Computing (2015) 97:483–507
DOI 10.1007/s00607-014-0433-6

A hybrid artificial immune network for detecting communities in complex networks

Amir-Mohsen Karimi-Majd · Mohammad Fathian · Babak Amiri

Received: 14 February 2014 / Accepted: 29 October 2014 / Published online: 12 November 2014

Abstract One of the challenging problems when studying complex networks is the detection of sub-structures, called communities. Network communities emerge as dense parts, while they may have a few relationships to each other. Indeed, communities are latent among a mass of nodes and edges in a sparse network. This characteristic makes the community detection process more difficult. Among community detection approaches, modularity maximization has attracted much attention in recent years. In this paper, modularity density (D value) has been employed to discover real community structures. Due to the inadequacy of previous mathematical models in finding the correct number of communities, this paper first formulates a mixed integer non-linear program to detect communities without any need of prior knowledge about their number. Moreover, the mathematical models often suffer from NP-Hardness. In order to overcome this limitation, a new hybrid artificial immune network (HAIN) has been proposed in this paper. HAIN aims to use a network's properties in an efficient way. To do so, this algorithm employs major components of the pure artificial immune network, hybridized with a well-known heuristic, to provide a powerful and parallel search mechanism. The combination of cloning and affinity maturation components, a strong local search routine, and the presence of network suppression and diversity are the main components. The experimental results on artificial and real-world complex networks illustrate that the proposed community detection algorithm provides a useful paradigm for robustly discovering community structures.

A.-M. Karimi-Majd · M. Fathian (✉)
Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran
e-mail: fathian@iust.ac.ir

B. Amiri
The University of Sydney, New South Wales, Australia

Keywords Complex network · Community detection · Mixed integer non-linear programming · Artificial immune network · Modularity-based maximization

Mathematics Subject Classification 91D30 · 90C27 · 68T20

1 Introduction

Recently, complex networks have been studied and developed in many areas of science. Internet, biological networks, power grids, social networks, food webs, and communication networks are well-known examples of complex networks. One of the main issues in modelling such networks is to extract hidden structures, called communities [18]. Detecting communities helps us to make sense of complex networks [9].

Communities consist of objects, called nodes, and their relationships, called edges. They emerge as dense parts in a network while they may have a few relationships to each other. Indeed, communities are latent among a mass of nodes and edges in a sparse network. This characteristic makes the community detection process more difficult. Moreover, we do not usually have any prior knowledge about the number of existent communities and their sizes (i.e., the number of nodes included).

In this paper, we focus on one of the well-known approaches for detecting communities, called modularity-based optimization, initiated from the Newman and Girvan (NG) method [44]. Their method is based on the maximization of a measure, called the modularity function Q, to distinguish the edges of a community from random by means of a heuristic search algorithm. In order to improve the NG method, some researchers proposed to optimize modularity function through a mathematical programming model [1,60,66]. However, solving mathematical models may fail to complete the optimization process for medium and large networks in proper time because of the NP-Hardness of those models, although they find the best solution for small ones. To tackle this issue, researchers proposed to employ different heuristic methods (e.g., [36,57,62]), and many metaheuristic algorithms (e.g., [3,21,23,27,33]).

The Q measure can work well if the respective network is truly modular [51] and communities do not significantly differ in terms of their sizes [31]. Unfortunately, no guarantee exists for network modularity and community uniformity; therefore, this measure may not provide reliable results [15,52]. Consequently, the abovementioned models inherit the stated limitations and drawbacks of the Q function.

Li et al. [34] have proposed another measure called modularity density or D measure in order to cope with this problem. D measure targets the average in-degree (i.e., the average number of edges coming to nodes), denoted by d_{in}, and the average out-degree (i.e., the average number of edges going out of nodes), denoted by d_{out}, of each potential community. This measure aims to find communities so that the maximum of d_{in} and minimum of d_{out} occurs. They also developed a mixed integer non-linear programming (MINLP) model for detecting communities based on the D measure. The number of communities must be known as prior information for their mathematical model. This limitation would be problematic in case of medium and large networks.

In this paper, first, a new mathematical model is proposed to tackle the problems so that automatically discovering the correct number of communities would be possible. Then, as a core contribution of this paper, a new metaheuristic is developed

based on D measure. Since the quality of detected communities extremely depends on initial grouping and method of exploring the search space, a metaheuristic algorithm that provides a highly robust and parallel search is needed for successful detection. The desired algorithm should be able to employ a dynamic local search for a discrete optimization problem because, in the community detection problem, a small change in grouping nodes might have a big effect on the quality of grouping. These characteristics, i.e., a highly robust and parallel search with a dynamic local search, can be found in artificial immune systems [55]. Thus, we select the artificial immune system algorithm family and develop a hybrid artificial immune network (HAIN), not only for detecting communities but also for finding the correct number of them simultaneously.

The rest of the paper is organized as follows. The next section presents related studies. Section 3 describes our new mathematical model. We illustrate components of the proposed metaheuristic in Sect. 4. Section 5 shows and analyses the experimental results of employing the proposed HAIN on artificial networks and well-known real ones. Finally, Sect. 6 provides some concluding remarks.

2 Related studies

Community is one of the most important features of networks, and community detection has been focused by different disciplines such as sociology, biology and computer science. The community detection problem has not reached a satisfactory level, despite huge studies on this problem in different fields over recent years. Community detection studies can be classified into seven major groups, as follows [16]:

Traditional methods: The traditional approaches for finding the natural grouping of given objects could be classified as graph partitioning, hierarchical clustering, partitional clustering, and spectral clustering methods. The graph partitioning methods try to divide nodes into a predefined number of clusters. Some graph partitioning algorithms that can work well include the Kernighan and Lin [29] algorithm, Fiedler vector [13], Goldberg and Tarjan [19], and Flake et al. [14] algorithms. Hierarchical clustering methods aim to find a hierarchy of clusters (groups of objects). In this group of methods, two strategies can be taken for clustering nodes: (1) grouping nodes in the same clusters with high levels of similarity, and (2) connecting nodes with low similarity. In this group of methods, similarity measure is very important because nodes will be grouped based on this [10]. Partitional clustering methods are another popular category of clustering data. In these methods, nodes would be clustered based on the distance as a measure of dissimilarity between nodes. One of the most famous methods in this category is [63]. The last group of methods in this class is spectral clustering, which consists of all methods that partition nodes into communities by using the eigenvectors of matrices [11].

Divisive algorithms: The idea of this group of methods is to detect and remove links that connect nodes of different communities in order to disjoin them from each other. Algorithms in this category are initiated by the research of Girvan and Newman [18]. These methods are based on the concept of edge betweenness: betweenness of an edge is defined as the total number of shortest paths between all node pairs in the network involving that edge [17]. The mentioned divisive algorithms could not detect

overlapping communities. To cope with this issue, Nepusz et al. [43] proposed the fuzzy community detection method in networks.

Modularity based methods: The word 'modularity' refers to the degree of separation of a network's components. Newman and Girvan proposed a modularity measure Q [44] to ascertain the quality of network clusters. Based on modularity assumptions, the higher value of modularity indicates a higher quality of the clustering. Thus, they used a simple greedy search method to find communities in a way that maximizes Q function. Unlike this approach, some studies have used mathematical modelling to focus on optimality. For example, Xu et al. [60] put forward a mixed integer quadratic programming (MIQP) method, Agrawal and Kempe [1] proposed a rounding linear programming (RLP) method, and Zhang and Wang [64] developed a mixed integer non-linear programming method to maximize modularity function. Another group of researchers proposed metaheuristic algorithms in order to find a trade-off between optimality and speed, such as simulated annealing [23,40,41], genetic algorithm [33, 37,53,56], memetic algorithm [20,21], immune system [22], ant colony optimization [27], and firefly algorithm [3].

Dynamic algorithms: Dynamic processes inspire these algorithms. Indeed, these algorithms can detect groups of similar nodes as communities based on their behaviour in the presence of a dynamic process. Random walks [28,67] and synchronization [4] are two well-known processes used for detecting communities in graphs. Other similar methods, called spin models [50], are based on the Potts model [59] developed in statistical mechanics. These algorithms, although inspired by nature, are slow in general.

