Received: 25 June 2018 | Revised: 10 May 2019 | Accepted: 27 December 2019

DOI: 10.1111/coin.12273

ORIGINAL ARTICLE

Computational Intelligence WILEY

Detecting community structure in complex networks using genetic algorithm based on object migrating automata

Bagher Zarei[1] | **Mohammad Reza Meybodi**[2]

[1]Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

[2]Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

Correspondence
Mohammad Reza Meybodi, Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran.
Email: mmeybodi@aut.ac.ir

Abstract

Community structure is an important topological feature of complex networks. Detecting community structure is a highly challenging problem in analyzing complex networks and has great importance in understanding the function and organization of networks. Up until now, numerous algorithms have been proposed for detecting community structure in complex networks. A wide range of these algorithms use the maximization of a quality function called *modularity*. In this article, three different algorithms, namely, MEM-net, OMA-net, and GAOMA-net, have been proposed for detecting community structure in complex networks. In GAOMA-net algorithm, which is the main proposed algorithm of this article, the combination of genetic algorithm (GA) and object migrating automata (OMA) has been used. In GAOMA-net algorithm, the MEM-net algorithm has been used as a heuristic to generate a portion of the initial population. The experiments on both real-world and synthetic benchmark networks indicate that GAOMA-net algorithm is efficient for detecting community structure in complex networks.

KEYWORDS

community structure, community structure detection, complex networks, modularity optimization

1 | INTRODUCTION

In the context of network theory, a complex network is a graph with nontrivial topological features—features that do not occur in simple networks such as lattices or random graphs but often occur in graphs that modeling real-world systems. Most social, biological, and technological networks display substantial nontrivial topological features, with patterns of connection between their elements that are neither purely regular nor purely random. Such features include a heavy tail (power law) in the degree distribution, a high clustering coefficient, assortativity or disassortativity among vertices, community structure, and hierarchical structure.[1-3]

Among the above features, community structure detection has become an important problem in complex networks analysis. The purpose of community structure detection problem is to identify groups of nodes such that the density of links within the groups is high and between the groups is low.[4,5] This is a qualitative definition. In Reference 6, two quantitative definitions for a community are given. The subgraph C is a community in a strong sense, if $K_i^{\text{in}}(C) > K_i^{\text{out}}(C), \forall i \in C$. In a strong community, each node has more connections within the community than with the rest of the graph. The subgraph C is a community in a weak sense, if $\sum_{i\in C} K_i^{\text{in}}(C) > \sum_{i\in C} K_i^{\text{out}}(C), \forall i \in C$. In a weak community, the sum of all degrees within community C is larger than the sum of all degrees toward the rest of the network. Clearly, a community in a strong sense is also a community in a weak sense, while the converse is not true. In Reference 7, another quantitative definitions for a community is given as follows: $K_i(C) \geq K_i(C'), \forall\, i \in C, \forall\, C' \neq C$. According to this definition, the connections of each node within the community to which it belongs should not be less than its connections to other communities. When there are only two communities in the network, this definition is equivalent to the definition of a community in strong sense. When there are more than two communities in the network, the constraint of this definition is weaker than the strong community.

In recent years, many approaches have been proposed to detect communities in complex networks. Among them, methods based on the optimization of an objective function have become more popular. Modularity[8] is one of the widely used objective functions for detecting communities. Notion of modularity is based on the idea that a random graph is not expected to have a community structure. Modularity measures "the possible existence of communities by comparison between the actual density of edges in a subgraph and the density one would have in the subgraph if the vertices of the graph were attached regardless of a community structure". This expected edge density depends on the chosen null model, that is, a copy of the original graph keeping some of its structural properties but without community structure. The choice of the null graph is in principle arbitrary and several possibilities exist. The usual choice of the null graph is to choose a model with the same size and degree distribution as of the original graph. The larger modularity indicates that there are more connections within the communities than the expected number in the null model. If one takes modularity as the objective function, the problem of detecting community structure can be modeled as a modularity optimization problem. It has been proven that modularity optimization problem is an NP-hard problem.[9] Hence, approximate algorithms are required while dealing with large networks.

In this article, three different algorithms, namely, MEM-net, OMA-net, and GAOMA-net are proposed for detecting community structure in complex networks. In MEM-net algorithm, each network node has a memory with a specific depth and a label is assigned to it. As a result of interacting with neighboring nodes, label of each node either moves to different states of its memory or it changes. Finally, the optimal community structure is determined by the labels of the network

nodes. MEM-net algorithm has linear time complexity in terms of the number of network edges. In OMA-net algorithm, the entire network is modeled as an object migrating automaton and each network node is mapped to one of its actions; also, an object (label) is assigned to each action. Finally, the optimal community structure is found through the evolution of the object migrating automaton. Like MEM-net algorithm, OMA-net algorithm has linear time complexity in terms of the number of network edges. In GAOMA-net algorithm, which is the main proposed algorithm of this article, the combination of genetic algorithm (GA) and object migrating automata (OMA) has been used. By combining GA with the OMA, it largely overcomes the problem of premature convergence. In addition, in GAOMA-net algorithm, the MEM-net algorithm has been used as a heuristic to generate a portion of the initial population. This will accelerate the convergence of the algorithm. The experiments on both real-world and synthetic benchmark networks indicate that GAOMA-net algorithm is efficient for detecting community structure in complex networks.

The main contributions of this work are summarized as follows:

- A method with linear time complexity, named MEM-net, is proposed to solve the problem of community structure detection (see Section 3). This method has a moderate performance on different benchmark networks.
- Standard object migrating automaton[10] is modified so that it can be useable in locus-based adjacency representation.[11] In addition, a method, named OMA-net, is proposed that uses modified object migrating automaton to solve the problem of community structure detection (see Section 4). This method also has a moderate performance on different benchmark networks.
- GA and OMA are integrated as a single framework (GAOMA-net) to solve the problem of community structure detection (see Section 5). To the best of our knowledge, the integration of GA and OMA is the first work in the community structure detection.
- As we know, one of the major drawbacks of population-based evolutionary algorithms, including GAs, is premature convergence (falling in the local optimum) due to the loss of diversity in the population. So far, different solutions have been proposed to solve this problem. In GAOMA-net, this problem has been largely overcome using the learning capability of OMA.
- As we know, smart population generation plays a significant role in the convergence of population-based evolutionary algorithms. For this reason, in the proposed hybrid framework (GAOMA-net), MEM-net algorithm is used as a heuristic to generate a portion of the initial population. This will accelerate the convergence of the algorithm.

The rest of the article is organized as follows: in Section 2, some works related to the problem of community structure detection in complex networks are briefly reviewed. In Sections 3, 4, and 5, the proposed MEM-net, OMA-net, and GAOMA-net algorithms are described, respectively. The experimental results and discussion are reported in Section 6. Finally, Section 7 gives the conclusion of this article.

2 | RELATED WORKS

2.1 | Categories of community detection algorithms

In this section, a general categorization is presented for community detection methods. Given the nature of the existing algorithms, community detection methods can be divided into four major

categories: traditional methods, divisive methods, quality optimization-based methods, and label propagation-based methods. In the following, each of these categories is briefly explained.

A. Traditional methods: These methods use graph clustering techniques to detect communities. Traditional methods based on similarity measures, such as spectral bisection, the Kernighan-Lin algorithm, and hierarchical clustering are not ideal for real-world networks (eg, biological and social networks) and may fail in detecting communities.[12] These methods can be further classified as:

I. Graph partitioning: The problem of graph partitioning consists in dividing the vertices of a network into two or, in general, into g groups of predefined size, such that the number of edges lying between the groups, usually named the cut size, is minimal. Various measures such as max-flow min-cut, conductance, ratio cut (RC), and normalized cut are used to determine the cut size. Specifying the number and size of modules is necessary, which is generally not the case in many real-world networks. Besides, from the methodological point of view, using iterative bisectioning to split the graph into more clusters is not a reliable procedure.[1,3,4]

II. Hierarchical clustering: Graphs may have a hierarchical structure, that is, may display several levels of grouping of the vertices, with small clusters included within large clusters, which are in turn included in larger clusters, and so on. In such cases, one may use hierarchical clustering algorithms, that is, clustering techniques that reveal the multilevel structure of the graph. The starting point of any hierarchical clustering method is the definition of a similarity measure between vertices. Various similarity measures such as Euclidean distance (L_2-norm), Manhattan distance (L_1-norm), L_∞-norm, and cosine similarity can be used. After a measure is chosen, one computes the similarity for each pair of vertices, no matter if they are connected or not. At the end of this process, one is left with a $n \times n$ matrix S, the similarity matrix. Hierarchical clustering techniques aim at identifying groups of vertices with high similarity, and can be classified in two categories:

1. *Agglomerative algorithms*, in which clusters are iteratively merged if their similarity is sufficiently high;
2. *Divisive algorithms*, in which clusters are iteratively split by removing edges connecting vertices with low similarity.

The two classes refer to opposite processes: agglomerative algorithms are bottom-up, as one starts from the vertices as separate clusters (singletons) and ends up with the graph as a unique cluster; divisive algorithms are top-down as they follow the opposite direction. Hierarchical clustering has the advantage that it does not require a preliminary knowledge on the number and size of the clusters. However, it does not provide a way to discriminate between the many partitions obtained by the procedure, and to choose that or those that better represent the community structure of the graph. The results of the method depend on the adopted similarity measure. The procedure also yields a hierarchical structure by construction, which is rather artificial in most cases, since the graph at hand may not have a hierarchical structure at all. Moreover, vertices may not be correctly classified, and in many cases some vertices are missed even if they have a central role in their clusters.[12] Another problem is that vertices with just one neighbor are often classified as separated clusters, which in most cases do not make sense. Finally, a major weakness of agglomerative hierarchical clustering is that it does not scale well.[4]

III. Partitional clustering: Partitional clustering indicates another class of methods to find clusters in a set of data points. Here, the number of clusters is predefined, say k. The points are embedded in a metric space, so that each vertex is a point and a distance measure is defined

between pairs of points in the space. The distance is a measure of dissimilarity between vertices. The goal is to separate the points in k clusters such that maximize/minimize a given cost function based on distances between points and/or from points to centroids, that is, suitably defined positions in space. Some of the most used cost functions are: minimum k-clustering, k-clustering sum, k-center, and k-median.[4]

IV. Spectral clustering: Spectral clustering includes all methods that partition the set of objects into clusters by using the spectral properties such as eigenvectors of similarity matrix or other matrices (eg, the Laplacian matrix, the normalized Laplacian matrix, the modularity matrix, etc.) derived from it. In particular, the objects could be points in some metric space, or the vertices of a graph. Spectral clustering consists of a transformation of the initial set of objects into a set of points in a metric space, by using the entries of eigenvectors. The i-th entries of the eigenvectors are the coordinates of vertex i in a k-dimensional Euclidean space, where k is the number of eigenvectors used. The set of points is then clustered via standard techniques, like k-means clustering. One may wonder why it is necessary to cluster the points obtained through the eigenvectors, when one can directly cluster the initial set of objects, based on the similarity matrix. The reason is that the change of representation induced by the eigenvectors makes the cluster properties of the initial dataset much more evident. In this way, spectral clustering is able to separate data points that could not be resolved by applying directly k-means clustering.[2,4,5]

B. Divisive methods: A simple way to identify communities in a graph is to detect the edges that connect vertices of different communities and remove them, so that the clusters get disconnected from each other. The crucial point is to find a property of intercommunity edges that could allow for their identification. Divisive methods do not introduce substantial conceptual advances with respect to traditional techniques, as they just perform hierarchical clustering on the graph at study. The main difference with divisive hierarchical clustering is that here one removes intercluster edges instead of edges between pairs of vertices with low similarity and there is no guarantee that intercluster edges connect vertices with low similarity.[2,4]

C. Quality optimization-based methods: Communities can also be identified based on the quality of a partition. In this case, the problem of detecting community structure can be modeled as an optimization problem. The latter is not trivial, as the number of possible partitions of a network in clusters increases exponentially with the size of the network, making exhaustive optimization computationally unreachable even for relatively small graphs. Therefore, these class of methods focus on to achieve fairly good approximations of quality function with the least computational cost. Quality optimization-based methods can be classified as[4]:

I. Greedy techniques: Greedy techniques merge vertices iteratively to form large communities such that their overall quality is maximum. These methods are fast but not very accurate.

