A Closed Asynchronous Dynamic Model of Cellular Learning Automata and its Application to Peer-to-Peer Networks

Ali Mohammad Saghiri[*], Mohammad Reza Meybodi[†]

Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Ave., Tehran, Iran

Abstract: *Cellular Learning Automata (CLAs)* are hybrid models obtained from combination of *Cellular Automata (CAs)* and *Learning Automata (LAs)*. These models can be either open or closed. In closed *CLAs*, the states of neighboring cells of each cell called local environment affect the action selection process of the *LA* of that cell whereas in open *CLAs* each cell, in addition to its local environment has an exclusive environment which is observed by the cell only and the global environment which can be observed by all the cells cell in *CLA*. In dynamic models of *CLAs,* one of their aspects such as structure, local rule or neighborhood radius may change during the evolution of the *CLA*. *CLA* can also be classified as synchronous *CLA* or asynchronous *CLA*. In synchronous *CLA,* all learning automata in different cells are activated synchronously whereas in asynchronous *CLA* the *LA* in different cells is activated asynchronously. In this paper, a new closed asynchronous dynamic model of *CLA* whose structure and the number of *LAs* in each cell may vary with time has been introduced. To show the potential of the proposed model, a landmark clustering algorithm for solving topology mismatch problem in unstructured peer-to-peer networks has been proposed. To evaluate the proposed algorithm, computer simulations have been conducted and then the results are compared with the results obtained for two existing algorithms for solving topology mismatch problem. It has been shown that the proposed algorithm is superior to the existing algorithms with respect to communication delay and average round-trip time between peers within clusters.

Keywords: cellular learning automata, dynamic cellular learning automata, peer-to-peer networks, landmark clustering algorithm.

1. INTRODUCTION

Cellular Automata (*CAs*) are discrete dynamical models which are composed of a large number of independent and identical cells. In these models, the cells are arranged into a lattice. Each cell selects a state from a set of states. In a cell, the previous states of a set of cells, including that cell itself, and their neighbors determines the new state of that cell [1]. *CAs* have the same computational power as Turing machines and found applications in some areas such as peer-to-peer networks[2], and social modeling [3]. On the other hand, *Learning Automata* (*LAs*) refer to

[*] Email address: a_m_saghiri@aut.ac.ir
[†] Email address: mmeybodi@aut.ac.ir

models for adaptive online decision making in unknown environments. A *LA* has a finite set of actions. Its goal is to learn the optimal action through repeated interaction with the environment. An optimal action is an action with the highest probability of getting reward from the environment. *LAs* have found applications in several areas such as sensor networks [4], channel assignment[5], and peer-to-peer networks[6]–[10], to mention a few.

Cellular Learning Automata (CLAs) are hybrid models based on *CAs* and *LAs*[11]. In a *CLA,* each cell of the *CA* is equipped with a *LA*. In a cell, the action selected by the *LA* of the cell determines the state of the cell. In the *CLA*, a local rule is defined to determine the response of the *LA* of each cell. Each cell utilizes the local rule to generate the response of its *LA* according to the state of itself and the states of its neighbors. Since, *CLAs* inherit the distributed computation from the *CAs* and learning in unknown environments from the *LAs*, they can be useful in many problems in distributed systems such as computer networks because many of distributed systems have distributed and dynamic nature[11]–[14].

Several models of *CLAs* are reported in the literature. The reported models can be classified into two main classes as described below.

- **Static *CLAs (SCLAs)*:** In a *SCLA*, the structure of the cells remains fixed during the evolution of the *SCLA* [15]–[20]. *SCLAs* can be either closed or open. In closed *SCLAs*, the states of neighboring cells of each cell called local environment affects on the action selection process of the *LA* of that cell whereas in open *SCLAs*, the local environment of each cell, a global environment, and an exclusive environment effects on 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 defined for the whole *SCLA*. *SCLAs* can be further classified as either synchronous or asynchronous. In synchronous *SCLA*, all cells perform their local rules at the same time [15]. This model assumes that there is an external clock which triggers synchronous events for the cells. In asynchronous *SCLA*, at a given time only some cells are activated and the state of the rest of cells remains unchanged [16]. In [17], a model of *SCLA* with multiple learning automata in each cell was reported. In this model, the set of *LAs* of a cell remains fixed during the evolution of the *SCLA*. *SCLA* depending on its structure can be also classified as regular [11] or irregular [18]. In Irregular *SCLA,* the structure regularity assumption is removed.
- **Dynamic *CLAs* (*DCLAs*):** In a *DCLA*, one of its aspects such as structure, local rule or neighborhood radius may change over time. *DCLAs* can be classified as closed either closed *DCLAs* [14],[13], [21] or open *DCLAs*. *DCLAs* can also be also classified as synchronous *DCLAs* or asynchronous *DCLAs.* In synchronous *DCLAs* all learning automata in different cells are activated synchronously whereas in asynchronous *DCLA*s the *LAs* in different cells are activated asynchronously. All the reported *DCLAs* are closed and asynchronous [14],[13], [21].

Since the goal of this paper is to propose a *CLA* based landmark clustering algorithm for solving the topology mismatch problem in peer-to-peer networks, the rest of the introduction section is devoted to more detailed discussion about the topology mismatch problem.

Peer-to-peer networks construct overlay networks over underlay networks, and can be structured or unstructured. In these networks, each peer directly communicates with other peers, and continually joins and leaves the network. Designing topology management algorithms for these networks is a challenging problem because the network dynamicity caused by joining and leaving peers affects the performance of the management algorithm. Topology mismatch problem is an important problem in unstructured peer-to-peer networks[22]–[27]. In these networks, the peers choose their neighbors without considering the underlay positions. Therefore the topology of overlay network may have mismatches with its underlying network. The management algorithms for solving the topology mismatch problem can be classified into three classes[10]: the algorithms in which each peer in the network uses some services such as *GPS* to gather information about the locations of peers, Landmark clustering algorithms, and the algorithms in which each peer in the network uses local information about its neighbors. Since in this paper, we propose a new landmark clustering algorithm, the landmark clustering algorithms are described in more details in the next paragraph.

In Landmark clustering algorithms such as those reported in[22], [24], [28]–[34], the overlay network is organized by clusters in which each cluster has one landmark peer and several ordinary peers. In the landmark clustering algorithms, the peers organize themselves into clusters such that peers that get in within a given cluster are relatively closed to each other in terms of communication delay. These algorithms can be classified into two subclasses: algorithms that use static landmark selection methods [24], [32]–[34] and algorithms that use dynamic landmark selection methods [6], [22], [31]. The dynamic landmark selection methods can be further classified into two subclasses: algorithms that use non-adaptive landmark selection methods such *mOverlay* reported in [22], and algorithms that use adaptive landmark selection methods such as *lOverlay* reported in[6]. A drawback of *lOverlay* is that it has many parameters that must be tuned.

In this paper, a new closed asynchronous dynamic model of *CLA* whose structure and the number of *LAs* of each cell may vary with time is suggested and then is used to design an adaptive landmark clustering algorithm for solving the topology mismatch problem in unstructured peer-to-peer networks. In this model, in contrast to the existing models of *CLA,* the number of *LAs* in each cell may vary with time. To show the superiority of the proposed landmark clustering algorithm, it is compared with *mOverlay*[22] and *lOverlay*[6] algorithms. Both the proposed algorithm and the *lOverlay* algorithm are adaptive landmark selection methods. The *lOverlay* algorithm uses a hybrid mechanism based on *mOverlay* and learning automata whereas the proposed algorithm uses a mechanism based on *CLA* and *Voronoi* diagrams. One of the advantages of the proposed algorithm is that it has fewer parameters than *lOverlay* algorithm. Simulation results have shown that the proposed algorithm performs better than the existing algorithms with respect to communication delay and average round-trip time between peers within clusters.

The rest of this paper is organized as follows. Section 2 we briefly introduce learning automata, landmark clustering algorithms, and *Voronoi* diagrams. In section 3 a new model of dynamic cellular learning automata is introduced and in section 4, an adaptive landmark clustering algorithm for solving topology mismatch problem for unstructured peer-to-peer networks based on this model is described. Section 5 reports the results of experiments, and section 6 concludes the paper.

2. PRELIMINARIES

In this section, in order to provide basic information for the remainder of the paper, we present a brief overview of learning automata, landmark clustering algorithms, and *Voronoi* diagrams.

A. *Learning Automata*

Learning Automata (*LAs*) refer to models for adaptive online decision making in unknown environments. The relationship between a *LA* and its environment is shown in Fig. 1. A *LA* has a finite set of actions. Its goal is to learn to choose the optimal action through repeated interaction with the environment utilizing a learning process. The learning process of the *LA* is described as follows. The *LA* randomly chooses an action from its action set and performs it on the environment. The environment then evaluates the chosen action and responds with a reinforcement signal (reward or penalty) to the *LA*. According to the reinforcement signal of the environment to the selected action, the *LA* updates its action probability vector and then the learning procedure is repeated. The updating algorithm for the action probability vector is called the learning algorithm. The aim of the learning algorithm of the *LA* is to find an appropriate action from the set of actions so that the average reward received from the environment is maximized.

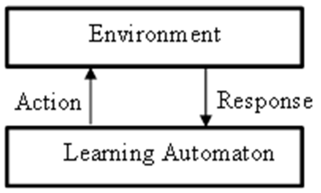

Fig. 1. Learning Automaton (LA)

LAs can be classified into two classes, fixed and variable structure *LAs* [35], [36]. Variable structure *LAs* which is used in this paper is represented by sextuple $\langle \underline{\beta}, \underline{\phi}, \underline{\alpha}, \underline{P}, G, T\rangle$, where $\underline{\beta}$

a set of input actions (called response or reinforcement signal), $\underline{\phi}$ is a set of internal states, $\underline{\alpha}$ a set of outputs, P denotes the state probability vector, G is the output mapping, and T is learning algorithm. The learning algorithm is used to modify the probability vector.

$$\begin{aligned} p_i(k+1) &= p_i(k) + a\big(1 - p_i(k)\big) \\ p_j(k+1) &= p_j(k) - a p_j(k), \qquad \forall j \neq i \end{aligned} \tag{1}$$

$$\begin{aligned} p_i(k+1) &= (1-b)p_i(k) \\ p_j(k+1) &= \frac{b}{r-1} + (1-b)p_j(k), \qquad \forall j \neq i \end{aligned} \tag{2}$$

Let α_i be the action chosen at step k as a sample realization from distribution $P(k)$. In a linear learning algorithm, equation for updating probability vector $P(k)$ is defined by (1) for a favorable response (β=1), and (2) for an unfavorable response (β=0). Two parameters a and b represent reward and penalty parameters, respectively. The parameter a (b) determines the amount of increase (decreases) of the action probabilities. r denotes the number of actions that can be taken by the learning automaton. If a=b, the above learning algorithm is called linear reward penalty (L_{RP}); if a>>b the learning algorithm is called linear reward-ε penalty ($L_{R\varepsilon P}$); and finally if b=0, it is called linear reward inaction (L_{RI}) algorithm.

