International Journal of Modern Physics B
Vol. 30, No. 0 (2016) 1650042 (20 pages)

DOI: 10.1142/S0217979216500429

# Distributed learning automata-based algorithm for community detection in complex networks

Mohammad Mehdi Daliri Khomami

*Computer Engineering and Information Technology Department,*
*Islamic Azad University, Qazvin branch, Qazvin, Iran*
*daliri.mojtaba@gmail.com*

Alireza Rezvanian* and Mohammad Reza Meybodi†

*Soft Computing Laboratory, Computer Engineering and Information Technology,*
*Department Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran*
**a.rezvanian@aut.ac.ir*
*†mmeybodi@aut.ac.ir*

Received 30 August 2015
Revised 2 December 2015
Published MM DD 2016

Community structure is an important and universal topological property of many complex networks such as social and information networks. The detection of communities of a network is a significant technique for understanding the structure and function of networks. In this paper, we propose an algorithm based on distributed learning automata for community detection (DLACD) in complex networks. In the proposed algorithm, each vertex of network is supposed to be isomorphic with learning automata. According to the cooperation among network of learning automata and updating action probabilities of each automaton, the algorithm interactively tries to identify high-density local communities. The performance of the proposed algorithm is investigated through a number of simulations on popular synthetic and real networks. Experimental results in comparison with popular community detection algorithms such as walk trap, Danon greedy optimization, Fuzzy community detection, Multi-resolution community detection and label propagation demonstrated the superiority of DLACD in terms of modularity, NMI, performance, min-max-cut and coverage.

*Keywords*: Complex networks; online social networks; social network analysis; community detection; distributed learning automata.

PACS numbers: Author to provide

## 1. Introduction

Many complex phenomena in real-world systems are modeled and represented as networks such as ecological, biological, technological and information and social networks.[1–3] These real-world networks can be defined as a graph with a set of vertices (e.g., users in an online social networks) and edge-sets (e.g., a particular type of relationship between actors in an online social networks).[4] It is shown that in variants of real-world networks, there are common universal characteristics such as small world phenomena,[5] small shortest path lengths in average manner,[6] power low degree distribution[7] and specially existence of community structures in networks.[8] Community structure refers to a set of vertices whose edges inside are more dense than edges outside.[9] Finding community in a network known as a community detection problem (also called community structure identification or cluster finding). Community detection plays a significant role for studying and understanding the structure and function of real-world networks including several various domains such as clustering web stations having similar functionality and are geographically near to each other may improve the performance of services provided on the World Wide Web.[10] Online social networks often consist of a set of communities based on common properties of users such as locations, hobbies, interests, activities and carriers.[11] Citation networks form communities on the basis of similar research interest topics between authors.[12] Communities (modules) in protein–protein interaction (PPI) networks may correspond to known or even unknown functional modules/protein complexes with significant functionality in cellular biological systems.[13] Therefore, the detection of the community structure in a network has important practical applications and can help researchers to understand the organization and function of complex network systems.

Detecting the communities in complex networks due to the wide spread of applications have received a great attentions in literature by scholars.[14–17] A good review for community detection consists of techniques and applications is the one presented by Fortunato.[8] This paper classified community detection methods into five categories including traditional algorithms, hierarchical algorithms, modularity-based methods, spectral algorithms and dynamic algorithms. Among all types of community detection approaches, hierarchical clustering techniques are widely used techniques which put similar vertices into larger communities. Hierarchical clustering algorithms including two categories of divisive and agglomerative form the communities gradually in a hierarchical manner. Several scholars improved the hierarchical algorithms using some metrics to select a suitable partition or a proper set of partitions that satisfies particular metrics such as the number of desired communities, the maximum (or minimum) number of vertices in each community and optimize an objective function.[8] Divisive techniques try to find the edges that connect vertices of different communities and iteratively eliminate them, so that the communities separated from each other. Girvan and Newman have introduced a famous divisive method[12] which includes the removal of the edges based on their values of edge betweenness. An agglomerative method, however, tries to form communities in a bottom up manner. In general, the common notion of agglomerative methods is to partition vertices into communities iteratively starting from a partition in which communities are composed of a single vertex. The process of partitioning vertices continues until a single community consists of all vertices of input network is achieved.[18] Most of the agglomerative algorithms select the best partition that maximizes a typical quality objective function. One of the best known quality function and the most widely used quality function is a *modularity* metric which is proposed by Newman *et al.*[18] Using modularity, Clauset *et al.* proposed a fast greedy modularity optimization method[19] which starting from a set of isolated vertices and then a pair of vertices iteratively connected to each other such that it achieves the maximum possible value of modularity at each step. Although using this measure by modularity optimization achieves many promising results for community detection, it is shown that it has some limitations such as resolution limit. For example in the extreme case, the modularity optimization algorithms are failed for a network with several cliques connected by a single edge.[20]

Another direction of research for community detection algorithms is devoted to random walk-based methods. Random walk-based method consider not only the topological structure of network but also the local neighborhood properties of vertices. So it can be proper for community detection in some types of complex networks. Hagen *et al.*[21] used random walks to detect cluster circuits in VLSI design. They defined a concept of cycle to find communities in circuit network. A new clustering method for complex network called CONCLUDE is presented by De Meo*et al.*[22] which combines the scalability of local algorithms and accuracy of global approaches. Harel *et al.*[23] introduced a new definition based on random walks for community. Then, they proposed two operators that change the weights of the edges and reduce the weights of inter-community edges. By enforcing them iteratively the inter-community edge weights are reduced until they are trivial. Finally, the community structure is revealed by omitting the near-zero weight edges. In Ref. 25, a divisive methods presented by employing edge centrality based on random walks. It first removes the connected edges with a leaf vertex, and then finds edge bridges between communities and removes them. Some methods try to enhance the performance of random walk=based methods by combining random walks with well-known algorithms such as $k$-path edge-centrality.[26] However, these hybrid methods have a high time complexity which is infeasible for real application of real-world networks. Another efficient random walk-based algorithm is walk trap (WT)[27] in which the distance between two vertices is defined in terms of random walk process. The basic idea is that if two vertices, $i$ and $j$, are in the same community, the probability to get a third vertex $k$, in the same community through a random walk should not be very different for $i$ and $j$.

In this paper, an algorithm based on distributed learning automata is proposed for detecting communities in complex networks. Since the distributed learning automata serves in a stochastic environment equivalent with unknown communities in the environment of complex networks, according to the capabilities of learning automata for solving many complicated problems, it can be useful for finding community in complex networks. In the proposed algorithm, a learning automaton is assigned to each vertex of the network, during the execution of the proposed algorithm, a set of learning automata cooperate with each other to find proper communities in the given network. In order to study the performance of the proposed algorithm, several experiments conducted on the well-known complex network datasets. And then, the proposed algorithm is compared with several community detection algorithms such as WT[27] *Danon* greedy optimization (DGO),[28] fuzzy community detection (FCD),[29] multi-resolution community detection (MRCD)[30] and label propagation.[31] Experimental results show that the proposed algorithm outperforms the other community detection algorithms in terms of modularity, NMI, performance, min-max-cut and coverage.