Statistical inference: The aim of the methods based on statistical inference is to fit a model to the actual structure of a graph using observations and hypotheses. Generative models and block modelling models are the two most popular groups. The basis of generative models is to maximize the likelihood of reaching fitness in comparison with the actual graph. Bayesian inference [25] and expectation-maximization clustering technique [65] are two popular methods for this group. Additionally, two commonly used approaches for methods of the block modelling models are to define communities based on their connectivity patterns, called regular equivalence [12], or as groups of nodes that have common neighbours, called structural equivalence [38].

Spectral algorithms: In the traditional methods group, we defined a group of clustering algorithms that use the eigenvectors of adjacency matrices to find partitions, called spectral clustering methods. The idea of using spectral properties of adjacency matrices would not be limited to clustering algorithms: previous works have addressed the capability of the eigenvectors of these matrices to show useful information on latent structures. Thus, many algorithms have been developed based on such idea [42,61]. The main drawback of these algorithms is that they would work well if the network is modular.

Alternative methods: Other algorithms exist that have quite different ideas in comparison with the previous groups, although they may seem to overlap with some above-mentioned approaches. A method based on the minimal spanning tree [7], agglomerative technique [5], clique percolation method [48], label propagation method [49], and bridge-bounding method [47] are some of the well-known alternatives.

3 Proposed mathematical model

As stated before, Li et al. [34] put forward the D measure. Based on this measure, they also proposed a simple MINLP where the number of communities must be known in order for it to start working. Their MINLP is the basis of our proposed MINLP. Thus, before describing new models, we illustrate the D measure and its respective MINLP.

Given a network G = (V, E) with N nodes and M edges, where V depicts the set of nodes and E implies the edges between nodes, Li et al. [34] proposed the general D measure for C communities as follows:

$$D = \Sigma_{i=1}^{C} d(G_i) = \Sigma_{i=1}^{C} (1-\lambda) L(V_i, V_i) - \lambda L(V_i, \neg V_i) \Big/ |V_i| . \tag{1}$$

where λ is an adjusting parameter, V_i implies the set of nodes in community i, $\neg V_i$ denotes the set of nodes exist in network, but not included in V_i. $|V_i|$ represents the cardinality of set V_i. The function L is defined as follows:

$$L(V_a, V_b) = \Sigma_{i \in Va} \Sigma_{j \in Vb} a_{ij} \tag{2}$$

where a_{ij} denotes the element respective to the ith row and jth column of adjacency matrix A. That is, both i and j are indices that imply nodes.

In order to build a mathematical model, Li et al. [34] defined a binary variable x_{il} so that it equals one if community l includes node i, and zero if otherwise. Consequently, their MINLP is as follows:

$$\max \quad z = \Sigma_{l=1}^{C} \left[\left((1-\lambda) \Sigma_{i=1}^{N} \Sigma_{j=1}^{N} a_{ij} x_{il} x_{jl} - \lambda \Sigma_{i=1}^{N} \Sigma_{j=1}^{N} a_{ij} x_{il} (1 - x_{jl}) \right) \Big/ \Sigma_{j=1}^{N} x_{il} \right] \tag{3}$$

Subject to

$$0 < \Sigma_{i=1}^{N} x_{il} < N \qquad l = 1, 2, \ldots, C. \tag{4}$$

$$\Sigma_{l=1}^{C} x_{il} = 1 \qquad i = 1, 2, \ldots, N. \tag{5}$$

$$x_{il} = \{0, 1\} \qquad i = 1, 2, \ldots, N, \; l = 1, 2, \ldots, C. \tag{6}$$

$$0 < \lambda < 1 \tag{7}$$

Here, before introducing our models, we simplify and rewrite objective function as follows:

$$z = \Sigma_{l=1}^{C} \left[\left(\Sigma_{i=1}^{N} \Sigma_{j=1}^{N} a_{ij} x_{il} x_{jl} - \lambda \Sigma_{i=1}^{N} \Sigma_{j=1}^{N} a_{ij} x_{il} \right) \Big/ \Sigma_{j=1}^{N} x_{il} \right]. \tag{8}$$

After simplification, the adjusting parameter λ appears as a coefficient of the negative terms. The negative terms of Eq. 8 imply the maximum number of edges that the model can score for each community. In the case of detecting large communities, we should assign a large value to λ to create a big penalty for all missed edges. Furthermore, detecting small natural communities will be possible if that penalty approaches

zero. However, for simplicity, we consider λ to equal 0.5, although the model can resolve the problem by finding the optimal λ.

In order to utilize this model, the correct number of communities, called C*, should be known in advance, but we do not often have any knowledge about this. To cope with this problem, a new solution has been proposed in this paper that specifies the maximum possible number of communities, called C_{max}, as the upper bound of C. Since each community should have at least three members, C_{max} equals the absolute value of N/3. Then, the model needs changeability in terms of the number of available communities, i.e., from 2 to N/3. This requirement dictates that some communities will not have any members if $C^* < C_{max}$. This means that some denominators of objective function (Eq. 3) must take zero, so we have a 'divided by zero' statement, which is undefined and computationally impossible. To escape from this situation, we define another set of binary variables, noted by b_l, one for each community l, and add them to their respective fractions. These variables will take one if the size of the respective community is zero, and equal to zero if otherwise. By this way, a new proposed MINLP can find the correct number of communities and detect them in only one execution while the Li et al. [34] model needs N/3-1 executions to do the same. Our model has an objective function, as follows with Eqs. 4–6:

$$\max \quad z = \Sigma_{l=1}^{C}\left[\left(\Sigma_{i=1}^{N}\Sigma_{j=1}^{N}a_{ij}x_{il}x_{jl} - \Sigma_{i=1}^{N}\Sigma_{j=1}^{N}a_{ij}x_{il}\right)\Big/\left(\Sigma_{j=1}^{N}x_{il} + b_l\right)\right] \quad (9)$$

The optimal number of communities are equal to the absolute value of N/3 minus the sum of b_l over l.

4 Proposed hybrid artificial immune network

Metaheuristic algorithms often provide a suitable trade-off between the solutions' quality and the cost of optimization. They utilize a combination of intensification and diversification mechanisms to search the solution space. As they are different in terms of using intensification and diversification strategies, hybridization of two or more metaheuristic may be helpful to lead better solutions [55].

Artificial immune systems (AIS), introduced by Castro and Timmis [8], are a powerful group of bio-inspired metaheuristics. This group of metaheuristic is adaptive, parallel, and highly robust. In order to solve a NP-Hard problem, AIS posses several biological operations, such as clonal selection, negative selection, affinity maturation, danger theory, and immune network theory. The two latter operations are of a well-known variation of AIS, called artificial immune network (AIN). Each of these operations can provide at least one of the mechanisms mentioned (i.e., intensification and diversification).

AIN can utilize the components mentioned for optimization problems as well as for data mining tasks. This method works on a population of solutions, called antibodies, against a number of goal solutions, called antigens. Antibodies try to make their paratopes, i.e., current solutions, as similar as possible to antigens' epitopes, i.e., goal solutions. The algorithm calculates this similarity, called affinity, for improvement and clonal expression. Successful antibodies will proliferate as a matter of degree of

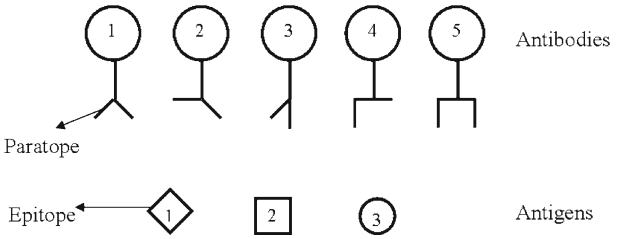

Fig. 1 Antibodies try to match their paratopes with epitopes of antigens for cloning

matching: Fig. 1 illustrates this issue. According to this figure, antibodies 1 and 5 completely match their paratopes with epitopes of antigens 1 and 2, respectively.

The next section describes the proposed hybrid metaheuristic algorithm (HAIN), and its application to solve the proposed MINLP model.

4.1 The HAIN procedure

A crucial characteristic of complex networks is sparsity. Sparsity implies that each node often has links only to a few nodes, called its neighbours. Although this characteristic makes mathematical modelling of a network more problematic, it provides a great opportunity for search methods. That is, for each node, these methods should explore only a small subset of a network, i.e., a node's neighbours. Therefore, the success of search methods depends on enhancing intensification and diversification in the search space using sparsity characteristics.