B. ***Landmark Clustering Algorithms***

Consider n peers $peer_1, peer_2, \dots, peer_n$ that are connected to each other through an overlay network over an underlay network. In the landmark clustering algorithms, the overlay network is organized by clusters. Graph $G = (V, E)$ which $V = \{peer_1, peer_2, \dots, peer_n\}$ is a set of peers and $E \subseteq V \times V$ is a set of links connecting the peers is used to represent the topology of the overlay network. This graph is called overlay graph. Each peer in the overlay network is mapped to a position in the underlay network according to a one-to-one function. In the underlay network, The set of positions is denoted by $V^u = \{pos_1, pos_2,, \dots, pos_n\}$. Graph $G^u = (V^u, E^u)$ which $E^u \subseteq V^u \times V^u$ is a set of links connecting the positions is used to represent the topology of the undelay network. This graph is called underlay graph. In the landmark clustering algorithm, the peers are divided into several clusters which each cluster has a landmark peer. In this algorithm, each peer plays one of two roles: *Ordinary* and *Landmark*. Each ordinary peer communicate with other ordinary peers using landmark peers. The set of clusters is denoted by $V^c = \{cluster_1, cluster_2,, \dots, cluster_p\}$. Graph $G^c = (V^c, E^c)$ which $E^c \subseteq V^c \times V^c$ is a set of links connecting the landmark peer of clusters that is used to represent the topology of the network of the landmark peers. Note that, each cluster in V^c is mapped to several peers in V according to a one-to-many function. The goal of landmark clustering algorithms is to select some peers as landmark peers and reconfigure the links among peers in order to improve the performance of the overlay network in terms of communication delay. The problem of finding a set of landmark peers to minimize the total communication delay is similar to the problem of selecting super-peers in a super-peer network that is proven to be an NP-hard problem[33], [37]. The landmark clustering algorithms try to find an overlay network such that (3) is minimized.

$$\text{Min } z = \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \sum_{k=1}^{n} \sum_{m=1}^{n} (d_{ik} + d_{km} + d_{mj}) \times y_{ik} \times y_{jm} \tag{3}$$

$s.t,$

$$y_{ij} \leq y_{jj} \qquad i,j = 1, \dots, n \tag{4}$$

$$\textstyle\sum_{j=1}^{n} y_{ij} = 1 \qquad i,j = 1, \dots, n \tag{5}$$

$$y_{ij} \in \{0,1\} \qquad i,j = 1, \dots, n \tag{6}$$

In (3), z is the total all-pairs end-to-end communication delay of the overlay network in which d_{ik} is the end-to-end delay from *peer*$_i$ and *peer*$_k$ in the underlay network. In (6), y_{ij} indicates whether there is a connection between *peer*$_i$ and *peer*$_j$ or not. If there is a connection between *peer*$_i$ and *peer*$_{j,}$ the value of variable y_{ij} is equal to 1. If *peer*$_j$ is chosen as a landmark peer, the value of variable y_{jj} is equal to 1. Constraint (4) indicates that an ordinary peer cannot connect to another ordinary peer. Constraint (5) indicates that each ordinary peer connects to exactly one landmark peer. The aim of the landmark clustering algorithm proposed in this paper is to minimize (3) subject to (4)(5)(6).

C. *Voronoi Diagrams*

In the field of computational geometry, constructing the *Voronoi* diagram for a set of *n* points in the Euclidean plane is one of well-known problems. In a Euclidean plane, the *Voronoi* diagram of a set of points is a collection of cells that divide up the plane. Fig. 2 shows an example for a *Voronoi* diagram for six points in a plane. Each cell corresponds to one of the points. All of the points in one cell are closer to the corresponding cell than to any other cell. This problem has an centralized optimal algorithm with O(nlogn) complexity [38], but the optimal algorithm is not applicable to distributed systems. In a distributed system, computing the *Voronoi* diagram where the nodes are modeled as points in the plane leads to a challenging problem. This is because, in some distributed systems, a node cannot gather information about all nodes of the system and the *Voronoi* construction algorithms for building the *Voronoi* diagram among the nodes use information of all nodes. In the next paragraph, we focus on a type of *Voronoi* construction algorithms reported for distributed systems.

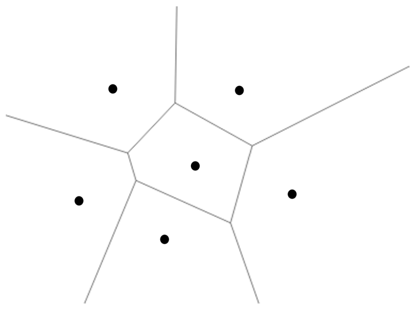

Fig. 2.An example for *Voronoi* diagram for six points in a plane

Several distributed algorithms for constructing *Voronoi* diagrams in distributed manner are reported in [39]–[42]. In [39], [41], a type of algorithms for constructing *Voronoi* diagram in distributed manner is reported for sensor networks. In this type of algorithms, each node of the network uses information about 1-hop neighbors to build initial local *Voronoi* cell. Then the node modifies it corresponding *Voronoi* cell gradually as it gathers messages containing location information from other neighboring nodes.

3. CLOSED ASYNCHRONOUS DYNAMIC CELLULAR LEARNING AUTOMATON WITH VARIIYNG NUMBER OF LEARNING AUTOMATA IN EACH CELL (*CADCLA-VL*)

A *CADCLA-VL* is a network of cells whose structure changes with time and each cell contains a set of *LAs* and a set of attributes. In this model, the connections among cells, and the number of *LAs* of each cell changes during the course of evolution of *CLA*. This model can be formally defined by 9 tuples as follows

CADCLA-VL= $(G, \mathrm{A}, \Psi, \Phi, \mathrm{N}, F_1, F_2, F_3, F_4)$,

where:

- $G = (V, E)$ is an undirected graph which determines the structure of *CADCLA-VL* where $\mathrm{V} = \{cell_1, cell_2, \dots, cell_n\}$ is the set of vertices and E is the set of edges.
- $\mathrm{A} = \{LA_1, LA_2, \dots, LA_m\}$ is a set of *LAs.* A subset of set A is assigned to a cell.
- $\mathrm{N} = \{\mathrm{N}_1, \mathrm{N}_2, \dots, \mathrm{N}_n\}$ where $N_i = \{cell_j \in V | dist(cell_i, cell_j) < \theta_i\}$ where θ_i is the neighborhood radius of $cell_i$ and $dist(cell_i, cell_j)$ is the length of the shortest path between $cell_i$ and $cell_j$ in G. N_i^1 determines the immediate neighbors of $cell_i$ which constitute its **local environment**.
- $\Psi = \{\Psi_1, \Psi_2, \dots, \Psi_n\}$, $\Psi_i = \{(j, X_j, C_j) |\ cell_j \in N_i\}$ denotes the attribute of $cell_j$ where $X_j \subseteq \{x_1, x_2, \dots, x_s\}$ and $C_j \subseteq A$. $\{x_1, x_2, \dots, x_s\}$ is the set of allowable attributes. Ψ_i^1 determines the attribute of $cell_i$ when $\theta_i = 1$.
- $\Phi = \{\Phi_1, \Phi_2, \dots, \Phi_n\}$ where

$\Phi_{\mathrm{i}} = \{(j, k, \alpha_l) | cell_j \in N_i \; and \; action \; \alpha_l$ has been chosen by $LA_k \in C_{\mathrm{i}}\}$ denotes the state of *cell_i*. Φ_i^1 determines the state of *cell_i* when $\theta_{\mathrm{i}} = 1$.

- $F_1 \colon (\underline{\Psi}) \rightarrow (\underline{\zeta})$ is the restructuring function. In each cell, the restructuring function computes the restructuring signal based on the attributes of the cell and its neighboring cells. For example, in *cell_i*, the restructuring function takes $< \Psi_i >$ and then returns a value from the closed interval [0,1] for ζ_i^1 which is the restructuring signal of *cell_i*. we define $\zeta_i = \{(j, \zeta_j^1) | cell_j \in N_i\}$ to be the set of restructuring signals of neighbors of *cell_i*. In a cell, depending the application the value of the restructuring signal determines whether the neighbors of that cell should be changed or not. If the ζ_i^1 is equal to zero (one) this means that the neighbors of cell_i are appropriate (not appropriate).
- $F_2 \colon (\underline{N}, \underline{\Psi}, \underline{\zeta}) \rightarrow (\underline{N}^1, \underline{\Psi}^1)$ is the structure updating rule. In each cell, the structure updating rule finds the immediate neighbors and attribute of the cell based on the restructuring signal computed by the cell, the attributes of the neighbors of the cell, and the neighbors of the cell.
- $F_3 \colon (\underline{\zeta}) \rightarrow (\underline{\nu})$ is the automaton trigger function. Upon activation of a cell, automaton trigger function is called to determine whether the learning automata residing in that cell to be triggered or not. If the automaton trigger function returns true then the learning automata of the cell will be triggered. The automaton trigger function in *cell_i* takes $< \zeta_i >$ and returns a value from {true, false} for ν_i where ν_i is called automaton trigger signal. In a cell, since the value of the restructuring signal affects the changes in the composition of the neighboring cells of that cell, the value of the restructuring signal is used to determine the value of the automaton trigger signal. In *cell_i*, If the ν_i is equal to true (false) then the learning automata of cell_i are triggered (not triggered).
- $F_4 \colon (\underline{\Phi}, \underline{\Psi}) \rightarrow (\underline{\beta})$ is the local rule of *CADCLA-VL*. In each cell, the local rule computes the reinforcement signal for the learning automata of that cell based on the states and the attributes of that cell and its neighboring cells. For example, in *cell_i*, local rule takes $< \Phi_i, \Psi_i >$ and then computes the reinforcement signal $< \beta_i >$ for the learning automata of *cell_i*.

