Journal of Computational Science 24 (2018) 413–426

Contents lists available at ScienceDirect

Journal of Computational Science

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

A new cellular learning automata-based algorithm for community detection in complex social networks

Mohammad Mehdi Daliri Khomami [a,*], Alireza Rezvanian [a,b], Mohammad Reza Meybodi [a]

[a] Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
[b] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

ARTICLE INFO

Article history:
Received 27 February 2017
Received in revised form 12 October 2017
Accepted 14 October 2017
Available online 26 October 2017

Keywords:
Complex networks
Social network analysis
Community detection
Learning automata
Cellular learning automata

ABSTRACT

Community structure is one of the common and fundamental characteristics of many real-world networks such as information and social networks. The structure, function, evolution and dynamics of complex social networks can be explored through detecting the community structure of networks. In this paper, a new community detection algorithm based on cellular learning automata (CLA), in which a number of learning automata (LA) cooperate with each other, is proposed. The proposed algorithm taking advantage of irregular CLA finds a partial spanning tree and then forms the local communities on the found the partial spanning tree at each step in order to reduce the network size. As the proposed algorithm proceeds, LA are interacted with both local and global environments to modify the found communities that gradually yielded the near-optimal community structure of the network through the evolution of the CLA. To evaluate the efficiency of the proposed algorithm, several experiments are conducted on synthetic and real networks. Experimental results confirm the superiority and effectiveness of the proposed CLA-based algorithm in terms of various evaluation measures comprising Conductance, Modularity, Normalized Mutual Information, Purity and Rand-index.

1. Introduction

Many real-world phenomena and systems can be represented as a network with a set of nodes and links, where the node set indicates the entities and the link set indicates any type of connection between nodes. Studies on complex network models [1–4] and real-world networks [5] help researchers to understand and reveal the structure, dynamics and behavior of many real network systems, such as resilience [6], controllability [7] and network design [8]. Different universal characteristics comprising of small-world phenomena [2], small shortest path length [9], power-law degree distribution [10] and existing community structure [11] have been observed in many real-world networks. One of the most important and common features of networks, reported in many studies is community structure [5,11,12]. A community can be represented as a group of nodes with dense inner connections and relatively sparse connections outside the group. Detecting the structure of communities plays a significant role in the analysis of complex social networks because it contains useful information and important characteristics for analyzing the organization and function of systems. Furthermore, the community detection problem is a well-known problem in different domains such as sociology, biology and computer science, which all provide example of systems that can be modeled by networks or graphs.

A *Cellular Learning Automaton* (CLA) [13] is a hybrid model based on cellular automata and learning automata. A cellular automaton (CA) is composed of a set of uniform components called cells, each of which corresponds to a node of a regular network. CA is a discrete dynamical model, which is composed of a large number of independent and identical cells, each of which corresponds to a node of a regular network. Each cell selects a state from a set of states. In a cell, the previous states of the set of cells, including that cell itself, and its neighbors, determines the new state of that cell [14]. A learning automaton (LA) [15] is an abstract model for adaptive online decision making in unknown environments. An LA has a finite set of actions and its goal is to learn the optimal action through repeated interactions with the environment. An optimal action is an action with the highest probability of obtaining reward from the environment. The real power of an LA is obtained when a group of them

* Corresponding author.
E-mail addresses: m.daliri@aut.ac.ir (M.M.D. Khomami), a.rezvanian@aut.ac.ir (A. Rezvanian), mmeybodi@aut.ac.ir (M.R. Meybodi).

https://doi.org/10.1016/j.jocs.2017.10.009
1877-7503/

cooperate with each other. Beigy et al. [13] presented a new model obtained from the combination of learning automata and cellular automata, which they called cellular learning automata. It is a cellular automaton with one or more learning automata residing in each of its cells. Therefore, the cells of CLA have learning abilities and can also interact with each other in a cellular manner. Because of its learning capabilities and the cooperation among its elements, this model is considered superior to both CA and LA. Since CLA models inherit the distributed computation from the CA and the ability to learn in unknown environments from the LA, they can be useful in many problems relating to complex networks and systems [16].

In this paper, we propose a new cellular learning automata-based algorithm to detect communities in a network called CLACD. Briefly, the CLACD algorithm works as follows: the whole network is regarded as an isomorphic to an irregular cellular learning automaton (ICLA), where each node corresponds to a cell of the cellular learning automaton and each cell has a learning automaton assigned to it. Through interactions with both the global and local environments, a partial spanning tree finds that forms local communities, the CLA evolves and gradually detects the near-optimal community structure in the network.

The rest of the paper is organized as follows. In Section 2, an overview of related work on community detection algorithms is provided. In Section 3, learning automata and cellular learning automata are briefly introduced. The proposed cellular learning automata-based algorithm for community detection (CLACD) is described in Section 4. Section 5 shows the performance of the CLACD algorithm through computer simulation on synthetic and real networks and comparison with existing community detection algorithms. Finally, Section 6 gives the concluding remarks.

2. Related work

Since detecting community structure in a network is an important research area for analysis of real-networks, in recent years, many community detection algorithms have been developed to reveal community structures in complex social networks [5,17]. In this section, some existing community detection algorithms are briefly introduced. These algorithms can be classified into six different categories including spectral and clustering methods, hierarchical algorithms, modularity-based methods, model-based methods, local methods and feature-based methods [5]. Among all the types of community detection algorithms, there exist some stochastic models such as stochastic block model (SBM) for investigating communities. SBMs are model-based and probabilistic algorithms that explain the connection between pairs of nodes in the networks.

Hierarchical clustering algorithms are widely used techniques, which can be classified into two major classes, i.e., agglomerative and divisive. An effective agglomerative algorithm for clustering networks was first introduced by Girvan and Newman [18]. The Girvan and Newman (GN) algorithm computes the link betweenness centrality value of each link, then sorts and removes links with large betweenness values in an iterative manner, until the community structures of the networks are detected. The intuition behind the GN algorithm is used links correspond to bridges connecting different modules, whereas low-betweenness links are correspond to internal modules. Using of the edge clustering value, Rahman et al. [19] developed an agglomerative algorithm called HC-PIN, for discovering communities in protein interaction networks (PINs), which achieves promising results.

Most agglomerative algorithms select the best partition that maximizes a typical quality objective function. One of the best-known and the most widely used quality functions is the modularity Q metric, which is proposed by Newman and Girvan [20]. A partition with high modularity should be one where the number of links within communities is significantly higher than random expectation. Although using modularity optimization has achieved many promising results, it has been shown that the technique is subject to a serious problem known as a resolution limit [21]. For example, in the extreme case, modularity optimization algorithms fail to perform correctly for a network with several cliques connected by a single link. Li et al. [22] proposed a new quantitative function called bipartite partition density for evaluating the community partition of bipartite networks. They showed that bipartite partition density can overcome the problem of resolution limit of modularity. The authors also designed a heuristic and adapted label propagation algorithm (BiLPA) to optimize the bipartite partition density function for solving the community detection problem. Elyasi et al. [23] presented a two-phase community detection algorithm using Louvain method and applying a belonging matrix. In this algorithm belonging matrix determines how much a node belongs to a community and finally, some of the found communities are merged based on the modularity measure.

Raghavan et al. [24] proposed label propagation algorithm (LPA) with nearly linear time complexity for community detection. However, the performance of LPA is dependent to the update order of label information, whose assigned to the nodes of graph randomly in initialization. Hosseini et al. [25] proposes an improved label propagation algorithm called memory-based label propagation algorithm (MLPA) for finding community structure in social networks. In their algorithm, a simple memory element is designed for each node of graph and this element store the most frequent common adoption of labels iteratively. Le Martelot et al. [26] investigate a stability measure for the quality of partitioning, as an optimization criterion that exploits a Markov process view of networks. Another approach to detecting community structures in networks is the clique percolation method (CPM) [27]. Briefly, this algorithm works by finding all the maximal cliques in a network and then forming communities by merging cliques with common nodes.

Due to the wide spread of applications for community detection, evolutionary and swarm intelligence-based algorithms could be used for solving the community detection problem using an appropriate objective function. Compared to earlier algorithms, meta-heuristic optimization algorithms can effectively find a proper, high-quality solution within a reasonable period of time [28]. In addition, in [29] an ant colony-based algorithm is presented for discovering communities, where each node is modeled by an ant. To detect a community, a new fitness function is proposed, which updates the pheromone diffusion by considering the feedback signal, to investigate the interaction between ants. A multi-objective genetic algorithm to uncover community structure in complex networks, called MOGA-Net, was proposed in [30]. MOGA-Net optimizes two objective functions able to identify densely connected communities of nodes having sparse inter connections. MOGA-Net generates a set of network divisions at different hierarchical levels such that solutions at deeper levels are contained within solutions having a lower number of communities. The number of modules is automatically determined by the objective functions. In [31] a multi-objective evolutionary algorithm, called MOEA/D-Net, was proposed to investigate community detection in networks. MOEA/D-Net optimizes two conflicting objective functions made up from modularity density. MOEA/D-Net maximizes the density of internal degrees, and minimizes the density of external degrees concurrently. Recently, many community detection algorithms based on reinforcement learning have been developed.