The hybrid artificial immune network (HAIN) strategy is to move across the neighbourhood of each node to maximize the total value of the D measure. This algorithm takes its strategy for generating initial population, intensification, and diversification. Similar to parallel methods, HAIN initiates with multiple start points. In the case of generating an initial population of antibodies (AB) and antigens (AG), HAIN avoids putting two non-adjacent nodes in one community. The next subsection describes this approach. Then, the algorithm calculates the affinity of each AB. Another component of HAIN is clonal selection and expansion. According to the basic idea of this component, the larger value of the affinity, the greater a population of new antibodies. In order to maturate new AB, the next component, called affinity maturation, starts to work after cloning. This component maturates all AB by applying one of two different moves with equal probability. The moves, described in the affinity maturation subsection, try to find the best neighbour of a random selected node for adding to (or removing from) its respective community to increase the total D measure. Indeed, they use sparsity characteristics and maximize the D measure simultaneously. This component particularly emphasizes the intensification mechanism. As a weak AB may exist, any algorithm needs cleaning and suppressing components. HAIN uses metadynamics and network suppression for those objectives. The diversity component takes a diversification strategy by splitting the largest community into two smaller ones, and merging two connected communities at random. The last subsection describes these procedures. Finally, the pseudocode of the proposed HAIN algorithm has been

Algorithm 1- The Hybrid Artificial Immune Network algorithm

Input: The adjacency matrix A
Output: The optimal communities
Begin
Generate random population using Algorithm 2.
Repeat
  For each AB and AG
    Do Affinity calculation (Section 4.3)
  End
  For each antigenic pattern AG
    Do Clonal selection and expansion (Section 4.4)
    For each new AB
      Do Affinity maturation using Algorithm 3.
    End
    Do Meta-dynamics (Section 4.6)
  End
  Suppress duplicates (Section 4.6)
  Do Diversity using Algorithm 4.
Until Converges
End

Fig. 2 The HAIN algorithm

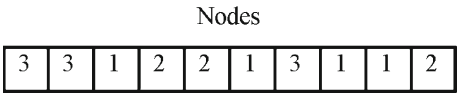

Fig. 3 Representation of AB (or AG)

illustrated in Fig. 2. A computer program in the MATLAB language can be obtained on request from the authors.

4.2 Representation and initial population

In order to apply a metaheuristic algorithm, the first question is how to represent a solution. In the case of the HAIN algorithm, this question states the representation of antibodies (AB) and antigens (AG). Based on our knowledge, there are three ways for representing a community detection solution in the literature: locus-based adjacency or graph-based encoding [2,24,54], binary matrix encoding [35], and string-of-group encoding [37]. The latter proposes to assign a specific cell of solution vector to each node, and put its respective community label into that cell. Note that the length of each AB (or AG) vector equals N (i.e., the number of nodes). Figure 3 shows this simple encoding for a case of ten nodes and three communities. This proposed algorithm uses this representation to illustrate both AB and AG.

Since the optimal communities are not available to use as goal solutions, the algorithm will choose the best AB related to each AG, called AB*, as an updated AG if AB* is greater than the current AG in each iteration. The number of AG can be more than one, i.e., the algorithm starts and proceeds with multiple initial points in parallel.

Initial populations, crucially, need two factors: randomness and completeness. Randomness provides an opportunity to search all of the search space. Completeness

denotes that all solution vectors should be able to include all admissible numbers, i.e., between 1 and the absolute value of N/3 in the case of our representation. The latter factor gives the opportunity for the algorithm to define different communities. Indeed, the algorithm can illustrate not only the optimal communities but also the correct number of communities. By this way, in the first step, the algorithm consecutively chooses nodes and assigns a specific integer, called the community number, to each selected node and subsequently to its neighbours in order to avoid putting two non-adjacent nodes in one community. This process repeats for each N_{AB} antibody of each N_{AG} antigen. Next, the algorithm fires a reiteration of the label propagation method (for more information about this method see [36]). According to this heuristic method, each node chooses the most frequent community number of its neighbours using the following formula:

$$x(i) = \text{mode}\ (\eta(i)) \tag{10}$$

where x(i) denotes the integer assigned to node i and η(i) shows the set of neighbours of node i. The mode function implies that the most frequent number of the set appears in its argument. Other components of HAIN use the output of this heuristic method. Thus, based on [55], HAIN is a high-level relay heuristic (HRH) because another heuristic method (i.e., label propagation) provides a set of initial solutions for HAIN instead of randomly generated antibodies.

The abovementioned processes supply both randomness and completeness for each solution. Figure 4 shows this algorithm.

4.3 Affinity calculation

Each AB (or AG) shows a complete solution and illustrates the members of each community by an identical number. Note that these numbers are neither values nor orders: they work as labels. In other words, each AB (or AG) shows variables x_{il} of MINLP, which take one. As the abovementioned representation satisfies all mathematical model constraints, in order to evaluate the solutions based on the D measure, the affinity of each AB (or AG) is equivalent to the respective objective function of the MINLP addressed by Li et al. [34]. Thus, we employ Eq. 9 (the objective function of the proposed MINLP) for affinity calculation. Note that we consider λ to equal 0.5.

4.4 Clonal selection and expansion

In the main variation of AIN, called aiNET, the clonal selection and expansion (CSE) component includes cloning each antibody with respect to each antigen based on its affinity. The cloning process follows a famous rule: the larger the affinity of an antibody, the greater a new generated clone. By this definition, each antigen is of a different type of output pattern that is expected to be recognized by a respective set of antibodies. However, since the modularity optimization is an optimization problem, no pattern exists for taking into account an antigen. In these cases, an antigen can be defined as the best antibody in each set of antibodies in terms of its affinity (i.e.,

Algorithm 2- Generating initial Antibodies and Antigens

Input: N_{AG}, N_{AB}, and C_{max}
Output: The N_{AG} initial Antigens and their respective Antibodies
Begin
Generate N_{AG} empty antigens
For each antigen
    Generate N_{AB} new empty antibodies
    For each antibody
        $k \leftarrow 1$
        Choose C_{max} nodes at random
        For each node
            Put k in its respective cell
            Put k in its neighbors' cells
            $k \leftarrow k + 1$
        End
        For each node
            Change its respective cell using Formula (10)
        End
    End
    Add the new generated antibodies to the respective pool
    Calculate the affinity of each of them
    Choose the best of them as non-empty antigen
End
End

Fig. 4 The proposed algorithm for generating initial antibodies

objective function). Thus, in each iteration, our algorithm works on N_{AG} different sets of antibodies in parallel. In other words, this mechanism works as a parallel population-based metaheuristic.

In order to employ a CSE component for each antigen, the algorithm ranks respective antibodies and calls them respectively. Then, the algorithm generates c_i identical antibodies similar to the one selected using the following formula:

$$c_i = \text{Round}\ (\beta \times N_{AB}/r_i) \tag{11}$$

where β is a constant coefficient and r_i is the rank of the respective antibody i. Formula 11 implies that better antibodies in terms of affinity have larger colonies in the HAIN. This component can provide a proper intensification if it uses a local search for each newborn antibody.

4.5 Affinity maturation

Affinity maturation works as a powerful local search to exploit the search space. Maturation of new clones often occurs by random mutation. Here, our algorithm utilizes two new greedy procedures for maturating newborn clones called firing and hiring. The algorithm randomly selects one of two procedures for each antibody with equal chance. Then, the selected procedure works on the respective antibody to maturate it.

The first procedure, i.e., firing, chooses one of the present communities in the respective antibody at random. Then, it determines which of its members has the minimum effect on improving the D measure of their community using the following formula:

$$F_i = D_i - \neg D_i \tag{12}$$

where D_i and $\neg D_i$ denote the number of internal and external edges of the community of node i, including node i, respectively. The algorithm chooses the node that has the largest value of F_i. Since the measure (F_i) calculates the changes in the nominator of the D measure of the respective community, this could contribute to the D measure. Note that the changes in the dominator of the D measure could be estimated as negligible because this refers to plus or minus one. Then, the firing procedure fires that member. The fired node must be placed in one of other communities. Thus, the procedure seeks the best new community for the fired node based on the maximum number of edges between this node and members of other present communities. Finally, the fired node joins to its new community.