II. Simulated annealing: Simulated annealing[13] is a probabilistic procedure for global optimization. It consists in performing an exploration on the space of possible states, looking for the global optimum of a function F, say its maximum. Transitions from one state to another occur with probability 1 if F increases after the change, otherwise with a probability $\exp(\beta\Delta F)$, where ΔF is the decrease of the function and β is an index of stochastic noise, a sort of inverse temperature, which increases after each iteration. The noise reduces the risk that the system gets trapped in local optima. At some stage, the system converges to a stable state, which can be good approximation of the maximum of F, depending on how many states were explored and how slowly is varied. Simulated annealing was first employed for modularity optimization by Guimera et al.[14]

III. Extremal optimization: Extremal optimization is a heuristic search procedure proposed by Boettcher and Percus,[15] in order to achieve an accuracy comparable with GA and simulated

annealing, but with a substantial gain in computer time. It is based on the optimization of local variables, expressing the contribution of each unit of the system to the global function at study. This technique was used for modularity optimization by Duch and Arenas.[16]

IV. Spectral optimization: Spectral optimization refers to the use of eigenvectors and eigenvalues of the modularity matrix for modularity optimization. The spectral optimization of modularity is quite fast.[4]

V. Evolutionary algorithms: Evolutionary algorithms are based on the evolution of the species. In general, they are based on the evolution theory of Charles Darwin. The way the evolutionary algorithms are implemented varies considerably; however, the basic idea behind all these variations is similar. Evolutionary algorithms are characterized by the existence of a population of individuals exposed to environmental pressure, which leads to natural selection, that is, the survival of the fittest, and in turn the increase of the average fitness of the population. They are known for their effective local learning and global searching capabilities.

D. Label propagation-based methods: Unlike other community detection algorithms, label propagation-based methods do not optimize any given objective function and do not require to have a priori information about the network structure. Initially, a distinct label is assigned to each node. Then, during an iterative process, each node adopts a label in agreement with the majority of its neighbors. When all nodes have the label of the majority of their neighbors, algorithm stops. At the end, the connected nodes with the same label establish a community. The time complexity of these algorithms is almost linear. The main drawback of these methods is that they may produce different solutions in different runs.[7]

In Table 1, the advantages and disadvantages of each category along with some of the well-known algorithms in each category are given.

2.2 | Evolutionary community detection algorithms

Given that the proposed algorithm is an evolutionary algorithm, in the following, only some of the evolutionary algorithms are described. Single-objective evolutionary approaches to community detection can be found in References 43-49 and multiobjective evolutionary approaches to community detection can be found in References 50-58.

In References 43 and 44, the authors present a GA that uses *modularity* of Newman and Girvan[8] as fitness function. An individual is constituted by n genes, where n is the number of network nodes. The i-th gene corresponds to the i-th node, and its value is the community identifier of node i. In these algorithms, by using a simple heuristic, the initial population is improved. This leads to fast convergence of the algorithms. In addition, they use nonstandard one-way crossover operator as follows: In one-way crossover, one of the parent chromosomes is called the source chromosome and the other is called the destination chromosome. A node is randomly selected and its community identifier is determined in the source chromosome. Suppose that its community identifier is c_r. Then, all nodes with community identifier c_r are found in the source chromosome and their community identifier is copied to the corresponding nodes in the destination chromosomes.

In Reference 45, Pizzuti proposed the concept of *community score* (CS) for evaluating the quality of a partition and used a GA, named GA-Net, to optimize it. In this algorithm, locus-based adjacency representation is used. The main advantage of this representation is that the number of communities is automatically determined in the decoding process. Moreover, in this

TABLE 1 Advantages and disadvantages of different categories of community detection algorithms

	Method	Advantage/disadvantage	Example algorithms
Traditional methods	Graph partitioning	Disadvantages[4]: • need a predefined number of communities • need a predefined size of communities • using iterative bisectioning to split the graph into more clusters is not a reliable procedure	• Kernighan-Lin algorithm[17] • Spectral bisection method[18] • Flake et al[19] • Flake et al[20]
	Hierarchical clustering	Advantages[4,21]: • do not need a predefined number of communities • do not need a predefined size of communities • can find large number of partitions Disadvantages[4,22,23]: • need to define similarity measure between vertices • it does not provide a way to discriminate between the obtained partitions, and to choose that or those that better represent the community structure of the graph • dependency of the results to the adopted similarity measure • the procedure yields a hierarchical structure by construction, which is rather artificial in most cases, since the graph at hand may not have a hierarchical structure at all • poor results in most cases • agglomerative hierarchical clustering is not scalable (use of matrix-based computation creates issue of scalability)	• Hierarchical clustering algorithms[24]
	Partitional clustering	Disadvantages[4]: • need a predefined number of communities	• k-Mean clustering[25] • Fuzzy k-mean clustering[26] • Fuzzy k-mean clustering[27]

(Continues)

TABLE 1 (Continued)

	Method	Advantage/disadvantage	Example algorithms
	Spectral clustering	Advantages[1]: • fast Disadvantages[4,5]: • need a predefined number of communities • not always reliable (when the network is very sparse, the separation between the eigenvalues of the community-related eigenvectors and the bulk is not sharp)	• Adjacency matrix spectral clustering method[28] • Laplacian matrix spectral clustering method[29] • Shi and Malik[30] • Ng et al[31]
Divisive methods		Advantages[4,22]: • do not need a predefined number of communities • do not need a predefined size of communities • more efficient Disadvantages[4]: • finding a property of intercommunity edges that could allow for their identification is crucial	• Girvan and Newman[23] • Newman and Girvan[8] • Radicchi et al[6] • Fortunato et al[32]
Quality optimization-based methods	Greedy methods	Advantages[21]: • fast Disadvantages[21]: • poor accuracy	• Newman[33] • CNM algorithm[34] • Wakita and Tsurumi[35] • Louvain Algorithm[36]
	Simulated annealing	Disadvantages[37]: • slow	• Guimera et al[14]

(Continues)

TABLE 1 (Continued)

	Method	Advantage/disadvantage	Example algorithms
	Extremal optimization		• Duch and Arenas[16]
	Spectral optimization	Advantages[4,38]: • quite fast	• Newman[39,40] • Richardson et al[41] • White and Smyth[42]
	Evolutionary algorithms	Disadvantages: • premature convergence (falling in the local optimum) due to the loss of diversity in the population	• see Section 2.2[43-58]
Label propagation-based methods		Advantages[7]: • Simple • very fast • no parameter needed Disadvantages[7]: • may produce different solutions in different runs	• LAP algorithm[7] • SLPA algorithm[59] • LabelRank algorithm[60] • COPRA algorithm[61]

algorithm, specialized variation operators (recombination and mutation) are used. These operators reduce the search space by generating safe solutions; hence, the convergence of the algorithm is improved. In a safe solution, the *i*-th gene can have *j* value if and only if there is an edge between *i* and *j* nodes.

In Reference 47, the authors proposed a memetic algorithm, named Meme-Net, for optimizing the *modularity density (D)*.[62] Modularity density includes a tunable parameter that allows one to explore the network at different resolutions and reveal the hierarchical structure of the network. This algorithm is a synergy of a GA with a hill-climbing strategy as the local search procedure. In this algorithm, the initial population is improved with the same heuristic as in References 43 and 44 and two-way crossover operator is used.

In Reference 48, the authors proposed a memetic algorithm, named MA-COM, for detecting community structure in complex networks. This algorithm combines a GA with a dedicated crossover operator (priority-based crossover) and a multilevel local optimization procedure. To maintain population diversity and avoid premature convergence, MA-COM additionally uses a quality-and-distance-based population updating strategy. In this updating strategy, not only the quality of the offspring but also its distance to the solutions of the population is considered to decide whether the offspring can be inserted into the population or not.

In References 50 and 51, Pizzuti proposed a multiobjective GA, named MOGA-Net, to uncover community structure in complex networks. This algorithm adopts the nondominated sorting GA (NSGA-II)[63] as its optimization mechanism and optimizes two contradictory objective functions termed as *CS* and *Community Fitness (*CF*)*. *CS* to some extent measures the link density inside communities, while CF measures the link density between communities. Both the objective functions have a positive real-valued parameter controlling the size of the communities. The higher the value of the parameter, the smaller the size of the communities found. By maximizing CS and negative of CF, we can ensure that the links inside communities are dense and the links between communities are sparse, which is in accordance with the basic nature of the communities in a network. Thus, the proposed optimization model is as follows:

$$\max \begin{Bmatrix} f1 = \text{CS} \\ f2 = -\text{CF} \end{Bmatrix}.$$

In Reference 52, the authors proposed a multiobjective evolutionary algorithm, named MOEA/D-Net, to reveal community structure in complex networks. This algorithm adopts the decomposition[64] as its optimization mechanism. The highlight of this work is the newly cranked out multiobjective community optimization model, which optimizes two objectives termed as *negative ratio association (NRA)* and *RC*. RA calculates the link density within communities, while RC measures the link density between communities. By minimizing NRA and RC, we can ensure that the links within communities are dense and the links between communities are sparse. Thus, the proposed optimization model is as follows:

$$\min \begin{Bmatrix} f1 = \text{NRA} \\ f2 = \text{RC} \end{Bmatrix}.$$

In Reference 53, a discrete framework of the particle swarm optimization algorithm is proposed. Based on the proposed discrete framework, a multiobjective discrete particle swarm optimization algorithm, named MODPSO, based on decomposition is proposed to solve the community structure detection problem by minimizing two objectives termed as *Kernel K-Means*

(KKM) and *RC*. The motivations to define the above objectives are that, the KKM is a decreasing function of the number of communities while the opposite trend happens to the RC function, in other words, they are two conflicting objectives. From the definitions of KKM and RC, we can notice that the KKM can be considered as the negative sum of the intracommunity links and RC can be considered as the sum of the intercommunity links. By minimizing KKM and RC, we can ensure that the links within communities are dense and the links between communities are spare. Thus, the proposed optimization model is as follows:

$$\min \begin{Bmatrix} f1 = \text{KKM} \\ f2 = \text{RC} \end{Bmatrix}.$$

In References 54 and 55, the authors proposed a multiobjective evolutionary algorithm, named SN-MOGA, to reveal community structure in signed networks. In a signed network, the weight of the edges can be positive or negative. This algorithm uses the NSGA-II framework to optimize two objectives termed as Q_{signed} and *frustration*. Q_{signed} measures the quality of the communities in a signed network. By minimizing frustration, we can ensure that the sum of the negative links within communities and the sum of the positive links between difference communities are minimum, which is in accordance with the basic nature of the communities in a signed network. Thus, the proposed optimization model is as follows:

$$\max \begin{Bmatrix} f1 = Q_{\text{signed}} \\ f2 = -\text{frustration} \end{Bmatrix}.$$

In Reference 56, the authors proposed a multiobjective evolutionary algorithm based on node similarity, named MEAs-SN, for community detection in signed social networks by optimizing two objectives termed as $f_{\text{pos}-\text{in}}$ and $f_{\text{neg}-\text{out}}$. By maximizing $f_{\text{pos}-\text{in}}$, we can ensure high positive similarities within communities, and by maximizing $f_{\text{neg}-\text{out}}$we can ensure high negative similarities between different communities. Thus, the proposed optimization model is as follows:

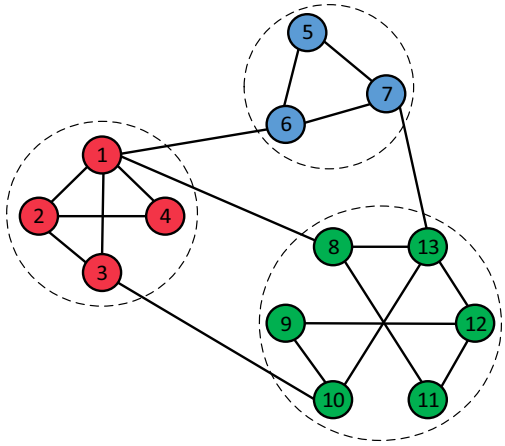
$$\max \begin{Bmatrix} f1 = f_{\text{pos-in}} \\ f2 = f_{\text{neg-out}} \end{Bmatrix}.$$

3 | THE PROPOSED MEM-NET ALGORITHM

In this section, a memory-based algorithm, named MEM-net, is proposed for detecting community structure in complex networks. In this algorithm, each network node has a memory with a specific depth and a label is assigned to it. As a result of interacting with neighboring nodes, label of each node either moves to different states of its memory or it changes. Finally, the optimal community structure is determined by the labels of the network nodes. The detailed description of MEM-net algorithm is described below.

3.1 | Solution representation

In MEM-net algorithm, locus-based adjacency representation[11] is used for detecting community structure in complex networks. In this graph-based representation, the solution is represented by a solution vector, $S = (s_1, s_2, \ldots, s_n)$, where s_i corresponds to the node i and it can take label of the



	Node	1	2	3	4	5	6	7	8	9	10	11	12	13
State	1 (Least Stable)													
	2	2												
	3					5		13		9				
	4			2			6		12		8			10
	5 (Most Stable)		4		1	6							13	

FIGURE 1 (Top) A sample network with three communities. Different communities are shown in different colors. (Bottom) An example of nodes' label and their memory state (the memory depth of nodes is considered to be 5). Solid circles indicate the current state and label of nodes. For example, node 9 is in state 4 and its label is 12. Typically, nodes that there is no edge between them and the nodes of other communities are in stable states of their memory; and nodes that there is edge between them and the nodes of other communities, are in unstable states of their memory [Color figure can be viewed at wileyonlinelibrary.com]

one of the adjacent nodes of node i (including node i itself). Loci and their values (labels) represent nodes of the graph $G = (V, E)$, and a value j assigned to the s_i (i-th locus) is interpreted as a link between the nodes i and j. This means that in the resulting solution, they will be in the same community. The decoding procedure of this representation is to identify all the connected components of graph G. The nodes belonging to the same connected component are assigned to one community. Using a simple backtracking scheme, the decoding procedure can be done in linear time. After decoding, the solution vector is converted into the membership vector, $C = (c_1, c_2, \ldots, c_n)$, which indicates the community structure. In the membership vector, c_i indicates the index of the community to which node i belongs. The main advantage of this representation is that it can represent community structure with dynamic number of communities. The number of communities is automatically determined in the decoding process. A solution vector for the network of Figure 1 in locus-based adjacency representation and its corresponding graph structure and membership vector are shown in Figure 2.