The order by which the cells of *CLA* will be activated is application dependent. Upon the activation of a cell, the cell performs a process which has three phases: **preparation**, **structure updating** and **state updating**. The pesoudo code of this process is given in Fig. 3. These three pahes are described bellow.

1) Preparation phase: In this phase, a cell performs the following step

Step.1: The cell and its neighboring cells compute their *restructuring signals* using the *restructuring function (F_1)*.

2) Structure updating phase: In this phase, a cell performs the following steps

Step.1: The neighborhood structure of the cell is updated using the structure updating rule (F_2) if the value of the restructuring signal of that cell is 1.

Step.2: The automata trigger function (F_3) depending on the restructuring signal and the signal received from the global environment determines whether the set of learning automata of the cell must be triggered or not.

Step.3: If the learning automata are triggered then the cell goes to the **state updating phase**

Step.4: If the set of learning automata of the cell are not triggered then the activation process terminates.

3) State updating phase: In this phase, a cell performs the following steps

Step.1: Each learning automaton of the cell selects one of its actions. The set of actions selected by the set of learning automata in the cell determines the new state for that cell.

Step.2: The local rule (F_4) is applied and a reinforcement signal is generated according to which the action probability vectors of the learning automata of the cell are updated.

```
Algorithm Activate ()
Input:      cell$_i$
```

```
Notations:  F_1 denotes the restructuring function
            F_2 denotes the structure updating rule
            F_3 denotes the automata trigger function
            F_4 denotes the local rule
            $\Psi_i$ denotes the attribute of $cell_i$
            $\Phi_i$ denotes the state of $cell_i$
            $\zeta_i^1$ denotes the restructuring signal of $cell_i$
            $\nu_i$ denotes the automata trigger signal of $cell_i$
            $N_i$ denotes the set of neighbors of $cell_i$
            $\beta_i$ denotes the reinforcement signal of the learning
                 automata of $cell_i$
---------------------------------------------------------------------------
Begin
                         // preparation phase//
  Compute $\zeta_i^1$ using $F_1$;
  Ask from neighboring cells of $cell_i$ to compute their restructuring
                                                              signals;
  Gather the restructuring signals of the Neighboring cells;

                         // structure updating phase//
  If ($\zeta_i^1$ is 1) Then
     Compute $N_i$ and $\Psi_i$ using $F_2$;
  EndIf
  Call $F_3$ to determine the value of $\nu_i$;

                         //state updating phase//
   If ($\nu_i$ is true) Then
      Each learning automaton of $cell_i$ chooses one of its actions;
      Set $\Phi_i$;// set $\Phi_i$ to be the set of actions chosen by the set of
                               learning automata in $cell_i$//
      Compute $\beta_i$ using $F_4$;
      Update the action probabilities of learning automata of $cell_i$ using
                                                                  $\beta_i$;
   EndIf
End
---------------------------------------------------------------------------
```

Fig. 3.The pseudo code of the process which is executed upon the activation of a cell

The internal structure of $cell_i$ and its interaction with local environments is shown in Fig. 4. In this model, each cell has a set of learning automata which may vary during the operation of CLA and three components: **restructuring signal generator**, **structure updater** and **automata trigger function** as explained below.

- **Restructuring signal generator:** this unit generates the restructuring signal using the restructuring function.
- **Structure updater:** this unit updates the set of neighboring cells of the cell according to the restructuring signal and structure updating rule.
- **Automata trigger function:** this unit determines whether the set of *LAs* of the cell to be triggered or not according to the restructuring signal.

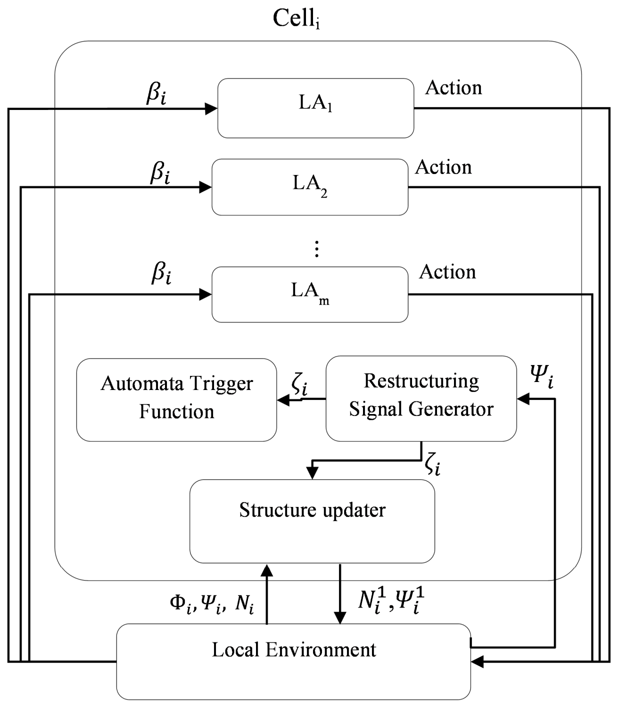

Fig. 4 Internal structure of *cell$_i$* and its interaction with local and global environments

Performance of *CADCLA-VL* will be studied using two metrics: *Entropy*, and *Potential Energy*, which are described in the rest of this section.

Potential energy: The potential energy of the CLA is defined by equation (7) given below

$$A(t) = \sum_{i=1}^{n} \nu_i(t) \qquad (7)$$

where $\nu_i(t)$ is the restructuring signal of *cell$_i$* at iteration *t*.

Potential energy can be used to study the changes in the structure of *CLA* as it interacts with the environment. If the value of *A(t)* becomes zero then no further change needs to be made to the structure. Higher value of *A(t)* indicates higher disorder in the structure of *CLA*.

Entropy: The entropy of the CLA at iteration *t* is defined by equation (8) given below

$$H(t) = -\sum_{l=1}^{n} H^l(t) \qquad (8)$$

where *n* is the number of cells of the *CLA*, $H^l(t)$ is the entropy of cell *cell$_l$* of the *CLA*. The entropy of cell *cell$_l$* is defined by equation (9)

$$H^l(t) = \sum_{i \in C_l(t)} H_i(t) \qquad (9)$$

where $H_i(t)$ is the entropy of learning automaton LA_i and $C_l(t)$ denotes the set of indexes of learning automata of cell $cell_l$ at iteration t. $H_i(t)$ is the entropy of learning automaton LA_i defined equation (10),

$$H_i(t) = \sum_{j=1}^{r_i} p_{ij}(t).\ln(p_{ij}(t)) \qquad (10)$$

where r_i is the number of actions of the learning automaton LA_i. p_{ij}(t) is the probability of selecting action α_j of learning automaton LA_i at iteration t.

Entropy of the CLA can be used to study the changes occur in the states of the cells *of CLA*. The value of zero for *H(t)* means that the learning automata of the cells no longer change their action. Higher values of *H(t)* mean higher rates of changes in the actions selected by learning automata residing in the cells of the *CLA* [8][9].

4. A CLA BASED LANDMARK CLUSTERING ALGORITHM FOR UNSTRUCTURED PEER-TO-PEER NETWORKS: *XOVERLAY*

Initially, a CLA isomorphic to the overlay graph is created which involves defining the initial structure, local rule, structure updating rule, automata trigger function, restructuring function, and local environments. The overlay graph is then mapped into the CLA in which each cell is equipped to a learning automaton. Each peer plays one of three roles: *Undecided*, *Ordinary* and *Landmark*. The initial role of each peer is set to *Undecided*. The learning automata in each cell have two actions "change the role to Landmark" and "change the role to Ordinary". We call the set of nodes in each cell a cluster of peers. Clearly, at the beginning of the algorithms each cluster contains one peer of the overlay graph and hence the number of cluster is equal to the number of cells in the CLA. Each cell of the CLA has an attribute which contains the geographical positions of the peers of the cell in the underlay graph. Fig. 5 shows a snapshot of the CLA at a particular time. As shown, $cluster_i$, $cluster_j$, and $cluster_k$ are corresponding clusters to $cell_x$, $cell_y$, and $cell_z$ respectively. $peer_i$, $peer_j$, and $peer_k$ are the landmark peers of $cluster_i$, $cluster_j$, and $cluster_k$ respectively. Once the CLA is created, the proposed algorithm utilizes it to manage the roles of the peers. The process executed by each peer $peer_i$ when joining to the network consists of three phases: *construction* phase, *organization* phase, and *maintenance* phase (Fig. 6). These phases are briefly described as below.

- ***Construction* phase**: During the *construction* phase performed by a peer, the peer tries to find an appropriate cluster and sets an initial role for itself. If the peer couldn't find any cluster, it sets its role to *Landmark*, and goes to *maintenance* phase. If the peer found a cluster, it connects to the cluster, sets its role to *Ordinary*, and goes to *organization* phase.

- ***Organization* phase**: During the *organization* phase performed by a peer, the peer and other peers that they have the same cluster as the cluster of that peer try to find an appropriate landmark peer for their cluster by activating their corresponding cell in the *CADCLA-VL*. In a cell, the changes were made by the activation function in the *CADCLA-VL* may change the connections of the peer and also select a new landmark peer according to feedbacks received from the network. In other word, As the Algorithm proceeds, the structure of CLA changes and as a result the number of peers in a cell may change. At the end of *organization* phase, the peer goes the *maintenance* phase.
- ***Maintenance* phase**: In this phase, the peer monitors (and broadcast) required information from (to) all peers of its corresponding cluster, and then execute appropriate management operation. In this phase, the peer may jump to organization or construction phases in some situations.