The second procedure, called hiring, starts by choosing a community at random. Then, it seeks a node out of the selected community, which can increase the total D measure more than other nodes. The idea behind this procedure is to increase the total D measure by means of a greedy search. Finally, this node joins the selected community. The algorithm determines which node is more suitable to be hired using the following formula:

$$H_i = (I_{New} - O_{New})/N_{New} - (I_{Old} - O_{Old})/N_{Old} \tag{13}$$

where I (O) denotes the number of edges between node i and its neighbours in (out of) its respective community. The 'Old' and 'New' indices imply the community of node i, before and after the hiring process, respectively. N is the number of nodes included in the community of node i with respect to its index. Indeed, these fractions calculate the change in the total D value resulting from the movement of node i from one community to another. The algorithm chooses the node i with the largest value of H_i.

Although these two moves consider both sparsity characteristics and maximizing the D measure, hiring or firing only one node may provide too slow an intensification. Thus, each of these procedures works on a set of antibodies in succession. This means that after maturating the first newborn antibody, the procedures continue to work on this antibody to generate the second matured one. Then, the procedures work on the second matured antibody to generate the third matured one, and so on. Consequently, matured antibodies, except the first one, might be maturated using more than one move. This strategy provides more rapid intensification. On the other hand, since these procedures are greedy, after affinity calculation, we can determine which of them is better. Figure 5 depicts these procedures in one algorithm, called Algorithm 3.

4.6 Metadynamics and network suppression

After using the affinity maturating component, some maturated antibodies may address identical or useless solutions. In order to reduce calculating time, the algorithm

Algorithm 3- Maturating newborn Antibodies

Input: c_i newborn antibodies (NB)
Output: A set of maturated antibodies
Begin:
NABS ← NB(1)
For k from one to c_i
    Generate a random number using Uniform distribution over [0,1], called r
    If r ≥ 0.5 % Employ firing procedure in fifty percent of times
        Choose a community number from NABS at random, called CR
        Find a node using Formula (12), called CRf
        Find a proper community for CRf, called CN
        Update NABS by moving CRf from CR to CN
        NB(k) ← NABS
    Else % Employ hiring procedure in other times
        Choose a community number from NABS at random, called CR
        Make a list of nodes not included in CR
        Find the best node for hiring in CR using Formula (13), called CRf
        Update NABS by moving CRf to CR
        NB(k) ← NABS
    End
End
End

Fig. 5 The proposed algorithm for maturating newborn antibodies

employs two cleaning components: metadynamics, respective to each antigen, and network suppression, respective to all antibodies.

The metadynamics component eliminates all antibodies whose affinities, with their respective antigen, are lower than a given threshold in the main variation of aiNET. As the proposed algorithm considers the best antibodies as antigens, it should eliminate only newly matured antibodies that have a lower affinity in comparison with other ones.

Bringing all newly matured antibodies together makes a network itself. This network would provide powerful exploration of the solution space if empowered by a proper diversification procedure. In the case of our algorithm, this kind of diversification could be provided by removing antibodies that have twins in the network. In other words, the algorithm saves unique antibodies for the next iteration. Removing duplicate antibodies give a chance for the remaining ones to enjoy a better local search—using a CSE component—in the next iteration. This is because these antibodies could reach a better rank if those duplicate antibodies with higher affinities are removed in the previous iteration. This task also reduces the number of solutions and the cost of calculation.

4.7 Diversity

The algorithm may suffer from early convergence if there is no good diversification mechanism. However, aiNET includes a great diversification procedure called the diversity component. The diversity component states the generation of new antibodies at random. Despite the main aiNET, which generates new antibodies randomly, our

Algorithm 4- Diversity

Input: A set of antibodies (AB)
Output: A set of new antibodies
Begin:
For each antibody
    Call it OB
    Generate a random number using Uniform distribution over [0,1], called r
    If r ≥ 0.5 % Employ splitting procedure in fifty percent of times
        Choose the largest community present in OB, called LC
        Select a half of LC nodes randomly, called NC
        Assign a new label to all nodes of NC
    Else % Employ merging procedure in other times
        Choose a node of OB at random, called RN
        Select the neighbors of RN, called NN
        Assign the most frequent label of NN to all nodes of NN
    End
End
End

Fig. 6 The proposed algorithm for the diversity component

algorithm takes two smart procedures for diversification: splitting the largest community into two smaller ones with equal size and merging all neighbours of a node to make a bigger community. In the case of a splitting procedure, the larger the size of community, the higher a probability of selecting this community for dividing into two partitions. It is fair because larger communities need more changes than smaller ones. Figure 6 illustrates the procedures of the diversity component:

5 Experimental results

This section provides experiments to compare the effectiveness of HAIN with some other search methods of real world and artificial networks. All of the datasets are undirected, unweighted, and unattributed. To evaluate the results obtained by different methods, we used the normalized mutual information (NMI) measure. Given two sets A and B, this measure is as follows:

$$I(A,\ B) = \left(-2\ \Sigma_{i=1}^{CA}\Sigma_{j=1}^{CB}C_{ij}\log\ \left((C_{ij}N)\Big/\left(C_{i.}C_{.j}\right)\right)\right)\Big/\left(\Sigma_{i=1}^{CA}C_{i.}\log\left(C_{i.}/N\right)\right.$$
$$\left.+\Sigma_{j=1}^{CB}C_{.j}\log\left(C_{.j}/N\right)\right) \tag{14}$$

where N is the number of nodes, CA and CB are the number of communities in sets A and B, respectively, and C_{ij} denotes the number of nodes of community i of set A, which appear in community j of set B. $C_{i.}$ ($C_{.j}$) is the number of nodes of community i of set A (community j of set B). I (A, B) will be at its maximum, i.e., 1, if A = B.

NMI will be helpful if we know exactly which nodes belong to which community. Coping with this issue is different for facing artificial or real world networks. In the case of artificial networks, we often employ a computer program to build a dataset. Most of these programs determine which node belongs to which community before making

the respective adjacency matrix. To make it clearer, consider a program that makes 2N nodes while the first N nodes are called community 1, and the second N nodes called community 2. Then, the program makes edges between nodes with probability p such that p for nodes in the same community is bigger than p for nodes of different communities. However, in the case of real world networks, more information must be in hand to know the community structures, while this desired information cannot be reached in most real world cases. In such cases, in order to make a fair comparisons between methods, we attempt to discover natural patterns in a network and compare their obtained D measure. Therefore, in this paper, we use calculated the D measure on real world datasets, and employed NMI only on artificial networks.

Based on our knowledge, only two proposed search methods in the literature, called Genetic Algorithm to Optimize D value (GAOD) [37] and Improved Memetic algorithm (iMeme-Net) [21], which employ modularity density (D measure) as their optimization function. The GAOD [37] algorithm hybridizes a simple genetic algorithm as its main algorithm, with a heuristic method that is employed to avoid generating invalid initial chromosomes. Another algorithm, called iMeme-Net, employs a heuristic method for generating a new initial population, a variation of the simulated annealing algorithm as its local search and a new memetic algorithm as its main body. We select these algorithms and our MINLP for comparison of real network datasets. In the case of artificial networks, since the actual communities present by a computer program, there is no need to include our MINLP; instead, we add one of the best methods in the literature, called Multi-Objective Community Detection algorithm with Max-Q model selection (MOCDQ) [54] to compare the obtained results. The MOCDQ method is a multiobjective genetic algorithm employed variations of Q function as its objective functions. Here, we solve the MINLP model by GAMS/BARON in http://www.neos-server.org/ and run algorithms in MATLAB 7.9 on a computer with Intel Core 2 Duo, 2.50 GHz, and 4.0 GB RAM.

5.1 Real world network

5.1.1 Datasets

The Zachary karate club network: Zachary [64] generated this network after he studied the friendship network of 34 members of a karate club over a period of two years. Although he reported that the network divided into two groups during this period, some researchers have discovered more than two communities in this network [1,24,34,35,37].

The Bottlenose dolphins network: Lusseau et al. [39] collected this social network. This network is of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand.

Political books: V. Krebs compiled this network (available: http://www.orgnet.com/), which includes books about US politics published around the time of the 2004 presidential election. The online bookseller Amazon sold these 105 books and show frequent co-purchasing of books by means of the edges between purchased books.

American college football: This dataset [18] represents a network of American football games between colleges during the 2000 season. Nodes and edges demonstrate teams and regular season games, respectively.