3.2 | Solution vector construction

In MEM-net algorithm, the solution vector is formed by the label of each network node. Hence, at each cycle t, the solution vector can be represented as

$$S(t) = (L_1(t), L_2(t), \ldots, L_n(t)), \tag{1}$$

where $L_i(t)$ is the label of node i at cycle t.

3.3 | Reward and penalty of a node

In MEM-net algorithm, each network node has a memory with a specific depth and its label either moves to different states of its memory or changes as a result of the reward or penalty. Assuming

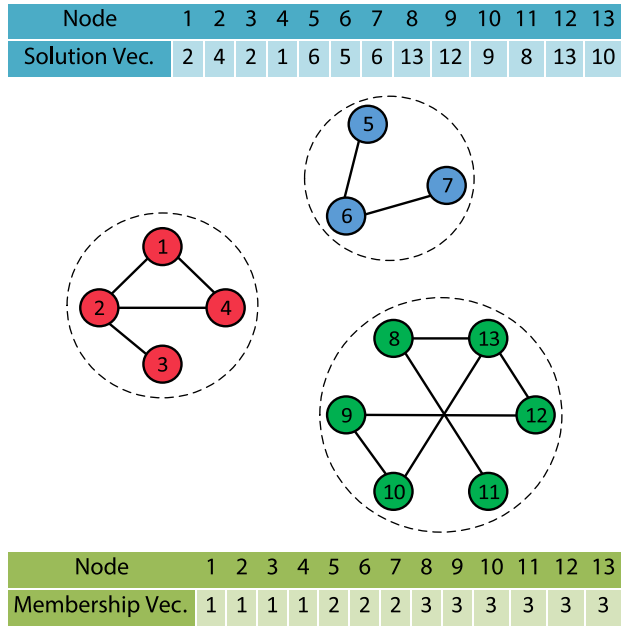


FIGURE 2 (Top) A solution vector for the network of Figure 1 in locus-based adjacency representation. (Middle) The graph structure corresponding to the solution vector. (Bottom) The membership vector corresponding to the solution vector [Color figure can be viewed at wileyonlinelibrary.com]

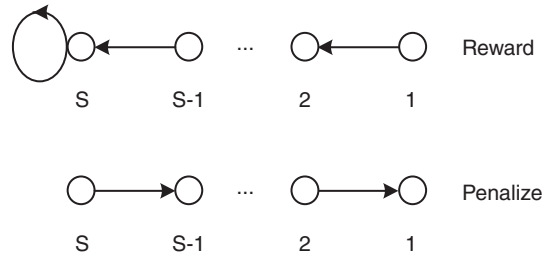


FIGURE 3 The way of rewarding and penalizing of a node.

that the memory depth of nodes is S, each node will have different memory states, that is, 1, 2, … , S. State 1 is called the most unstable state and state S is called the most stable state. By rewarding a node, its label moves towards stable states of memory and by penalizing a node, its label moves towards unstable states of memory. If the label of a node is in the most stable state and the related node is rewarded, the label of the node remains in the same state. If the label of a node is in the most unstable state and the related node is penalized, the label of the node will change through the majority voting of its neighbors' label. Figure 3 depicts the way of rewarding and penalizing of a node.

3.4 | Framework of the MEM-net algorithm

In MEM-net algorithm, in each cycle, all nodes are randomly selected and evaluated. Based on the evaluation result, the selected node is rewarded or penalized (asynchronous updating). For evaluating the selected node, majority voting of its neighbors' label can be used. If the label of the selected node is equal to the label of the majority of its neighbors, the selected node is rewarded; otherwise, it is penalized. The process of randomly selecting a node and its rewarding/penalizing is repeated until the node labels do not change in several consecutive cycles. At the

Algorithm MEM-net

Inputs
$G = (V, E)$: The graph of the network
md: Nodes memory depth

Output
S: The community structure

Variables
L_i: The label of the node i

Initialization
Step 1:
for each node i of the graph, $1 \leq i \leq |V|$
$L_i = i$;
end

repeat
Step 2:
for each node i of the graph in random order
Evaluate the node i;
If evaluation result is positive **then**
Reward the node i;
else
Penalize the node i;
end
end
until the node labels does not change in several consecutive cycles

Solution Vector = label of nodes;
Membership Vector = **Convert**(Solution Vector);
return Membership Vector;

FIGURE 4 Main framework of the MEM-net algorithm.

end, the solution vector is formed by using the label of nodes. Then, after decoding, it is converted into the membership vector, which indicates the community structure. In Figure 4, the main framework of the MEM-net algorithm for detecting community structure in complex networks is given.

3.5 | Time complexity of the MEM-net algorithm

In this section, the time complexity of the MEM-net algorithm is analyzed. Let n and m denote the number of nodes and edges in the network, respectively. Hence, the average degree of nodes is $O(\bar{k}) = O(m/n)$. In step 1, the label of each network node is initialized. Doing this task for each node requires $O(1)$ time. Since the number of network nodes is equal to n, the total time complexity of step 1 will be $O(n)$. In step 2, each node is firstly evaluated; then, based on the evaluation result, it is rewarded or penalized. Since the time complexity of evaluating a node is $O(\bar{k})$ (because the majority voting of neighbors' label is used for evaluating of a node) and the reward/penalty of a node requires $O(1)$ time, the total time complexity of step 2 will be $O(n\bar{k}) = O(m)$. Hence, according to the above analysis, the total time complexity of MEM-net algorithm is linear in terms

of the number of network edges and equal to $O(n) + O(Tm) = O(Tm)$. T stands for the number of cycles.

4 | THE PROPOSED OMA-NET ALGORITHM

In this section, an algorithm based on the object migrating automaton, named OMA-net, is proposed for detecting community structure in complex networks. In this algorithm, the entire network is modeled as an object migrating automaton and each network node is mapped to one of its actions; also, an object (label) is assigned to each action. Finally, the optimal community structure is found through the evolution of the object migrating automaton. The detailed description of OMA-net algorithm is described below.

4.1 | Object migrating automaton

The number of ways in which a network with n nodes can be partitioned into k nonempty communities is equal to Stirling number of the second kind, S_n^k. Hence, the number of possible distinct partitions of a network with n node is $\sum_{k=1}^{n} S_n^k$. This sum is not known in the closed form but it can be seen that $S_n^1 + S_n^2 = 2^{n-1}$ for all $n > 1$; hence, the sum must increase at least exponentially in terms of n.[33] If the learning automaton is used for solving the problem of community structure detection in complex networks, the number of learning automaton actions must be equal to the number of possible distinct partitions of network (at least exponentially). Large numbers of actions reduce the convergence speed of learning automaton. For solving this problem, object migrating automaton was proposed by Oommen and Ma.[10]

The object migrating automaton is defined as a quintuple $\{\alpha, \phi, \beta, F, G\}$ in which $\alpha = \{a_1, \ldots, a_r\}$ is the set of learning automaton allowable actions; $\phi = \{\phi_1, \ldots, \phi_{rN}\}$ is the set of learning automaton states; $\beta = \{0, 1\}$ is the set of inputs; $F = \phi \times \beta \rightarrow \phi$ is the state mapping function; and $G = \phi \rightarrow a$ is the output mapping function. This type of automaton is used for classifying objects, assigning letters to keys and partitioning graph. In this automaton, each action indicates a class.

In the fixed structure automaton, the environment response to automaton makes it transfer from one state to another state. However, in object migrating automaton, objects are assigned to states and the environment response to automaton leads to the movement of objects among the states of automaton. In this way, objects classification is carried out. If object O_i is in action a_k of object migrating automaton, it indicates that this object will be in the class numbered K.

For each action a_k, there is a set of states $\{\phi_{(k-1)N+1}, \ldots, \phi_{kN}\}$, where N is the depth of memory and $1 \leq k \leq r$ in which r is the number of actions (classes). In general, state $\phi_{(k-1)N+1}$ can be considered as the most internal state (stable state) and state ϕ_{kN} can be considered as the most external state (unstable or boundary state) of that action. If two objects O_i and O_j are in states $\phi_{(k-1)N+n}$ and $\phi_{(k-1)N+m}$ (for $n < m$), respectively, in this case, certainty of belonging object O_i to class numbered K is more. Hence, for action a_k, state $\phi_{(k-1)N+1}$ is called the state with the most degree of certainty and state ϕ_{kN} is called the state with the least degree of certainty. Function F produces the next state of an objetc from the current state of the object and input of automaton. In Sections 4.1.1 to 4.1.3, this function is described for different automata. Function G determines the output (action/class) of an object from the current state of the object.

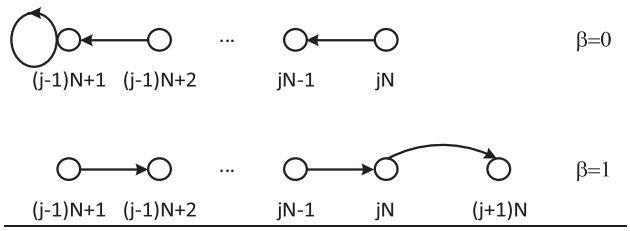


FIGURE 5 The way of rewarding ($\beta = 0$) and penalizing ($\beta = 1$) an action in object migrating automaton based on Tsetlin automaton

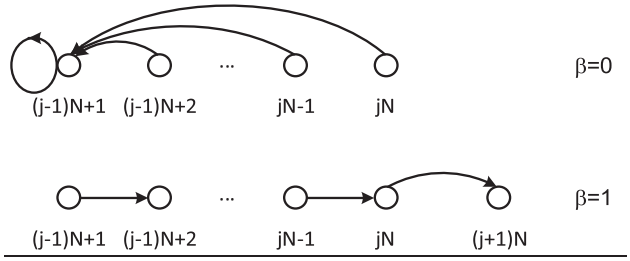


FIGURE 6 The way of rewarding ($\beta = 0$) and penalizing ($\beta = 1$) an action in object migrating automaton based on Krinsky automaton

4.1.1 | Object migrating automaton based on Tsetlin automaton

Object migrating automaton based on Tsetlin automaton uses Tsetlin automaton[65] for rewarding and penalizing actions. In this automaton, by rewarding an action, its object moves towards stable states; if the object is in the most stable state, it will remain in the same state. In addition, by penalizing an action, its object moves towards unstable states; if the object is in the most unstable state, it will migrate to the most unstable state of next action. Figure 5 depicts the way of rewarding and penalizing an action in this automaton.

4.1.2 | Object migrating automaton based on Krinsky automaton

Object migrating automaton based on Krinsky automaton uses Krinsky automaton[65] for rewarding and penalizing actions. In this automaton, by rewarding an action, its object moves to the most stable state; if the object is in the most stable state, it will remain in the same state. The way of penalizing an action in this automaton is similar to that of Tsetlin automaton. Figure 6 depicts the way of rewarding and penalizing an action in this automaton.

4.1.3 | Object migrating automaton based on Krylov automaton

Object migrating automaton based on Krylov automaton uses Krylov automaton[65] for rewarding and penalizing actions. In this automaton, by penalizing an action, it will be rewarded with the probability of 50% similar to the Tsetlin automaton, and it will be penalized with the probability of 50% similar to the Tsetlin automaton. The way of rewarding an action in this automaton is similar to that of Tsetlin automaton. Figure 7 depicts the way of rewarding and penalizing an action in this automaton.

Figure 8 shows a sample network with six nodes and the corresponding object migrating automaton based on Tsetlin/ Krinsky/Krylov automaton. Each action has an object that is in one of the memory states of that action. For example, action a_4 has object 6 that is in state 20 of memory.