Khomami et al. [32] investigated a distributed learning automaton-based algorithm for detecting communities in deterministic graphs. According to this algorithm, a set of learning

automata interact with each other in order to identify high-density local communities by updating the action probability vector of a network of cooperative learning automata. The improved version of this algorithm for community detection has also been developed by Ghamgosar et al. [33] using extended distributed learning automata. Mirsaleh et al. [34] suggested a new Michigan memetic algorithm called MLAMA-Net for solving the community detection in complex networks. MLAMA-Net is a distributed evolutionary algorithm in which each chromosome, without any prior information, locally evolves by evolutionary operators and improves by a local search. MLAMA-Net also introduces the priority concept to solve the resolution limit of modularity optimization in community detection problem.

Zhao et al. [35] proposed an open cellular learning automaton called CLA-Net for solving the community detection problem. Based on CLA-Net, the network is formulated with the aid of cellular learning automata. Then, the solution is constituted from current actions chosen by the learning automata in the network. The authors have shown experimentally that their algorithm can solve the problem of the resolution limit of modularity optimization. However, in this paper, a new version of cellular learning automata based algorithm is proposed for community detection. In the proposed algorithm, local communities are formed on a partial spanning tree of network found by the algorithm in order to reduce the network size. Furthermore, in the proposed community detection algorithm, two important types of information related to the network topology are used by ICLA: global information, which relies on the importance of nodes in the whole network and local information, which presents the similarities between nodes. It would be of great value to determine the two types of information in ICLA simultaneously, making it more accurate and scalable.

3. Theory of automata

In this section, we first outline cellular automata (CA) and learning automata (LA) in brief. Then, cellular learning automaton (CLA) as a combination of cellular automata and learning automata will also be introduced. Finally, we present irregular cellular learning automaton (ICLA), which differs from traditional CLA in that the restriction of a rectangular grid structure is removed.

3.1. Cellular automata

Cellular automaton (CA) [36] is a mathematical model for system with a large number of simple and similar components that interacts locally. A CA is a nonlinear space/time-discretized dynamical system. In the term cellular automaton, the cellular part implies that it consists of a grid of points in a lattice (made up of cells like points in a lattice or like squares on a checker-board) and the automaton part denotes the fact that it performs based on a simple rule.

By interacting, these simple components generate complicated patterns of behavior. CA is shown to be robust and efficient for complex computations. More specifically, they perform well for modeling natural systems including massive collections of simple objects with local interact [37]. Informally, a d-dimensional CA consists of an infinite d-dimensional lattice of identical cells. Each cell can choose a state from a finite set of states. The cells update their states synchronously in discrete steps according to a local rule. The new state of each cell depends on the previous states of itself and the cells in its neighborhood. The state of all cells in the lattice is described by a configuration. The rule and the initial configuration of the CA specify the evolution of CA to indicate how each configuration is changed in each step.

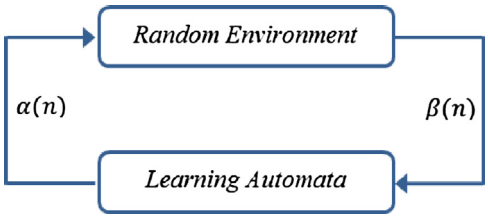

Fig. 1. A learning automata and the relationship with its random environment.

3.2. Learning automata

A learning automaton [15] is defined to be an adaptive decision-making unit whose performance is improved by learning how to choose the optimal action from a finite set of allowed actions during repeated interactions with a random environment. The action is chosen randomly based on a probability distribution over the action set and at each instant the given action serves as the input to a random environment. The environment responds in turn to the action taken with a reinforcement signal. The action probability vector is updated with regard to reinforcement feedback from the environment. The goal of a learning automaton is to find the optimal action from the action set so that the average penalty received from the environment is minimized.

The environment is modeled by a triple $E = \{\alpha, \beta, c\}$ where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the finite set of inputs, $\beta = \{\beta_1, \beta_2, \ldots, \beta_r\}$ denotes the set of values that can be taken by the reinforcement signal and $c = \{c_1, c_2, \ldots, c_r\}$ refers to the set of penalty probabilities, which is associated with the given action α_i. A random environment is called a stationary environment if the penalty probabilities are constant and if they vary with time it is called a non-stationary environment. The environments are classified into three groups, *i.e.*, P-models, Q-models and S-models depending on the nature of the reinforcement signal β. A P-model is an environment, in which the reinforcement signal only takes one of two binary values 0 and 1. The second class of environments allowing a finite number of values in the interval [0, 1] to be taken by the reinforcement signal is referred to as a Q-model. In S-model environments, the reinforcement signal lies in the interval [0, 1]. Fig. 1 shows the relation between a learning automaton and its random environment.

In systems with incomplete information about the environment, the learning automaton has been shown to perform appropriately. Learning automata have even proved to be helpful in modeling complex, dynamic and random environments with significant uncertainties. A wide variety of applications in complex social networks [38–42] are approached using learning automata.

Learning automata can be characterized by a triple with elements β, α and T, where β denotes the set of inputs, α is the set of actions and T is the learning algorithm. The learning algorithm T is an iterative recurrence relation, which is used to update the action probability vector. Let $\alpha_i(t)$ be the action selected by the learning automaton and $p(t)$ denote the probability vector defined for the action set at instant t. Let r be the number of possible actions to be taken by the learning automaton, at each instant t the action probability vector $p(t)$ is updated by the linear learning algorithm as given in (1) when the selected action $a_i(t)$ is rewarded (*i.e.*, $\beta(t) = 0$) by the random environment and it is updated using (2) if the action taken is penalized (*i.e.*, $\beta(t) = 1$).

$$p_j(t+1) = \begin{cases} p_j(t) + a(t)\left(1 - p_j(t)\right) & j = i \\ (1 - a(t))\, p_j(t) & \forall j \neq i \end{cases} \quad (1)$$

$$p_j(t+1) = \begin{cases} (1-b(t))\,p_j(t) & j = i \\ \left(\frac{b(t)}{r-1}\right)(1-b(t))\,p_j(t) & \forall j \neq i \end{cases} \quad (2)$$

where $a(t)$ and $b(t)$ denote the reward and penalty parameters respectively, thus determining the increases and decreases of the action probabilities. If $a(t) = b(t)$, the recurrence consisting of (1) and (2) is called a linear reward-penalty ($L_{R\text{-}P}$) algorithm. If $a(t) \gg b(t)$, the given equations form a linear reward-penalty ($L_{R\text{-}\varepsilon P}$) algorithm and finally, if $b(t) = 0$, they represent a linear reward-inaction ($L_{R\text{-}I}$) algorithm. In the latter group, the action probability vectors remain constant in the case where the action is penalized by the environment.

In recent years, several learning automata-based algorithms have been developed to solve graph problems and successful results have been reported in the literature, for such problems as shortest path problem [43], maximum independent set problem [44], vertex covering problem [45], vertex coloring problem [46]. More recently, LA have been utilized in optimization [47,48], grid computing [49], image processing [50], machine vision [51] and network sampling [52,53].

3.3. Cellular learning automata

Combinations of cellular automata and learning automata are called cellular learning automata (CLA). CLA is served as a proper mathematical model for many decentralized problems and events. In a CLA, the state transition probability is updated using the learning automata. Hence, CLA is a mathematical tool for modeling dynamical complex systems consisting of a large number of simple components. These components cooperate to produce complicated behavioral patterns. In fact, the learning capability helps to produce such complicated patterns.

More precisely, a CLA is composed of a CA with one or more learning automata assigned to each cell. The learning automaton corresponding to a particular cell determines its state (action) based on its action probability vector. Similar to CA, a CLA also operates under a simple rule. This rule, together with the actions selected by the multiple neighboring learning automata of any particular LA determines the reinforcement signal. In a CLA, any particular learning automaton and its neighbors together constitute the local environment. The action probability vector of the neighboring learning automata varies during the evolution of the CLA, and therefore the local environment is non-stationary. CLA has been shown to perform extremely well in a variety of applications such as network sampling [54], data mining [55], peer to peer networks [56], optimization [57] and adaptive Petri net [58] to name only a few.

The main operational steps of cellular learning automata can be described as follows. First, the internal state of every cell is determined on the basis of the action probability vectors of the learning automata residing in that cell. The initial value of the states (action probability vectors) may also be chosen based on past experiences, or even randomly. In the second step, the rules of cellular automata determine the reinforcement signal of each learning automaton residing in that cell. Finally, each learning automaton updates its action probability vector according to both the reinforcement signal obtained and the chosen action. This process is continued until the desired result is obtained. In the following paragraph, we formally explain a d-dimensional CLA

A d-dimensional cellular learning automaton is a structure $A = \left(Z^d, \Phi, A, N, F\right)$ such that:

1. Z^d is a lattice of d-tuples of integer numbers.
2. Φ is a finite set of states.
3. A is the set of LAs, each of which is assigned to the corresponding cell of the CA.
4. $N = \{x_1, x_2, \ldots, x_m\}$ is a finite subset of Z^d called the neighborhood vector, where $x_i \in Z^d$.
5. F: $\Phi_m \rightarrow \beta$ is the local rule of cellular learning automata, where β is the set of values that the reinforcement signals can take. It determines the reward (reinforcement) signal for each LA from the current actions selected by its neighboring. In the following subsection, we will review in detail the different types of CLA reported in the literature.