Word adjacencies: This dataset [45] includes an adjacency network. This network is of common adjectives and nouns in the novel David Copperfield by Charles Dickens.

Les Miserables: This dataset [30] contains a network of characters in the novel Les Miserables, which appear together.

Neural network: White and Southgate [58] compiled this dataset, which is a network representing the neural network of C. Elegans.

Co-authorship network: This dataset contains a network of scientists who have researched a network theory area of science [45]. This dataset has 128 isolated nodes out of 1589 nodes.

Power grid: The structure of the Western States power grid of the US is illustrated in this network. White and Southgate [58] provided its respective dataset.

Table 1 illustrates more details about the abovementioned datasets.

Table 2 illustrates the parameter values of HAIN method after trial and error tuning. The tuning task involves running the algorithm with different combinations of three parameters' values on various datasets several times and finding which combination obtains the best results. This table also shows the parameter values of GAOD, iMeme-Net, and MOCDQ, extracted from their respective papers.

Table 1 Real world networks in details

Dataset	N	M	$d_{average}$
Karate club	34	78	4.5882
Dolphins	62	159	5.1290
Political books	105	441	8.4000
American football	115	613	10.6609
Word adjacencies	112	425	7.5893
Les miserables	77	254	6.5974
Neural network	297	2, 148	14.4646
Co-authorship	1, 461	2, 742	3.7536
Power grid	4, 941	6, 594	2.6690

Table 2 Parameter values of the methods

HAIN		GAOD		iMeme-Net		MOCDQ	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
N_{AG}	10	N_{pop}	20	N_{pop}	100	N_{pop}	50
N_{AB}	10	k_{max}	2	I_{LP}	5	$S_{Survive}$	40
N_{div}	5	m_{max}	10	β	0.16	I_{max}	300
		I_{max}	350				

5.1.2 Results

Table 3 shows the best results obtained by the proposed MINLP, HAIN, GAOD, and iMeme-Net methods on real world networks. Columns C denote the number of communities respective to the best-obtained result, illustrated in the D columns. The bold entries correspond to the best obtained result(s) respective to each dataset. Figures 7 and 8 also depict the discovered communities using the HAIN algorithm in the karate club network and co-authorship network, respectively.

In the case of the karate club dataset, all methods report the best number of communities equalling three respective to $D_{measure} = 3.9225$. HAIN and GAOD could establish better results than other methods in the case of the dolphin dataset. One might expect that the MINLP model would find the global optimum solution. Here, we note that the MINLP model is non-convex as it contains only integer variables [26]. Therefore, this kind of model may be trapped in a local optimum. Another drawback of this kind of model is that it fails in cases of medium and large networks. This issue is depicted in Table 3 by means of an F character.

Table 3 Number of communities and modularity density results obtained by the methods

Dataset	MINLP		HAIN		GAOD		iMeme-Net	
	C	D	C	D	C	D	C	D
Karate club	**3**	**3.9225**	**3**	**3.9225**	**3**	**3.9225**	**3**	**3.9225**
Dolphins	4	5.7495	**5**	**6.0626**	**5**	**6.0626**	5	5.8107
Political books	6	10.8079	**6**	**10.9576**	**6**	10.4778	5	10.5991
American football	10	22.1702	**11**	**22.1940**	12	21.4232	11	22.1212
Word adjacencies	**2**	**3.9125**	**2**	3.8937	4	3.0739	1	3.7946
Les miserables	8	12.1814	**8**	**12.2737**	9	12.0256	**8**	11.3784
Neural network	**3**	10.1043	**3**	10.1143	9	−0.9843	**3**	**10.1537**
Co-authorship	F	F	**350**	**368.9737**	393	332.4802	223	252.9365
Power grid	F	F	401	212.5005	1, 842	−294.25	**321**	**213.3118**

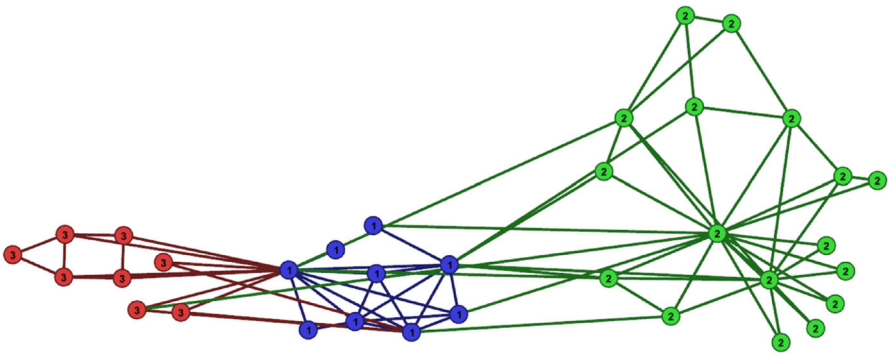

Fig. 7 Discovered communities in karate club network using HAIN algorithm

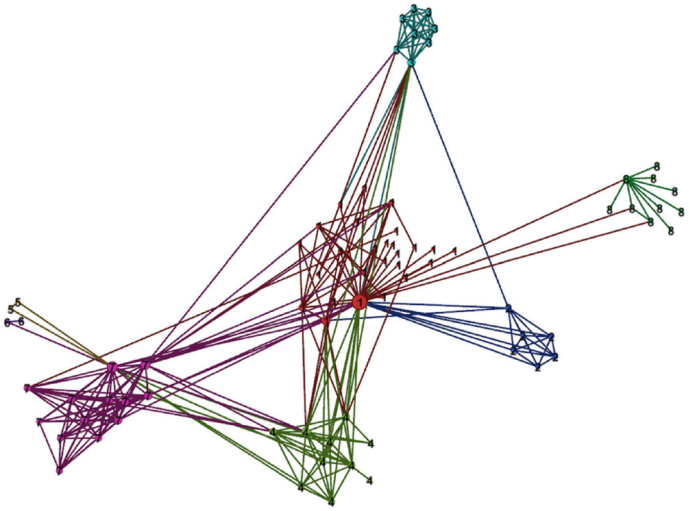

Fig. 8 Discovered communities in co-authorship network using HAIN algorithm

Generally, HAIN achieves the best results in six out of nine experiments. In the case of two out of the three remaining experiments (i.e., word adjacencies and neural network datasets), this method finds the best-reported number of communities correctly, and differs slightly from the best results. In the case of the neural network and power grid dataset, the GAOD method reports an excellent result, although this method reaches near-best results on other datasets. Consequently, HAIN demonstrates appropriate performance on real world datasets.

Other question arising here points to which of these algorithms can provide a better trade-off between the quality of solutions and CPU time. In order to reply such question we need criteria to fairly measure the performance of algorithms. Fortunately, for evaluating algorithms developed for maximization models, Osman and Al-Ayoubi [46] have defined such criteria, called Marginal Improvement per unit of Computational effort (MIC), as follows:

$$\mathrm{MIC} = \left((\mathrm{D}(\mathrm{A}) - \mathrm{LB}) \Big/ (\mathrm{OP} - \mathrm{LB}) \right) \Big/ CPU \tag{15}$$

where D(A) implies the value of objective function obtained using the metaheuristic, CPU is the consumed time in seconds for running the metaheuristic, and LB and OP are the lower bound and global optimum value of objective function, respectively. These latter two values could be determined by the user if they are unavailable. The higher the value of MIC, the greater quality an algorithm has.

For comparison purposes, from each level of network size (i.e., <50 nodes, ~ 100 nodes, and $>1{,}000$ nodes) a representative dataset is selected. Table 4 illustrates the best results obtained by the methods using MIC criteria over three selected datasets:

According to Table 4, the HAIN algorithm provides better results in two out of three experiments. However, there are other crucial issues arising for algorithm evaluation that are not addressed by MIC criteria, such as robustness and method of programming

Table 4 The MIC results obtained by methods

Dataset	HAIN			GAOD			iMeme-Net		
	D(A)	CPU	MIC	D(A)	CPU	MIC	D(A)	CPU	MIC
Karate club	3.9225	0.3328	**3.0048**	3.9225	1.8603	0.5375	3.9225	0.6691	1.4945
Political books	10.9576	7.5623	0.1307	10.4778	5.9317	0.0841	10.5991	2.5244	**0.2478**
Co-authorship	368.9737	493.0172	**0.0020**	332.4802	663.2013	0.0010	252.9365	78.6916	0.0003

algorithms. Moreover, claims about the superiority of the HAIN algorithm need further research. To do so, in the next section, we describe and employ a set of computer-generated datasets.