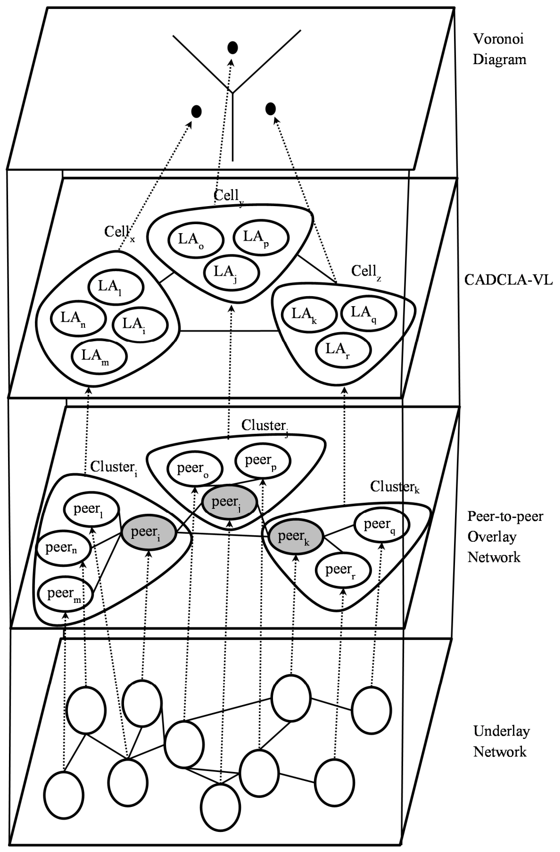

Fig. 5. A snapshot of the CLA at a particular time

```
Algorithm xOverlay ()
Notation: Cluster_x denotes the cluster that the peer belongs to at any time.
          cell denotes the cell corresponds to the cluster Cluster_x.

01  Begin
02                         //Construction phase//
03    If (there is an appropriate cluster cluster_x) Then
04      Connect to the Cluster_x;
05      Set the role of the peer to Ordinary;
06    Else
07      Set the role of the peer to Landmark;
08      goto Maintenance phase; // goto line 13//
09    EndIf
10                         //Organization phase
11    Call Activate(cell) ;
        // Algorithm activate given in Fig. 3 is  used to activate the cell//
12    Determine the role of the peers according to the state of cell
13                         //Maintenance phase
14    While (management phase is needed) Do
15      Broadcast  management information of cluster cluster_x;
16      If(reorganizing in the cluster is needed) Then
17        Goto Organization phase; // goto line line 10//
18      EndIf
19      If (reconstruction in the cluster is needed) Then
20        Goto Construction phase; // goto line line 02//
21      EndIf
22    EndWhile
23  End
```

Fig. 6. Simplified pseudo code of the proposed algorithm

Now, we complete the description of the algorithm by describing the **Local environment**, **Automata trigger function**, **Structure updater**, and **Restructuring signal generator** for the CLA used by function **Activate (cell)** called by Algorithm ***xOverlay*** given in Fig. 6. The detailed descriptions of this algorithm are given in appendix A.

- ***Restructuring function generator:*** The input and output of this unit are shown in Fig. 7. This unit takes information about neighbors of a cell as input and returns a *restructuring signal*. Let $cluster_i$ be the corresponding cluster to $cell_i$. The *restructuring signal* of $cell_i$ is set to 1 if there is a *departed* peer returned by the function *Voronoi_center_selector* for $cluster_i$, and 0 otherwise. The pseudo code for restructuring signal generator is given below.

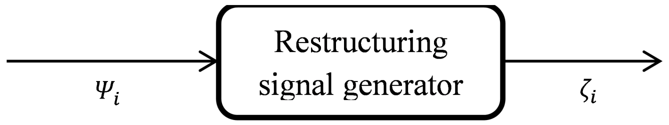

Fig. 7. Input and output of the restructuring signal generator of $cell_i$

```
Algorithm Restructuring signal generator
Input:      Attributes of a $cell_i$ and its neighboring cells
Output:     Restructuring signal
Notations:  Let $cluster_i$ be the set of nodes of $cell_i$.
            Let d-peers be the set of departed peers of $cell_i$.
Begin
  Find d-peers;// using function Voronoi_center_selector given in Fig. 9//
  If (d-peers is not empty) Then
      Return(1);
  Else
      Return(0) ;
  EndIf
End
```

Fig. 8. The pseudo code of the restructuring signal generator

```
Algorithm Voronoi_center_selector()
Input:   $group_i$ // the set of nodes of $cell_i$//
         r  // a threshold//
         u  // a vector containing distances from all nodes of $group_i$ to
            the hub nodes of the groups adjacent to $group_i$ //
Output:  c-node  // a candidate node//
         d-nodes  // set of departed nodes//
         m  // mean of distances computed for node c-node//
         v  // variance of distances computed for node  c-node//
Begin
    Find the c-node; // a c-node (candidate node) is a node that has the
                        minimum mean and variance of
                        distances to the hub nodes of the
                        groups adjacent to $group_i$ and spoke
                        nodes of $group_i$ //
    Find the d-nodes; // In $group_i$, a $node_d$ is a d-nodes (departed node)
                        if the distances from $node_d$ to the hub
                        nodes of the adjacent groups of   $group_i$
                         is not equal (based on the output of
                        function distance_ evaluator  given
                        in Fig. 10) to the distances from the
                        c-node to the hub nodes of the adjacent
                        groups of $group_i$ //

    Return (c-node, d-node, m, v);
End
```

Fig. 9. The pseudo code for function *Voronoi_center_selector*

```
Algorithm distance_ evaluator()
Input:   r // a threshold//
         x // a  real value//
         y // a real value//
Output: z // a value which determines whether two values x and y
           can be considered equal or not considering parameter r//
Begin
  If $(x \geq y\ and\ (1-r) \times x \leq y)$ or
     $(x < y\ and\ (1-r) \times y \leq x)$ Then
       z ← True;
  Else
       z ← False;
  EndIf
  Return (z);
End
```

Fig. 10. The pseudo code for function *distance_ evaluator*

Remark 1: in a peer-to-peer network, the information about delays among peers cannot be gathered with high accuracy. Inaccurate information results in creating inaccurate *Voronoi* cells. We solved this problem using a function called *distance_tester* which is described as follows. Function *distance_tester* takes two values (m and n), and threshold $q \in [0,1)$ as input and then return true if $(m \geq n\ and\ (1-q) \times m \leq n) or\ (m < n\ and\ (1-q) \times n \leq m\ )$ and false otherwise. This function determine whether two values of delays can be considered equal or not. This function is also used in landmark clustering algorithms such as those reported in [6], [31].

Remark 2: In each cluster, function *Voronoi_center_selector* categories the peers of that cluster and returns some information about that cluster. In *cluster*$_i$, this function takes a threshold *t*, the delays among all peers of *cluster*$_i$ and delays from all peers of *cluster*$_i$ to the landmark peers of the clusters adjacent to *cluster*$_i$ and then returns *m, v, c-peer,* and *d-peers*. *c-peer* is called *central candidate* peer and it has the minimum mean and variance of delays to the landmark peers of the clusters adjacent to *cluster*$_i$ and ordinary peers of *cluster*$_i$. In *cluster*$_i$, the *central candidate* peer is an appropriate landmark peer. *m* and *v* denote the mean and variance of delays computed for peer *c-peer* respectively. *d-peers* is called *departed* peers set which contains *departed* peers. In *cluster*$_i$, a *peer*$_d$ is called *departed* peer if the delays from *peer*$_d$ to the landmark peers of adjacent clusters of the *cluster*$_i$ is not equal (based on the output of function *distance_tester)* to the delays from the *c-peer* to the landmark peers of the adjacent clusters of the *cluster$_i$*. The algorithm used for finding *central candidate* peer is similar to an algorithm used for finding center point of *Voronoi* cells reported in [39], [41].

- ***Automata Trigger Function:*** The input and output of this unit are shown Fig. 11. This unit takes the *restructuring signal* of a cell as input and then returns true or false which determines the learning automata of the cell are activated or not. This unit returns true if the *restructuring signal* of the cell is equal to 0 and false otherwise. The pseudo code for this unit is given below.

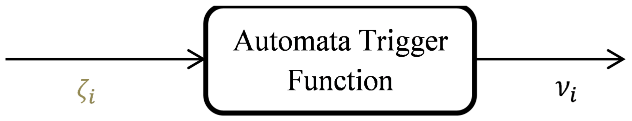

Fig. 11. Input and output of the automata trigger function of cell$_i$

```
Algorithm Automata Trigger Function
Input: The restructuring signal of cell_i
Output: Automata trigger signal
Begin
  If (the restructuring signal is equal to 0)Then
      Return (True);
    Else
      Return (False);
  EndIf
End
```

Fig. 12. The pseudo code of the automata trigger function

- ***Local environment:*** The input and output of this unit are shown in Fig. 13. The reinforcement signal β_i is computed by applying the local rule. This unit returns 1, if the learning automaton of *peer$_j$* (returned by the *Voronoi_center_selector* function as *central candidate* peer) has selected "set the role to landmark peer", other learning automata of *cell$_i$* have selected "set the role to ordinary peer" and there is no departed peer (returned by the *Voronoi_center_selector* function as d-peer) , and 0 otherwise. The pseudo code for this unit is given below.

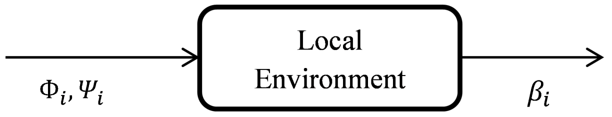

Fig. 13 . Input and output of the local environment of cell$_i$

```
Algorithm Local rule
Input:      The attribute and state of $cell_i$
Output:     The reinforcement signal
Notations:  Let $cluster_i$ be the set of nodes of $cell_i$.
            Let c-peers be the set of candidate peers of $cell_i$.
            Let d-peers be the set of departed peers of $cell_i$.
Begin
  Find c-peer;// using fuction Voronoi_center_selector given in Fig. 9//
  Find d-peer;// using fuction Voronoi_center_selector given in Fig. 9//
  If ((the learning automaton of a c-peer has selected "set the role to landmark
      peer") and (the other learning automata of $cell_i$ have selected "set the
          role to ordinary peer") and (d-peer is empty))Then
     Returns(1);
  Else
     Returns(0);
  EndIf
End
```

Fig. 14 . The pseudo code of the local environment

- ***Structure updating rule:*** The input and output of this unit are shown in Fig. 15. This unit is implemented using three operations called ***Migrate*** operation, ***split*** operation, and ***Merge*** operation which are described below. These operations are described as bellows. This rule use the concept of *Voronoi* diagrams to appropriately divide the peers among the clusters in such a way that an appropriate clustered network with low communication delay can be obtained.