The rest of this paper is organized as follows. In Sec. 2, the learning automata and distributed learning automata are introduced in brief. The proposed community detection algorithm based on distributed learning automata for complex networks is described in Sec. 3. In Sec. 4, the performance of the proposed algorithm is compared to other community detection algorithms on the well-known real and synthetic networks with respect to some well-known measures. Finally, Sec. 5 concludes this paper.

## 2. Learning Automata

In this section, we briefly describe the learning automata and the interested reader may refer to Ref. 32 for more information about learning automata models, algorithms, application and theoretical aspects. A learning automaton (LA)[32] is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. The action is chosen at random based on a probability distribution kept over the action-set and at each instant the given action is served as the input to the random environment. The environment responds the taken action in turn with a reinforcement signal. The action probability vector is updated based on the reinforcement feedback from the environment. The objective of a LA is to find the optimal action from the action-set so that the average penalty received from the environment is minimized. The environment can be described by a triple $E = \{\alpha, \beta, c\}$ where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the finite set of the inputs, $\beta = \{\beta_1, \beta_2, \ldots, \beta_m\}$ denotes the set of values can be taken by the reinforcement signal and $c = \{c_1, c_2, \ldots, c_m\}$ denotes the set of the penalty probabilities, where the element $c_i$ is associated with the given action $\alpha_i$. If the penalty probabilities are constant, the random environment is said to be a stationary random environment and if they vary with time, the environment is called a nonstationary environment. The environments depending on the nature of the reinforcement signal $\beta$ can be classified into P-model, Q-model and S-model. The environments in which the reinforcement signal can only take two binary values 0 and 1 are referred to as P-model environments. Another class of the environment allows a finite number of the values in the interval $[0, 1]$ can be taken by the reinforcement signal. Such an environment is referred to as Q-model environment. In S-model environments,

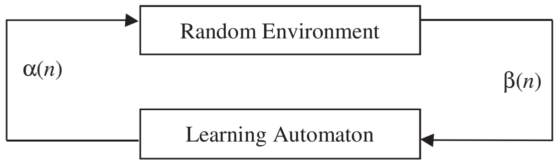



Fig. 1. The relationship between the learning automata and its random environment.

the reinforcement signal lies in the interval $[0, 1]$. The relationship between the LA and its random environment has been shown in Fig. 1.

Learning automata can be classified into two main families: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $\langle \beta, \alpha, T \rangle$ where $\beta$ is the set of inputs, $\alpha$ is the set of actions and $T$ is learning algorithm. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let $\alpha(k)$ denotes the action chosen at instant $k$ and $p(k)$ being the corresponding action probability vector. Suppose $a(k)$ and $b(k)$ denote the reward and penalty parameters, respectively and let $\alpha_i(k)$ be the action chosen by the automata at instant $k$. When the taken action is rewarded by the environment (i.e., $\beta(k) = 0$), the action probability vector, $p(k)$, is updated by the recurrence equation

$$p_j(k+1) = \begin{cases} p_j(k) + a(k)[1 - p_j(k)] & \text{if } i = j\,, \\ p_j(k) - a(k)p_j(k) & \text{if } i \neq j \end{cases} \tag{1}$$

which is a linear learning algorithm. Similarly, in case of penalizing (i.e., $\beta(k) = 1$), the update formula would be

$$p_j(k+1) = \begin{cases} p_j(k)(1 - b(k) & \text{if } i = j \\ \dfrac{b}{r-1} + p_j(k)(1 - b(k)) & \text{if } i \neq j\,. \end{cases} \tag{2}$$

In (1) and (2), $r$ is the number of actions that can be chosen by the automaton.

Based on the value of $a(k)$ and $b(k)$, three types of learning algorithm have been defined. If the $a(k)$ equals to $b(k)$ then recurrence Eqs. (1) and (2) is called linear reward penalty ($L_{R-P}$) algorithm, if $a(k) \gg b(k)$ the given equations are called linear reward-$\varepsilon$ penalty ($L_{R-\varepsilon P}$) and finally if $b(k) = 0$ they are called linear reward-Inaction ($L_{R-I}$). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment.

### 2.1. *Variable action set learning automata*

If the number of action set for a learning automata change over the time, it is called a variable action set learning automata. In variable action set a learning automata consist of finite set of $n$ actions, $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. Let $A = \{A_1, A_2, \ldots, A_m\}$ be the set of actions which can be selected and $A(k) \subseteq \alpha$ be the subset of all actions can be selected by a specific learning automata, at each instance $k$. According to the probability distribution $q(k) = \{q_1(k), q_2(k), \dots, q_m(k)\}$, an external agency chooses a particular action subset randomly, where, $q_i(k) = \text{prob}[A(k) = A_i | A_i \in A, 1 \le i \le 2^n - 1]$. Moreover, $\hat{p}_i(k)$ is defined as the probability of choosing action $\alpha_i$, if the action subset $A(k)$ has already been selected and $\alpha_i \in A(k)$. The scaled probability $\hat{p}_i(k)$ is also defined as

$$\hat{p}_i(k) = \frac{p_i(k)}{\mathbf{K}(k)}, \tag{3}$$

where $\mathbf{K}(k) = \sum_{\alpha_i \in A(k)} p_i(k)$ is the sum of the probabilities of the actions in subset $A(k)$ and $p_i(k) = \text{prob}[\alpha(k) = \alpha_i]$.

### 2.2. *Distributed learning automata*

In this section, we introduce a new model of interconnected automata calling distributed learning automata (DLA).[33] Distributed learning automata is a network of learning automata which collectively cooperate to solve a particular problem with many applications including: Vehicle Routing Problem,[34] wireless networks,[35–37] data mining,[38] information retrieval,[39] complex networks[2,40] and grid computing.[41] A DLA can be modeled by a directed graph in which the set of vertices of graph constitutes the set of actions for corresponding automaton. When a LA selects one of its actions, another LA on the other end of edge corresponding to the selected action will be activated. Formally, a DLA with $n$ learning automata can be defined by a graph $(A, E)$, where $A = \{A_1, A_2, \dots, A_n\}$ is the set of learning automata and $E \subset A \times A$ is the set of edges in the graph for which an edge $(i, j)$ corresponds to action $\alpha_i^j$ of automaton $A_i$. Let action probability vector for LA $A_j$ be represented by $p_j$ where a component $p_j^m$ of $p^j$ denotes the probability of choosing action $\alpha_j^m$ that is the probability of choosing edge $(j, m)$. An example of DLA is given in Fig. 2, in which every automaton has two actions. If automatons $A_1$ select action $\alpha_1^2$ then automaton $A_2$ will be activated. Activated automaton $A_2$ chooses one of its actions which in turn it activates one of the automata connected to $A_2$. At any time only one automaton in the network will be activated.

