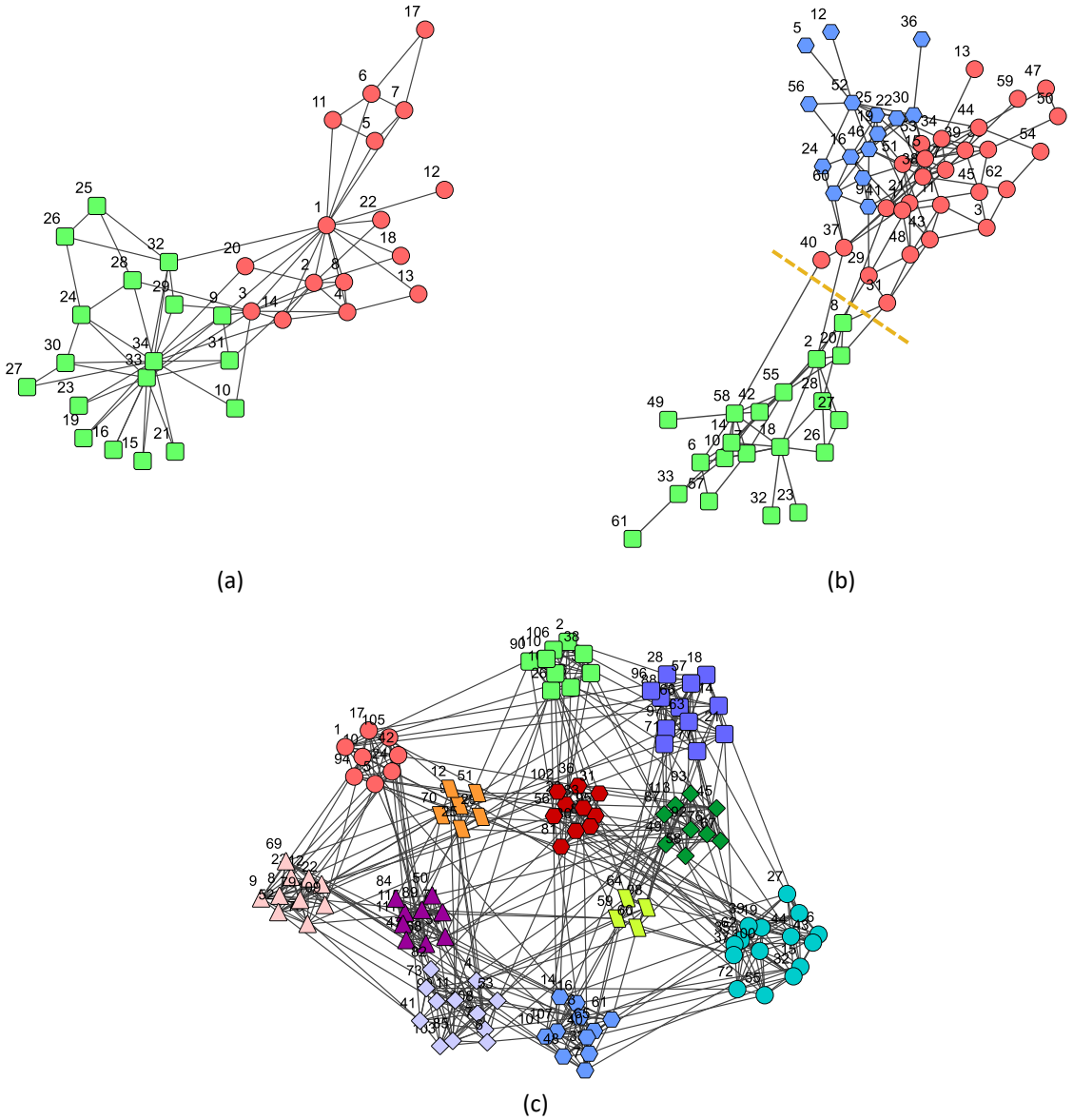


Figure 11. Community structure found by the WLPA on the unsigned real-world networks. Different communities are shown with different shapes and colors. (a) Zakary karate club network, (b) bottlenose dolphins network, and (c) American football network.

4-5. The memory complexity of algorithms

The amount of memory required for an algorithm depends on how the input network is stored in a typical implementation. In the implementations used in this paper, the edge-list method has been used for storing the input network. In the edge-list method, the memory required for storing a network with m edges is $2bm$ bit, which b is the number of bits required to store the ID of a node, e.g., 16. In algorithms WLPA, GenLouvain, CPMap, LPA, and SLPA, most of the memory consumption is related to the input network storage. The memory required for the other variables of the algorithms is negligible compared to the memory required to store the input network. Therefore, the memory complexity of these algorithms is $2bm \in O(m)$. In algorithms MEAs-SN and SN-MOGA, most of the memory consumption is related to the input network and population of solutions storage. The memory required for the other variables of the algorithms is also negligible. Therefore,

the memory complexity of these algorithms is $(2bm + pbn) \in O(m + n)$, which p and n are the population size and number of network nodes, respectively.