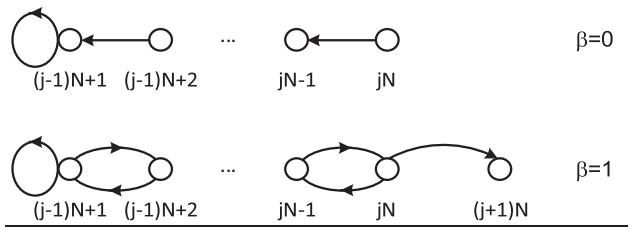


FIGURE 7 The way of rewarding ($\beta = 0$) and penalizing ($\beta = 1$) an action in object migrating automaton based on Krylov automaton

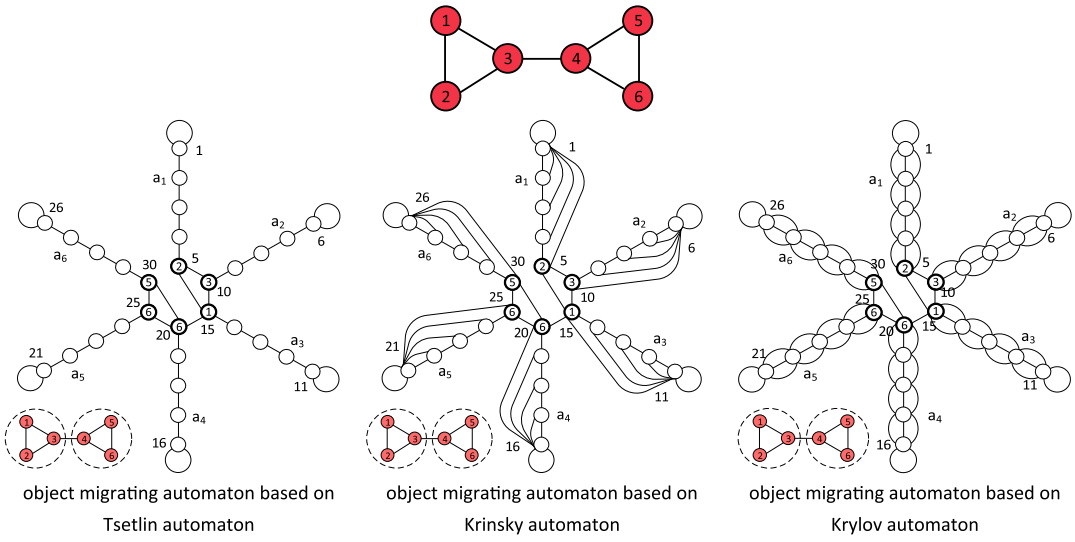


FIGURE 8 (Top) A sample network with six nodes. (Bottom) Object migrating automata corresponding to the network [Color figure can be viewed at wileyonlinelibrary.com]

4.1.4 | Object migrating automaton for detecting community structure in complex networks

In object migrating automaton for detecting community structure in complex networks, by rewarding/penalizing an action, that action will be rewarded/penalized according to one of the above OMA. In other words, by rewarding/penalizing an action, its object will move towards stable/unstable states. The only difference is that, if an object is in the most unstable state and its action is penalized, the object will not migrate to the most unstable state of next action; rather, the object will be replaced with object of one of its neighboring actions. However, in this replacement, a neighboring object is selected, which leads to the maximum increase in modularity. Figure 9 depicts the way of rewarding and penalizing an action in the object migrating automaton based on Tsetlin automaton.

4.2 | Solution representation

In OMA-net algorithm, locus-based adjacency representation[11] is used for detecting community structure in complex networks. Locus-based adjacency representation is fully described in Section 3.1. A solution vector for the network of Figure 8 in locus-based adjacency representation and its corresponding membership vector are shown in Figure 10.

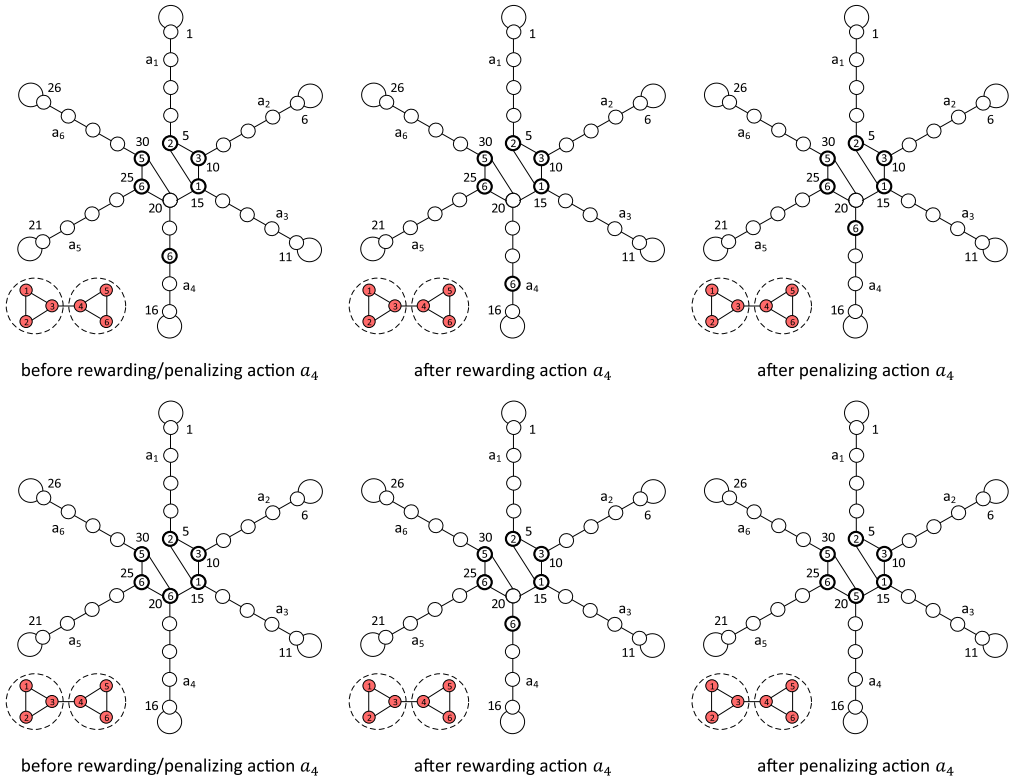


FIGURE 9 (Top) The way of rewarding and penalizing an action that its object is not in the most unstable state. (Bottom) The way of rewarding and penalizing an action that its object is in the most unstable state. In bottom figure for penalizing action a_4, its object is replaced with object of one of the its neighbors (in this example, object of action a_6) [Color figure can be viewed at wileyonlinelibrary.com]

Node	1	2	3	4	5	6
Solution Vector	2	3	1	6	6	5

Node	1	2	3	4	5	6
Membership Vector	1	1	1	2	2	2

FIGURE 10 (Top) A solution vector for the network of Figure 8 in locus-based adjacency representation. (Bottom) The membership vector corresponding to the solution vector [Color figure can be viewed at wileyonlinelibrary.com]

4.3 | Solution vector construction

In OMA-net algorithm, the solution vector is formed by the object of each action of the object migrating automaton. Hence, at each cycle t, the solution vector can be represented as

$$S(t) = (O_1(t), O_2(t), \ldots, O_n(t)), \tag{2}$$

where $O_i(t)$ is the object of action a_i at cycle t.

4.4 | Framework of the OMA-net algorithm

In OMA-net algorithm, in each cycle, an action is randomly selected and evaluated. Based on the evaluation result, the selected action is rewarded or penalized. For evaluating the selected action, the amount of changes in modularity by transferring that action to the community of each of its neighbors can be used. If the maximum amount of changes in modularity is smaller than or equal to zero, it means that, the community in which the selected action is located, is a desirable community; hence, that action is rewarded. Otherwise, the selected action will be penalized. Depending on the learning automaton type, the way of rewarding and penalizing an action is different, which is fully described in Sections 4.1.1 to 4.1.4. The process of randomly selecting an action and its rewarding/penalizing is repeated until the action objects does not change in several consecutive cycles. At the end, the solution vector is formed by using the object of actions of the object migrating automaton. Then, after decoding, it is converted into the membership

Algorithm OMA-net

Inputs
$G = (V, E)$: The graph of the network
$automatonType$: Object Migrating Automaton type
md: Actions memory depth.

Output
S: The community structure

Variables
O_i: The object (label) of the action α_i
S_i: The state of the action α_i

Initialization
Step 1:
for each action α_i of the OMA, $1 \leq i \leq |V|$
$O_i = id$ of random neighbor;
$S_i = i \times md$;
end

repeat
Step 2:
α_i = randomly selected action;
Evaluate the action α_i;
If evaluation result is positive **then**
Reward the action α_i;
else
Penalize the action α_i;
end
until the action objects does not change in several consecutive cycles

Solution Vector = object of actions;
Membership Vector = **Convert**(Solution Vector);
return Membership Vector;

FIGURE 11 Main framework of the OMA-net algorithm

vector, which indicates the community structure. In Figure 11, the main framework of the OMA-net algorithm for detecting community structure in complex networks is given.

4.5 | Time complexity of the OMA-net algorithm

In this section, the time complexity of the OMA-net algorithm is analyzed. Let n and m denote the number of nodes and edges in the network, respectively. Hence, the average degree of nodes is $O(\bar{k}) = O(m/n)$. In step 1, the object of each object migrating automaton action is initialized. Doing this task for each action requires $O(1)$ time. Since the number of object migrating automaton actions is equal to n, the total time complexity of step 1 will be $O(n)$. In step 2, an action is firstly evaluated; then, based on the evaluation result, it is rewarded or penalized. Since the time complexity of evaluating an action is $O(\bar{k})$ (because the amount of changes in modularity by transferring an action to the community of each of its neighbors is used for evaluating it) and the reward/penalty of an action requires $O(1)$ time, the total time complexity of step 2 will be $O(\bar{k})$. Hence, according to the above analysis, the total time complexity of OMA-net algorithm is linear in terms of the number of network nodes and equals to $O(n) + O(T\bar{k}) = O(n)$. T stands for the number of cycles.

5 | THE PROPOSED GAOMA-NET ALGORITHM

In this section, a hybrid algorithm (combination of GA and OMA), named GAOMA-net, is proposed for detecting community structure in complex networks. Unlike classic GAs, in the GAOMA-net algorithm, binary coding for chromosomes is not used. In 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. Finally, the optimal community structure is found through the evolution of GA.

By combining GA and learning automata and the concepts of gene, chromosome, action, and depth, one can achieve an efficient search method for detecting community structure in complex networks. By simultaneously using GA and learning automata in the search process, the speed of achieving a solution is significantly increased and avoids falling algorithm in the local optimum. Self-healing, reproduction, rewarding and penalizing (guiding), and no need prior information about the network such as the number and size of the communities are main features of hybrid GAOMA-net algorithm. The detailed description of GAOMA-net algorithm is described below.

5.1 | Initial population

Assuming that the number of population members is n, hence, n chromosomes (solution vector or object migrating automaton) are randomly generated. At the beginning, objects are placed in the most unstable state of its action. For increasing convergence speed, a portion of the initial population can be generated using heuristics such as MEM-net Algorithm. Since the time complexity of the MEM-net algorithm is linear, the generation of a portion of the initial population using it does not have a significant time overhead. In Figure 12, an example of initial population with $n = 6$ members for the network of Figure 8 is shown.

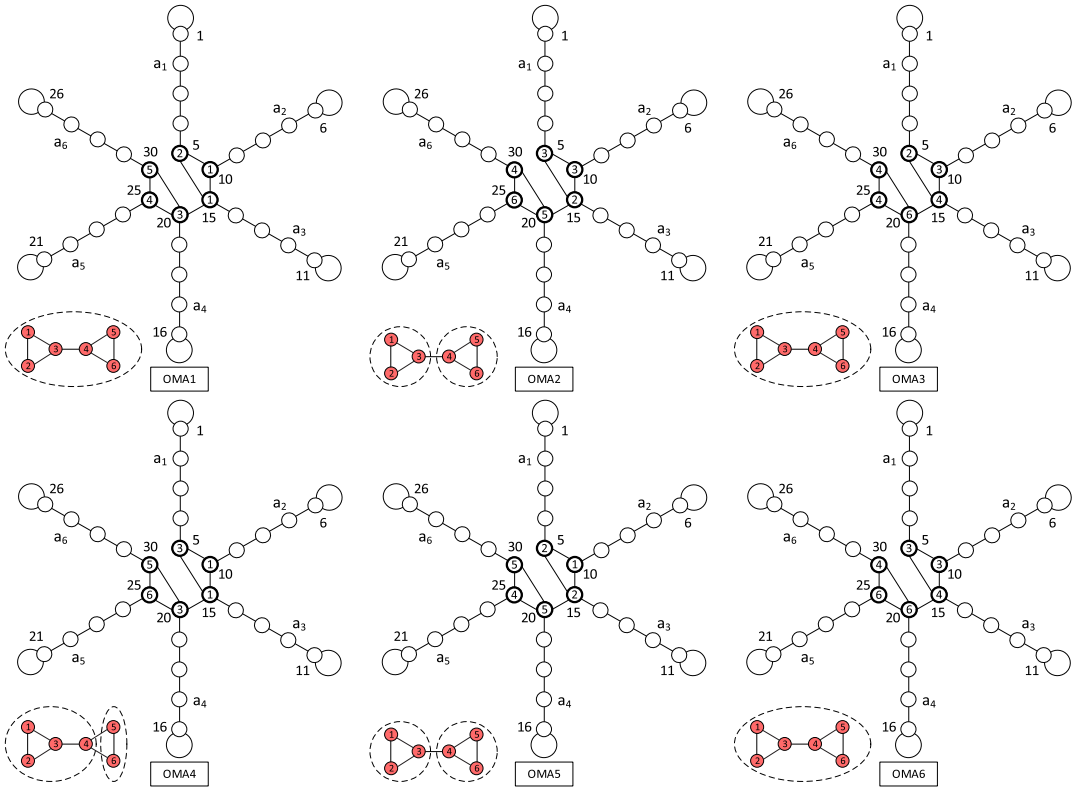


FIGURE 12 An example of initial population with $n = 6$ members for the network of Figure 8 [Color figure can be viewed at wileyonlinelibrary.com]

5.2 | Fitness function

In GAs, fitness function indicates the survival of chromosomes. In the problem of community structure detection in complex networks, the purpose is to find a partition with maximum modularity. Hence, fitness function is defined as follows:

$$\text{Fitness (chromosome or community structure)} = \text{Modularity (community structure)}. \quad (3)$$

5.3 | Operators

In this section, the operators used in GAOMA-net algorithm are described. These operators include the common operators in GAs (selection, recombination, and mutation) and an extra operator called reward/penalize.