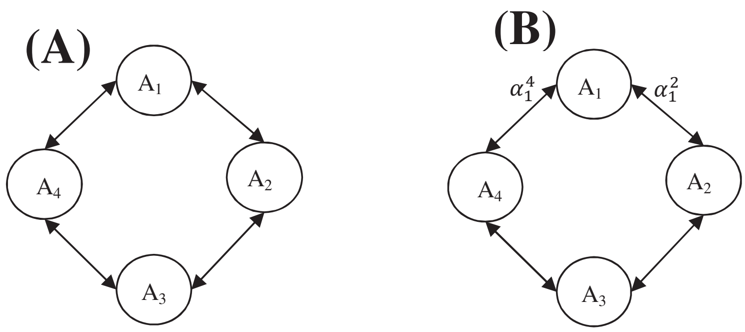



Fig. 2. Distributed learning automata.

## 3. Proposed Community Detection Algorithm Based on Distributed Learning Automaton

In this section, we describe the proposed algorithm based on distributed LA for finding communities in complex networks. It is assumed that the input network $G = \langle V, E\rangle$ is an undirected and un-weighted network where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges in the given network. The proposed distributed LA algorithm including four steps tries to find iteratively a set of communities that are more densely connected internally to each other than to the rest of the network. After the initialization step is performed, by assigning a LA to the vertices of input network, the proposed algorithm repeats community finding by doing a guided traversal in the network with the help of DLA, evaluates the set of found communities and updates their action probability vectors iteratively until stopping criteria are satisfied. We describe four steps of the proposed algorithm in the following subsections in detail.

### 3.1. *Initialization*

In the first step, a DLA $\langle A, \alpha\rangle$ which is isomorphic to the input network is constructed. The resulting network can be defined by 2-tuple $\langle A, \alpha\rangle$ where $A = \{A_1, A_2, \ldots, A_n\}$ is the set of learning automata corresponding to the set of vertices, $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ denotes the set of actions in which $\alpha_i = \{\alpha_i^1, \alpha_i^2, \ldots, \alpha_i^{r_i}\}$ are the set of actions that can be taken by LA $A_i$ and $r_i$ is the number of actions that can be taken by LA $A_i$. An action of a LA $A_i$ corresponds to choosing an adjacent vertex of the corresponding vertex $v_i$. Let $p(v_i) = \{p_i^1, p_i^2, \ldots, p_i^{r_i}\}$ be the action probability vector of LA $A_i$ and $p_i^j = 1/r_i$ equally initialized for all $j$. At all iterations each LA can be in either active or inactive mode. At the beginning of the proposed algorithm all learning automata initially are set in inactive mode. Let $C_k$ be the set of vertices in the $k$th community and initially is set to be empty, $G'$ represent the set of unvisited vertices in the execution of algorithm and is initially equal to $G$ and also $\pi^t$ is the path of visited vertices at the iteration $t$.

### 3.2. *Finding communities*

At $t$th iteration of this step, the algorithm finds $k$ communities in such a way that the proposed algorithm starts with randomly selecting vertex $v_i$ among unvisited vertex set $G'$ and the selected vertex $v_i$ is inserted in the set of current community $C_k$ and current path $\pi^t$. Then, LA $A_i$ corresponds to the starting vertex $v_i$ is activated and chooses one of adjacent vertex $v_i$ according to its action probability vector. Let the chosen action by LA $A_i$ be vertex $v_j$. If the number of internal connections for union of selected vertex $v_j$ and current community $C_k$ is greater than the number of internal connections for current community $C_k$ then $v_j$ is inserted to the set of current community $C_k$, $C_k$ is removed from set $G'$ and also visited vertex $v_j$ is updated in path $\pi^t$. The process of activating an automaton, choosing an action, checking the condition of inserting chosen vertex $v_j$ in the current community $C_k$, inserting new vertex $v_j$ to $C_k$, updating visited vertex $v_j$ in path $\pi^t$ and removing $C_k$ from set $G'$ is repeated until total number of edges inside the current community $C_k$ is more than the total number of edges outside the current community $C_k$ or active LA could not select any action. The process of finding new communities according to the above description and updating path of visited vertices at the current iteration $\pi^t$ is continued when the union of all vertex-set of found communities is equal to the input network $G$.

### 3.3. *Computing objective function*

Let $C^t = \{C_1, C_2, \ldots, C_k\}$ be the set of $k$ communities found at the iteration $t$. The quality of the set of communities found at the iteration $t$ is evaluated via normalized cut as objective function[42] by following equation

$$\mathrm{NC}(C^t) = \frac{1}{k}\sum_{i=1}^{k}\frac{\mathrm{cut}(C_i, \bar{C}_i)}{\mathrm{vol}(C_i)}\,, \tag{4}$$

where $\mathrm{cut}(C_i, \bar{C}_i)$ denotes the number of edges between communities $C_i$ and $\bar{C}_i = GC_i, \mathrm{vol}(C_i)$ is the total degree of vertices that are the members of community $C_i$ and also $k$ is the number of communities. Since mentioned in Ref. 47, normalized cut with low complexity considers extracting the global impression of the network, instead of local features and measures both the total dissimilarity between the different communities as well as the total similarity within the communities. So, using the proposed algorithm gradually decreased normalized cut which means that the algorithm gradually converges to the minimum normalized cut and approach to the proper set of communities.

### 3.4. *Updating action probability vectors*

In this step, the set of $k$ communities found at the iteration $t$ is evaluated via corresponding normalized cut and if the value of the normalize cut at current iteration $\mathrm{NC}(C^t)$ is less than or equal to the value of normalized cut at previous iteration $\mathrm{NC}(C^{t-1})$, then the chosen action along the path $\pi^t$ by all the activated learning automata are rewarded according to the learning algorithm described in Sec. 2 and penalized otherwise.

### 3.5. *Stopping criteria*

The proposed algorithm iterates steps 2, 3 and 4 until the number of iterations exceeds a given threshold $T$ or $\in P_t = \prod_{v_i \in C^t} (p_i^j)$ at iteration $t$ becomes greater than a particular threshold $\tau$ where $p_i^j$ is the probability of choosing neighboring vertex $v_j$ by LA $A_i$ residing in vertex $v_i$ and $N(v_i)$ is the set of neighboring vertex $v_i$.

Figure 3 shows the pseudo-code for the proposed community detection algorithm for complex networks.