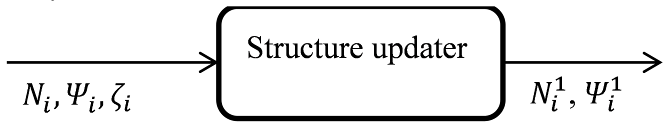

Fig. 15. Input and output of the structure updater of cell$_i$

```
Algorithm Structure updater
Input:      The neighbors, attributes and restructuring signals of $cell_i$ and its
            neighbors
Output:     Immediate neighbors and attribute of $cell_i$
Notations:  Let $cluster_i$ be the set nodes of $cell_i$.
            Let $cluster_j$ be the set nodes of $cell_j$.
            Let d-nodes be the set of departed nodes of $cell_i$.
            Let c-nodes be the candidate node of $cell_i$.
Begin
  If (the restructuring signal of $cell_i$ is equal to 1) Then
     Find set d-peers;// using function Voronoi_center_selector given in Fig. 9//
     If (d-peers is not empty) Then
        Select a peer $peer_k$ from set d-peers randomly;
```

```
            Select an appropriate cluster $cluster_j$ for $peer_k$;
            If ($cluster_j$ has been found) Then
              If ($cluster_i$ has one peer) Then
                Execute Merge operation;
              Else
                Execute Migrate operation;
              EndIf
            Else
            Execute Split operation;
          EndIf
        EndIf
      EndIf
    End
```

Fig. 16. The pseudo code of the structure updater

1. ***Migrate operation***: Fig. 15 shows an example of the *Migrate* operation. In this figure, *cell*$_i$ and *cell*$_j$ have participated in a *Migrate* operation. Let *cluster*$_i$ and *cluster*$_j$ denote the clusters of peers for *cell*$_i$ and *cell*$_j$ respectively and *peer*$_i$ and *peer*$_j$ are the *landmark* peers of *cluster*$_i$ and *cluster*$_j$, respectively and also *peer*$_k$ is returned by function *Voronoi_center_selector* as a *departed* peer for *cluster*$_i$. In *cluster*$_i$, the *structure updating rule* of *cell*$_i$ uses *Migrate* operation to assign the *LA*$_k$ (the learning automaton of *peer*$_k$) of *cell*$_i$ to cell *cell*$_j$. *cell*$_j$ is a neighboring cell of *cell*$_i$ whose landmark peer has the least delay to the *landmark* peer of *cell*$_i$. Note that, assigning the *LA*$_k$ to *cell*$_y$ leads to changing the connection of *peer*$_k$ from *peer*$_i$ to *peer*$_j$.

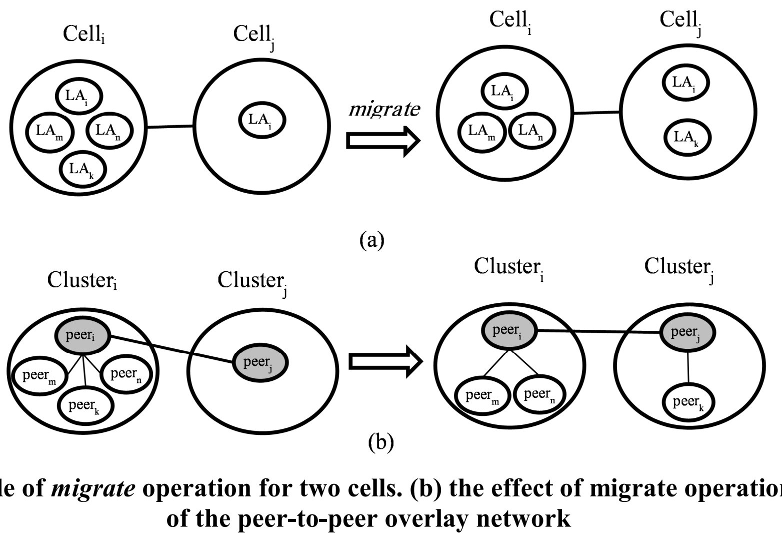

Fig. 17. (a) example of *migrate* operation for two cells. (b) the effect of migrate operation on the structure of the peer-to-peer overlay network

2. ***Split operation***: Fig. 18 shows an example of the *Split* operation. In this figure, *cell*$_i$, *cell*$_j$, *cell*$_w$, and *cell*$_z$ have participated in a *Split* operation. Let *cluster*$_i$ and

cluster$_j$ are the clusters of peers for *cell$_i$* and *cell$_j$* respectively, and *peer$_j$* is returned by function *Voronoi_center_selector* as a *departed* peer for *cluster$_i$*. In *cluster$_i$*, if *node$_i$* cannot find an appropriate cell for executing the *Migrate* operation with it, the *structure updating rule* of *cell$_i$* uses *Split* operation to split the *cell$_i$* into two cells; a new cell which contains the learning automaton of *peer$_j$* and a cell which contains the rest of learning automata of *cell$_i$*. Note that the neighbors of the new cell are the neighbors of *cell$_i$* (before execution of *Split* operation).

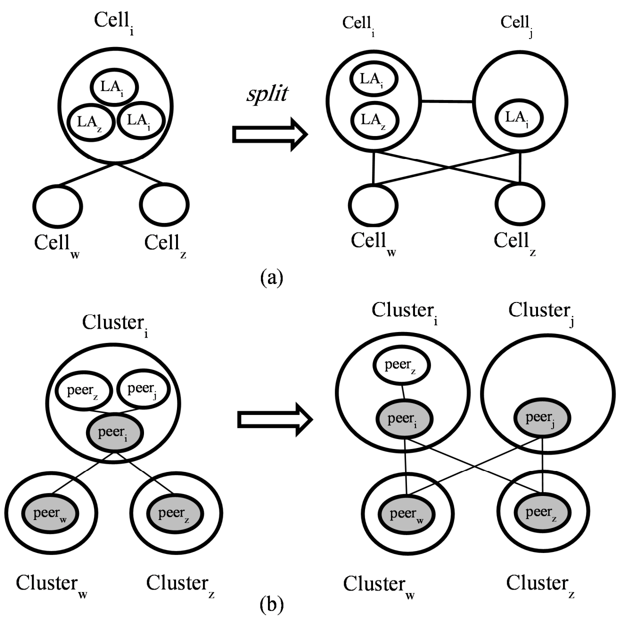

Fig. 18.(a) example of *split* operation. (b) the effect of *split* operation on the structure of the peer-to-peer overlay network

3. ***Merge operation***: Fig. 19 shows an example of the *Merge* operation. In this figure, $cell_a$, $cell_b$, $cell_i$ and $cell_j$ have participated in a *Merge* operation. The *Merge* operation is performed if after the execution of *Migrate* operation there is no learning automaton in $cell_i$ ($cell_j$)

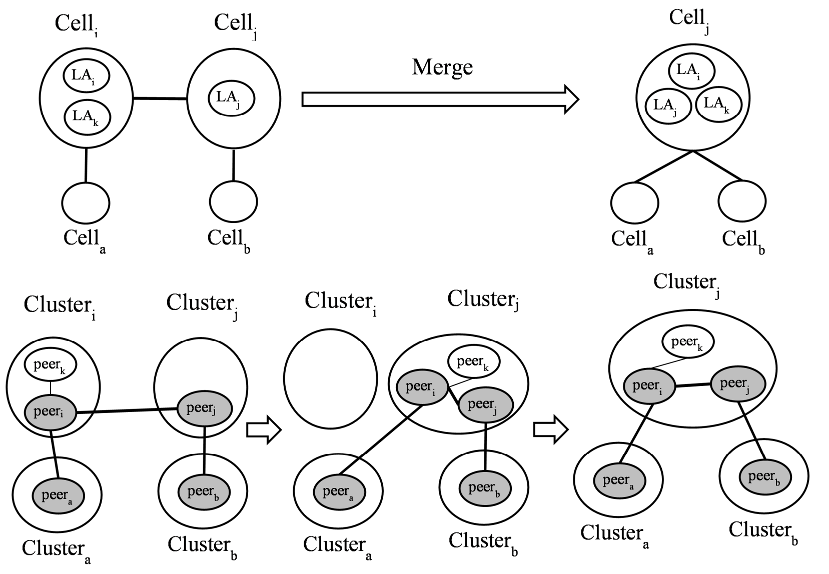

Fig. 19. (a) example of *merge* operation. (b) the effect of *merge* operation on the structure of the peer-to-peer overlay network

5. Experimental Results

In this section, computer simulations have been conducted to evaluate the performance of the proposed algorithm (*xOverlay*) and then the results are compared with the results obtained for other existing algorithms. All simulations have been implemented using the *OverSim* which is a flexible overlay network simulation framework [43]. Different types of underlay network models such as *Simple* and *ReaSE* are supported in *Oversim*. The *Simple model* is the most scalable model and the *ReaSE model* is able to generate different types of underlay networks. We used the *ReaSE model* to generate router-level topologies[44]. We will use two underlying network topologies: *T.1(k)* as an example of the *Simple model* where *k* determines the size of the network, and *T.2, T.3, and T.4* as examples of the *ReaSE model*. The detailed descriptions of these topologies are given below.

- T.1(k): In this underlay topology, k peers are placed on a N-Dimensional Euclidean space and the Internet latencies are based on CAIDA/Skitter data [45][46].
- T.2: Consists of 10 autonomous systems, and about 1000 router –level peers.
- T.3: Consists of 50 autonomous systems, and about 200 router –level peers.
- T.4: Consists of 100 autonomous systems, and about 100 router –level peers.

To evaluate the performance of the *xOverlay algorithm*, it is compared with two groups of algorithms. The first group contains the algorithms reported in the literature for solving the landmark clustering problem that is listed below.

- The *mOverlay* algorithm [22] which is a well-known landmark clustering algorithm in unstructured peer-to-peer networks.
- The *lOverlay* algorithm [6] reported recently is an extension of the *mOverlay* algorithm.

The second group contains different versions of the *xOverlay algorithm*. These algorithms are used to study different aspects of the proposed learning model used in *xOverlay algorithm* (experiment 2 and experiment 3). These visions of *xOverlay* are briefly described below.

- The *nOverlay* algorithm is a version of the *xOverlay algorithm* in which *CADCLA-VL* is replaced with a *pure-chance CADCLA-VL*. A pure-chance *CADCLA-VL* is a *CADCLA-VL* in which, each *learning automaton* is replaced with a *pure-chance* automaton. In this case, each peer participates in the landmark selection procedure with probability 0.5. *nOverlay* will be used to study the learning capability of the *CLA* on the performance of *xOverlay algorithm*.
- The *hOverlay* algorithm is a version of the *xOverlay algorithm* in which the *learning automata* of *CADCLA-VL* are deactivated. In this algorithm, each peer always participates in the landmark selection procedure without considering the actions selected by the *learning automata*. This means that restructuring of the network is performed based on the positions of the peers only. This is in contrast to *xOverlay* in which the restructuring of the network depends on both the positions of the peers and the actions selected by the *learning automata*. *hOverlay* will be used to study the performance of the landmark clustering algorithm when the learning automata are absent.
- *kOverlay*(p) algorithm is a version of *xOverlay* in which the information about each peer such as its position and the actions selected by the *learning automata* residing the cells containing that peer is used with probability p. *kOverlay*(p) will be used to study the effect of the above mentioned information on the performance of *xOverlay* algorithm.