5.3.1 | Selection operator

In order to select parents for recombination, mutation, and reward/penalize operators, one of the common methods such as fitness-based selection, rank-based selection, etc. can be used.

5.3.2 | Recombination operator

For doing this operator, one of the common methods such as n-point crossover, uniform crossover, etc. can be used. For example, uniform crossover is described here. In this method, children (new solution vector or new object migrating automaton) are generated according to a random binary array called mask. In this way, if a bit in mask array is equal to 1, the corresponding action in child one is selected from parent one and the corresponding action in child two is selected from parent two. In case that a bit in mask array is equal to 0, the corresponding action in child one is selected from parent two and the corresponding action in child two is selected from parent one. In Figure 13, this operator is illustrated.

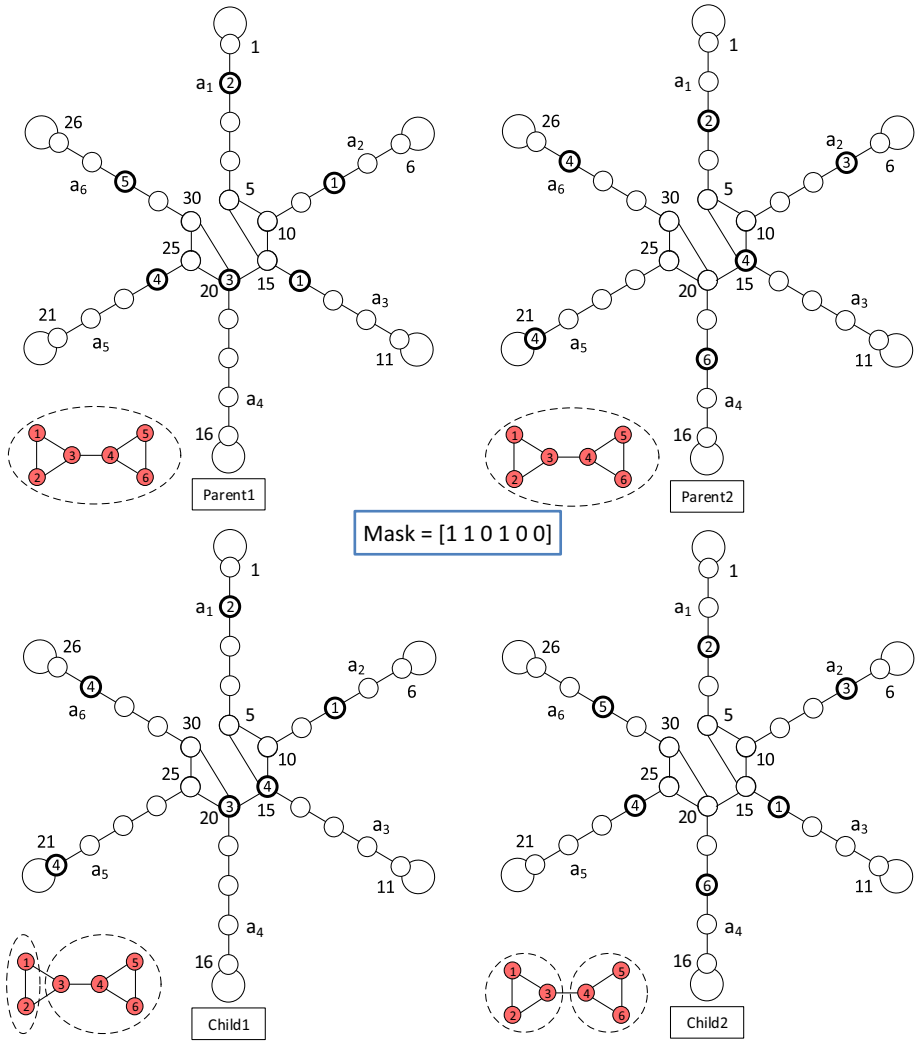


FIGURE 13 The way of doing uniform crossover. According to the mask array, in child1, a_1, a_2, and a_4 actions have been selected from parent1 and a_3, a_5, and a_6 actions have been selected from parent2. Similarly, in child2, a_1, a_2, and a_4 actions have been selected from parent2 and a_3, a_5, and a_6 actions have been selected from parent1 [Color figure can be viewed at wileyonlinelibrary.com]

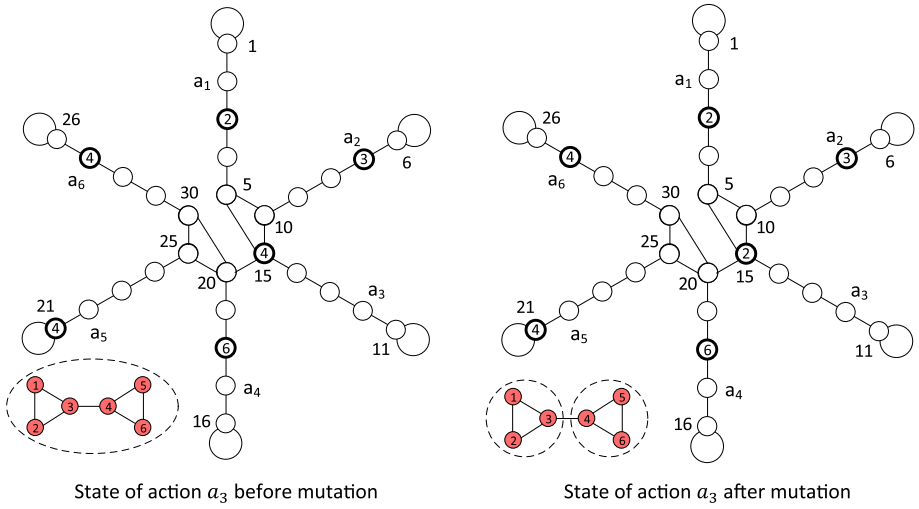


FIGURE 14 The way of doing mutation operator. Neighbor a_2 (2) is selected as object of action a_3 [Color figure can be viewed at wileyonlinelibrary.com]

5.3.3 | Mutation operator

For doing this operator, an action (gene) is randomly selected and its object is replaced. In other words, another neighbor of that action is selected as an object of that action. In Figure 14, this operator is illustrated.

5.3.4 | Reward and penalty operator

Since in the GAOMA-net algorithm, each chromosome is denoted by a learning automaton of object migrating type, in each of the automata, after evaluating the randomly selected action (gene), that action is rewarded or penalized. As a result of rewarding or penalizing of an action, the state of the object of that action in the set of states of related action is changed or object of that action is replaced. Depending on the learning automaton type, the way of rewarding and penalizing an action is different, which is fully described in Sections 4.1.1 to 4.1.4.

5.4 | Framework of the GAOMA-net algorithm

In GAOMA-net algorithm, after the initial population is generated, in each cycle, the parents are selected and genetic operators (recombination, mutation, and reward/penalize) are applied on them to generate a new generation of solutions. Then, the new solutions replace the previous population. This process is repeated until the obtained best community structure does not change in several consecutive cycles. At the end, the solution vector is formed by using the object of actions of the best object migrating automaton in the population. Then, after decoding, it is converted into the membership vector, which indicates the community structure. In Figure 15, the main framework of the GAOMA-net algorithm for detecting community structure in complex networks is given.

Algorithm GAOMA-net

Inputs
$G = (V, E)$: The graph of the network
PS: Population size
$EvaluationMethod$: Evaluation method
$SelectionMethod$: Parent selection method
$crossoverMethod$: Crossover operator type
P_c: Crossover rate
$mutationMethod$: Mutation operator type
P_m: Mutation rate
$ReplacementMethod$: Replacement operator type
$MaxGen$: Maximum number of generations
$automatonType$: Object Migrating Automaton type
md: Actions memory depth.

Output
S: The community structure

Initialization
CreateInitialPopulation();
Evaluate();

repeat
SelectParents();
Crossover();
Mutation();
Reward/Penelaize();
Replacement();
Evaluate();
until the best community structure in the population remains fixed in some consecutive cycles **OR** $MaxGen$ is reached
return best community structure of the current generation

FIGURE 15 Main framework of the GAOMA-net algorithm

6 | EXPERIMENTAL RESULTS AND DISCUSSION

The proposed algorithms are tested on both real-world and synthetic benchmark networks. The performance of the proposed algorithms are compared with other community structure detection algorithms including CNM,[34] CONCLUDE,[66] COPRA,[61] GA-net,[45] ICLA-net,[67] Infomap,[68] LPA,[7] LabelRank,[60] Louvain,[36] MOGA-net,[50,51] and SLPA.[59]

6.1 | Experimental environment and parameter settings

The proposed algorithms are implemented by MATLAB R2016a and are run on a PC with a dual-core 3 GHz processor and 4 GB of RAM. In GAOMA-net, GA-net, and MOGA-net algorithms, population size is set to 100, recombination rate is set to 0.8, mutation rate is set to 0.2, and the maximum number of generations is set to 100. In GA-net and MOGA-net algorithms, exponent r is set to 1.5. In GAOMA-net algorithm, 20% of the initial population is generated by MEM-net algorithm and the rest are generated randomly. In OMA-net and GAOMA-net algorithms, object migrating automaton based on Tsetlin automaton with depth 5 is used. In OMA-net algorithm, maximum number of iterations is set to 500. In MEM-net algorithm, nodes

memory depth is set to 5 and maximum number of iterations is set 50. In ICLA-net algorithm, reward parameter is set to 0.1; initial number of iterations is set 20 and maximum number of iterations is set to 500. In LabelRank algorithm, inflation parameter is set to 4; cutoff threshold is set to 0.1; conditional update coefficient is set to 0.7, and maximum number of iterations is set to 50.

6.2 | Evaluation criteria

For evaluating the performance of different community structure detection algorithms, criteria of modularity (Q),[8] normalized mutual information (NMI),[69] and adjusted Rand index (ARI)[70] are used. Q is an internal criterion, while NMI and ARI are external criteria. The external criteria are used when the ground truth community structure is known.

Modularity criterion is defined as follows:

$$Q = \frac{1}{2m} \sum_{v,w \in V} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w), \tag{4}$$

where, A is the adjacency matrix of the network; $m = \frac{1}{2}\sum_{vw} A_{vw}$ is the total number of network edges; k_v is the degree of node v; c_v is the community of node v and $\delta(i, j) = 1$ if $i = j$, otherwise $\delta(i, j) = 0$. The value of Q lies in the range $[-0.5, 1]$. Greater values of Q indicate that there are more connections within the communities than the expected number in the null model.

NMI criterion operates according to a matrix called confusion matrix. The rows of the confusion matrix correspond to the ground truth communities and its columns correspond to the found communities. Element n_{ij} of the confusion matrix indicates the number of nodes which are common in the i^{th} ground truth community and j^{th} found community. Let A and B are two different partitions of a network with n nodes, c_A indicates the number of communities of partition A and $n_{i.}(n_{.j})$ denotes the sum of the elements of the i^{th} row (j^{th} column) of the confusion matrix. Based on information theory, NMI criterion is defined as follows:

$$\text{NMI}(A, B) = \frac{-2 \sum_{i=1}^{c_A} \sum_{j=1}^{c_B} n_{ij} \log\left(\frac{n_{ij} n}{n_{i.} n_{.j}}\right)}{\sum_{i=1}^{c_A} n_{i.} \log\left(\frac{n_{i.}}{n}\right) + \sum_{j=1}^{c_B} n_{.j} \log\left(\frac{n_{.j}}{n}\right)}. \tag{5}$$

The value of NMI lies in the range [0, 1]. Greater values of NMI indicate that the found communities are more matched with ground truth communities.