**Algorithm 1.** proposed algorithm for community detection in complex networks

**Input**: A network $G$=($V$, $E$), Thresholds $\tau$, $T$ // $\tau$: stopping threshold for product of probabilities, T:maximum iteration number
**Output**: Set of found communities $C^*$
**Assumptions**
Assign an automaton $A_i$ to each vertex $v_i$;
Let $k$ is the number of communities;
Let $C_k$ is the set of $k^{\text{th}}$ community and initially set to empty;
Let $t$ is the iteration number of algorithm and initially set to 0;
Let $NC(C^t)$ is the normalized cut value for set of communities found at iteration $t$ and initially set to 0;
Let $P_t$ is the product of maximum probabilities in probability vector of learning automata the vertices of a set of communities at iteration $t$ and initially set to 0;
Let $\pi^t$ is the path of visited vertices by the algorithm at iteration $t$ and initially set to empty;
**Begin**
$G' \leftarrow G$; //set of unvisited vertices
All learning automata initially are set inactive mode;
**While** ($t<T$ **or** $P_t<\tau$)
**Repeat**
$t \leftarrow t+1$;
$k \leftarrow 1$;
Vertex $v_i$ is selected randomly from $G'$;
$C_k \leftarrow C_k \cup v_i$;
$\pi^t \leftarrow \pi^t \cup v_i$;
**While** ($d_{\text{in}}(C_k) < d_{\text{out}}(C_k)$ **AND** $|\alpha_i| \neq 0$) **Do** // Finding $s^{\text{th}}$ community
Automaton $A_i$ is activated and then choose an action using its action probability vector;
Let the chosen action by $A_i$ be $v_j$;
**If** ($d_{\text{in}}(C_k \cup v_j) > d_{\text{in}}(C_k)$ **AND** $d_{\text{out}}(C_k \cup v_j) < d_{\text{out}}(C_k)$) **then** //checking internal and external connections
$C_k \leftarrow C_k \cup v_j$;
$\pi^t \leftarrow \pi^t \cup v_j$;
$v_i \leftarrow v_j$;
**End If**
**End while**
$G' \leftarrow G' \setminus C_k$;
$k \leftarrow k+1$;
**Until** ($|G'| \neq 0$)
Compute $NC(C^t)$ according to equation (4);
**If** ($NC(C^t) < NC(C^{t-1})$)
Reward the actions chosen along the path $\pi_t$ by all the activated learning automata;
**Else**
Penalize the actions chosen along the path $\pi_t$ by all the activated learning automata;
**End If**
Compute $P_t = \prod_{v_i \in C^t} \max_{v_j \in N(v_i)} (p_i^j)$;
Set all learning automata in inactive mode;
**End while**

$C^* \leftarrow C^{t;}$
**End Algorithm**

Fig. 3. Pseudo-code of proposed algorithm for community detection in complex network.

Table 1. Description of the test networks used for the experiments.

| Networks | Vertex | Edge | Description |
|---|---|---|---|
| Karate | 34 | 78 | Zachary karate club network. |
| Dolphins | 62 | 159 | The network of dolphins. |
| Les-miserables | 77 | 254 | Co-appearance network of characters in the novel Les Miserables. |
| Books | 105 | 441 | The network of American Politics Books. |
| Football | 115 | 615 | Network of American college football teams. |
| Reactome | 6327 | 147,547 | Network of PPI in Humans. |
| LFR1 | 5000 | 38,160 | Synthetic modular benchmark. |
| LFR2 | 5000 | 250,000 | Synthetic modular benchmark. |

## 4. Simulation Results

In order to study the performance of the proposed DLA based algorithm for community detection (DLACD), we conducted a number of experiments on the well-known real and synthetic modular networks. Table 1 describes the set of real test networks that are used for experiments including the popular real networks: Karate,[43] Dolphins,[44] Books,[45] Football[12] Reactome[41] and Les-miserables[45] and also LFR1 and LFR2 as LFR benchmark networks[42] for synthetic modular networks. To assess the performance of the proposed algorithm and other community detection algorithms, we applied several commonly used standard measures consist of modularity, coverage, min-max-cut, performance and NMI. In the next subsection these measures for assessing finding communities will be described in brief.

### 4.1. *Evaluation measures*

In the following subsections, we will describe evaluation measures for assessing found communities by the algorithms.

#### 4.1.1. *Modularity*

The first measure is *Modularity* $Q$[18] as the popular measure for evaluating set of communities in network found by the algorithm. This measure is defined as follows

$$Q = \frac{1}{2m} \sum_{C \in P} \sum_{v_i, v_j \in C} \left[ A_{i,j} - \frac{k_i k_j}{2m} \right], \tag{5}$$

where $A$ is the adjacency matrix that $A_{i,j}$ is equal to 1 if there is an edge between vertex $v_i$ and vertex $v_j$ and zero otherwise. $k_i = \sum_j A_{ij}$ is the degree of vertex $v_i$ and $m$ is the total number of edges in the network. The summation is over all pairs of vertices that member of the same community $C$ of partitioning $P$.

#### 4.1.2. *Coverage*

The *coverage* measure[44] defined as the fraction of edges inside a community compared with total number of edges in the whole network and defined as following

$$\text{Coverage} = \frac{\sum_{i=1}^{k} E_i}{E}\,, \tag{6}$$

where $k$ is the number of communities obtained from algorithm. In this fashion a good partitioning of the network could maximize the value of coverage.

#### 4.1.3. *Min-max-cut*

The *min-max-cut*[45] is the fraction of between community edges compared to the number of edges associated to the different communities. The minimizing value of the Min-Max-Cut results in having a good partitioning of communities. The min-max-cut is defined by

$$\min - \max - \text{cut} = \sum_{i=1}^{k} \frac{E'_i}{E_i}\,, \tag{7}$$

where $k$ is the number of communities obtained by the algorithm, $E_i$ is the number of edges of community $C_i$ and $E'_i$ is the number of edges between community $C_i$ and the rest of communities.

#### 4.1.4. *Performance*

The *performance*[44] evaluates the rate of edges that set on the basis of the partitioning of the network well. Performance is calculated as follows

$$\text{performance} = 1 - \frac{2(T+F)}{n(n-1)}\,, \tag{8}$$

where $T$ is the number of nonadjacent pairs $(v_i, v_j)$ in the same community and $F$ is the number of edges between community. Hence, a good partitioning could maximized the performance.

#### 4.1.5. *Normalized mutual information*

*Normalized mutual information* (NMI)[43] measures the similarity between known set of communities and set of communities found by the algorithm, where $A$ and $B$ are two partitions of the input network, and NMI is a value between $[0, 1]$, the higher value indicates the partitions $A$ and $B$ are totally independent.

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

This measure is useful for artificial networks with a prior knowledge about built-in communities.

### 4.2. *Experimental setup*

To investigate the performance of the proposed algorithm, we employed several simulations on the well-known real and synthetic networks. In all experiments presented in this paper, the learning scheme is $L_{R-I}$ and the learning rate is set to be 0.02. The maximum threshold $\tau$ is set to 0.9 and maximum iteration $T$ is set to $n \times 1000$ where $n$ is the number of vertices of graph. For LFR benchmark parameters, $N$ is the number of vertices, $K$ is average degree, $\max_k$ is maximum degree of nodes, $\min_c$ is minimum size of communities, $\max_c$ is maximum size of communities and $\mu$ is mixing parameter. Each vertex shares a fraction of $1-\mu$ connections with other vertices in its community and a fraction $\mu$ with other vertices of the network. For synthetic modular networks, we set LFR benchmark network parameters as $N = 5000$, $K \in \{15, 100\}$, $\max_k = 50$, $\mu = \{0.1, 0.5\}$, $\min_c = 10$ and $\max_c = 50$.