3.4. Irregular cellular learning automata

A CLA is represented by a rectangular structure, in which each cell is equipped with a learning automaton. To remove the restriction of the rectangular grid structure in traditional CLA, an irregular cellular learning automaton (ICLA) [59] is proposed. We note that in many applications such as wireless sensor networks, immune network systems and graph-related applications, it is not possible to model the environment adequately with rectangular grids. An ICLA is defined as an undirected graph with the same cell representation for the nodes as CLA. The learning automaton corresponding to a particular cell determines its state (action) based on the action probability vector. In addition, there is a rule under which the ICLA operates. The reinforcement signal to the LA residing in a cell is determined according to this rule and by the actions selected by the learning automata in the neighboring cells. The local environment is called non-stationary because the action probability vectors of the neighboring learning automata vary during the evolution of the ICLA.

The CLA models studied so far are regarded as closed, because they do not take into account the interaction between the CLA and the external environments. In this paper, a new class of CLA called open CLA (OSCLA), in which the evolution of a CLA is influenced by both the internal and external environments, is investigated. Two types of environments can be considered in the open CLA, i.e., global environment and the local environment. Each CLA has one global environment that influences all cells and a local environment for each particular cell. The interconnections of a typical cell in the open CLA is shown in Fig. 2.

Recently several models of CLA have been suggested in the literature, based on their structure and function as summarized below. The reported models can be classified into two main classes as described in the following paragraphs.

- **Static CLA (SCLA)**: In an SCLA, the structure of the cells remains unchanged during the evolution process of the SCLA [35,60]. An SCLA can be either closed or open. In a closed SCLA, the local environment (i.e., the states of the neighboring cells of each cell) impacts on the action selection process of the LA of that cell, whereas in an open SCLA, the local environment of each cell, the global environment and an exclusive environment all affect the action selection process of the LA of that cell. In an open SCLA, each cell has its own exclusive environment and one global environment is defined for the whole SCLA. An SCLA can be further classified as either synchronous or asynchronous. In a synchronous SCLA, all cells implement their local rules at the same time [61]. This model assumes that there is an external clock, which triggers synchronous events for the cells. In an asynchronous SCLA, at a given time, only some cells are activated and the states of the rest of the cells remain unchanged [62]. In [60], a model of SCLA with multiple learning automata in each cell was reported. In this model, the set of learning automata in a cell remains fixed during the evolution of the SCLA. An SCLA can also be classified as regular [13] or irregular [59], depending on its

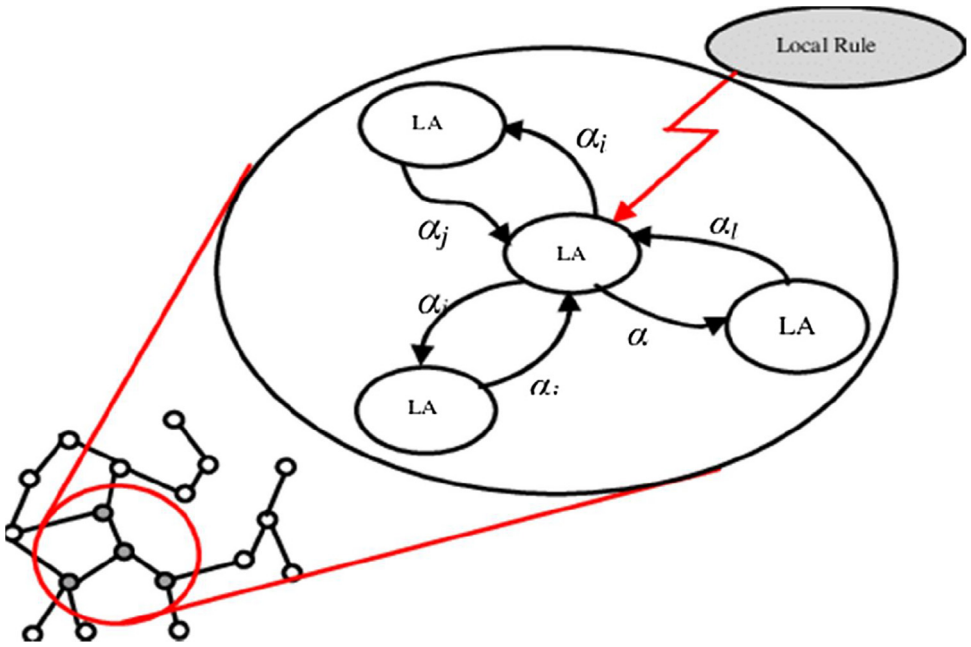

Fig. 2. The structure of irregular cellular learning automaton.

structure. In irregular SCLA, the structural regularity assumption is removed.

- **Dynamic CLA (DCLA)**: In a DCLA, one of its aspects such as the structure, local rule, attributes or neighborhood radius may change over time. A DCLA can be classified as either a closed DCLA [63,64,56] or an open DCLA. A DCLA can also be classified as synchronous or asynchronous DCLA. In a synchronous DCLA, all learning automata in different cells are activated synchronously whereas in an asynchronous DCLA the learning automata in different cells are activated asynchronously. An asynchronous DCLA can be either time-driven or step-driven [63]. In a time-driven asynchronous DCLA, each cell is assumed to have an internal clock, which wakes up the LA associated with that cell, while in a step-driven asynchronous DCLA; a cell is selected in a fixed or random sequence. Note that the problem of the definition of an asynchronous DCLA is dependent on the application. All the reported DCLAs are closed and asynchronous [56,63]. A DCLA can also be classified as an interest-based DCLA [63] or an attribute-based DCLA [64]. In an interest-based DCLA, a set of interests is defined for describing the dynamicity of the CLA, and in an attribute-based DCLA, a set of attributes is defined for describing the dynamicity of the CLA. A main drawback of both interest-based DCLA and attribute-based DCLA is that there are no formal definitions for the rules, which determine the changes in the attributes or interests of the cells. Therefore, the existing models of DCLA are unable to support dynamicity in a wide range of applications with changing attributes or interests.

4. Proposed algorithm

In this section, we propose an irregular cellular learning automata-based algorithm called CLACD for revealing the communities in a network. The CLACD algorithm is locally and independently run at each cell of the CLA, which means the decision made by each cell is local and is independent of others. The CLACD algorithm is also a fully distributed algorithm, which avoids remaining in local solutions. We will further detail how it is possible to detect communities with the aid of cellular learning automata in more depth. To achieve this goal, we first describe the whole structure of the algorithm in brief. The CLACD algorithm, consisting of four steps, tries to reveal the community structures in the networks. It is assumed that $G = V$, E is undirected and unweighted graph, in which $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes and $E \subseteq V \times V$ is the set of links in the input network. For a graph G, there are many possible partitions. The main goal of the community detection in the graph G is to reveal sub-graphs $CP = \{C_1, \ldots, C_k\}$ of the set V of nodes divided into k disjoint partition such that a quality function $\phi(CP)$ is optimal. We note that there is no assumption that provides either the number or the size of the communities of the proper partition. To achieve this goal, after the initialization step, the CLACD algorithm tries to find communities in an iterative manner by finding partial spanning tree. The process of detecting communities from the nodes is guided by the set of learning automata residing at the nodes and by selecting neighbor nodes as an action selection of the input networks. With the aid of cellular learning automata, the set of obtained communities is evaluated by both the reinforcement signal of local and global environments, and their action probability vectors are updated until the stopping criteria are satisfied. It is necessary to point out that, the asynchronous structure for the cellular learning automata is adopted for the implementation aspect in the interests of simplicity. The pseudo-code of the CLACD algorithm is given in Fig. 3.

4.1. Community detection algorithm based on cellular learning automata (CLACD)

The proposed algorithm called community detection algorithm based on cellular learning automata (CLACD) as described as the following steps.

4.1.1. Initialization

To initialize the algorithm, an asynchronous ICLA is created, which is isomorphic to the input network. To construct such an ICLA, each node is associated with a cell of CLA, and then an LA is assigned to each cell (hereafter v_i may be used interchangeably for cell v_i or node v_i). The resulting ICLA can be described by a duple $< \underline{A}, \underline{a} >$, where $\underline{A} = \{A_1, A_2, \ldots, A_n\}$ denotes the set of LAs residing in each cell (node) of ICLA (network) and $\underline{a} = \{\underline{a_1}, \underline{a_2}, \cdots, \underline{a_n}\}$ denotes the action set, in which $\underline{a_i} = \left\{\underline{a_i^1}, \underline{a_i^2}, \cdots, \underline{a_i^{r_i}}\right\}$ (for each

Cellular learning automata-based community detection algorithm (CLACD)

Input: The graph $G = (V, E)$, Threshold P_{min}, T_{max}
Output: A set of communities $CP = \{C_1, \ldots, C_k\}$
Initialization
Create *ICLA* isomorphic to the graph G by associating with a cell in each node and then assigning an LA in each cell
Let $\underline{a} = \{\underline{a_1}, \ldots, \underline{a_n}\}$ be the action set of CLA in which $\underline{a_i} = \{\underline{a_i^1}, \ldots, \underline{a_i^{r_i}}\}$ be the set of actions for LA A_i in cell v_i
Let $p(v_i) = \{p_i^1, p_i^2, \ldots, p_i^{r_i}\}$ be the action probability of A_i in cell v_i and equally initialized $p_j^i = \frac{1}{r_i}$ for all j.
Begin algorithm
Let t be the iteration number of the algorithm and initially set to 1
Let T_t be the spanning tree found at iteration t