Table 1. The parameters of *lOverlay* algorithm

Parameters	Values
A	0.1
T	0.3
MAX_ITERATION_LEARNING	1
MAX_ITERATION	1000

The parameters of the *lOverlay* algorithm are set according to To evaluate the performance of the *xOverlay algorithm*, it is compared with two groups of algorithms. The first group contains the algorithms reported in the literature for solving the landmark clustering problem that is listed below.

- The *mOverlay* algorithm [22] which is a well-known landmark clustering algorithm in unstructured peer-to-peer networks.
- The *lOverlay* algorithm [6] reported recently is an extension of the *mOverlay* algorithm.

The second group contains different versions of the *xOverlay algorithm*. These algorithms are used to study different aspects of the proposed learning model used in *xOverlay algorithm* (experiment 2 and experiment 3). These visions of *xOverlay* are briefly described below.

- The *nOverlay* algorithm is a version of the *xOverlay algorithm* in which *CADCLA-VL* is replaced with a *pure-chance CADCLA-VL.* A pure-chance *CADCLA-VL* is a *CADCLA-VL* in which, each *learning automaton* is replaced with a *pure-chance* automaton. In this case, each peer participates in the landmark selection procedure with probability 0.5. *nOverlay* will be used to study the learning capability of the *CLA* on the performance of *xOverlay algorithm*.
- The *hOverlay* algorithm is a version of the *xOverlay algorithm* in which the *learning automata* of *CADCLA-VL* are deactivated. In this algorithm, each peer always participates in the landmark selection procedure without considering the actions selected by the *learning automata*. This means that restructuring of the network is performed based on the positions of the peers only. This is in contrast to *xOverlay* in which the restructuring of the network depends on both the positions of the peers and the actions selected by the *learning automata*. *hOverlay* will be used to study the performance of the landmark clustering algorithm when the learning automata are absent.
- *kOverlay*(p) algorithm is a version of *xOverlay* in which the information about each peer such as its position and the actions selected by the *learning automata* residing the cells containing that peer is used with probability p. *kOverlay*(p) will be used to study the effect of the above mentioned information on the performance of *xOverlay* algorithm.

Table 1. The *xOverlay* algorithm has two main parameters: *t,* and *THRSHOLD*. Parameter *THRSHOLD* must be set to a large value in order to provide enough time for the peers of the network to communicate with their neighbors. For the experiments, *THRSHOLD* is set to 5s. For

all experiments, each peer is equipped with a variable structure learning automaton of type L_{RI} with reward parameter a=0.1 and the neighborhood radius of all cells of the *CADCLA-VL* is set to 2.

The results reported are averages over 60 different runs. In each run, the algorithms are executed for 100 rounds. The algorithms are compared with respect to six metrics: Total Communication Delay (*TCD*), Mean Round-trip Time (*MRT*), Control Message Overhead (*CMO*), *Entropy*, and Potential Energy (*PE*). The definitions of *Entropy*, *PE* were previously given in section 4. The definitions of other metrics are given below.

- **Total Communication Delay** is the total of all-pairs end-to-end communication delay of the overlay network. This metric is measured using (3).
- **Mean Round-trip Time** is the mean round-trip time between members of the same cluster. Clusters with only one peer are ignored in this case.
- **Control Message Overhead** is the number of extra control messages generated for the purpose of reconfiguration of overlay network.

The design of experiments is given as follows. Experiment 1 is designated to study the impact of parameter *q* on the performance of the *xOverlay* algorithm. Experiment 2 is conducted to study the impact of the learning capability of the learning automata of the *CADCLA-VL* on the performance of the *xOverlay* algorithm with respect to *TCD, MRT* and *CMO*. Experiment 3 is designated to study the performance of the *CADCLA-VL* of the *xOverlay* algorithm with respect to *Entropy, and PE*. Experiment 4 and experiment 5 are conducted to study the effect of network size and underlay topology on the *mOverlay*, the *lOverlay* and the *xOverlay* algorithm.

A. *Experiment 1*

This experiment is conducted to study the impact of the parameter *q* on the performance of the *xOverlay* algorithm. For this purpose, the *xOverlay* algorithm is tested for five values for parameter *q*=0.1, 0.2, 03, 0.4 and 0.5. In this experiment, *T.1(10000)* is used as the underlay topology. The results are compared with respect to *TCD, MRT,* and *CMO*. According to the results of this experiment that are given in Table 2, we may conclude that increasing parameter *q* leads to increasing *MRT* and *CMO*. In terms of *TCD*, *xOverlay* algorithm performs well for q=0.3.

Table 2. The impact of the parameter q on the performance of xOverlay

	Parameter q				
	q=0.1	**q=0.2**	**q=0.3**	**q=0.4**	**q=0.5**
TCD	176881±812	174782±514	173891±213	173954±503	179941±913
MRT	5.8±0.69	5.89±0.49	6.6±0.21	8.4±0.79	10.4±1.12
CMO	110152± 15109	130897± 9201	170019± 8931	250210± 19201	342102± 21032

B. *Experiment 2*

This experiment is conducted to study the impact of the learning capability of the learning automata of the *CADCLA-VL* on the performance of *xOverlay* algorithm with respect to *TCD, MRT* and *CMO*. For this purpose, *xOverlay* algorithm is compared with *nOverlay* and *hOverlay* algorithms. For this experiment parameter *q* is set to 0.3, and *T.1(10000)* is used as underlay topology. The results are compared with respect to *TCD, MRT, and CMO.* According to the results of this experiment given in Table 3, we may conclude the followings.

- In terms of *TCD*, and *MRT, xOverlay* algorithm performs better than *nOverlay* because in *xOverlay* algorithm each peer decides on its role (to be an ordinary or landmark) adaptively based on the decisions made by its neighboring peers whereas in *nOverlay* at a given time the history of a peer about its role does not affect its decision regarding its role (chooses its role with probability 0.5).
- In terms of *TCD*, and *MRT,* the *xOverlay* algorithm performs better than *hOverlay*. This is because in *hOverlay*, unlike *xOverlay* all the learning automata are deactivated and as a result each peer always participates in the landmark selection procedure. In *hOverlay* restructuring of the network is performed based on the positions of the peers only whereas in *xOverlay* restructuring of the network depends on both the positions of the peers and the actions selected by the learning automata.

- The *xOverlay* algorithm has lower *CMO* as compared to *hOverlay* algorithm. This is because in *hOverlay*, each peer always participates in the landmark clustering which results in generating higher number of control messages. *xOverlay* has lower *CMO* as compared to *nOverlay* algorithm. This is because *nOverlay* performs an exhaustive search in order to find appropriate landmarks whereas the search performed by *xOverlay* is guided by *CLA*.

Table 3. Comparison of different versions of the proposed algorithm

	xOverlay	hOverlay	nOverlay
TCD	173980±223	174300±314	179980±1158
MRT	6.6±0.21	6.7±0.34	8.4±0.41
CMO	170113±2952	191232±1521	181132±2115

C. *Experiment 3*

This experiment is conducted to study the performance of the *CADCLA-VL* of *xOverlay* algorithm with respect to *Entropy, and PE*. For this purpose, *xOverlay* algorithm is compared with *nOverlay* and *kOverlay* (0.8) algorithms. For this experiment parameter *q* is set to 0.3, and *T.1(10000)* is used as underlay topology. The results obtained from different rounds of the simulation are reported in Fig.17 to Fig.18. According to the results of this experiment, one may conclude the followings.

- In terms of *Entropy, xOverlay* performs better than other algorithms. Fig. 20 indicates that the *Entropy* is high at initial rounds of the simulation and gradually decreases. This means that the changes in the states of *CLA* becomes less frequent as the set of landmark peers found by the algorithm becomes closer to the appropriate set of landmark peers.
- Fig. 21 shows the changes in *PE* during the execution of *xOverlay*. This figure shows that *PE* is initially high and approaching zero indicating the fact that the overlay network topology is approaching to a fixed structure. Experimentations have shown that for *nOverlay* and *kOverlay*, *PE* shows a gradual decrease but approaching to a fixed structure is very slow.

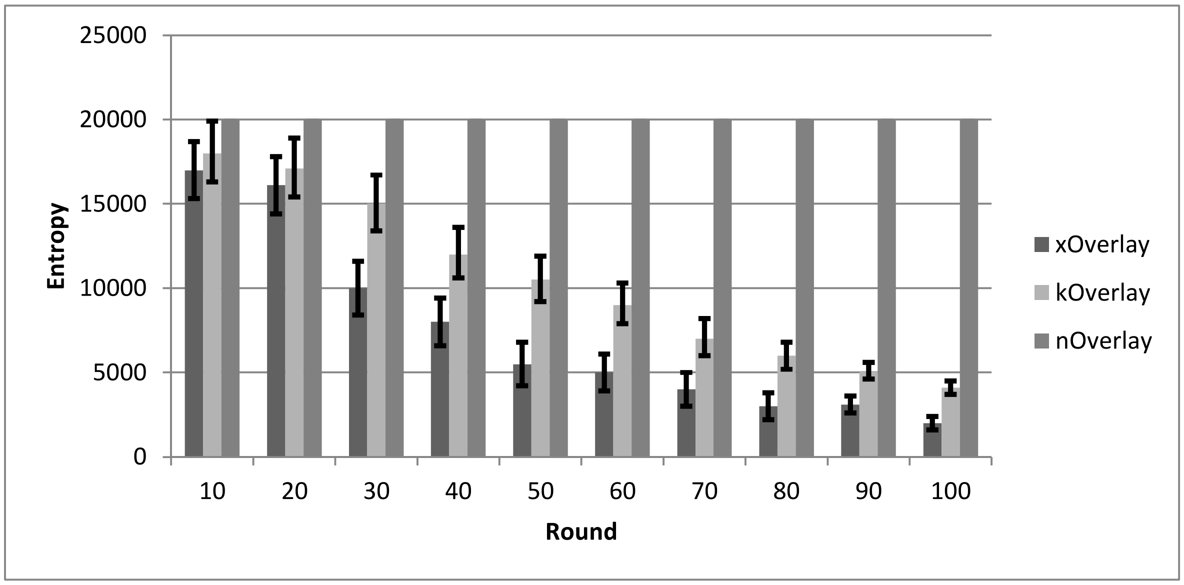

Fig. 20. Comparison of *xOverlay, kOverlay(0.8),* and *nOverlay* with respect to Entropy