### 4.3. *Experimental results*

#### 4.3.1. *Experiment I*

This experiment is carried out to investigate the impact of learning rate $a$ on the performance of the proposed community detection algorithm. The normalized cut value for the real and synthetic networks for varying learning rates from 0.01 to 0.1 with step size 0.01 is given in Table 2. From the results of this experiment, we may conclude that the performance of the proposed community detection algorithm completely depends on the choice of the learning rate $a$. For Karate, Dolphins, Books, Les-miserable networks, small values of learning rate results in higher performance while large values results in lower performance. However, for Football, Reactome and LFR2 networks, larger values for learning rate results in higher performance. This may be because of their structure of these networks. It is noted that for the rest of experiments conducted in this paper, the learning rate is set to 0.02.

#### 4.3.2. *Experiment II*

This experiment is performed to study the behavior of the proposed algorithm during the process of finding the set of communities and the effect of learning in the proposed algorithm by replacing the LA residing in each vertex by a pure chance automaton. In pure chance automaton, the initial probability of choosing actions is always remaining unchanged.[32] The comparison is made with respect to normalized cut $NC(C^t)$. Note that $NC(C^t)$ is the average of the normalized cut value for set of $k$ communities found up to iteration $t$ [Eq. (4)]. The plot of $NC(C^t)$ versus iteration for different real networks demonstrated in Fig. 4 indicates the important role of learning automata in guiding the process of finding communities in real

Table 2. Result of different learning rate in terms of normalized cut.

| Learning rate | Networks | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Karate | Dolphins | Books | Les-miserables | Football | Reactome | LFR1 | LFR2 |
| 0.01 | 0.4309 | 0.3877 | 0.4112 | 0.3680 | 0.4003 | 0.5643 | 0.5812 | 0.5858 |
| 0.02 | 0.3443 | 0.3622 | 0.3291 | 0.3273 | 0.4934 | 0.5642 | 0.5982 | 0.5925 |
| 0.03 | 0.4072 | 0.4082 | 0.3449 | 0.3796 | 0.5069 | 0.5647 | 0.5794 | 0.4635 |
| 0.04 | 0.3825 | 0.4635 | 0.3506 | 0.4643 | 0.3954 | 0.5353 | 0.5813 | 0.4423 |
| 0.05 | 0.4457 | 0.3094 | 0.3293 | 0.4032 | 0.4000 | 0.5217 | 0.5383 | 0.4024 |
| 0.06 | 0.4429 | 0.3765 | 0.4104 | 0.3720 | 0.4053 | 0.5225 | 0.5544 | 0.4310 |
| 0.07 | 0.4456 | 0.3863 | 0.3746 | 0.5375 | 0.3711 | 0.5141 | 0.5984 | 0.4211 |
| 0.08 | 0.4421 | 0.4171 | 0.3496 | 0.5877 | 0.3533 | 0.5111 | 0.5469 | 0.4214 |
| 0.09 | 0.4423 | 0.3654 | 0.4101 | 0.5119 | 0.3863 | 0.5082 | 0.5613 | 0.4199 |
| 0.10 | 0.3936 | 0.3783 | 0.3609 | 0.4348 | 0.4123 | 0.5025 | 0.5658 | 0.4214 |

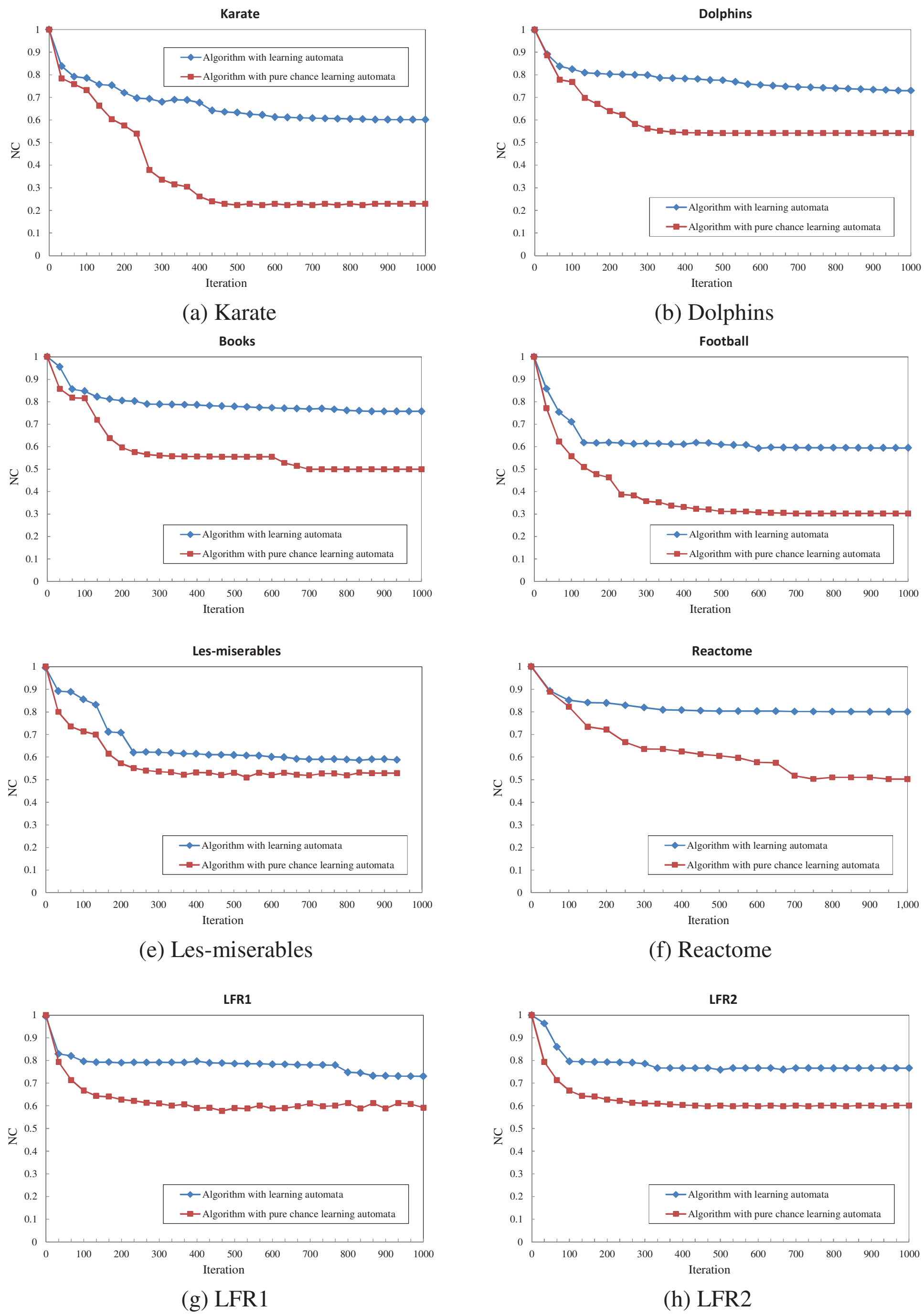
and synthetic networks. With the aid of learning automata, the process of finding communities from the network is done faster and more accurate as compared to the same case in the absence of learning. Also, since, $NC$ considers extracting the global impression of the network, instead of local features and measures both the total dissimilarity between the different communities as well as the total similarity within the communities. So, $NC$ gradually decreases which means that the algorithm gradually converges to the minimum normalized cut and approach to the proper set of communities.