Let $E(k)$ be the entropy value for set of communities CP and initially set to a MAX value
While ($t < T_{max}$ **OR** $E(p) < P_{min}$) **Do**
Let a random selected node be v_i
Repeat
A_i in cell v_i selects one of its action based on its action probabilities
Let the selected action by A_i be cell v_j
$T_t \leftarrow T_t + \{v_i\}$
v_i v_j
Until a partial spanning tree is created
q 1;
Let a random selected node be v_i from T_t
Repeat
Let a selected node be v_i
C_q v_i
If ($\sum_{e \in T_t} K_i^{in}(C_q \cup \{v_j\}) > \sum_{e \in T_t} K_i^{in}(C_q)$ **AND** $\sum_{e \in T_t} K_i^{out}(C_q \cup \{v_j\}) < \sum_{e \in T_t} K_i^{out}(C_q)$) **Then**
C_q $C_q \cup \{v_j\}$
v_i v_j
Else
q $q + 1$
C_q v_j
End If
Until all nodes assigned to a local community
Compute $\theta_t$using equation (4)
Compute β_g using equation (5)
Compute β_i using equation (6)
If ($\beta_i == 0$) and ($\beta_g == 0$) **Then**
Reward LA A_i of cell v_i based on equation (1)
End if
Compute summation of all entropy probability $E(p)$ according to equation (7)
$t \leftarrow t + 1$
End While
End Algorithm

Fig. 3. Pseudo-code of the proposed algorithm (CLACD).

$\underline{a_i} \in a$) represents the set of actions that can be taken by learning automaton A_i. Learning automaton A_i, which is residing in node v_i has r_i actions, each of which corresponds to selecting one of the adjacent nodes. Let $p(v_i) = \{p_i^1, p_i^2, \ldots, p_i^{r_i}\}$ be the action probability vector of learning automaton A_i and $p_j^i = \frac{1}{r_i}$ be equally initialized for all j.

4.1.2. Construction of communities

In this step, a partial spanning tree of network is constructed and several local communities are formed on the partial spanning tree found by the algorithm. At first, learning automaton A_i in cell v_i selects an action based on its action probability vector, which corresponds to selecting learning automata A_j. Let T_t be the spanning tree at iteration t. The current node v_i and the selected node v_j is added to T_t if this addition constructs a partial spanning tree. Then, learning automaton A_j in cell v_j selects an action. The process of selecting an action by each LA in each cell of CLA, adding the selected node v_i to T_t is continued until either the set of remaining available LA is empty or the new selected action is one of the actions that is previously added to T_t. After selecting an action by all LAs, the algorithm constructs local communities on the found partial spanning tree of the network. The partial spanning tree of

the network is used in the algorithm in order to reduce the network size and computational cost for detecting communities due to the low time complexity of finding partial spanning tree. A set of local communities $CP = \{C_1, \ldots, C_q, \ldots, C_k\}$ are formed based on local connectivity of cells, in which C_q is the q^{th} local community by merging neighboring cells, in such a way that the number of internal connections for merging cell v_i with cell v_j or a local community C_q is greater than the number of internal connections for current community C_q then v_i and v_j are formed a local community or assigned to the current community C_q. In this step, most of nodes are formed a set of local communities CP, however the remaining unassigned nodes assigned to the k^{th} community (i.e., C_k).

4.1.3. Computation of the objective function

In this step, both reinforcement signals of local and global environments are used to evaluate the set of communities CP found by the algorithm. The conductance $\theta_t(CP)$ is used for generating reinforcement signal of global environment, in order to evaluate the quality of the current communities found by the algorithm at iteration t in the network. Let $vol(C_q)$ denote the total number of links within community C_q and $Cut(C_q)$ denote the number of links falling between different community partitions (cut sizes) where one endpoint is inside community C_q. The quality of the found community CP at iteration t can be calculated using conductance as an objective function, as defined by following equation:

$$\phi_t(CP) = \frac{1}{k}\sum_{q=1}^{k}\frac{Cut(C_q)}{\min\left(vol(C_q), vol(\overline{C_q})\right)} \tag{3}$$

where $\overline{C_q}$ refers to the complement community C_q (or the rest of the network). In brief, the conductance computes the fraction of the total link volume that points outside the community. The average conductance at iteration t (*i.e.*, θ_t), which is the average of all obtained conductance values found up to that point is calculated as follows:

$$\theta_t = \frac{(t-1)\theta_{t-1} + \phi_t(CP)}{t} \tag{4}$$

We used the conductance due to the fact that it can be simply calculated. Furthermore, it exhibits proper performance in general. We note that, the lower value of conductance reflects the better it is. Hence, the reinforcement signal for the global environment, β_g can be computed as follows

$$\beta_g = \begin{cases} 0 & \theta_t \le \theta_{t-1} \\ 1 & Otherwise \end{cases} \tag{5}$$

The local environment of a learning automaton is configured based on the cellular learning automata in the neighboring cells and incorporates the neighboring nodes in the network. The reinforcement signal for local environment, β_i for learning automaton A_i in cell v_i of cellular learning automata is defined according to

$$\beta_i = \begin{cases} 0 & \sum_{i \in C_q} K_i^{in}(C_q) > \sum_{i \in C_q} K_i^{out}(C_q) \\ 1 & Otherwise \end{cases} \tag{6}$$

where $K_i^{in}(C_q)$ and $K_i^{out}(C_q)$ are the summation of number of internal and external links for node v_i in community C_q with respect to the given input network, respectively.

4.1.4. Updating the action probabilities

At every iteration t, for each cell, if both the reinforcement signal of local and global environment are favorable (*i.e.*, $\beta_i = 0$ and $\beta_g = 0$), then the action probabilities of learning automaton in each cell are updated depending upon the internal state, the actions chosen by all learning automata are rewarded. Each learning automaton in each cell updates its action probability vector by using an L_{R-I} reinforcement scheme.

4.1.5. Stopping condition

Here, we aim to describe the stopping condition for the CLACD algorithm. A simple criterion for a stopping condition is the number of iterations, i.e., the algorithm terminates after a predefined number of iterations $K_{\max}$. Furthermore, Entropy also used as an another criterion for stopping condition for an LA, which is defined as

$$E(p) = -\sum_{i \in C} p_i \log p_i \tag{7}$$

where p_i is the probability of selecting action α_i by an LA. If the entropy value is less than a predefined threshold it can be concluded that the algorithm has converged to an action and it is therefore terminated. In the simulation, we compute Entropy for all LAs in CLA.

As mentioned in previous section, the CLACD algorithm can be locally performed at each cell independent of the other cells to create local communities. As the CLACD algorithm proceeds, each LA in each cell of CLA learns how to select a neighboring cell to create a local community. Since the local internal and external connection of each node in a local community is considered by local reinforcement signal and the set of communities is compared with the best set of communities which it has created so far by global reinforcement signal, the CLACD algorithm gradually yields the near-optimal community structures. Pseudo-code of the proposed algorithm (CLACD) is given in Fig. 3.

4.2. Complexity analysis

The space complexity of the CLACD algorithm is $O(n)$ where n is the number of nodes in the social network. Because, the data must be storing during the iterations of CLACD include: the probability vector, which consist of n elements, the computed values of φ, which are stored in a vector and two n-element vectors representing internal and external degree respectively. Hence the whole space complexity of CLACD is $S = 3n + t$ where t the number of necessary iterations is. In most cases the algorithm converges in $t < n$ iterations and in this case the space complexity would be $S \simeq O(n)$.

5. Experiments

To demonstrate the efficiency of the CLACD algorithm for detection of community structures, a number of experiments are conducted on several real (*i.e.*, *Karate, Dolphins, Books, Football, Netscience and Power-grid*) and synthetic (*i.e.*, LFR benchmark [65]) networks as described in Table 1. In LFR benchmark, N indicates the number of nodes in the network, k indicates the average degree of nodes, Max_k indicates maximum degree of nodes, $\min_c$ indicates the number of nodes that the smallest community contains, $\max_c$ indicates the maximum size of the communities and μ as mixing parameter indicates the probability that nodes are connected with nodes of the external community. For synthetic modular networks, we set the LFR benchmark parameters as $N = 1000$, $k \in \{15, 100\}$, $\max_k \in 50$, $\min_c = 10$, $\max_c = 50$ and $\mu = \{0.00, 0.50\}$ with a span of 0.05.

5.1. Evaluation measures

In this subsection, we will describe evaluation measures including: Modularity [20], Normalized Mutual information (NMI) [32],

Table 1
Characteristics of real test networks.

Networks	Nodes	Links	Description
Karate	34	78	Network of Zachary's karate club.
Dolphins	62	159	Network of Lusseau's dolphins.
Books	105	441	Network of books about US politics.
Football	*115*	*615*	Network of American College football union
Net-science	1589	2742	Coauthor-ship network of scientists working on network theory
Power-grid	4941	6594	The topology of the Power Grid of the United State

Purity [66] and Rand-index [17] for assessing found communities by the algorithms.

5.1.1. Modularity

The Modularity Q is another measure for evaluating the set of communities with a known structure in the network found by the algorithm. This measure is defined as follows:

$$Q = \frac{1}{2m}\sum_{C\in P}\sum_{v_i v_j \in C}\left[A_{i,j} - \frac{k_i k_j}{2m}\right] \tag{8}$$

where A is the adjacency matrix and $A_{i,j}$ is equal to unity if there is a link between node v_i and node v_j and zero otherwise. The degree of node v_i is $k_i = \sum_j A_{i,j}$ and m is the total number of links in the network. The summation is over all pairs of nodes that are member of the same community C with partitioning P [20].