5.2 Artificial network

5.2.1 Datasets

Lancichinetti et al. [32] proposed Lancichinetti-Fortunato-Radicchi (LFR) benchmark datasets including well-known artificial graphs. One can make them by varying the defined parameters via simulation. In order to evaluate our algorithm and analyse its behaviour, we employed a set of LFR datasets. These datasets are different in terms of the number of edges (M), the maximum and minimum size of communities (S_{Max} and S_{Min}), permitted interval of nodes' degrees (k_{min}, k_{max}), and the mixing parameter (μ) where two parameters β and γ are constant and equal 1 and 2, respectively. By means of these datasets, we designed three different experiment sets for comparing our HAIN with GAOD, iMeme-Net, and MOCDQ. The next sections show the experiments' results.

5.2.2 Results of varying the mixing parameter

The mixing parameter μ denotes the probability of making a connection between a node and nodes of other communities. The higher the value of μ, the more mixed a network is. Thus, we could study the behaviour of the methods by increasing the value of μ, from 0 to 0.6, through the following experiments. In all of these experiments, N = 350, C* = 6, k = [25, 50], and S = [50, 75]. Figures 9 and 10 show the results obtained using the methods.

According to these figures, in most of the experiments, HAIN reaches the correct number of the communities even when NMI <1. Other methods (i.e., GAOD, MOCDQ, and iMeme-Net), although competing with each other in terms of NMI, have a smaller NMI than HAIN. GAOD and MOCDQ report larger numbers as the number of communities; however, iMeme-Net has smaller numbers in most of the experiments. Consequently, HAIN shows higher reliability in comparison with other methods.

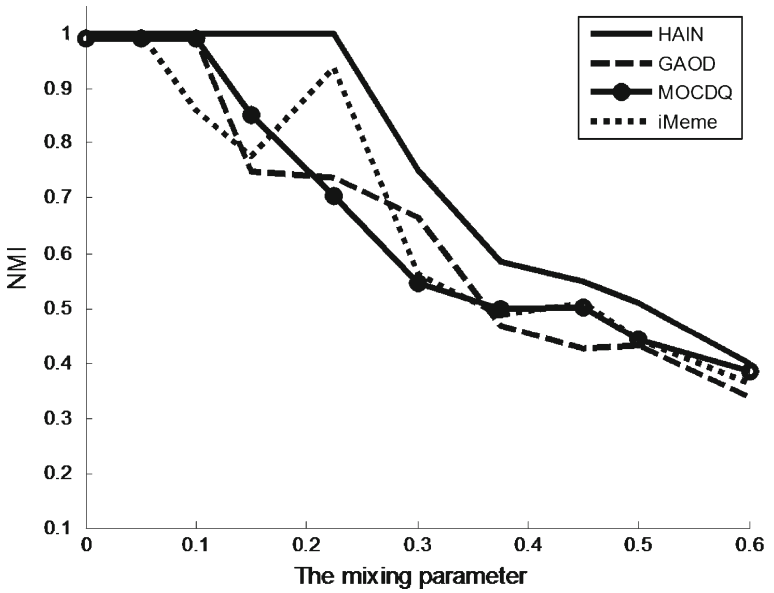

Fig. 9 NMI results obtained by the methods when the mixing parameter varies

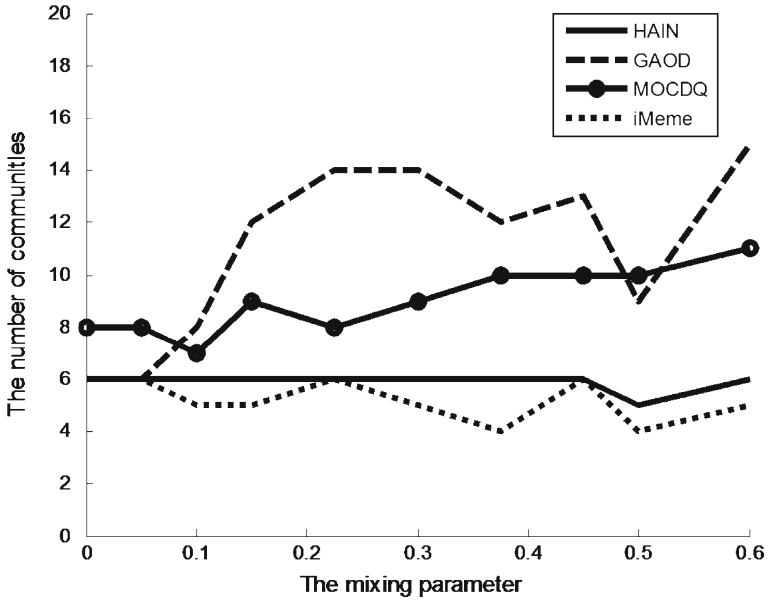

Fig. 10 Changing the obtained number of communities when the mixing parameter varies

5.2.3 Results of varying the allowed community sizes

A robust method shows effective performance when something changes in input. In order to analyse the robustness of the methods in detail, we provide six LFR datasets,

Table 5 The parameters of artificial LFR networks

Dataset	M	$d_{average}$	C*	k_{min}, k_{max}	S_{min}, S_{max}	μ
LFRSA	5,981	33.8171	7	25–50	49–51	0.1
LFRSB	5,829	33.3086	6	25–50	50–75	0.1
LFRSC	5,838	33.3600	5	25–50	40–90	0.1
LFRSD	5,769	32.9657	8	25–50	20–95	0.1
LFRSE	5,734	32.7657	4	25–50	20–120	0.1
LFRSF	5,657	32.3257	9	25–50	20–145	0.1

Table 6 The results obtained by methods in terms of the number of communities and NMI

Dataset	$C*$	HAIN		GAOD		MOCDQ		iMeme-Net	
		C	NMI	C	NMI	C	NMI	C	NMI
LFRSA	7	7	**1**	9	0.9682	7	0.9957	5	0.7374
LFRSB	6	**6**	**1**	8	0.9874	7	0.9912	5	0.8592
LFRSC	5	**5**	**1**	9	0.9161	6	0.9890	**5**	**1**
LFRSD	8	**8**	**1**	10	0.9907	9	0.9919	6	0.7960
LFRSE	4	**4**	**1**	10	0.7269	10	0.7283	7	0.7245
LFRSF	9	**9**	**1**	**9**	**1**	10	0.9920	7	0.7906

which are different in terms of the minimum and maximum allowed size of communities present. These datasets include 350 nodes. Tables 5 and 6 represent the details about the datasets and of the experimental results, respectively.

In all of the experiments, HAIN achieves the best results. GAOD and MOCDQ, although achieving near-optimal values, represent incorrect numbers as C* in five out of six experiments. GAOD could reach the correct number of communities in the dataset with the widest interval of community size. Despite the performance of GAOD, MOCDQ achieves the correct number of communities when communities are approximately identical in terms of size. Like the previous set of experiments (i.e., the results depicted in Fig. 10), iMeme-Net reaches values smaller than C* in most of the experiments; however, like GAOD and MOCDQ, this method reports far from the reality in terms of C*. Indeed, this method has problems when the sizes of communities are significantly different. The main reason for this is that iMeme-Net starts with a set of good solutions made by five iterations of the label propagation method, which converges rapidly toward a local optimum. Then, in the absence of great diversification, moving to better points is impossible. In general, the results over varying the permitted community sizes again confirm the effectiveness of our HAIN method.

5.2.4 Results of varying the allowed interval of degrees

In this section, we want to analyse the performance of the methods by varying the permitted interval of nodes' degrees. In this way, first, we make nine datasets by

changing k_{max}, from 10 to 50, and by keeping other parameters constant as follows: $k_{min} = 5$, S = [40, 90], $\mu = 0.1$, and N = 350. Then, we implement all methods using these datasets. Figures 11 and 12 depict the respective results. Note that for all datasets, C* = 7.

There exists a close competition between our HAIN method and the MOCDQ method according to Figs. 11 and 12. Despite the performance of HAIN and MOCDQ, two other methods (i.e., GAOD and iMeme-Net) provide worse results: they reached a lower NMI than HAIN and MOCDQ on most of the experiments. In the case of the number of communities, GAOD demonstrates the worst results. Among the methods, HAIN represents the best results.