#### 4.3.3. *Experiment III*

In this experiment, we investigate the impact of initial vertex selection of the proposed algorithm for finding communities in real and synthetic networks. For this purpose, we compared a high degree vertex and a random initial vertex as the initial selection in the proposed algorithm. The plot of $NC(C^t)$ for high degree initial vertex selection and random initial vertex selection versus iteration is presented in Fig. 5. From the results one can conclude that the proposed algorithm with high degree initial vertex selection has higher convergence speed than the proposed algorithm with random initial vertex selection for all networks. This may be because of process of initial formation of communities for each network affects the process of formation of rest of communities. In this regard, the vertex with highest degree may be member of the initial main community.

#### 4.3.4. *Experiment IV*

In order to evaluate the performance of different community detection algorithms, we provide an experiment on the known synthetic modular networks based on LFR benchmark. In this experiment, the LFR benchmark network parameters are set as $N = 5000$, $K = 15$, $\max_k = 50$, $\mu = \{0.1, 0.5\}$, $\min_c = 10$, $\max_c = 50$ and the mixing parameter is changed from 0.05 to 0.5 with 0.05 increment. Several community detection algorithms including WT,[27] LP,[46] DGO,[28] FCD[29] and MRCD[30] are

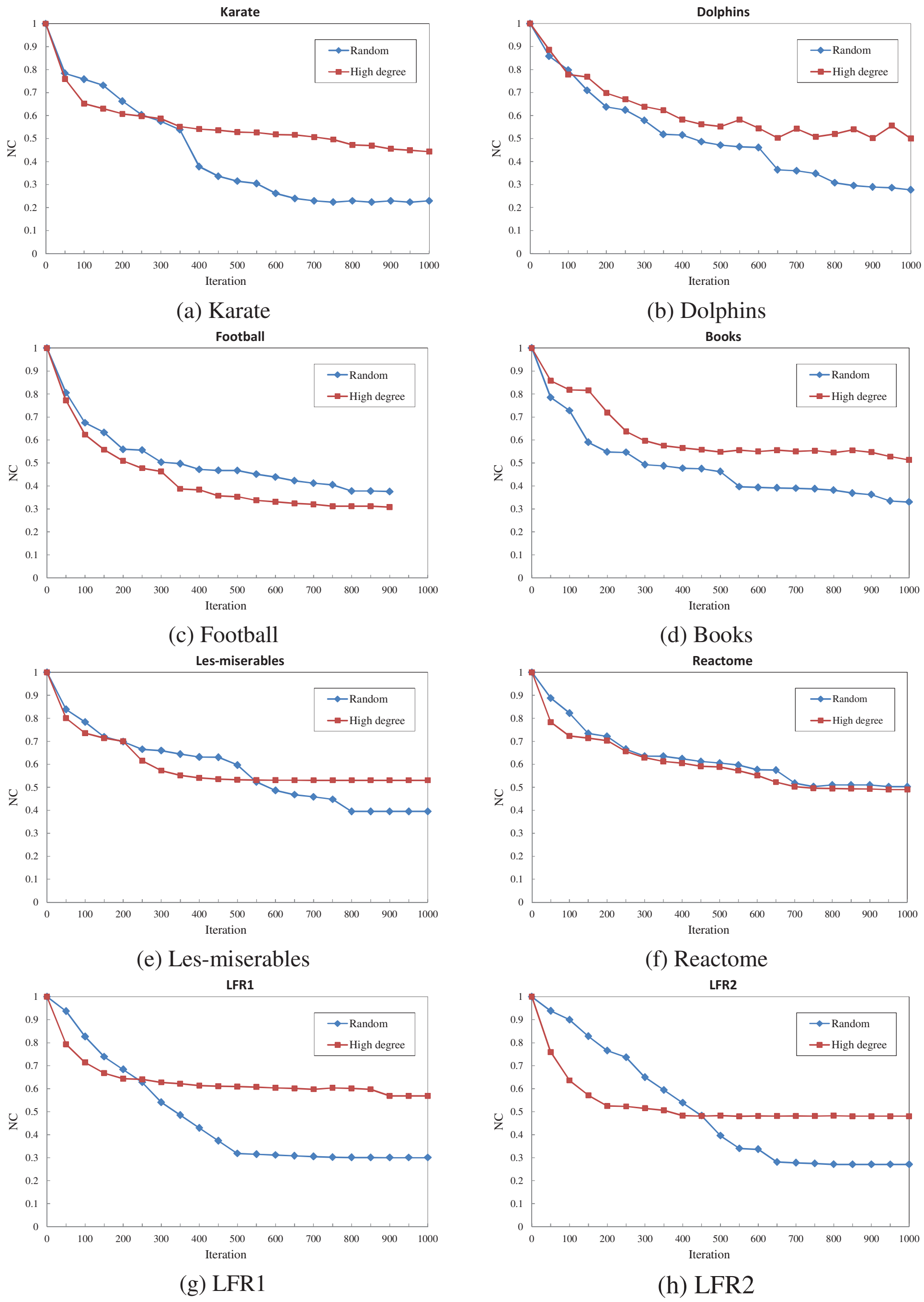



(a) Karate (b) Dolphins

(e) Les-miserables (f) Reactome

(g) LFR1 (h) LFR2

Fig. 4. Comparison of proposed community detection algorithm with proposed community detection algorithm in which learning automata are replaced with pure chance automata.




(a) Karate (b) Dolphins

(c) Football (d) Books

(e) Les-miserables (f) Reactome

(g) LFR1 (h) LFR2

Fig. 5. Comparison of the proposed algorithm with random initial vertex selection versus the proposed algorithm with high degree initial vertex selection.