5.1.2. Normalized mutual information (NMI)

NMI is used for networks with known community structures that measures the similarity between real community structure and community structure found by the algorithm. NMI is calculated as follows:

$$NMI(A,B) = \frac{-2\sum_{a\in A}\sum_{b\in B}|a\cap b|\log\left(\frac{|a\cap b|n}{|a||b|}\right)}{\sum_{a\in A}|a|\log\left(\frac{|a|}{n}\right) + \sum_{b\in B}|b|\log\left(\frac{|b|}{n}\right)} \tag{9}$$

where A and B are two partitions of the input networks, and NMI is a value in the range [0, 1], where the higher value indicates that the partitions A and B are totally independent. This measure is useful for synthetic networks where there is prior knowledge about built-in communities [32].

5.1.3. Purity

Purity is used to evaluate the prediction performance of community detection algorithm. This measure is defined as follows:

$$Purity(R,C) = \frac{1}{n}\sum_{m}\max_{k}(R_m\cap C_k) \tag{10}$$

where n is the number of nodes, R_m is the set of clusters and C_k is the set of classes. In order to compute the purity, a label is considered for each of the communities that occur most frequently, and then the cardinality of well-assigned nodes is computed with respect to ground-truth communities. Purity takes values in the range [0, 1]. A value of 1 shows the highest accuracy and low values indicate that communities are poorly detected [66].

5.1.4. Rand–index

The Rand-index measure is applied by investigating the similarity between the two types of partitions in the network. The index is bound in the range [0, 1] with the upper bound denoting a perfect match. Let A be defined as the set of communities obtained by the proposed community detection algorithm and B be the true communities of the network. During the execution of the algorithm two groups may show different behavior. The Rand-index for two partitions of A and B is defined as follows:

$$Rand-index(A,B) = \frac{(p+s)}{(p+q+r+s)} \tag{11}$$

where p is the number of pairs of nodes which are in the same community in both partitions, q and r are the number of pairs of nodes which are in the same community in A and in different communities in B and s is the number of pairs of nodes that are in different communities in both partitions [17].

5.2. Experimental results

5.2.1. Experiment I

This experiment is conducted to evaluate the performance of the CLACD algorithm for finding the communities in terms of the modularity Q. The results are compared with some popular algorithms such as community finding algorithm of Clauset, Newman and Moore termed as CNM [67], genetic algorithm for community detection termed as GA-Net [68], multi-objective evolutionary algorithm with decomposition, termed as MOEA/D-Net [31], multi-objective genetic algorithm, termed as MOGA-Net [30], community detection method based on modularity and an improved genetic algorithm, termed as MIGA [69], cellular learning automata based algorithm for community detection termed as CLA-Net [35] and michigan memetic algorithm for community detection termed as MLAMA-Net [34]. The results reported in Table 2 consist of both the maximum (Max) and average (Avg.) values of modularity. As can be seen from Table 2, for *Football*, *Net-science* and *Power-grid* networks, the *CLACD* reveals community structures with larger value of Modularity in comparison with CNM, GA-Net and MOGA-Net. For Dolphins network *CLACD* algorithm produces similar results to MLALA-Net and CLA-Net. The revealed similar results are mainly due to an overlapping community structures in some networks.

5.2.2. Experiment II

This experiment is conducted to study the convergence behavior of the CLACD algorithm during the process of community finding in terms of conductance and entropy. For this experiment, we present the plot of entropy and conductance versus iteration number for real and synthetic LFR benchmark network. For LFR network, the mixing parameter varies from $\mu = 0.0$ to $\mu = 0.5$ with a 0.1 interval. The results of both plot for entropy and conductance are taken average over 100 runs. The plots of entropy for real and synthetic networks are given in Figs. 4 and 5 respectively. Also the plots of conductance for real and synthetic networks are given in Figs. 6 and 7 respectively.

From the results in Figs. 4 and 6, once can observe that for all values of mixing parameters the entropy and conductance have high values equal to unity in early iterations and it gradually decreases to converge to a small value. This means that for all types of modular synthetic networks the algorithm smoothly reveals the communities after a finite number of iterations. From Figs. 5 and 7, one may conclude that the value of entropy and conductance starts from a value of unity and gradually decreases. Since a lower conductance value indicates that more links are within the community found by

Table 2
Comparison of the community detection algorithms on real networks in terms of maximum (Max) and average (Avg.) modularity.

Network	Q	CNM	GA-Net	MOGA-Net	MOEA/D-Net	MLAMA-Net	CLA-Net	CLACD
Karate	Max	0.3807	0.4059	0.4198	0.4198	0.4198	0.4188	0.4188
	Avg.	0.3807	0.4059	0.4158	0.4198	0.4136	0.4175	0.4039
Dolphins	Max	0.4955	0.5014	0.5258	0.5210	0.5277	0.5277	0.5277
	Avg.	0.4938	0.4046	0.5215	0.5189	0.5222	0.5268	0.5198
Football	Max	0.5733	0.5940	0.5280	0.6044	0.6058	0.6046	0.6044
	Avg.	0.5706	0.5830	0.5173	0.6032	0.6050	0.6042	0.5864
Book	Max	0.5181	0.5230	0.5272	0.5268	0.5272	0.5268	0.5223
	Avg.	0.5178	0.5230	0.5255	0.5236	0.5255	0.5254	0.5188
Net-science	Max	0.9555	0.8581	0.8916	0.9143	0.9550	0.9346	0.9553
	Avg.	0.9554	0.8473	0.8810	0.9060	0.9544	0.9177	0.9541
Power-grid	Max	0.9345	0.6660	0.7035	0.6880	0.9357	0.7505	0.8654
	Avg.	0.9338	0.6571	0.6949	0.6815	0.9336	0.7350	0.8124

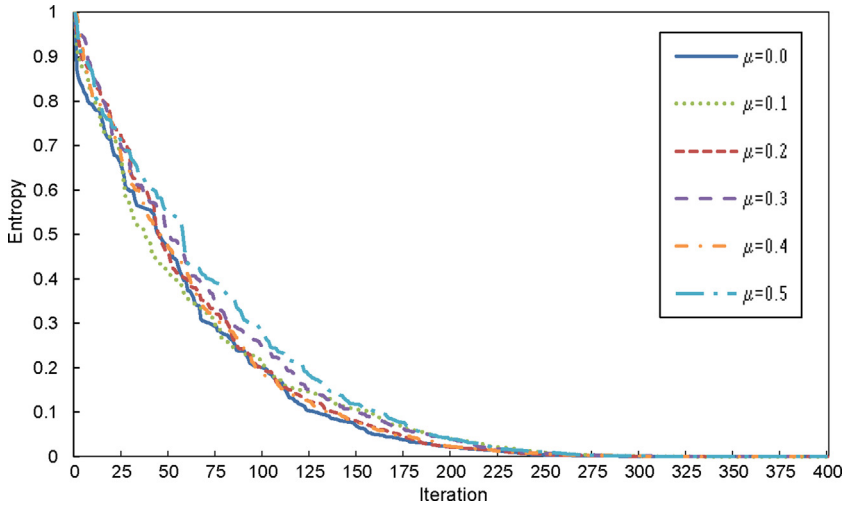

Fig. 4. The plot of entropy versus iteration for synthetic LFR benchmark networks.

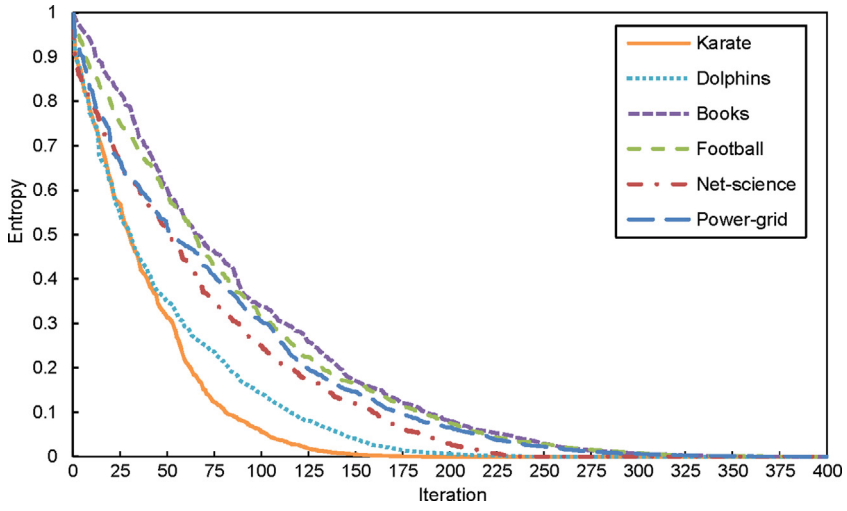

Fig. 5. The plot of entropy versus iteration for real networks.

the algorithm, the trend of these plots indicates that the community found by the algorithm has a proper quality.

Moreover, one may observe that in most of the curves, as mixing parameter μ increases, the detection of communities becomes a more difficult task; therefore it leads to higher values in the corresponding conductance plot. These plots for both conductance and entropy versus iteration number indicate the important role of the learning automata in guiding the process of finding communities in networks.