ARI is the corrected for chance version of the Rand index (RI). Although, the RI may only yield a value in the range [0, 1], the ARI can yield negative values, if the index is less than the expected index. Greater values of ARI indicate that the found communities are more matched with ground truth communities. ARI operates according to a matrix called contingency matrix. This matrix is the same as the confusion matrix in NMI. ARI is defined as follows:

$$\text{ARI}(A, B) = \frac{\sum_{ij} \binom{n_{ij}}{2} - \sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2} / \binom{n}{2}}{\frac{1}{2}\left(\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right) - \sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2} / \binom{n}{2}}. \tag{6}$$

6.3 | Real-world networks

The proposed algorithms (MEM-net, OMA-net, and GAOMA-net) along with the algorithms for comparison (CNM, CONCLUDE, COPRA, GA-net, ICLA-net, Infomap, LPA, LabelRank, Louvain, MOGA-net, and SLPA) are tested with several real-world networks, which have been widely used in the literature. General information of these networks is given in Table 2.

For the Karate, Dolphins, and Football networks, the ground truth community structure and an instance of the obtained community structure by the proposed GAOMA-net algorithm are shown in Figures 16,17, and 18, respectively. In all three networks, the modularity of the obtained community structure by the proposed GAOMA-net algorithm is larger than the modularity of the ground truth community structure. Because the ground truth community structure of other networks is unknown, such a comparison cannot be done for them.

Since the ground truth community structure of most real-world networks is unknown, modularity criterion is used for measuring the quality of the obtained communities by different algorithms. The average performance of different algorithms on real-world networks is given in Table 3. The largest obtained modularity value for each network is highlighted.

TABLE 2 General information of the real-world networks

Network	Description	Node	Edge	Avg. degree	Avg. local C.C.
Karate	Zachary"s karate club	34	78	4.5882	0.5706
Contiguous	USA states contiguous network	49	107	4.3673	0.4967
Dolphins	Dolphins social network	62	159	5.1290	0.2590
PolBooks	Books about US politics	105	441	8.4000	0.4875
AdjNoun	Copperfield word adjacencies network	112	425	7.5893	0.1728
Football	American college football network	115	613	10.6609	0.4032
Jazz	Jazz musicians" network	198	2742	27.697	0.6175
Email	URV email network	1133	5451	9.6222	0.2202
Facebook	EGO-Facebook friendship network	2888	2981	2.0644	0.0272
PowerGrid	US power grid network	4941	6594	2.6691	0.0801

Abbreviation: C.C., clustering coefficient.

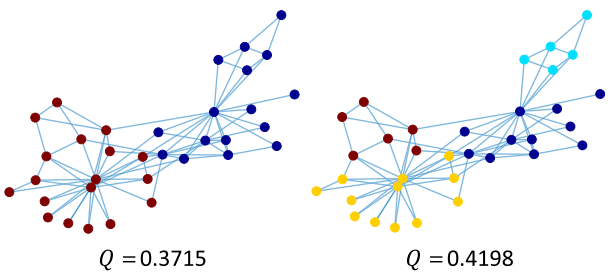


FIGURE 16 Karate network. (Left) Ground truth community structure. (Right) An instance of obtained community structure by the proposed GAOMA-net algorithm [Color figure can be viewed at wileyonlinelibrary.com]

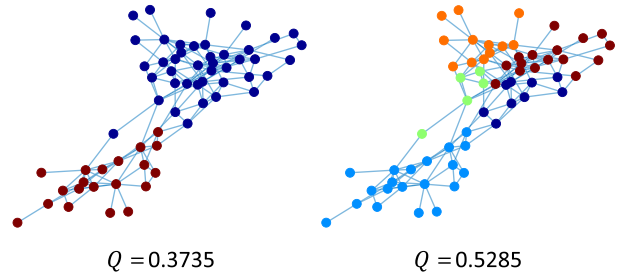


FIGURE 17 Dolphins network. (Left) Ground truth community structure. (Right) An instance of obtained community structure by the proposed GAOMA-net algorithm [Color figure can be viewed at wileyonlinelibrary.com]

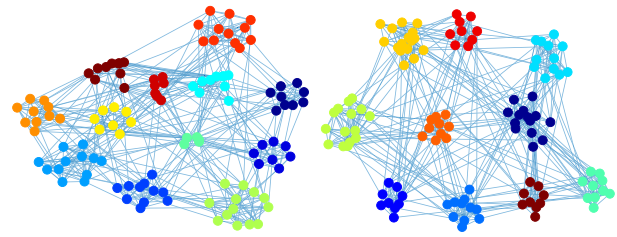


FIGURE 18 Football network. (Left) Ground truth community structure. (right) An instance of obtained community structure by the proposed GAOMA-net algorithm [Color figure can be viewed at wileyonlinelibrary.com]

As shown in Table 3, proposed GAOMA-net algorithm has found the community structure with largest modularity value on all networks except for Email and PowerGrid networks. In these networks, Louvain algorithm found the community structure with largest modularity value. The reason for the smaller modularity values of GAOMA-net algorithm on Email and PowerGrid networks than Louvain algorithm is that, in GAOMA-net algorithm, termination condition (reach to the maximum number of predefined generations) is satisfied before the convergence of the algorithm. By increasing the maximum number of allowed generations, the performance of GAOMA-net algorithm on Email and PowerGrid networks is improved.

Given the number of networks in which algorithms were able to find the community structure with largest modularity value, GAOMA-net algorithm has gained rank 1 (This algorithm has found the community structure with largest modularity value in eight out of 10 networks). In addition, GAOMA-net algorithm significantly outperforms than GA-net and MOGA-net genetic-based algorithms. Note that the MOGA-net algorithm is not applicable on large networks such as Email, Facebook, and PowerGrid due to the long running time. According to the results, it can be mentioned that proposed GAOMA-net algorithm is effective and efficient for detecting community structure in real-world networks.

6.4 | Synthetic networks

For evaluating the accuracy of the proposed algorithms, they are tested on different synthetic benchmark networks with known community structures.

6.4.1 | Evaluating with GN benchmark networks

For evaluating the accuracy of the proposed algorithms, experiments are first carried out on Girvan-Newman (GN) benchmark networks. GN benchmark networks were introduced by

TABLE 3 The average modularity obtained by different algorithms on real-world networks

	Algorithms													
Network	**CNM**	**CONCLUDE**	**COPRA**	**GA-net**	**ICLA-net**	**Infomap**	**LabelRank**	**Louvain**	**LPA**	**MEM-net**	**MOGA-net**	**OMA-net**	**SLPA**	**GAOMA-net**
Karate	0.3807	0.3717	0.1338	0.4025	0.4077	0.4020	0.3715	0.4190	0.3165	0.3735	0.4156	0.2755	0.1597	0.4198
Contiguous	0.5614	0.5150	0.4774	0.5174	0.5807	0.5937	0.4619	0.5850	0.4959	0.5291	0.5948	0.5319	0.4838	0.5970
Dolphins	0.4955	0.4775	0.3767	0.4293	0.4958	0.5186	0.3699	0.5198	0.4795	0.4565	0.5057	0.4337	0.4086	0.5280
PolBooks	0.5020	0.4382	0.4851	0.4715	0.5176	0.5266	0.4572	0.5260	0.4879	0.5041	0.5223	0.4464	0.4988	0.5272
AdjNoun	0.2953	0.2232	−0.0116	0.2090	0.1864	0.0445	0.0000	0.2949	0.0000	0.0014	0.1847	0.1745	0.0055	0.3010
Football	0.5773	0.5599	0.5919	0.5365	0.5590	0.6005	0.5918	0.6014	0.5880	0.5766	0.5740	0.4434	0.5637	0.6046
Jazz	0.4389	0.3950	0.3684	0.3322	0.3863	0.4421	0.4369	0.4440	0.3307	0.3163	0.3206	0.2740	0.2335	0.4440
Email	0.5130	0.4075	0.3589	0.2663	0.4006	0.5342	0.0029	0.5674	0.1852	0.4425	—	0.2839	0.4737	0.5625
Facebook	0.8087	0.6381	−0.0429	0.7931	0.8084	0.7981	0.7653	0.8086	0.7946	0.7947	—	0.7907	0.7966	0.8087
PowerGrid	0.9341	0.7276	0.2887	0.6371	0.7230	0.8308	0.6471	0.9356	0.8026	0.5827	—	0.6778	0.7930	0.8436

Girvan and Newman.[23] GN benchmark network contains 128 nodes, which are divided into four communities with 32 nodes within each community. The degree of each node in the network is exactly equal to $Z_{in} + Z_{out} = 16$. Each node has Z_{in} connections with nodes of its own community and Z_{out} connections with other network nodes. Parameter Z_{out} is used for controling the community structure in the network. Smaller value of Z_{out} indicates that the community structure in the network is more significant. When $Z_{out} < 8$, the network has relatively strong communities and when $Z_{out} > 8$, the communities will be relatively unclear. Parameter Z_{out} is changed from 0 to 8 and the proposed algorithms along with the algorithms for comparison are tested. Since the built-in communities in GN benchmark networks are already known, NMI and ARI criteria are used for evaluating the performance of different algorithms. Figures 19 and 20 show the average performance of different algorithms on GN benchmark networks. Each data point is the average of 10 runs.

Figure 19 shows the average NMI obtained from different algorithms on GN benchmark networks. As shown in Figure 19, when $Z_{out} < 3$, almost all algorithms except, CONCLUDE, LabelRank, OMA-net, and SLPA, can find the community structure corresponding to the correct partition (NMI ≈ 1). By increasing Z_{out}, the community structure in the network becomes unclear, resulting in that the NMI of all algorithms begins to decrease. When $Z_{out} > 4$, the NMI of the most algorithms decreased sharply so that when $Z_{out} \leq 6$, only three algorithms, GAOMA-net, Louvain, and Infomap, can find the community structure corresponding to the correct partition.

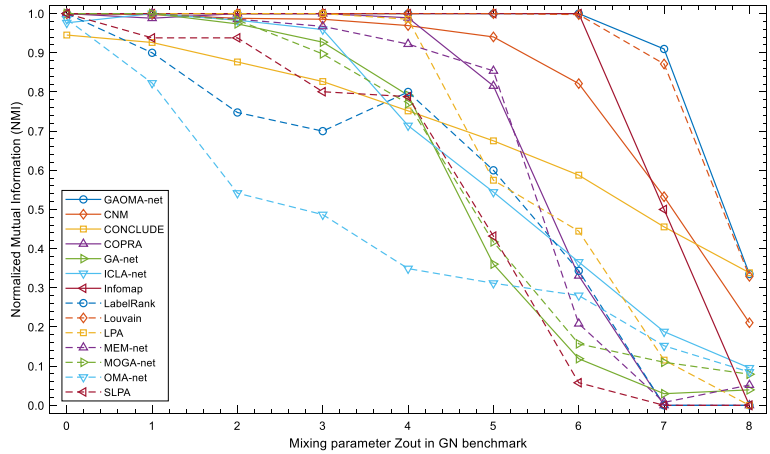


FIGURE 19 The average NMI obtained by different algorithms on GN benchmark networks [Color figure can be viewed at wileyonlinelibrary.com]

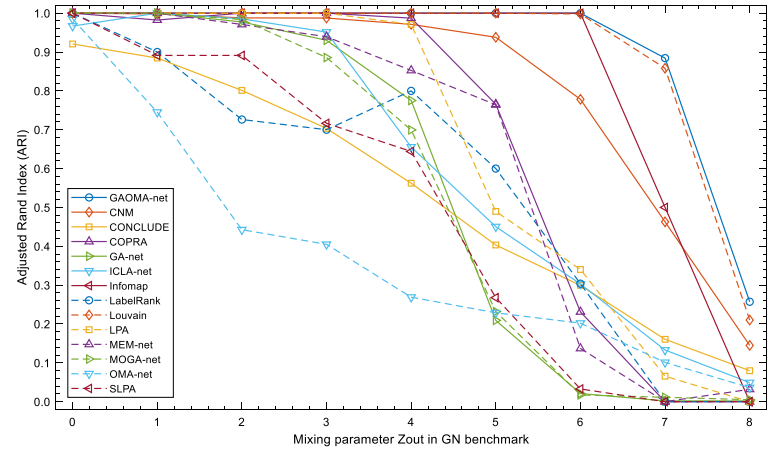


FIGURE 20 The average ARI obtained by different algorithms on GN benchmark networks [Color figure can be viewed at wileyonlinelibrary.com]

When $Z_{out} \geq 7$, GAOMA-net algorithm slightly outperforms than Louvain algorithm and significantly outperforms than all other algorithms. Figure 20 shows the average ARI obtained from different algorithms on GN benchmark networks. Similar descriptions can be taken for Figure 20.

We can see that the performance of the proposed GAOMA-net algorithm is in close proximity to that of the Louvain, and significantly outperforms than other algorithms. Moreover, it always gives excellent results until the mixing parameter Z_{out} reaches 6, beyond which its performance decreases sharply. According to the results, it can be mentioned that, proposed GAOMA-net algorithm has great performances on GN benchmark networks.

6.4.2 | Evaluating with LFR benchmark networks

GN benchmark networks do not have some important properties of the real-world networks, like the power-law distribution of node degrees and community sizes. Therefore, Lancichinetti et al.[71] proposed the LFR benchmark networks, which are more consistent with the properties of real-world networks. In Lancichinetti-Fortunato-Radicchi (LFR) benchmark networks, both node degrees and community sizes follow the power-law distribution with exponents τ_1 and τ_2, respectively. The significance of the community structure depends on the mixing parameter μ, which indicates the average connections of each node to other communities. The benchmark network with smaller mixing parameter μ has more significant community structure.