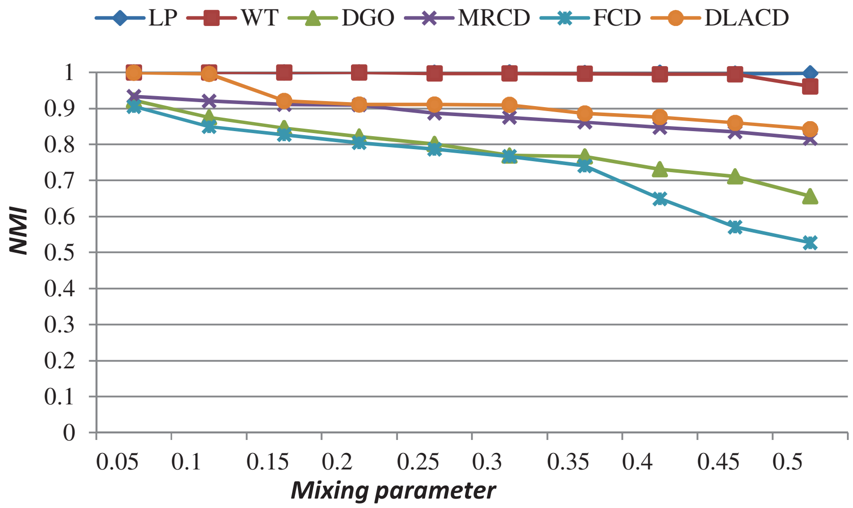



Fig. 6. The comparison results for synthetic LFR benchmark with different mixing parameter $\mu$.

conducted on mentioned synthetic modular networks and compared with respect to NMI results. As shown in Fig. 6, for the small size mixing parameter, all algorithms obtained almost same and high NMI values. As the complexity of the network is increased by the value of mixing parameter, the quality of each algorithm with respect to NMI is decreased. This figure also shows that the proposed community detection algorithm is better than MRCD, DGO and FCD in terms of NMI.

#### 4.3.5. *Experiment V*

This experiment is done to study the performance of the proposed community detection algorithm in comparison with other community detection algorithms including WT,[27] LP,[46] DGO,[28] FCD[29] and MRCD[30] in term of modularity, min-max-cut, coverage, performance value for real-world networks described in Table 1. The results of this experiment are given in Tables 3–6 for modularity, min-max-cut, coverage and performance, respectively. The best results for each network are highlighted in boldface. According to the results of Table 3, for Books and Les-miserables, the

Table 3. Comparison of the community detection algorithms in terms of modularity.

| | Networks | | | | | |
|---|---|---|---|---|---|---|
| Methods | Karate | Dolphins | Books | Les-miserable | Football | Reactome |
| MRCD | 0.267 | 0.376 | 0.337 | — | 0.504 | 0.621 |
| DGO | 0.383 | 0.522 | 0.523 | — | 0.566 | 0.652 |
| FCD | 0.360 | 0.389 | 0.442 | — | 0.476 | 0.524 |
| WqT | 0.407 | 0.519 | 0.52 | 0.521 | 0.602 | 0.733 |
| LP | 0.317 | 0.514 | 0.497 | 0.353 | 0.583 | 0.741 |
| DLACD | 0.384 | 0.349 | 0.526 | 0.524 | 0.584 | 0.651 |

Table 4. Comparison of the community detection algorithms of min-max-cut.

| | Networks | | | | | |
|---|---|---|---|---|---|---|
| Methods | Karate | Dolphins | Books | Les-miserable | Football | Reactome |
| MRCD | 1.094 | 1.012 | 1.227 | — | 0.707 | 1.475 |
| DGO | 0.452 | 0.303 | 0.179 | — | 0.338 | 1.842 |
| FCD | 0.149 | 0.063 | 0.067 | — | 0.197 | 1.756 |
| WT | 0.379 | 3.261 | 0.115 | 1.015 | 0.221 | 0.992 |
| LP | 0.379 | 3.170 | 0.067 | 0.426 | 0.445 | 0.845 |
| DLACD | 0.241 | 0.209 | 0.300 | 0.282 | 0.409 | 1.025 |

Table 5. Comparison of the community detection algorithms of coverage.

| | Networks | | | | | |
|---|---|---|---|---|---|---|
| Methods | Karate | Dolphins | Books | Les-miserable | Football | Reactome |
| MRCD | 0.477 | 0.498 | 0.433 | — | 0.585 | 0.641 |
| DGO | 0.683 | 0.769 | 0.848 | — | 0.747 | 0.645 |
| FCD | 0.864 | 0.940 | 0.936 | — | 0.835 | 0.654 |
| WT | 0.865 | 0.371 | 0.864 | 0.496 | 0.735 | 0.892 |
| LP | 0.865 | 0.477 | 0.936 | 0.700 | 0.691 | 0.880 |
| DLACD | 0.800 | 0.826 | 0.768 | 0.779 | 0.709 | 0.761 |

Table 6. Comparison of the proposed algorithm in term of performance value.

| | Networks | | | | | |
|---|---|---|---|---|---|---|
| Methods | Karate | Dolphins | Books | Les-miserable | Football | Reactome |
| MRCD | 0.921 | 0.947 | 0.954 | — | 0.936 | 0.759 |
| DGO | 0.812 | 0.852 | 0.833 | — | 0.877 | 0.755 |
| FCD | 0.694 | 0.697 | 0.755 | — | 0.782 | 0.789 |
| WT | 0.815 | 0.889 | 0.832 | 0.887 | 0.922 | 0.890 |
| LP | 0.815 | 0.855 | 0.755 | 0.750 | 0.926 | 0.899 |
| DLACD | 0.750 | 0.760 | 0.815 | 0.869 | 0.906 | 0.844 |

proposed community detection algorithm outperforms other algorithms in terms of modularity. Also, for other networks, the modularity of proposed algorithm is approximately in the same range as of other algorithms. It is noted that the DANON is designed for only modularity as objective function and thus it is biased toward high modularity's; therefore, it is not surprising that DANON reaches higher modularity in some networks, and this means that some more criteria must be considered to analyze the quality of algorithms accurately. The modularity of DLACD is low in some of the datasets, but it is still much higher than those of many metrics that are shown in the following. In Table 4, DLACD has the best value for Les-miserable real network and the proposed algorithm outperforms other algorithms in terms of min-max-cut for other networks such as Karate, Dolphins and Books. According to Table 5, DLACD has highest possible value compared to other algorithms for Dolphins and Les-miserable network meaning that the proposed algorithm could find an appropriate cut. The obtained results are negligible for other network datasets and most of the networks are in the same range. According to Table 6, the proposed algorithm reaches an acceptable value in set of the optimization-based algorithms. Note that the proposed community detection algorithm try to find optimal normalized cut value in the network and it is independent of some restrictions such as resolution limits.

## 5. Conclusion

Complex networks are composed by a large number of elements, organized into sub-communities. In this paper, an algorithm based on a distributed LA was proposed for community detection in complex networks. In the proposed algorithm, a set of learning automata was assigned into vertices of the given network. Then based on cooperation between automatons, the proposed algorithm tries to find optimal communities iteratively based on a learning mechanism. The performance of the proposed community detection algorithm was empirically investigated against the well-known community detection algorithms. According to the obtained results in several experiments, the proposed community detection algorithm outperforms the other algorithms in terms of the modularity, NMI, coverage, min-max-cut and performance in most cases.