5.2.3. *Experiment III*

This experiment is carried out to study the performance of the CLACD algorithm compared with other well-known community detection algorithms (CNM [67], GA-Net [68], MIGA [69], MOGA-

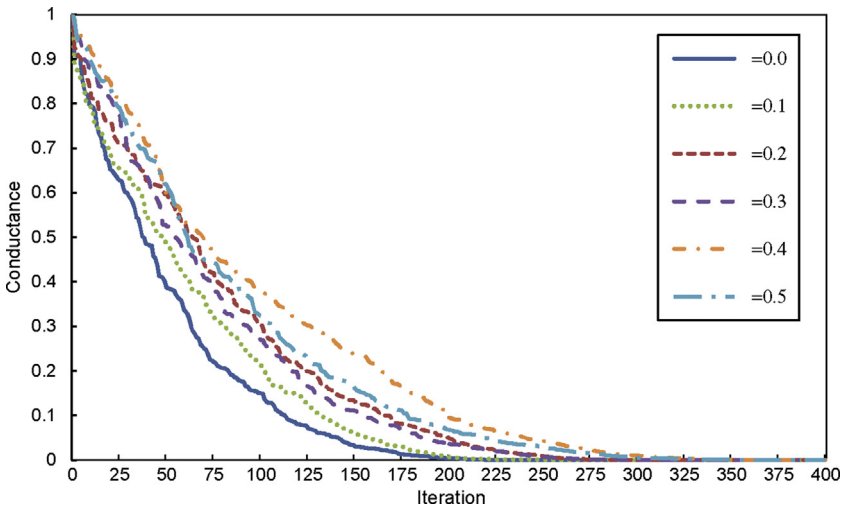

Fig. 6. The plot of conductance versus iteration for synthetic LFR benchmark networks.

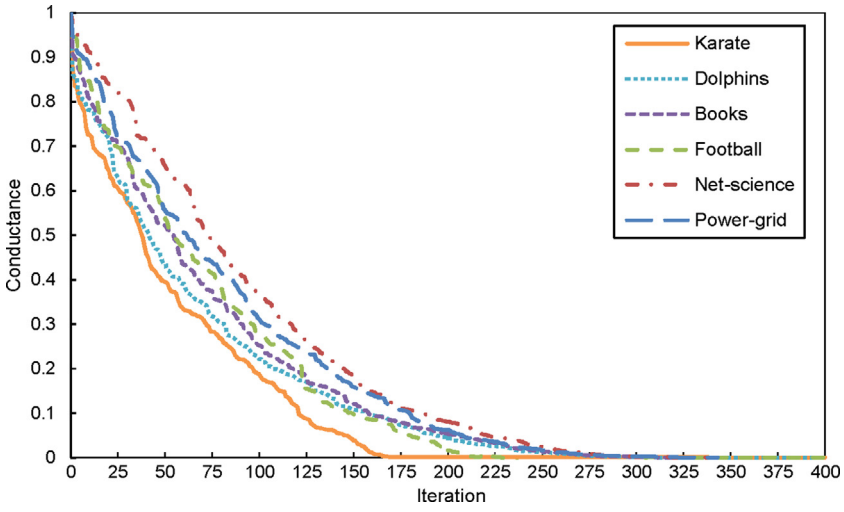

Fig. 7. The plot of conductance versus iteration for real networks.

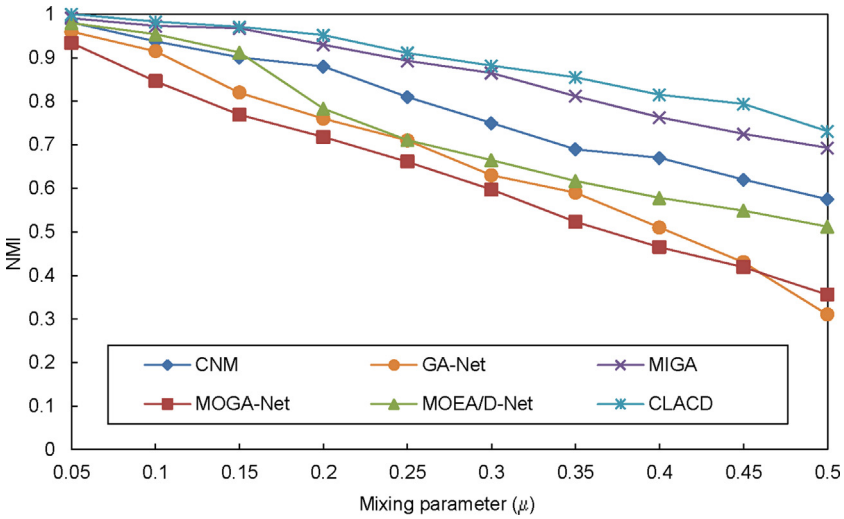

Fig. 8. Results of NMI for different algorithms on LFR benchmark networks.

Net [30] MOEA/D-Net [31]) for synthetic LFR benchmark network with varying values of the mixing parameters in terms of NMI, Purity and Rand-index. For this purpose, we plot these measures for varying values of the mixing parameters from $\mu = 0.00$ to $\mu = 0.50$

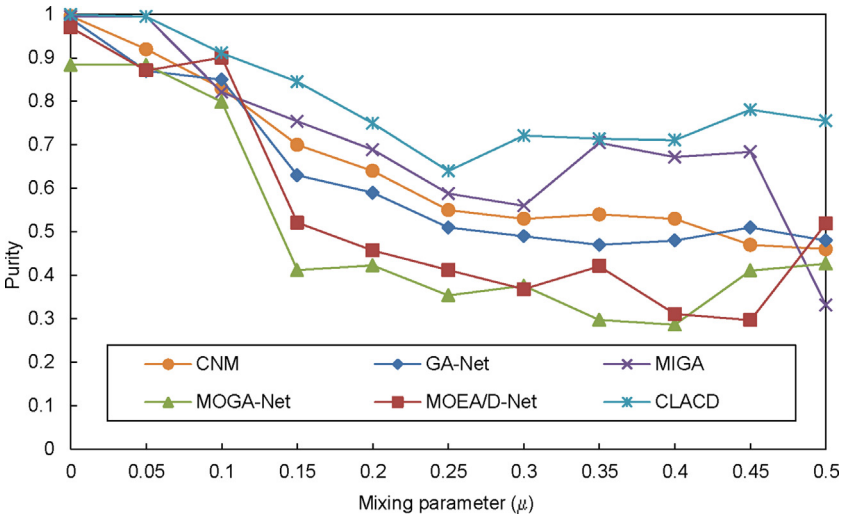

Fig. 9. Results of Purity for different algorithms on LFR benchmark networks.

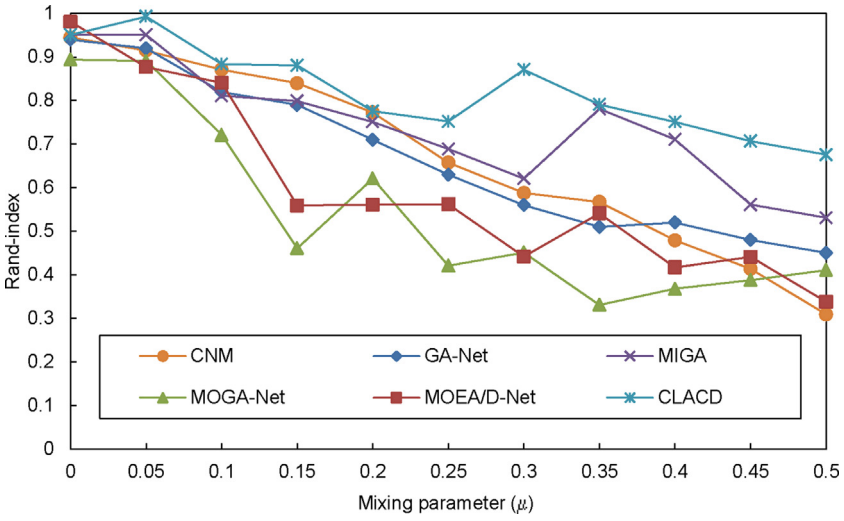

Fig. 10. Results of Rand-index for different algorithms on LFR benchmark networks.

with a 0.05 interval. The results obtained for different mixing parameters in terms of NMI, Purity and Rand-index are given in Figs. 8–10, respectively.

From the results shown in Figs. 8–10 the following observations can be made:

- For a mixing parameter value of less than 0.2 the community structure of networks is clear, and hence the obtained NMI, Purity and Rand-index values are close to unity.
- For mixing parameters between 0.2 and 0.4, it is expected that all algorithms will attain a lower NMI, Purity and Rand-index values because when the mixing parameter increases, the detection of communities becomes a more difficult task. However, CLACD outperforms all the other algorithms with respect to NMI, Purity and Rand-index.
- For a mixing parameter value greater than 0.4, the performance of CLACD is superior to all other algorithms in terms of NMI, Purity and Rand-index.

Overall, the CLACD algorithm outperforms the other algorithms for different mixing parameters in the range [0, 0.5] in terms of mentioned measures.

5.2.4. Experiment IV

This experiment is designed to study the impact of the resolution limit problem on the performance of the community structure found by the CLACD. For this purpose, we use two networks; one network as given in Fig. 11, consisting of 125 nodes and 25 distinct cliques as 25 communities; one network as given in Fig. 12, consisting of 2 cliques as 2 communities with 54 nodes and 4 nodes, respectively. In the second network for each community, a node is connected with all the other nodes in the same community and a single link connects the two communities to each other between node u in community 1 and node v in community 2.