The proposed algorithms along with the algorithms for comparison are tested with LFR benchmark networks. In our experiments, the network size is set to 500 and the power-law exponents τ_1 and τ_2 are set to 2 and 1, respectively. The average and maximum degree of nodes is set to 7 and 25, respectively. The minimum and maximum size of communities is set to 40 and 80, respectively. Again, NMI and ARI criteria are used for evaluating the performance of different algorithms. Figures 21 and 22 show the average performance of different algorithms on LFR benchmark networks. Each data point is the average of 10 runs.

Figure 21 shows the average NMI obtained from different algorithms on LFR benchmark networks. As shown in Figure 21, when $\mu < 0.10$, a few algorithms can find the community structure corresponding to the correct partition. By increasing μ, the NMI of the all algorithms, except GAOMA-net, begins to decrease so that when $\mu \leq 0.25$, only GAOMA-net algorithm can find the community structure corresponding to the correct partition. When $\mu > 0.25$, the NMI of the

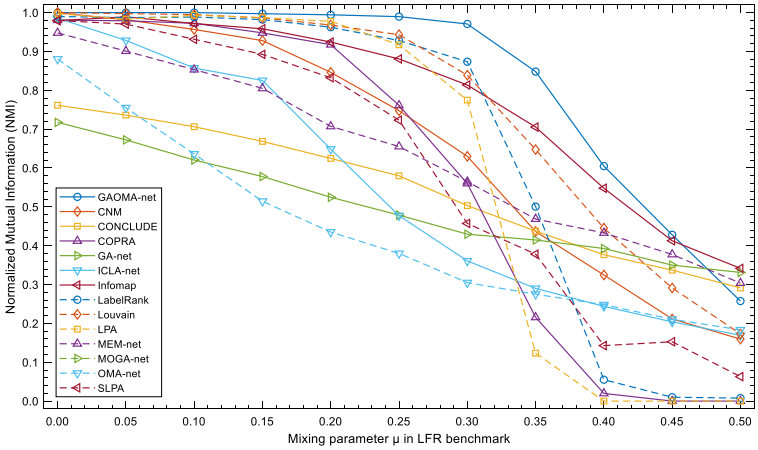


FIGURE 21 The average NMI obtained by different algorithms on LFR benchmark networks [Color figure can be viewed at wileyonlinelibrary.com]

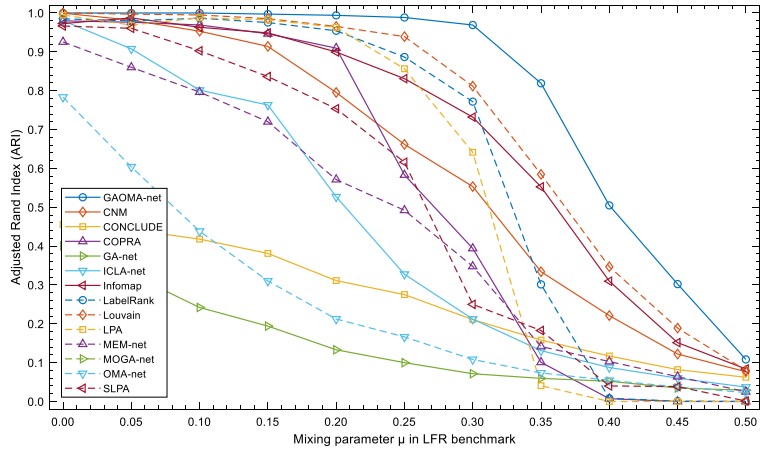


FIGURE 22 The average ARI obtained by different algorithms on LFR benchmark networks [Color figure can be viewed at wileyonlinelibrary.com]

most algorithms decreased sharply and GAOMA-net algorithm significantly outperforms than all algorithms. Figure 22 shows the average ARI obtained from different algorithms on LFR benchmark networks. Similar descriptions can be taken for Figure 22.

We can see that when $\mu > 0.20$, GAOMA-net algorithm significantly outperforms than all algorithms. Moreover, it always gives excellent results until the mixing parameter μ reaches 0.30, beyond which its performance decreases sharply. According to the results, it can be mentioned that, GAOMA-net algorithm has great performances on LFR benchmark networks.

6.4.3 | Evaluating with large-scale benchmark networks

In this section, the accuracy of the GAOMA-net algorithm is evaluated with two large-scale networks. The first network has 12 000 nodes and 57 770 edges, and the second network has 15 000 nodes and 75 290 edges. The average performance of different algorithms on these networks is given in Table 4. Given that the values of the ARI criterion are very close to the NMI criterion, in Table 4, instead of ARI, two other criteria purity (P)[72] and RI[73] are given. The largest obtained value for each network is highlighted. As shown in Table 4, the proposed GAOMA-net algorithm on both networks significantly outperforms than other algorithms and slightly outperforms than Infomap algorithm. In Network1, in terms of criterion RI, Infomap algorithm is slightly better than the GAOMA-net algorithm. According to the results, it can be mentioned that proposed GAOMA-net algorithm is efficient for detecting community structure in large-scale networks. Note that the ICLA-net and MOGA-net algorithms are not applicable on these large-scale networks due to the very long running time.

6.5 | Scalability analysis

In this section, the scalability of the GAOMA-net algorithm is investigated in terms of execution time, modularity, and NMI. Given that the GAOMA-net algorithm is a GA, its scalability is compared only with GA-net and MOGA-net algorithms. In Figures 23,24, and 25, average time required for the convergence, modularity, and NMI of these algorithms is shown for the networks with different sizes, respectively. As shown in Figure 23, for small networks ($n \leq 200$), the execution time of the GA-net algorithm is lower than the GAOMA-net algorithm. However, with

TABLE 4 The average NMI, purity, and Rand index obtained by different algorithms on two large-scale networks

	Network1 (12 000 nodes and 57 770 edges)			Network2 (15 000 nodes and 75 290 edges)		
Algorithm	**NMI**	**P**	**RI**	**NMI**	**P**	**RI**
GAOMA-net	0.5287	0.6909	0.9734	0.5375	0.6150	0.9815
CNM	0.1814	0.1468	0.9041	0.1037	0.0754	0.9110
CONCLUDE	0.3837	0.4239	0.9707	0.3438	0.3162	0.9780
COPRA	0.0010	0.0415	0.0276	0.0004	0.0359	0.0189
GA-net	0.4373	0.4676	0.9723	0.4286	0.3858	0.9801
ICLA-net[a]	—	—	—	—	—	—
Infomap	0.5227	0.5867	0.9740	0.4968	0.4613	0.9815
LabelRank	0.0603	0.0654	0.1028	0.4817	0.5248	0.9799
Louvain	0.2710	0.2930	0.9388	0.1442	0.1264	0.9394
LPA	0.2715	0.3249	0.6493	0.3418	0.2991	0.8830
MEM-net	0.3795	0.3799	0.9701	0.3639	0.3010	0.9782
MOGA-net[a]	—	—	—	—	—	—
OMA-net	0.3885	0.3913	0.9705	0.3795	0.3171	0.9790
SLPA	0.3853	0.3994	0.9605	0.3620	0.2928	0.9722

Abbreviations: NMI, normalized mutual information; P, purity; RI, Rand index.

[a]These two algorithms are not applicable on these large-scale networks due to the very long running time.

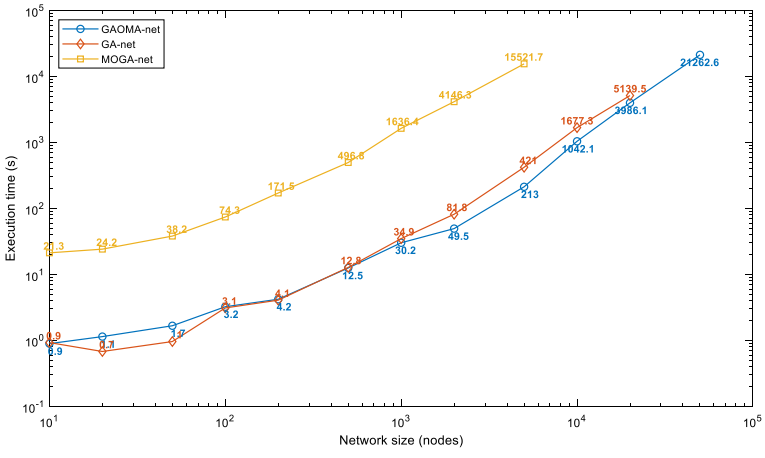


FIGURE 23 A log-log plot about a comparison of execution time between GAOMA-net, GA-net, and MOGA-net algorithms in benchmark networks with different sizes [Color figure can be viewed at wileyonlinelibrary.com]

increasing network size, the execution time of the GA-net algorithm increases gradually and becomes more than GAOMA-net algorithm. In addition, the execution time of the MOGA-net algorithm is far more than GAOMA-net and GA-net algorithms. The results in Figure 23 clearly show that for the networks with more than 5000 nodes, MOGA-net algorithm and for the networks with more than 20 000 nodes, GA-net algorithm cannot tackle it within a reasonable time. However, GAOMA-net algorithm can detect communities on the network with 50 000 nodes in a

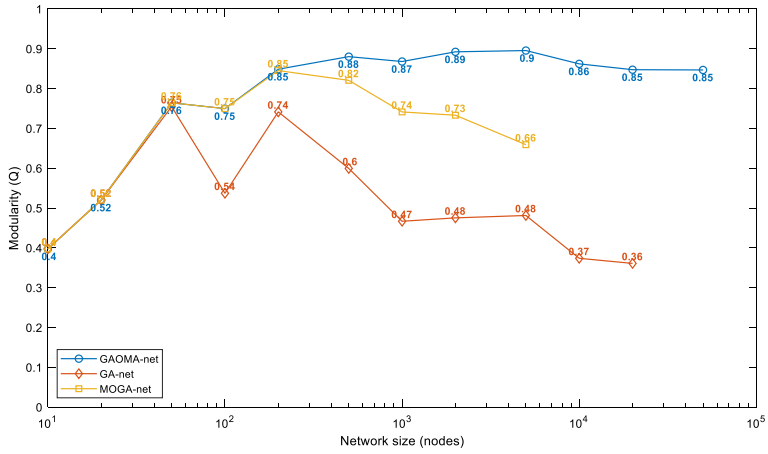


FIGURE 24 A log-log plot about a comparison of modularity between GAOMA-net, GA-net, and MOGA-net algorithms in benchmark networks with different sizes [Color figure can be viewed at wileyonlinelibrary.com]

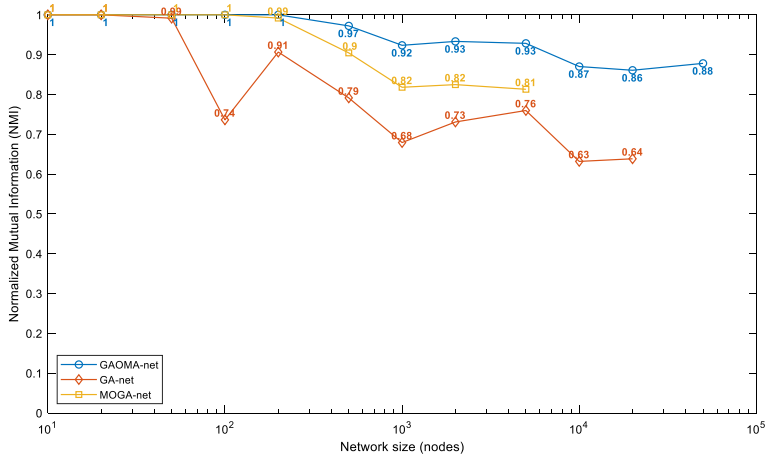


FIGURE 25 A log-log plot about a comparison of NMI between GAOMA-net, GA-net, and MOGA-net algorithms in benchmark networks with different sizes [Color figure can be viewed at wileyonlinelibrary.com]

reasonable time (21 262.6 seconds). As shown in Figures 24 and 25, for the networks with more than 200 nodes, GAOMA-net algorithm significantly outperforms than GA-net and MOGA-net algorithms. The results of Figures 23-25 demonstrate that the GAOMA-net algorithm is more scalable than GA-net and MOGA-net algorithms. For example, for the network with 5000 nodes, GAOMA-net algorithm converges in 213 seconds ($Q = 0.9$, NMI = 0.93), while GA-net and MOGA-net algorithms converge in 421 ($Q = 0.48$, NMI = 0.76), and 15 521.7 ($Q = 0.66$, NMI = 0.81) seconds, respectively. As another example, for the network with 50 000 nodes, GAOMA-net algorithm converges in 21 262.6 seconds ($Q = 0.85$, NMI = 0.88), while GA-net and MOGA-net algorithms are not applicable for this large-scale network due to the very long running time.