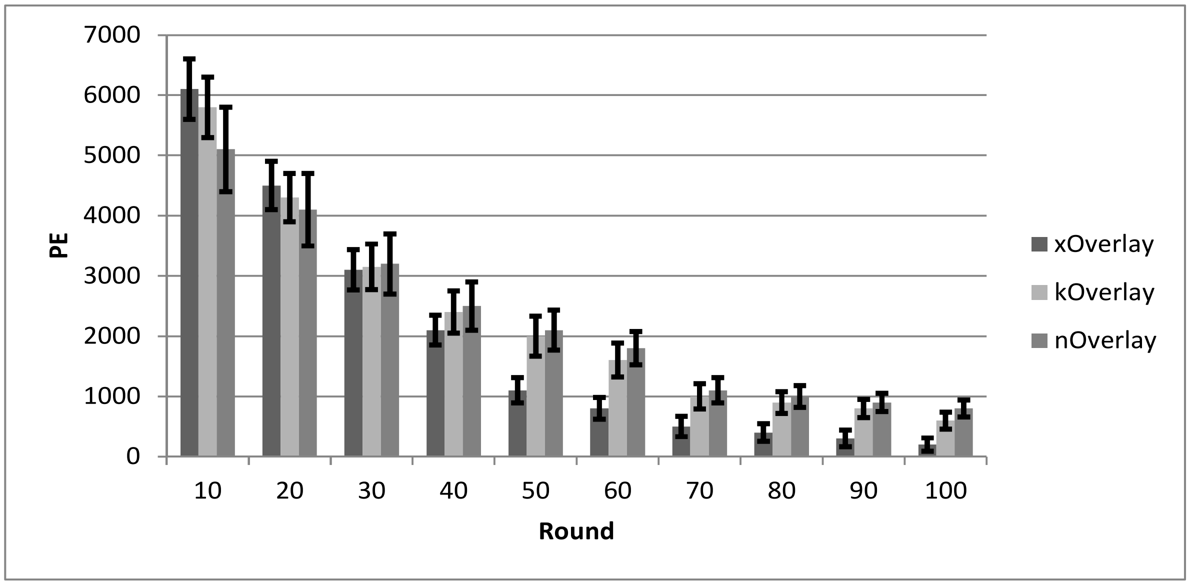

Fig. 21. Comparison of *xOverlay, kOverlay(0.8),* and *nOverlay* with respect to PE

D. *Experiment 4*

This experiment is conducted to study the impact of the network size on the performance of the *xOverlay* algorithm when the underlay network topology is generated by the simple model as described before. For this purpose, we generate five topologies from *T.1(k)* for *k*=10000, 20000, 30000, 40000 and 50000. For *xOverlay* algorithm, the parameter *q* is set to *0.3.* The results obtained are compared with the results obtained for *mOverlay* and *lOverlay* algorithms with respect to the criteria mentioned above. According to the results of this experiment that are shown in Fig. 22 to Fig. 24, one may conclude that in terms of *TCD* and *MRT*, *xOverlay* algorithm performs better than other algorithms because in *xOverlay* algorithm, *CADCLA-VL* gradually finds appropriate landmark peers for the clusters which results in lower *TCD,* and *MRT*. *mOverlay* performs better than other algorithm in terms of *CMO* but worse than other algorithms in terms of *TCD* and *MRT*. This is because *mOverlay* unlike other algorithms is not adaptive and does not have the burden of generating messages for the sake of overlay structure adaptation.

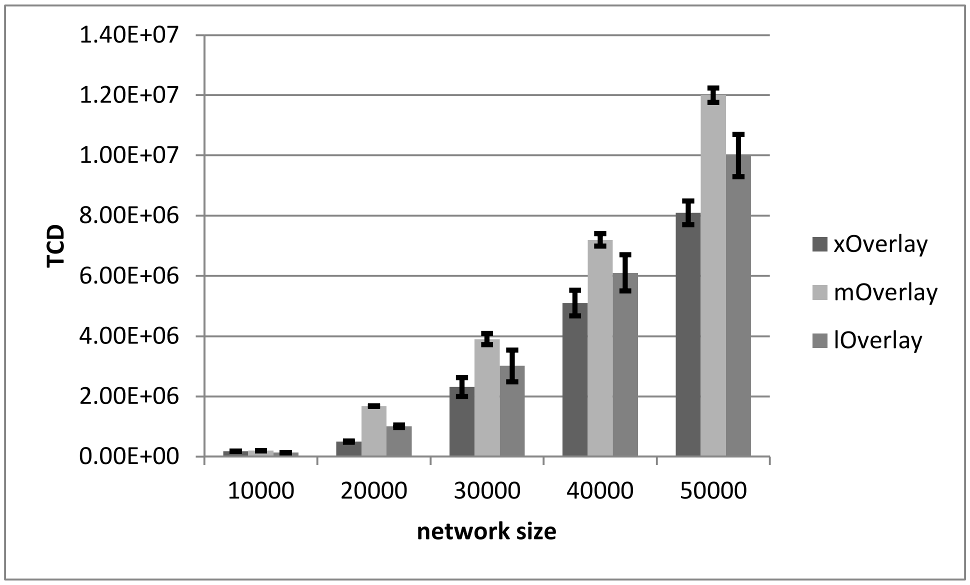

Fig. 22. Comparison of *xOverlay* with *mOverlay* and *lOverlay* with respect to *TCD*

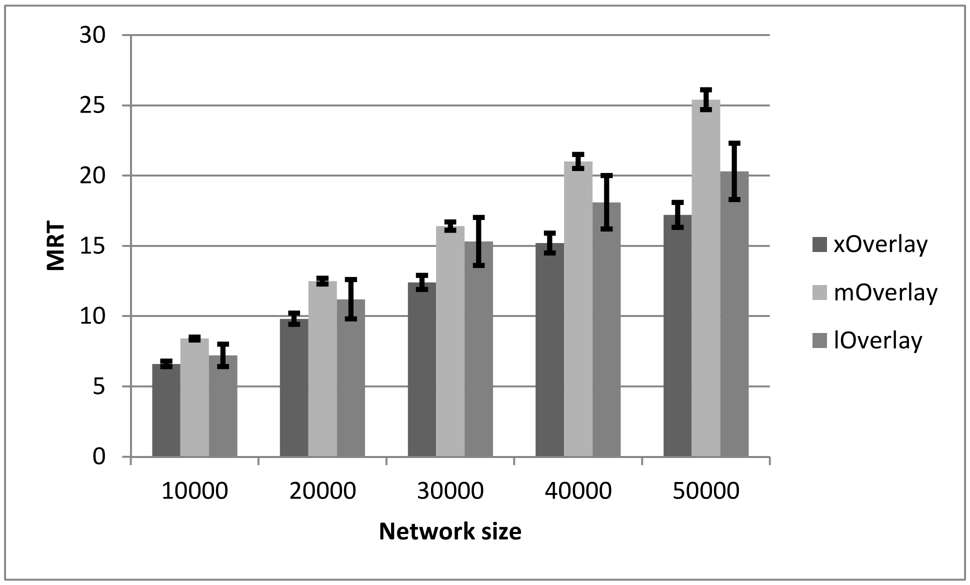

Fig. 23. Comparison of *xOverlay* with *mOverlay* and *lOverlay* with respect to *MRT*

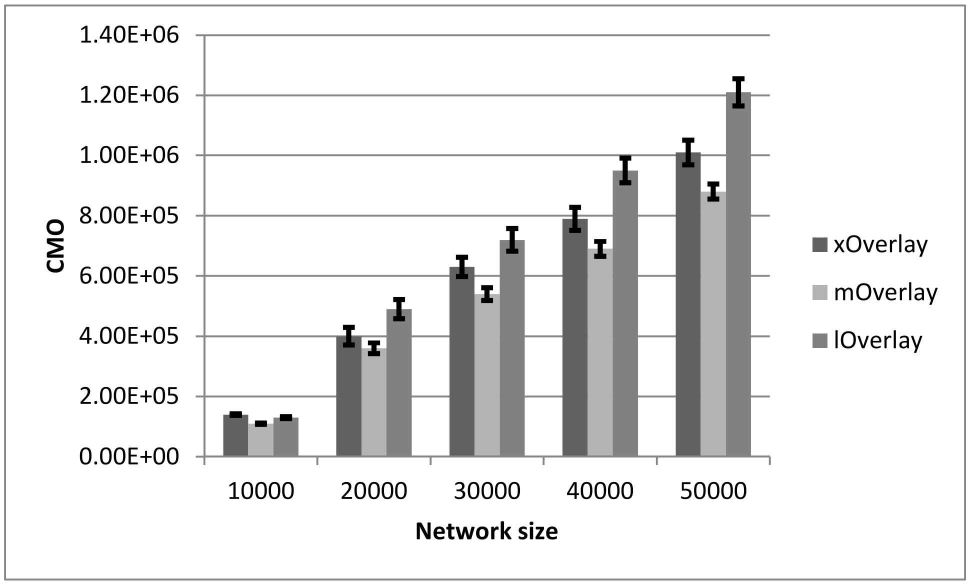

Fig. 24. Comparison of *xOverlay* with *mOverlay* and *lOverlay* with respect to *CMO*

E. ***Experiment 5***

In this experiment, we study the performance of *mOverlay, lOverlay* and *xOverlay* algorithms on topologies *T.2, T.3,* and *T.4* which all of them belong to the class of router level topologies[44]. The results obtained from the *xOverlay* algorithm are compared with the results obtained for *mOverlay* and *lOverlay* algorithms with respect to *TCD, MRT*, and *CMO* when parameter *q* is set to *0.3*. From the results of this experiment given in Table 4, Table 5, and Table 6, we may conclude that the *xOverlay* algorithm performs better than *mOverlay* and *lOverlay* algorithms in terms of *TCD* and *MRT*. This means that the proposed *CLA* based algorithm can be efficient even for router level topologies.