The output of the CLACD algorithm for the first network is visualized in Fig. 11 where the algorithm properly found the cliques as communities. Also, the visualization of the second test network is shown in Fig. 12. The modularity of the community structure in this network is 0.00845. However, according to basic modularity optimization, node u would be assigned to Community 2, because this partitioning of the network achieves a larger modularity value of 0.01203. This phenomenon is due to the resolution limit of modularity optimization. We run the CLACD algorithm on the test network 100 times. Since the other algorithm such as MIGA and CNM algorithms are based on modularity optimization, they always incorrectly find node u as a member of Community 2. On other hand,

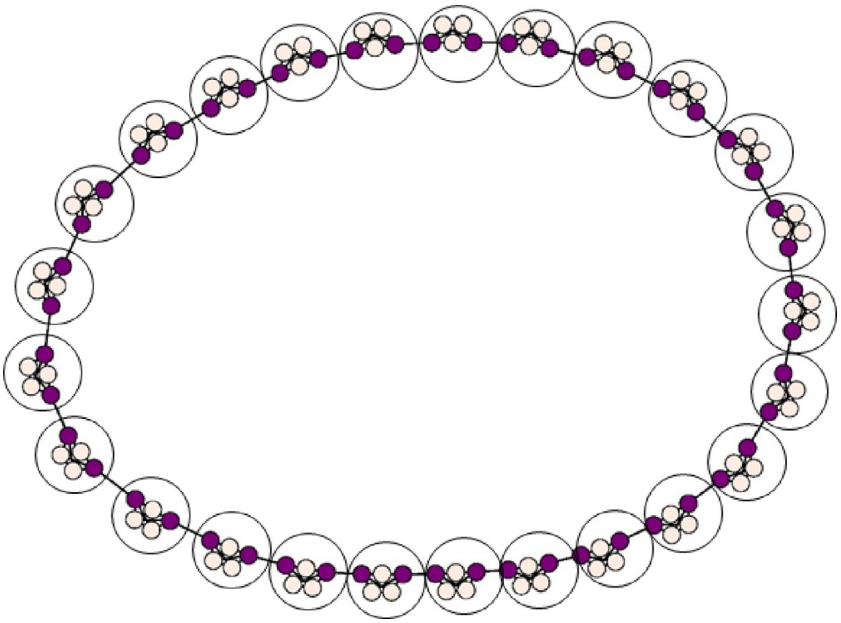

Fig. 11. The synthetic test networks with 25 communities with same size for evaluation of resolution limit problem. The communities found by the algorithm are surrounded by circles.

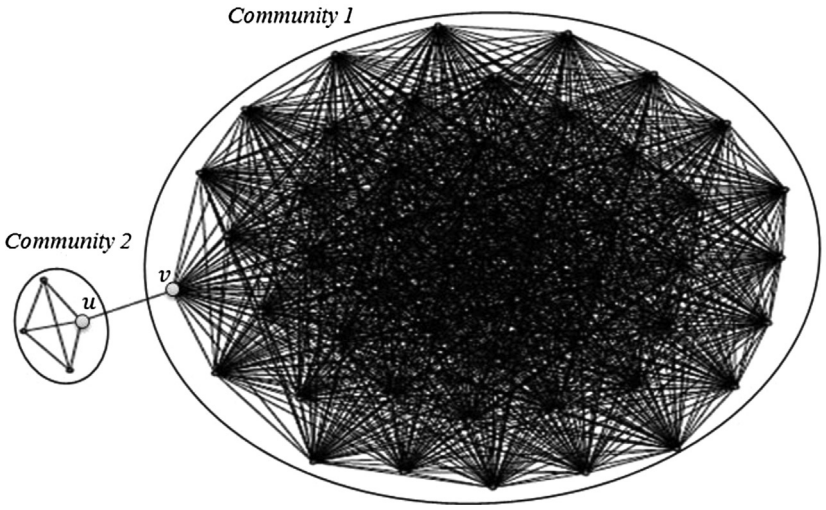

Fig. 12. The synthetic test networks with 2 communities with different size for evaluation of resolution limit problem. The communities found by the algorithm are surrounded by circles.

the CLACD algorithm always finds the correct communities in the network due to use of both local and global reinforcement signal for updating action probabilities of learning automata in cells of CLA.

6. Conclusion

In this paper, a new CLA-based algorithm called CLACD is proposed, to reveal community structure in complex social networks. The CLACD algorithm with the aid of CLA has many proper characteristics such as scalability, accuracy and applicability, and its most important advantage is its ability to consider both local and global environments, which makes detecting the communities possible even in complicated cases such as relatively small communities. In order to investigate the performance of the CLACD algorithm, a number of experiments conducted on real and synthetic modular networks. In the experimental, the CLACD algorithm is compared with some well-known community detection algorithms in terms of Conductance, Modularity, NMI, Purity and Rand-index. Simulation results reveals that the CLACD algorithm can successfully solve the resolution limit problem and also can detect communities precisely even in cases where existing basic algorithms fail to do so. The CLACD algorithm, not only provides a solution for community detection for deterministic and static networks, but also can be developed for dynamic and stochastic networks for future researches and developments of new directions of social network analysis.

Acknowledgment

This research was in part supported by a grant from IPM. (No. CS1395-4-67).

References

[1] P. Erdos, A. Rényi, On the evolution of random graphs, Publ. Math. Inst. Hung. Acad. Sci. 5 (1960) 17–61.