4-6. Discussion

One of the most important questions about the results of Sections 4-3 and 4-4 is what makes WLPA outperforms other algorithms. In this section, this important question is answered. As explained in sub-section 3-1, the introduced similarity index gives a more positive/negative weight for strong similar/dissimilar connections and less positive/negative weight for weak similar/dissimilar connections. Implicitly, the introduced similarity index maximizes positive inner-community links and negative intra-community links. In Phase 2-Step 2 of WLPA, the label of a node is updated by majority voting on the weighted label of its neighboring nodes. In other words, each node tries to update its label to that of the neighboring nodes, which overall has the most positive similarity and least negative similarity with them. In this way, in the final community structure, the density of positive edges inside the communities is maximized, and the density of negative edges inside/between the communities is minimized/maximized. This largely corresponds to the maximization of modularity and minimization of frustration. In short, WLPA assigns a weight to each link according to the similarity in neighborhoods, hence rapidly picks up the local clusters, differently from the compared algorithms. Unlike evolutionary algorithms that optimize one or more objective functions, WLPA does not directly optimize any objective function, but uses label propagation (network structural information) to find communities that are good in terms of community structure quality criteria. The use of network structural information has been the main motivation for presenting WLPA. Using network structural information leads to finding structurally meaningful communities. According to the results of Sections 4-3 and 4-4, it can be said that WLPA in both signed and unsigned benchmark networks has an excellent performance in terms of both criteria Q and NMI, and its efficiency in most networks is higher than compared algorithms. It should be noted that WLPA, unlike other algorithms and algorithms compared in this paper, is applicable in both signed and unsigned networks. The only drawback of WLPA is that its execution time is longer than algorithms GenLouvain, CPMap, LPA, and SLPA. However, because its execution time is linear in terms of network size, its execution time is acceptable. According to the results, the following can be summarized:

- In terms of criterion Q, WLPA performs better than other algorithms except for GenLouvain in all signed benchmark networks. In a small number of networks, GenLouvain performs better than WLPA.
- In terms of criterion NMI, WLPA performs better than other algorithms except for CPMap in all signed benchmark networks. In two networks (LFR_s1 and LFR_s9), CPMap performs better than WLPA.
- In terms of execution time, WLPA performs better than algorithms MEAs-SN and SN-MOGA and worse than algorithms GenLouvain and CPMap in signed benchmark networks. Furthermore, in small networks (networks with 1000 nodes) algorithm GenLouvain and large networks (networks with 5000 and 10000 nodes) algorithm CPMap has less execution time.
- In terms of both criteria Q and NMI, WLPA performs better than algorithm SLPA in all unsigned benchmark networks and better than algorithm LPA in most unsigned benchmark networks.

- In terms of execution time, WLPA performs worse than both algorithms LPA and SLPA in unsigned benchmark networks.
- In terms of memory complexity, algorithms WLPA, GenLouvain, CPMap, LPA, and SLPA are better than algorithms MEAs-SN and SN-MOGA.
- In short, WLPA has a good performance in both signed and unsigned benchmark networks with acceptable (linear) execution time.

5. Conclusion and future works

In this paper, by using MinHash and label propagation, an algorithm called WLPA has been proposed for detecting community structure in signed and unsigned social networks. WLPA takes into account the intensity of communications in addition to the communications. In WLPA, first, the similarity of all adjacent nodes is estimated by using MinHash. Then, each edge is assigned a weight equal to the estimated similarity of its end nodes. The weights assigned to the edges somehow indicate the intensity of communication between users. Finally, the community structure of the network is determined through the weighted label propagation. WLPA requires no prior knowledge about the community structure, such as the number and size of the communities, and identifies the communities of a network in linear time. Comprehensive experiments on the signed and unsigned benchmark networks with 1000, 5000, and 10000 nodes show that WLPA performs better on both signed and unsigned networks than compared algorithms. WLPA can only detect the communities in the single-layer networks. Extension of WLPA for detecting community structure in the multi-layer networks will be the focus of our work in the future.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

[1] J. Tang, Y. Chang, C. Aggarwal, and H. Liu, "A survey of signed network mining in social media," *ACM Computing Surveys (CSUR),* vol. 49, no. 3, p. 42, 2016.

[2] S. Fortunato, "Community detection in graphs," *Physics reports,* vol. 486, no. 3-5, pp. 75-174, 2010.