7 | DISCUSSION

One of the most important questions about the results of Sections 6.3 and 6.4 is that what makes the proposed GAOMA-net algorithm outperforms than other algorithms and even GA-net (single-objective GA) and MOGA-net (multiobjective GA) algorithms. In this section, this important question is answered. As explained in Section 5, in the GAOMA-net algorithm, the GA and

OMA are integrated as a single framework. In the GAOMA-net algorithm, common coding for the chromosomes is not used, but each chromosome is denoted by a learning automaton of object migrating type. As a result, in the proposed GA, in addition to the recombination and mutation operators, an additional operator (reward/penalize) is also considered. Reward/Penalize (learning) makes the chromosomes adopted with the environment and avoids falling algorithm in the local optimum. In other words, the learning capability of chromosomes largely overcomes the premature convergence problem of the proposed GA. In addition, in the proposed GA, a portion of the initial population is generated using MEM-net algorithm. This will accelerate the convergence of the algorithm. In brief, it can be said that overcoming the problem of premature convergence and smart generation of a portion of the initial population are the main reasons for the efficiency of the proposed GAOMA-net algorithm.

8 | CONCLUSION

In this article, three different algorithms, namely, MEM-net, OMA-net, and GAOMA-net, are proposed for detecting community structure in complex networks. The proposed algorithms do not need prior information about the network and dynamically identify communities in the networks. In GAOMA-net algorithm, which is the main proposed algorithm of this article, GA and OMA are integrated as a single framework and MEM-net algorithm is used as a heuristic to generate a portion of the initial population. This combination will accelerate the convergence of the algorithm and avoid falling it in the local optimum. The proposed algorithms along with CNM, CONCLUDE, COPRA, GA-net, ICLA-net, Infomap, LPA, LabelRank, Louvain, MOGA-net, and SLPA algorithms are tested on both real-world and synthetic benchmark networks. The experimental results indicate that GAOMA-net algorithm is efficient for detecting community structure in complex networks.

In this article, unweighted networks and disjoint communities are considered. However, most real-world networks are weighted and they have overlapping community structure. Therefore, generalization of the GAOMA-net algorithm for detecting community structure in different types of networks (weighted, signed, bipartite, etc.) and overlapping community structure detection will be the focus of our work in the future.

ORCID

Bagher Zarei https://orcid.org/0000-0002-3535-5093
Mohammad Reza Meybodi https://orcid.org/0000-0003-3775-5565

REFERENCES

1. Newman M. *Networks: An Introduction*. Oxford, New York: Oxford University Press; 2010.
2. Costa L d F, Rodrigues FA, Travieso G, Villas Boas PR. Characterization of complex networks: a survey of measurements. *Adv Phys*. 2007;56(1):167-242.
3. Latora V, Nicosia V, Russo G. *Complex Networks: Principles, Methods and Applications*. Cambridge; New York, NY: Cambridge University Press; 2017.
4. Fortunato S. Community detection in graphs. *Phys Rep*. 2010;486(3–5):75-174.
5. Fortunato S, Hric D. Community detection in networks: a user guide. *Phys Rep*. 2016;659:1-44.
6. Radicchi F, Castellano C, Cecconi F, Loreto V, Parisi D. Defining and identifying communities in networks. *Proc Natl Acad Sci*. 2004;101(9):2658-2663.
7. Raghavan UN, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks. *Phys Rev E*. 2007;76(3):036106.

8. Newman ME, Girvan M. Finding and evaluating community structure in networks. *Phys Rev E*. 2004;69(2):026113.
9. Brandes U, Delling D, Gaertler M, et al. On finding graph clusterings with maximum modularity. Paper presented at: International Workshop on Graph-Theoretic Concepts in Computer Science; 2007; Springer:121-132.
10. Oommen BJ, Ma DCY. Deterministic learning automata solutions to the equipartitioning problem. *IEEE Trans Comput*. 1988;37(1):2-13.
11. Park Y, Song M. A genetic algorithm for clustering problems. Paper presented at: Proceedings of the Third Annual Conference on Genetic Programming; 1998:568-575.
12. Newman ME. Detecting community structure in networks. *Eur Phys J B*. 2004;38(2):321-330.
13. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. *Science*. 1983;220(4598):671-680.
14. Guimera R, Sales-Pardo M, Amaral LAN. Modularity from fluctuations in random graphs and complex networks. *Phys Rev E*. 2004;70(2):025101.
15. Boettcher S, Percus AG. Optimization with extremal dynamics. *Complexity*. 2002;8(2):57-62.
16. Duch J, Arenas A. Community detection in complex networks using extremal optimization. *Phys Rev E*. 2005;72(2):027104.
17. Kernighan BW, Lin S. An efficient heuristic procedure for partitioning graphs. *Bell Syst Tech J*. 1970;49(2):291-307.
18. Barnes ER. An algorithm for partitioning the nodes of a graph. *SIAM J Algebraic Discrete Methods*. 1982;3(4):541-550.
19. Flake GW, Lawrence S, Giles CL. Efficient identification of web communities. Paper presented at: KDD; 2000:150-160.
20. Flake GW, Lawrence S, Giles CL, Coetzee FM. Self-organization and identification of web communities. *Computer*. 2002;35(3):66-71.
21. Didwania A, Narmawala Z. A comparative study of various community detection algorithms in the mobile social network. Paper presented at: 2015 5th Nirma University International Conference on Engineering (NUiCONE); 2015; IEEE:1-6.
22. Dhumal A, Kamde P. Survey on community detection in online social networks. *Int J Comput Appl*. 2015;121(9):35-41.
23. Girvan M, Newman ME. Community structure in social and biological networks. *Proc Natl Acad Sci*. 2002;99(12):7821-7826.
24. Hastie T, Tibshirani R, Friedman J, Franklin J. The elements of statistical learning: data mining, inference and prediction. *Math Intell*. 2005;27(2):83-85.
25. MacQueen J. Some methods for classification and analysis of multivariate observations. Paper presented at: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability; 1967; Oakland, CA:281-297.
26. Bezdek JC. *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York, NY: Plenum Press; 2013.
27. Dunn JC. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters; 1973.
28. Donath WE, Hoffman AJ. Lower bounds for the partitioning of graphs. Paper presented at: Selected Papers of Alan J Hoffman: with Commentary; 2003; World Scientific:437-442.
29. Fiedler M. Algebraic connectivity of graphs. *Czech Math J*. 1973;23(2):298-305.
30. Shi J, Malik J. Normalized cuts and image segmentation. *IEEE Trans Pattern Anal Mach Intell*. 2000;22(8):888-905.
31. Ng AY, Jordan MI, Weiss Y. On spectral clustering: analysis and an algorithm. *Advances in Neural Information Processing Systems*; Cambridge, MA: MIT Press; 2002:849-856.
32. Fortunato S, Latora V, Marchiori M. Method to find community structures based on information centrality. *Phys Rev E*. 2004;70(5):056104.
33. Newman ME. Fast algorithm for detecting community structure in networks. *Phys Rev E*. 2004;69(6):066133.
34. Clauset A, Newman ME, Moore C. Finding community structure in very large networks. *Phys Rev E*. 2004;70(6):066111.
35. Wakita K, Tsurumi T. Finding community structure in mega-scale social networks. Paper presented at: Proceedings of the 16th International Conference on World Wide Web; 2007; ACM:1275-1276.

36. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. *J Stat Mech Theory Exp*. 2008;2008(10):P10008.
37. Wadhwa P, Bhatia M. Community detection approaches in real world networks: a survey and classification. *Int J Virtual Commun Soc Netw*. 2014;6(1):35-51.
38. Khan BS, Niazi MA. Network community detection: a review and visual survey; 2017. arXiv:00977.
39. Newman ME. Finding community structure in networks using the eigenvectors of matrices. *Phys Rev E*. 2006;74(3):036104.
40. Newman ME. Modularity and community structure in networks. *Proc Natl Acad Sci*. 2006;103(23):8577-8582.
41. Richardson T, Mucha PJ, Porter MA. Spectral tripartitioning of networks. *Phys Rev E*. 2009;80(3):036111.
42. White S, Smyth P. A spectral clustering approach to finding communities in graphs. Paper presented at: Proceedings of the 2005 SIAM International Conference on data mining; 2005; SIAM:274-285.
43. Tasgin M, Bingol H. Community detection in complex networks using genetic algorithm; 2006. arXiv preprint cond-mat/0604419.
44. Tasgin M, Herdagdelen A, Bingol H. Community detection in complex networks using genetic algorithms; 2007. arXiv preprint.
45. Pizzuti C. GA-net: a genetic algorithm for community detection in social networks.Paper presented at: International Conference on Parallel Problem Solving from Nature; 2008; Springer:1081-1090.
46. Shi C, Wang Y, Wu B, Zhong C. A new genetic algorithm for community detection. Paper presented at: International Conference on Complex Sciences; 2009; Springer:1298-1309.
47. Gong M, Fu B, Jiao L, Du H. Memetic algorithm for community detection in networks. *Phys Rev E*. 2011;84(5):056101.
48. Gach O, Hao J-K. A memetic algorithm for community detection in complex networks. Paper presented at: International Conference on Parallel Problem Solving from Nature; 2012; Springer:327-336.
49. Ma L, Gong M, Liu J, Cai Q, Jiao L. Multi-level learning based memetic algorithm for community detection. *Appl Soft Comput*. 2014;19:121-133.
50. Pizzuti C. A multi-objective genetic algorithm for community detection in networks. Paper presented at: 2009 21st IEEE International Conference on Tools with Artificial Intelligence; 2009; IEEE:379-386.
51. Pizzuti C. A multiobjective genetic algorithm to find communities in complex networks. *IEEE Trans Evol Comput*. 2012;16(3):418-430.
52. Gong M, Ma L, Zhang Q, Jiao L. Community detection in networks by using multiobjective evolutionary algorithm with decomposition. *Phys A Stat Mech Appl*. 2012;391(15):4050-4060.
53. Gong M, Cai Q, Chen X, Ma L. Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition. *IEEE Trans Evol Comput*. 2014;18(1):82-97.
54. Amelio A, Pizzuti C. Community mining in signed networks: a multiobjective approach. Paper presented at: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining; 2013; ACM:95-99.
55. Amelio A, Pizzuti C. An evolutionary and local refinement approach for community detection in signed networks. *Int J Artif Intell Tools*. 2016;25(4):1650021.
56. Liu C, Liu J, Jiang Z. A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks. *IEEE Trans Cybern*. 2014;44(12):2274-2287.
57. Rahimi S, Abdollahpouri A, Moradi P. A multi-objective particle swarm optimization algorithm for community detection in complex networks. *Swarm Evol Comput*. 2018;39:297-309.
58. Shi C, Yan Z, Cai Y, Wu B. Multi-objective community detection in complex networks. *Appl Soft Comput*. 2012;12(2):850-859.
59. Xie J, Szymanski K, Liu X. SLPA: uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. Paper presented at: 2011 IEEE 11th International Conference on data mining workshops; 2011; IEEE:344-349.
60. Xie J, Szymanski BK. LabelRank: a stabilized label propagation algorithm for community detection in networks. Paper presented at: 2013 IEEE 2nd Network Science Workshop (NSW); 2013; IEEE:138-143.
61. Gregory S. Finding overlapping communities in networks by label propagation. *New J Phys*. 2010;12(10):103018.
62. Li Z, Zhang S, Wang R-S, Zhang X-S, Chen L. Quantitative function for community detection. *Phys Rev E*. 2008;77(3):036109.

63. Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput*. 2002;6(2):182-197.
64. Zhang Q, Li H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans Evol Comput*. 2007;11(6):712-731.
65. Narendra KS, Thathachar MA. *Learning Automata: An Introduction*. Mineola, NY: Dover Publications; 2012.
66. De Meo P, Ferrara E, Fiumara G, Provetti A. Mixing local and global information for community detection in large networks. *J Comput Syst Sci*. 2014;80(1):72-87.
67. Zhao Y, Jiang W, Li S, Ma Y, Su G, Lin X. A cellular learning automata based algorithm for detecting community structure in complex networks. *Neurocomputing*. 2015;151:1216-1226.
68. Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. *Proc Natl Acad Sci*. 2008;105(4):1118-1123.
69. Danon L, Diaz-Guilera A, Duch J, Arenas A. Comparing community structure identification. *J Stat Mech Theory Exp*. 2005;2005(09):P09008.
70. Hubert L, Arabie P. Comparing partitions. *J Classif*. 1985;2(1):193-218.
71. Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. *Phys Rev E*. 2008;78(4):046110.
72. Manning C, Raghavan P, Schütze H. Introduction to information retrieval. *Nat Lang Eng*. 2010;16(1):100-103.
73. Rand WM. Objective criteria for the evaluation of clustering methods. *J Am Stat Assoc*. 1971;66(336):846-850.

How to cite this article: Zarei B, Meybodi MR. Detecting community structure in complex networks using genetic algorithm based on object migrating automata. *Computational Intelligence*. 2020;1–37. https://doi.org/10.1111/coin.12273