[2] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (1998) 440–442.
[3] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (1999) 509–512.
[4] K.R. Harrison, M. Ventresca, B.M. Ombuki-Berman, A meta-analysis of centrality measures for comparing and generating complex network models, J. Comput. Sci. 17 (2016) 205–215.
[5] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (2010) 75–174.
[6] J. Gao, B. Barzel, A.-L. Barabási, Universal resilience patterns in complex networks, Nature 530 (2016) 307.
[7] Y.-Y. Liu, J.-J. Slotine, A.-L. Barabási, Controllability of complex networks, Nature 473 (2011) 167.
[8] J. Wu, K.T. Chi, F.C. Lau, Concept of node usage probability from complex networks and its applications to communication network design, IEEE Trans. Circuits Syst. Regul. Pap. 62 (2015) 1195–1204.
[9] J. Kleinberg, The small-world phenomenon: an algorithmic perspective, Proc. Thirty-Second Annu. ACM Symp. Theory Comput., ACM (2000) 163–170.
[10] A. Clauset, C.R. Shalizi, M.E. Newman, Power-law distributions in empirical data, SIAM Rev. 51 (2009) 661–703.
[11] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. U. S. A. 99 (2002) 7821–7826.
[12] I. Hamid, Y. Wu, Q. Nawaz, R. Zhao, A fast heuristic detection algorithm for visualizing structure of large community, J. Comput. Sci. (2017), http://dx.doi.org/10.1016/j.jocs.2017.07.002.
[13] H. Beigy, M.R. Meybodi, A mathematical framework for cellular learning automata, Adv. Complex Syst. 3 (2004) 295–319.
[14] S. Wolfram, Theory and Applications of Cellular Automata, World Scientific Publication, 1986.
[15] K.S. Narendra, M.A. Thathachar, Learning Automata: An Introduction, Prentice-Hall, 1989.
[16] P. Topa, J. Wąs, New trends in complex collective systems, J. Comput. Sci. 21 (2017) 395–396.
[17] S. Fortunato, D. Hric, Community detection in networks: a user guide, Phys. Rep. 659 (2016) 1–44.
[18] M.E. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (2004) 066133.
[19] M.S. Rahman, A. Ngom, A fast agglomerative community detection method for protein complex discovery in protein interaction networks, in: IAPR Int. Conf. Pattern Recognit. Bioinforma, Springer, 2013, pp. 1–12.
[20] M.E. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. U. S. A. 103 (2006) 8577–8582.
[21] S. Fortunato, M. Barthelemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. U. S. A. 104 (2007) 36–41.
[22] Z. Li, R.-S. Wang, S. Zhang, X.-S. Zhang, Quantitative function and algorithm for community detection in bipartite networks, Inf. Sci. 367 (2016) 874–889.
[23] M. Elyasi, M. Meybodi, A. Rezvanian, M.A. Haeri, A fast algorithm for overlapping community detection, 2016 Eighth Int. Conf. Inf. Knowl. Technol. IKT 2016 (2016) 221–226.
[24] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (2007) 036106.
[25] R. Hosseini, R. Azmi, Memory-based label propagation algorithm for community detection in social networks, 2015 Int. Symp. Artif. Intell. Signal Process. AISP, IEEE (2015) 256–260.
[26] E. Le Martelot, C. Hankin, Multi-scale community detection using stability optimisation, Int. J. Web Based Commun. 9 (2013) 323–348.
[27] S. Maity, S.K. Rath, Extended clique percolation method to detect overlapping community structure, 2014 Int. Conf. Adv. Comput. Commun. Inform., IEEE (2014) 31–37.
[28] D. Chen, F. Zou, R. Lu, L. Yu, Z. Li, J. Wang, Multi-objective optimization of community detection using discrete teaching–learning-based optimization with decomposition, Inf. Sci. 369 (2016) 402–418.
[29] J. Ji, X. Song, C. Liu, X. Zhang, Ant colony clustering with fitness perception and pheromone diffusion for community detection in complex networks, Phys. Stat. Mech. Appl. 392 (2013) 3260–3272.
[30] C. Pizzuti, A multiobjective genetic algorithm to find communities in complex networks, IEEE Trans. Evol. Comput. 16 (2012) 418–430.
[31] M. Gong, L. Ma, Q. Zhang, L. Jiao, Community detection in networks by using multiobjective evolutionary algorithm with decomposition, Phys. Stat. Mech. Appl. 391 (2012) 4050–4060.
[32] M.M.D. Khomami, A. Rezvanian, M.R. Meybodi, Distributed learning automata-based algorithm for community detection in complex networks, Int. J. Mod. Phys. B 30 (2016) 1650042.
[33] M. Ghamgosar, M.M.D. Khomami, N. Bagherpour, M.R. Meybodi, An extended distributed learning automata based algorithm for solving the community detection problem in social networks, 2017 Iran. Conf. Electr. Eng. ICEE, IEEE (2017) 1520–1526.
[34] M.R. Mirsaleh, M.R. Meybodi, A Michigan memetic algorithm for solving the community detection problem in complex network, Neurocomputing 214 (2016) 535–545.
[35] Y. Zhao, W. Jiang, S. Li, Y. Ma, G. Su, X. Lin, A cellular learning automata based algorithm for detecting community structure in complex networks, Neurocomputing 151 (2015) 1216–1226.
[36] S. Wolfram, Cellular Automata and Complexity: Collected Papers, Addison-Wesley, Reading, 1994.
[37] N.H. Packard, S. Wolfram, Two-dimensional cellular automata, J. Stat. Phys. 38 (1985) 901–946.
[38] A. Rezvanian, M.R. Meybodi, Stochastic graph as a model for social networks, Comput. Hum. Behav. 64 (2016) 621–640.
[39] A. Rezvanian, M.R. Meybodi, Stochastic Social Networks: Measures and Algorithms, LAP LAMBERT Academic Publishing, 2016.
[40] M.M.D. Khomami, A. Rezvanian, N. Bagherpour, M.R. Meybodi, Minimum positive influence dominating set and its application in influence maximization: a learning automata approach, Appl. Intell. (2017) 1–24, http://dx.doi.org/10.1007/s10489-017-0987-z, in-press.
[41] M.M.D. Khomami, A. Rezvanian, N. Bagherpour, M.R. Meybodi, Irregular cellular automata based diffusion model for influence maximization, 2017 5th Iran. Jt. Congr. Fuzzy Intell. Syst. CFIS, IEEE (2017) 69–74.
[42] A. Rezvanian, M.R. Meybodi, A new learning automata-based sampling algorithm for social networks, Int. J. Commun. Syst. 30 (2017) e3091.
[43] S.M. Vahidipour, M.R. Meybodi, M. Esnaashari, Finding the shortest path in stochastic graphs using learning automata and adaptive stochastic petri nets, Int. J. Uncertain. Fuzziness Knowl. Based Syst. 25 (2017) 427–455.
[44] M.M.D. Khomami, N. Bagherpour, H. Sajedi, M.R. Meybodi, A new distributed learning automata based algorithm for maximum independent set problem, in: Artif. Intell. Robot. IRANOPEN 2016, IEEE, Qazvin, Iran, 2016, pp. 12–17.
[45] A. Rezvanian, M.R. Meybodi, Finding minimum vertex covering in stochastic graphs: a learning automata approach, Cybern. Syst. 46 (2015) 698–727.
[46] S.M. Vahidipour, M.R. Meybodi, M. Esnaashari, Cellular adaptive Petri net based on learning automata and its application to the vertex coloring problem, Discrete Event Dyn. Syst. (2017) 1–32.
[47] M. Mahdaviani, J.K. Kordestani, A. Rezvanian, M.R. Meybodi, LADE: learning automata based differential evolution, Int. J. Artif. Intell. Tools 24 (2015) 1550023.
[48] F. Abtahi, M.R. Meybodi, M.M. Ebadzadeh, R. Maani, Learning automata-based co-evolutionary genetic algorithms for function optimization, Proc. 6th Int. Symp. Intell. Syst. Inform. SISY (2008) 1–5.
[49] M.H. Mofrad, O. Jalilian, A. Rezvanian, M.R. Meybodi, Service level agreement based adaptive grid superscheduling, Future Gener. Comput. Syst. 55 (2016) 62–73.
[50] M.H. Mofrad, S. Sadeghi, A. Rezvanian, M.R. Meybodi, Cellular edge detection: combining cellular automata and cellular learning automata, AEU-Int. J. Electron. Commun. 69 (2015) 1282–1290.
[51] B. Damerchilu, M.S. Norouzzadeh, M.R. Meybodi, Motion estimation using learning automata, Mach. Vis. Appl. 27 (2016) 1047–1061.
[52] A. Rezvanian, M.R. Meybodi, Sampling algorithms for stochastic graphs: a learning automata approach, Knowl. Based Syst. 127 (2017) 126–144.
[53] A. Rezvanian, M.R. Meybodi, Sampling algorithms for weighted networks, Soc. Netw. Anal. Min. 6 (2016) 1–22.
[54] M. Ghavipour, M.R. Meybodi, Irregular cellular learning automata-based algorithm for sampling social networks, Eng. Appl. Artif. Intell. 59 (2017) 244–259.
[55] M.K. Sohrabi, R. Roshani, Frequent itemset mining using cellular learning automata, Comput. Hum. Behav. 68 (2017) 244–253.
[56] A.M. Saghiri, M.R. Meybodi, A closed asynchronous dynamic model of cellular learning automata and its application to peer-to-peer networks, Genet. Program. Evolvable Mach. 18 (2017) 313–349.
[57] M. Mozafari, M.E. Shiri, H. Beigy, A cooperative learning method based on cellular learning automata and its application in optimization problems, J. Comput. Sci. 11 (2015) 279–288.
[58] S.M. Vahidipour, M.R. Meybodi, M. Esnaashari, Adaptive Petri net based on irregular cellular learning automata with an application to vertex coloring problem, Appl. Intell. 46 (2017) 272–284.
[59] M. Esnaashari, M.R. Meybodi, Irregular cellular learning automata, IEEE Trans. Cybern. 45 (2015) 1622–1632.
[60] H. Beigy, M.R. Meybodi, Cellular learning automata with multiple learning automata in each cell and its applications, IEEE Trans. Syst. Man Cybern. Part B Cybern. 40 (2010) 54–65.
[61] H. Beigy, M.R. Meybodi, Open synchronous cellular learning automata, Adv. Complex Syst. 10 (2007) 527–556.
[62] H. Beigy, M.R. Meybodi, Asynchronous cellular learning automata, Automatica 44 (2008) 1350–1357.
[63] M. Esnaashari, M.R. Meybodi, Deployment of a mobile wireless sensor network with k-coverage constraint: a cellular learning automata approach, Wirel. Netw. 19 (2013) 945–968.
[64] A.M. Saghiri, M.R. Meybodi, An approach for designing cognitive engines in cognitive peer-to-peer networks, J. Netw. Comput. Appl. 70 (2016) 17–40.
[65] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (2008) 046110.
[66] D.M. Christopher, R. Prabhakar, S. Hinrich, Introduction to Information Retrieval, Cambridge University Press, 2008.
[67] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (2004) 066111.
[68] C. Pizzuti, Ga-net: a genetic algorithm for community detection in social networks, in: Int. Conf. Parallel Probl. Solving Nat., Springer, 2008, pp. 1081–1090.
[69] R. Shang, J. Bai, L. Jiao, C. Jin, Community detection based on modularity and an improved genetic algorithm, Phys. Stat. Mech. Appl. 392 (2013) 1215–1231.

Mohammad Mehdi Daliri Khomami received the M.S. degree Computer Engineering from Department of electrical and computer engineering at Qazvin Islamic Azad University. He is currently pursuing Ph.D. degree in Computer Engineering at the Computer Engineering & Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Iran. His research interests include learning automata, social network analysis and optimization with application to problems from graph theory.

Alireza Rezvanian was born in Hamedan, Iran, in 1984. He received the B. Sc. degree from Bu-Ali Sina University of Hamedan, Iran, in 2007, the M. Sc. degree in Computer Engineering with honors from Islamic Azad University of Qazvin, Iran, in 2010, and the Ph.D. degree in Computer Engineering at the Computer Engineering & Information Technology Department from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2016. Currently, he works as a researcher in School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. Prior to the current position, he joined the Department of Computer Engineering and Information Technology at Hamedan University of Technology, Hamedan, Iran as a lecturer. He has authored or co-authored more than 70 research publications in reputable peer-reviewed journals and conferences including IEEE, Elsevier, Springer, Wiley and Taylor & Francis. He has organized various special sessions in international conferences in his area of expertise in Iran and abroad. He is a member of Technical Program Committees (TPCs) of various IEEE sponsored conferences in Iran and abroad. He has been a reviewer of many reputable conferences and journals. His research activities include soft computing, evolutionary algorithms, complex social networks and learning automata.

Mohammad Reza Meybodi received the B.S. and M.S. degrees in Economics from the Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degrees from the Oklahoma University, USA, in 1980 and 1983, respectively, in Computer Science. Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA.H is current research interests include, learning systems, cloud computing, soft computing and social networks.