6 Conclusion

This paper focused on modularity density maximization to detect communities in networks. The modularity density targets the average in-degree and the average out-degree of each potential community; therefore, this measure can avoid limited resolution. In this way, two models have been developed in this paper: the MINLP model and the HAIN algorithm. The MINLP model can discover not only latent communities but also the correct number of communities. The HAIN algorithm is able to detect optimal communities without any prior knowledge about the number of communities. This method combines the characteristics of complex networks, i.e., sparsity, with the excellent components of pure immune networks, i.e., cloning, affinity maturation, network suppression, and diversity. The HAIN algorithm uses a well-known local search method, called label propagation, to generate initial solutions.

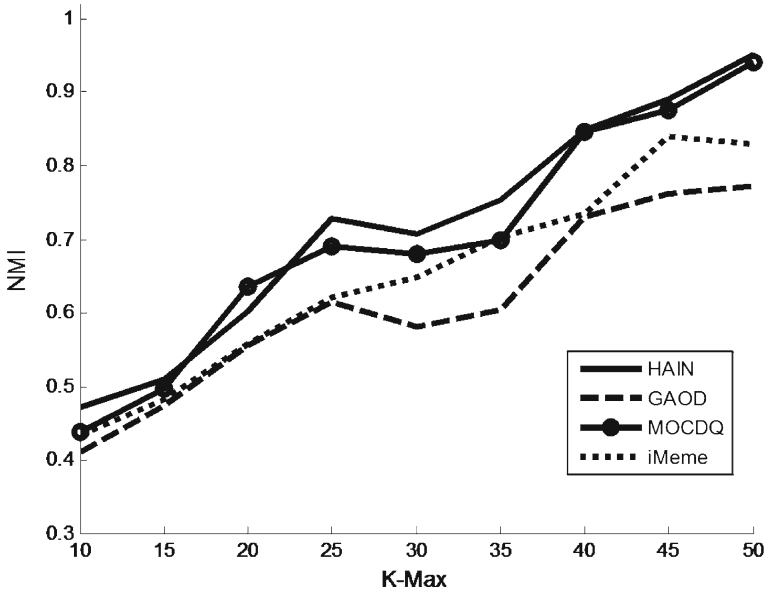

Fig. 11 NMI results reached by the methods when k_{max} varies

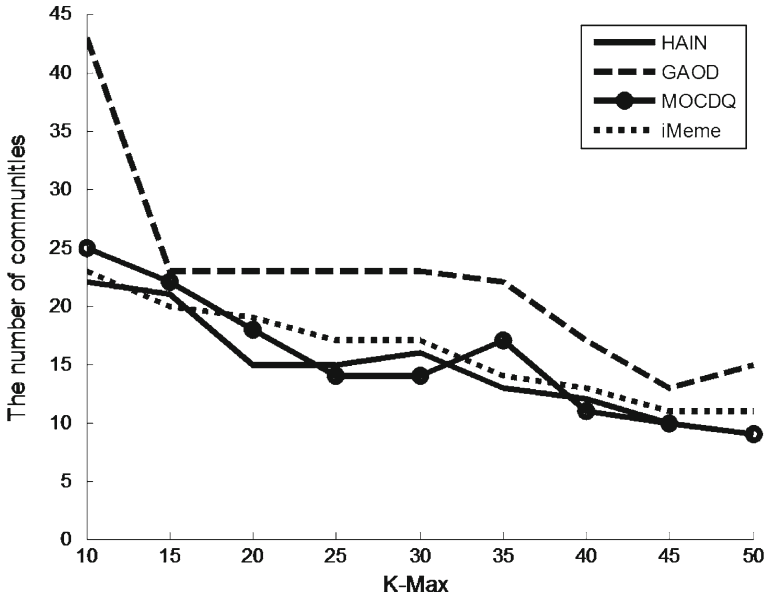

Fig. 12 The number of communities reached when k_{max} varies

The experimental results on real world complex networks suggest that the HAIN algorithm reaches the best results in comparison to the GAOD, MOCDQ, and iMeme-Net methods. We also built three sets of experiments to evaluate the performance of the HAIN algorithm on the LFR benchmark datasets. The first experiment set on LFR datasets showed that the proposed methods were quite efficient at discovering the community structure of complex networks with varying mixing parameters. By changing the permitted interval of community sizes, we designed another experiment. The results again confirmed the effectiveness of our HAIN method, although, in the case of the third experiment set, the differences between the results could be negligible. Consequently, the experiments on artificial datasets express that the HAIN algorithm can reach the best results in terms of NMI and the number of communities.

The proposed methods can achieve the same gains on directed and weighted networks by means of simple modifications. However, in the case of an attributed network, there exists a strong requirement for future research. Further future work could involve employing and improving state-of-the-art algorithms such as [6] in order to solve modularity-based optimization models.

References

1. Agrawal G, Kempe D (2008) Modularity-maximizing graph communities via mathematical programming. Eur Phys J B 66(3):409–418. doi:10.1140/epjb/e2008-00425-1
2. Amiri B, Hossain L, Crawford JW (2011) An efficient multiobjective evolutionary algorithm for community detection in social networks. In: Evolutionary computation (CEC). IEEE Congress, New Orleans, pp 2193–2199. doi:10.1109/CEC.2011.5949886

3. Amiri B, Hossain L, Crawford JW, Wigand RT (2013) Community detection in complex networks: multi-objective enhanced firefly algorithm. Knowl Syst 46:1–11. doi:10.1016/j.knosys.2013.01.004
4. Arenas A, Díaz-Guilera A, Pérez-Vicente CJ (2006) Synchronization reveals topological scales in complex networks. Phys Rev Lett 96(11):114102. doi:10.1103/PhysRevLett.96.114102
5. Bagrow JP, Bollt EM (2005) Local method for detecting communities. Phys Rev 72(4):046108. doi:10.1103/PhysRevE.72.046108
6. Bhagyesh VP, Nataraj PSV, Bhartiya S (2012) Global optimization of mixed-integer nonlinear (polynomial) programming problems: the Bernstein polynomial approach. Computing 94:325–343. doi:10.1007/s00607-011-0175-7
7. Bonanno G, Caldarelli G, Lillo F, Mantegna RN (2003) Topology of correlation-based minimal spanning trees in real and model markets. Phys Rev 68(4):046130. doi:10.1103/PhysRevE.68.046130
8. Castro LN, Timmis J (2002) Artificial immune systems: a new computational intelligence approach. Springer, Berlin
9. Chang MS, Hung LJ, Lin CR, Su PC (2013) Finding large k-clubs in undirected graphs. Computing 95:739–758. doi:10.1007/s00607-012-0263-3
10. Cheng Q, Liu Z, Huang J, Zhu C (2012) Hierarchical clustering based on hyper-edge similarity for community detection. In: Web intelligence and intelligent agent technology, IEEE, Macau. doi:10.1109/WI-IAT.2012.9
11. Donath W, Hoffman A (1973) Lower bounds for the partitioning of graphs. IBM J Res Dev 17(5):420–425. doi:10.1147/rd.175.0420
12. Everett MG, Borgatti SP (1994) Regular equivalence: general theory. J Math Soc 19(1):29–52
13. Fiedler M (1973) Algebraic connectivity of graphs. Czech Math J 23(2):298–305
14. Flake GW, Lawrence S, Giles CL, Coetzee FM (2002) Self-organization and identification of web communities. IEEE Comput 35:66–71. doi:10.1109/2.989932
15. Fortunato S, Barthélemy M (2007) Resolution limit in community detection. Proc Natl Acad Sci USA 104:36–41. doi:10.1073/pnas.0605965104
16. Fortunato S (2010) Community detection in graphs. Phys Rep 486:75–174. doi:10.1016/j.physrep.2009.11.002
17. Freeman LC (1977) A set of measures of centrality based on betweenness. Sociometry 40:35–41
18. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci USA 99(12):7821–7826. doi:10.1073/pnas.122653799
19. Goldberg AV, Tarjan RE (1988) A new approach to the maximum flow problem. J ACM 35:921–940. doi:10.1145/48014.61051
20. Gong M, Fu B, Jiao L, Du H (2011) Memetic algorithm for community detection in networks. Phys Rev E84. doi:10.1103/PhysRevE.84.056101
21. Gong M, Cai Q, Li Y, Ma J (2012) An improved memetic algorithm for community detection in complex networks. In: Evolutionary computations (CEC) IEEE congress on Brisbane. doi:10.1109/CEC.2012.6252971
22. Gong M, Zhang LJ, Ma JJ, Jiao LC (2012) Community detection in dynamic social networks based on multiobjective immune algorithm. J Comput Sci Technol 27(3):455–467. doi:10.1007/s11390-012-1235-y
23. Guimerà R, Sales-Pardo M, Amaral LAN (2004) Modularity from fluctuations in random graphs and complex networks. Phys Rev E 70(2):025101. doi:10.1103/PhysRevE.70.025101
24. Halalai R, Lemnaru C, Potolea R (2010) Distributed community detection in social networks with genetic algorithms. In: Intelligent communication and processing (ICCP), IEEE International Conference on Cluj-Napoca, pp 35–41. doi:10.1109/ICCP.2010.5606467
25. Handcock MS, Raftery AE, Tantrum JM (2007) Model based clustering for social networks. J R Stat Soc A 170(2):301–354. doi:10.1111/j.1467-985X.2007.00471.x
26. Hemmecke R, Köppe M, Lee J, Weismantel R (2010) Nonlinear integer programming. In: Jünger M et al (eds) 50 Years of integer programming 1958–2008. Springer, Berlin, pp 561–618
27. Honghao C, Zuren F, Zhigang R (2013) Community detection using ant colony optimization. In: Evolutionary computation (CEC) IEEE Congress on Cancun. doi:10.1109/CEC.2013.6557944
28. Hughes BD (1995) Random walks and random environments: random walks, vol 1. Clarendon Press, Oxford
29. Kernighan BW, Lin S (1970) An efficient heuristic procedure for partitioning graphs. Bell Syst Tech J 49:291–307. doi:10.1002/j.1538-7305.1970.tb01770.x