## References

1. D. Easley and J. Kleinberg, *Networks, Crowds and Markets: Reasoning About a Highly Connected World* (Cambridge University Press, 2010).
2. A. Rezvanian, M. Rahmati and M. R. Meybodi, Sampling from complex networks using distributed learning automata, *Physica A* **396**, 224 (2014).
3. R. M. Tripathy, A. Bagchi and S. Mehta, Towards combating rumors in social networks: Models and metrics, *Intell. Data Anal.* **17**, 149 (2013).
4. R. Albert and A. L. Barabási, Statistical mechanics of complex networks, *Rev. Mod. Phys.* **74**, 47 (2002).
5. D. J. Watts and S. H. Strogatz, Collective dynamics of "small-world" networks, *Nature* **393**, 440 (1998).
6. J. Kleinberg, The small-world phenomenon: An algorithmic perspective, in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing* (ACM, 2000), pp. 163–170.
7. A. Clauset, C. R. Shalizi and M. E. Newman, Power-law distributions in empirical data, *SIAM Rev.* **51**, 661 (2009).
8. S. Fortunato, Community detection in graphs, *Phys. Rep.* **486**, 75 (2010).
9. R. Rabbany *et al.*, Communities validity: Methodical evaluation of community mining algorithms, *Soc. Netw. Anal. Min.* **3**, 1039 (2013).
10. B. Krishnamurthy and J. Wang, On network-aware clustering of web clients, in *ACM SIGCOMM Comput. Commun. Rev.* **30**, 97 (2000).
11. A. Ranjbar and M. Maheswaran, Using community structure to control information sharing in online social networks, *Comput. Commun.* **41**, 11 (2014).
12. M. Girvan and M. E. J. Newman, Community structure in social and biological networks, *Proc. Natl. Sci.* **99**, 7821 (2002).

13. L. Gao, P.-G. Sun and J. Song, Clustering algorithms for detecting functional modules in protein interaction networks, *J. Bioinform. Comput. Biol.* **7**, 217 (2009).
14. M. E. Newman, Modularity and community structure in networks, *Proc. Natl. Acad. Sci.* **103**, 8577 (2006).
15. A. Clauset, M. E. J. Newman and C. Moore, Finding community structure in very large networks, *Phys. Rev. E* **70**, 066111 (2004).
16. J. Kumpula *et al.*, Limited resolution and multiresolution methods in complex network community detection, *Fluct. Noise Lett.* **7**, L209 (2007).
17. L. Hagen and A. B. Kahng, A new approach to effective circuit clustering in *1992 IEEE/ACM International Conference Computer-Aided Design* (*ICCAD-92*) (IEEE, 1992), pp. 422–427.
18. P. De Meo *et al.*, Mixing local and global information for community detection in large networks, *J. Comput. Syst. Sci.* **80**, 72 (2014).
19. D. Harel and Y. Koren, *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science* (Springer, 2001), pp. 18–41.
20. P. G. Sun and Y. Yang, Methods to find community based on edge centrality, *Physica A* **392**, 1977 (2013).
21. A. Azran and Z. Ghahramani, A new approach to data driven clustering, in *Proceedings of the 23rd International Conference on Machine Learning* (ACM, 2006), pp. 57–64.
22. P. Pons and M. Latapy, Computing communities in large networks using random walks, *Computer and Information Sciences — ISCIS 2005* (2005), pp. 284–293.
23. A. Arenas *et al.*, Communities in complex networks: Identification at different levels, *Enciclopedia of Life Support System* (*EOLSS*) (EOLSS Publishers, Oxford, 2010).
24. J. Reichardt and S. Bornholdt, Detecting fuzzy community structures in complex networks with a Potts model, *Phys. Rev. Lett.* **93**, 218701 (2004).
25. P. Ronhovde and Z. Nussinov, Multiresolution community detection for megascale networks by information-based replica correlations, *Phys. Rev. E* **80**, 016109 (2009).
26. X. Liu and T. Murata, Advanced modularity-specialized label propagation algorithm for detecting communities in networks, *Physica A* **389**, 1493 (2010).
27. K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction* (Printice-Hall, 1989).
28. H. Beigy and M. R. Meybodi, Utilizing distributed learning automata to solve stochastic shortest path problems, *Int. J. Uncertainty Fuzziness and Knowl. Based Syst.* **14**, 591 (2006).
29. M. M. Alipour, A learning automata based algorithm for solving capacitated vehicle routing problem, *Int. J. Comput. Sci. Issues* **9**, 138 (2012).
30. H. Mostafaei, Stochastic barrier coverage in wireless sensor networks based on distributed learning automata *Comput. Commun.* **55**, 51 (2015).
31. J. A. Torkestani, An adaptive backbone formation algorithm for wireless sensor networks, *Comput. commun.* **35**, 1333 (2012).
32. J. A. Torkestani and M. R. Meybodi, Weighted Steiner connected dominating set and its application to multicast routing in wireless MANETs, *Wireless Personal Commun.* **60**, 145 (2011).
33. S. M. Mehr *et al.*, Determining web pages similarity using distributed learning automata and graph partitioning, in *2011 International Symposium on Artificial Intelligence and Signal Processing* (AISP, 2011), pp. 129–134.
34. M. Bazarganigilani and A. Syed, Web page classification using distributed learning automata and partitioning graph algorithm, in *2010 Workshop on Database and Expert Systems Applications* (DEXA, 2010), pp. 302–304.

35. A. Rezvanian and M. R. Meybodi, Finding maximum clique in stochastic graphs using distributed learning automata, *Int. J. Uncertainty, Fuzziness Knowl. Based Syst.* **23**, 1 (2015).
36. M. Hasanzadeh and M. R. Meybodi, Grid resource discovery based on distributed learning automata, *Computing* **96**, 909 (2014).
37. I. S. Dhillon, Y. Guan and B. Kulis, Kernel $k$-means: Spectral clustering and normalized cuts, in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2004), pp. 551–556.
38. W. W. Zachary, An information flow model for conflict and fission in small groups, *J. Anthropol. Res.* 452 (1977).
39. D. Lusseau *et al.*, The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations, *Behav. Ecol. Sociobiol.* **54**, 396 (2003).
40. Newman dataset. Network data (2015) available http://www-personal.umich.edu/∼mejn/netdata/.
41. G. Joshi-Tope *et al.*, Reactome: A knowledgebase of biological pathways, *Nucl. Acids Res.* **33**, D428 (2005).
42. A. Lancichinetti, S. Fortunato and F. Radicchi, Benchmark graphs for testing community detection algorithms, *Phys. Rev. E* **78**, 046110 (2008).
43. L. Danon, A. Diaz-Guilera and J. Duch, Comparing community structure identification, *J. Statist. Mech.: Theory Exp.* **2005**, P09008 (2005).
44. U. Brandes, M. Gaertler and D. Wagner, *Experiments on Graph Clustering Algorithms* (Springer, 2003).
45. C. H. Ding *et al.*, A min-max cut algorithm for graph partitioning and data clustering, in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference* (IEEE, 2001), pp. 107–114.
46. U. N. Raghavan, R. Albert and S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, *Phys. Rev. E* **76**, 036106 (2007).
47. J. Shi and J. Malik, Normalized cuts and image segmentation, *Pattern Anal. Mach. Intel. IEEE Trans.* **22**(8), 888 (2000).