[3] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics reports,* vol. 659, pp. 1-44, 2016.

[4] B. Yang, W. Cheung, and J. Liu, "Community mining from signed social networks," *IEEE transactions on knowledge and data engineering,* vol. 19, no. 10, pp. 1333-1348, 2007.

[5] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences,* vol. 99, no. 12, pp. 7821-7826, 2002.

[6] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the national academy of sciences,* vol. 101, no. 9, pp. 2658-2663, 2004.

[7] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E,* vol. 69, no. 6, p. 066133, 2004.

[8] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E,* vol. 70, no. 6, p. 066111, 2004.

[9] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment,* vol. 2008, no. 10, p. P10008, 2008.

[10] C. Pizzuti, "GA-net: A genetic algorithm for community detection in social networks," in *International conference on parallel problem solving from nature*, 2008: Springer, pp. 1081-1090.

[11] Y. Park and M. Song, "A genetic algorithm for clustering problems," in *Proceedings of the third annual conference on genetic programming*, 1998, vol. 1998, pp. 568-575.

[12] C. Pizzuti, "A multiobjective genetic algorithm to find communities in complex networks," *IEEE Transactions on Evolutionary Computation,* vol. 16, no. 3, pp. 418-430, 2011.

[13] B. Zarei and M. R. Meybodi, "Detecting community structure in complex networks using genetic algorithm based on object migrating automata," *Computational Intelligence,* vol. 36, no. 2, pp. 824-860, 2020.

[14] B. Zarei, M. R. Meybodi, and B. Masoumi, "Chaotic memetic algorithm and its application for detecting community structure in complex networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science,* vol. 30, no. 1, p. 013125, 2020.

[15] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E,* vol. 76, no. 3, p. 036106, 2007.

[16] J. Xie, B. K. Szymanski, and X. Liu, "SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process," in *2011 ieee 11th international conference on data mining workshops*, 2011: IEEE, pp. 344-349.

[17] P. Doreian and A. Mrvar, "A partitioning approach to structural balance," *Social networks,* vol. 18, no. 2, pp. 149-168, 1996.

[18] C. Liu, J. Liu, and Z. Jiang, "A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks," *IEEE transactions on cybernetics,* vol. 44, no. 12, pp. 2274-2287, 2014.

[19] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation,* vol. 11, no. 6, pp. 712-731, 2007.

[20] J. Huang, H. Sun, Y. Liu, Q. Song, and T. Weninger, "Towards online multiresolution community detection in large-scale networks," *PloS one,* vol. 6, no. 8, p. e23829, 2011.

[21] A. Amelio and C. Pizzuti, "An evolutionary and local refinement approach for community detection in signed networks," *International Journal on Artificial Intelligence Tools,* vol. 25, no. 04, p. 1650021, 2016.

[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation,* vol. 6, no. 2, pp. 182-197, 2002.

[23] Y. Li, J. Liu, and C. Liu, "A comparative analysis of evolutionary and memetic algorithms for community detection from signed social networks," *Soft Computing,* vol. 18, no. 2, pp. 329-348, 2014.

[24] P. Jaccard, "Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines," *Bull Soc Vaudoise Sci Nat,* vol. 37, pp. 241-272, 1901.

[25] A. Z. Broder, "On the resemblance and containment of documents," in *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, 1997: IEEE, pp. 21-29.

[26] R. Rotta and A. Noack, "Multilevel local search algorithms for modularity clustering," *Journal of Experimental Algorithmics (JEA),* vol. 16, pp. 2.1-2.27, 2011.

[27] L. Jeub, M. Bazzi, I. Jutla, and P. Mucha, "A generalized Louvain method for community detection implemented in MATLAB," ed. http://netwiki.amath.unc.edu/GenLouvain, https://github.com/GenLouvain/GenLouvain, 2011-2019.

[28] P. Esmailian and M. Jalili, "Community detection in signed networks: the role of negative ties in different scales," *Scientific reports,* vol. 5, p. 14339, 2015.

[29] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E,* vol. 69, no. 2, p. 026113, 2004.

[30] S. Gómez, P. Jensen, and A. Arenas, "Analysis of community structure in networks of correlated data," *Physical Review E,* vol. 80, no. 1, p. 016114, 2009.

[31] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment,* vol. 2005, no. 09, p. P09008, 2005.

[32] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E,* vol. 78, no. 4, p. 046110, 2008.

[33] Y. Su, B. Wang, F. Cheng, L. Zhang, X. Zhang, and L. Pan, "An algorithm based on positive and negative links for community detection in signed networks," *Scientific reports,* vol. 7, no. 1, p. 10874, 2017.