30. Knuth DE (1993) The Stanford graph base: a platform for combinatorial computing. Addison-Wesley, Reading
31. Kumpula JM, Saramäki J, Kaski K, Kertész J (2007) Limited resolution and multiresolution methods in complex network community detection. In: Noise and stochastics in complex systems and finance in SPIE Conference Series, vol 6601
32. Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. Phys l Rev E 78:046110. doi:10.1103/PhysRevE.78.046110
33. Li X, Li D, Wang S, Tao Z (2007) Effective algorithm for detecting community structure in complex networks based on GA and clustering. Proc. Comput. Sci. ICCS, Beijing, China, pp 657–664. doi:10.1007/978-3-540-72586-2_95
34. Li Z, Zhang S, Wang RS, Zhang XS, Chen L (2008) Quantitative function for community detection. Phys Rev E 77:036109. doi:10.1103/PhysRevE.77.036109
35. Li J, Song Y (2013) A genetic algorithm for community detection in complex networks. Soft Comput 17(6):925–937. doi:10.1007/s11771-013-1611-y
36. Liu X, Murata T (2010) Advanced modularity-specialized label propagation algorithm for detecting communities in networks. Phys A Stat Mech Appl 389(7):143–150. doi:10.1016/j.physa.2009.12.019
37. Liu JX, Zeng J (2010) Community detection based on modularity density and genetic algorithm. In: 2010 International Conference on Computational Aspects of Social Networks (CASoN), pp 29–32. doi:10.1109/CASoN.2010.14
38. Lorrain F, White H (1971) Structural equivalence of individuals in social networks. J Math Sociol 1:49–80. doi:10.1080/0022250X.1971.9989788
39. Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM (2003) The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. Behav Ecol Sociobiol 54:396–405. doi:10.1007/s00265-003-0651-y
40. Massen CP, Doye JPK (2005) Identifying communities within energy landscapes. Phys Rev E 71 doi:10.1103/PhysRevE.71.046101
41. Medus A, Acuña G, Dorso CO (2005) Detection of community structures in networks via global optimization. Phys A Stat Mech Appl 358:593–604. doi:10.1016/j.physa.2005.04.022
42. Mitrović M, Tadić B (2009) Spectral and dynamical properties in classes of sparse networks with mesoscopic in homogeneities. Phys Rev E 80(2):026123. doi:10.1103/PhysRevE.80.026123
43. Nepusz T, Petróczi A, Négyessy L, Bazsó F (2008) Fuzzy communities and the concept of bridgeness in complex networks. Phys Rev E. doi:10.1103/PhysRevE.77.016107
44. Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E. doi:10.1103/PhysRevE.69.026113
45. Newman MEJ (2006) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E. doi:10.1103/PhysRevE.74.036104
46. Osman IH, Al-Ayoubi B (2005) MIC Analysis for Comparing Metaheuristics. In: Proceedings of the 6th Meta-heuristics International Conference, Vienna, Austria, August 22–26, pp 725–732
47. Papadopoulos S, Skusa A, Vakali A, Kompatsiaris Y, Wagner N (2009) Bridge bounding: a local approach for efficient community discovery in complex networks. eprint arXiv:0902.0871
48. Palla G, Derényi I, Farkas I, Vicsek T (2005) Uncovering the overlapping community structure of complex networks in nature and society. Nature 435:814–818. doi:10.1038/nature03607
49. Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E. doi:10.1103/PhysRevE.76.036106
50. Reichardt J, Bornholdt S (2004) Detecting Fuzzy community structures in complex networks with a Potts model. Phys Rev Lett. doi:10.1103/PhysRevLett.93.218701
51. Reichardt J, Bornholdt S (2006) When are networks truly modular? Phys D 224:20–26. doi:10.1016/j.physd.2006.09.009
52. Rosvall M, Bergstrom CT (2007) An information-theoretic framework for resolving community structure in complex networks. Proc Natl Acad Sci USA 104:7327–7331. doi:10.1073/pnas.0611034104
53. Shang R, Bai J, Jiao L, Jin C (2010) Community detection based on modularity and an improved genetic algorithm. Phys A Stat Mech Appl 392(5):1215–1231. doi:10.1016/j.physa.2012.11.003
54. Shi C, Yan Z, Cai Y, Wu B (2012) Multi-objective community detection in complex networks. Appl Soft Comput 12(2):850–859. doi:10.1016/j.asoc.2011.10.005
55. Talbi E (2009) Metaheuristics from design to implementation. Wiley, Hoboken
56. Tasgin M, Bingol H (2006) Community detection in complex networks using genetic algorithm. In: ECCS '06. Proceedings of the European Conference on Complex Systems

57. Traag VA, Bruggeman J (2009) Community detection in networks with positive and negative links. Phys Rev. doi:10.1103/PhysRevE.80.036115
58. White JG, Southgate E, Thompson JN, Brenner S (1986) The structure of the nervous system of the nematode C. elegans (aka "The Mind of a Worm"). Phil Trans R Soc Lond 314:1–340. doi:10.1098/rstb.1986.0056
59. Wu FY (1982) The Potts model. Rev Mod Phys 54(1):235–268. doi:10.1103/RevModPhys.54.235
60. Xu G, Tsoka S, Papageorgiou LG (2007) Finding community structures in complex networks using mixed integer optimization. Eur Phys J B 60:231–239. doi:10.1140/epjb/e2007-00331-0
61. Yang B, Liu J (2008) Discovering global network communities based on local centralities. ACM Trans Web 2(1):1–32
62. Ye Z, Hu S, Yu J (2008) Adaptive clustering algorithm for community detection in complex networks. Phys Rev. doi:10.1103/PhysRevE.78.046115
63. Yuruk N, Mete M, Xu X, Schweiger TAJ (2007) A divisive hierarchical structural clustering algorithm for networks. In: Data mining workshops, ICDM, Omaha, pp 441–448. doi:10.1109/ICDMW.2007.73
64. Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33(4):452–473
65. Zanghi H, Ambroise C, Miele V (2008) Fast online graph clustering via Erdös-Rényi mixture. Pattern Recognit 41(12):3592–3599. doi:10.1016/j.patcog.2008.06.019
66. Zhang XS, Wang RS (2008) Optimization analysis of modularity measures for network community detection. The Second International Symposium on Optimization and System Biology (OSB'08). Lijiang, China
67. Zhou H (2003) Network landscape from a Brownian particle's perspective. Phys Rev. doi:10.1103/PhysRevE.67.041908