Table 4. The impact of the underlay topology on the performance of *mOverlay, lOverlay* and the *xOverlay* with respect to TCD

	Topology		
	Topology T.2	**Topology T.3**	**Topology T.4**
lOverlay	197542±14185	167780±10105	169794±9311
mOverlay	240164±10058	197880±6258	227961±5201
xOverlay	190145±12124	138668±8111	148792±7162

Table 5. The impact of the underlay topology on the performance of *mOverlay, lOverlay* and the *xOverlay* with respect to MRT

	Topology		
	Topology T.2	**Topology T.3**	**Topology T.4**
lOverlay	23.1±1.9	21.4±1.8	18.2±1.4
mOverlay	25.6±1.3	22.3±1	19.7±0.8
xOverlay	19.5±1.1	18.3±0.7	15.7±0.5

Table 6. The impact of the underlay topology on the performance of *mOverlay, lOverlay* and the *xOverlay* with respect to CMO

	Topology		
	Topology T.2	**Topology T.3**	**Topology T.4**
lOverlay	112164±7157	102106±4911	92203±5301
mOverlay	91223±6254	89187±2521	79105±4237
xOverlay	102145±7189	98055±3936	88011±5154

6. CONCLUSIONS

In this paper, a new closed asynchronous dynamic model of *CLA* whose structure and the number of *LAs* of each cell may vary with time was introduced. In this model, in contrast to the existing models of *CLA,* the number of *LAs* in each cell may vary with time. To show the potential of this new model in designing algorithms an adaptive landmark clustering algorithm for solving topology mismatch problem in unstructured peer-to-peer networks based on this model was designed. To show the superiority of the proposed landmark clustering algorithm, it is compared with two existing algorithms *mOverlay* and *lOverlay.* The results of experimentations showed that the proposed algorithm performs better than the existing algorithms with respect to communication delay and average round-trip time between peers within clusters. One of the advantages of the proposed algorithm is that it has fewer parameters than *lOverlay* algorithm.

REFERENCES

[1] S. Wolfram, "Theory and applications of cellular automata," *World Scientific Publication*, 1986.

[2] N. Ganguly and A. Deutsch, "A cellular automata model for immune based search algorithm," in *6 th International conference on Cellular Automata for Research and Industry*, Amsterdam, Netherlands, 2004, pp. 142–150.

[3] J. M. Benito and P. Hernández, *Modelling Segragation Through Cellular Automata: a Theoretical Answer*. Instituto Valenciano de Investigaciones Económicas, 2007.

[4] M. Esnaashari and M. R. Meybodi, "Data aggregation in sensor networks using learning automata," *Wireless Networks*, vol. 16, no. 3, pp. 687–699, 2010.

[5] H. Beigy and M. R. Meybodi, "A learning automata-based adaptive uniform fractional guard channel algorithm," *The Journal of Supercomputing*, vol. 71, no. 3, pp. 871–893, 2015.

[6] A. M. Saghiri and M. R. Meybodi, "A distributed adaptive landmark clustering algorithm based on mOverlay and learning automata for topology mismatch problem in unstructured peer-to-peer networks," *International Journal of Communication Systems*, 2015.

[7] S. Gholami, M. R. Meybodi, and A. M. Saghiri, "A Learning Automata-Based Version of SG-1 Protocol for Super-Peer Selection in Peer-to-Peer Networks," in *Proceedings of the 10th International Conference on Computing and Information Technology*, Angsana Laguna, Phuket, Thailand, 2014, pp. 189–201.

[8] M. Ghorbani, A. Saghiri, and M. Meybodi, “A Novel Learning based Search Algorithm for Unstructured Peer to Peer Networks,” *Technical Journal of Engineering and Applied Sciences*, vol. 3, no. 2, pp. 145–149.

[9] M. Ghorbani, M. R. Meybodi, and A. M. Saghiri, “A novel self-adaptive search algorithm for unstructured peer-to-peer networks utilizing learning automata,” in *3rd Joint Conference of AI & Robotics and 5th RoboCup Iran Open International Symposium*, Qazvin,Iran, 2013, pp. 1–6.

[10] A. M. Saghiri and M. R. Meybodi, “A Self-adaptive Algorithm for Topology Matching in Unstructured Peer-to-Peer Networks,” *Journal of Network and Systems Management*, 2015.

[11] H. Beigy and M. R. Meybodi, “A mathematical framework for cellular learning automata,” *Advances in Complex Systems*, vol. 3, no. 4, pp. 295–319, 2004.

[12] M. Asnaashari and M. R. Meybodi, “Irregular Cellular Learning Automata and Its Application to Clustering in Sensor Networks,” in *Proceedings of 15th Conference on Electrical Engineering*, Tehran, Iran, 2007, pp. 21–28.

[13] M. Esnaashari and M. Meybodi, “A cellular learning automata-based deployment strategy for mobile wireless sensor networks,” *Journal of Parallel and Distributed Computing*, vol. 71, no. 5, pp. 988–1001, 2011.

[14] M. Esnaashari and M. Meybodi, “Deployment of a Mobile Wireless Sensor Network with k-Coverage Constraint: A Cellular Learning Automata Approach,” *Wireless Networks*, vol. 19, no. 5, pp. 945–968, 2013.

[15] H. Beigy and M. R. Meybodi, “Open synchronous cellular learning automata,” *Advances in Complex Systems*, vol. 10, no. 4, pp. 527–556, 2007.

[16] H. Beigy and M. R. Meybodi, “Asynchronous cellular learning automata,” *Automatica*, vol. 44, no. 5, pp. 1350–1357, 2008.

[17] H. Beigy and M. R. Meybodi, “Cellular learning automata with multiple learning automata in each cell and its applications,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, no. 1, pp. 54–65, 2010.

[18] M. Esnaashari and M. R. Meybodi, “Irregular Cellular Learning Automata,” *IEEE Transactions on Cybernetics*, no. 99, p. 1, 2014.

[19] M. Mozafari, M. E. Shiri, and H. Beigy, “A cooperative learning method based on cellular learning automata and its application in optimization problems,” *Journal of Computational Science*, 2015.

[20] Y. Zhao, W. Jiang, S. Li, Y. Ma, G. Su, and X. Lin, “A cellular learning automata based algorithm for detecting community structure in complex networks,” *Neurocomputing*, vol. 151, pp. 1216–1226, 2015.

[21] A. M. Saghiri and M. R. Meybodi, “An Approach for Designing Cognitive Engines in Cognitive Peer-to-Peer Networks,” *Journal of Network and Computer Applications*, vol. 70, pp. 17–40, 2016.

[22] X. Y. Zhang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu, “A construction of locality-aware overlay network: mOverlay and its performance,” *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 18–28, 2004.

[23] T. Qiu, E. Chan, M. Ye, G. Chen, and B. Y. Zhao, “Peer-exchange schemes to handle mismatch in peer-to-peer systems,” *The Journal of Supercomputing*, vol. 48, no. 1, pp. 15–42, 2009.

[24] H.-J. Ju and L.-J. Du, “Nodes Clustering Method in Large-Scale Network,” in *8th International Conference on Wireless Communications, Networking and Mobile Computing*, Shanghai, China, 2012, pp. 1–4.

[25] Y. Liu, “A two-hop solution to solving topology mismatch,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, pp. 1591–1600, 2008.

[26] H. C. Hsiao, H. Liao, and P. S. Yeh, “A near-optimal algorithm attacking the topology mismatch problem in unstructured peer-to-peer networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 7, pp. 983–997, 2010.

[27] H. C. Hsiao, H. Liao, and C. C. Huang, “Resolving the topology mismatch problem in unstructured peer-to-peer networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 11, pp. 1668–1681, 2009.

[28] Y. Li and Z. Yu, “An improved genetic algorithm for network nodes clustering,” in *Proceedings of the Second International Conference on Information Computing and Applications*, Qinhuangdao, China, 2011, pp. 399–406.

[29] Y. Jiang, J. You, and X. He, “A particle swarm based network hosts clustering algorithm for peer-to-peer networks,” in *International Conference on Computational Intelligence and Security*, Guangzhou, China, 2006, vol. 2, pp. 1176–1179.

[30] R. Tian, Y. Xiong, Q. Zhang, B. Li, B. Y. Zhao, and X. Li, “Hybrid overlay structure based on random walks,” in *Peer-to-Peer Systems IV*, Springer, 2005, pp. 152–162.

[31] M. Scheidegger and T. Braun, “Improved locality-aware grouping in overlay networks,” in *Kommunikation in Verteilten Systemen*, Bern, Switzerland, 2007, pp. 27–38.

[32] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, “Topologically-aware overlay construction and server selection,” in *The 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, USA, 2002, vol. 3, pp. 1190–1199.

[33] S. Wolf and P. Merz, “Evolutionary local search for the super-peer selection problem and the p-hub median problem,” in *Proceedings of the 4th International Conference on Hybrid Metaheuristics*, Berlin, Heidelberg, 2007, pp. 1–15.

[34] Z. Xu, C. Tang, and Z. Zhang, “Building topology-aware overlays using global soft-state,” in *23rd International Conference on Distributed Computing Systems*, Providence, RI, USA, 2003, pp. 500–508.

[35] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[36] M. Thathachar and P. S. Sastry, *Networks of Learning Automata: Techniques for Online Stochastic Optimization*. Dordrecht, Netherlands: Kluwer Academic Publishers, 2004.

[37] M. E. O’kelly, “A quadratic integer program for the location of interacting hub facilities,” *European Journal of Operational Research*, vol. 32, no. 3, pp. 393–404, 1987.

[38] F. Aurenhammer, “Voronoi diagrams—a survey of a fundamental geometric data structure,” *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.

[39] B. A. Bash and P. J. Desnoyers, “Exact distributed Voronoi cell computation in sensor networks,” in *Proceedings of the 6th international conference on Information processing in sensor networks*, New York, NY, USA, 2007, pp. 236–243.

[40] W.-P. Chen, J. C. Hou, and L. Sha, “Dynamic clustering for acoustic target tracking in wireless sensor networks,” *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, pp. 258–271, 2004.

[41] M. Sharifzadeh and C. Shahabi, “Supporting spatial aggregation in sensor network databases,” in *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, Washington DC, USA, 2004, pp. 166–175.

[42] Z. Zhou, S. R. Das, and H. Gupta, “Variable radii connected sensor cover in sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, p. 8, 2009.

[43] I. Baumgart, B. Heep, and S. Krause, “OverSim: A scalable and flexible overlay framework for simulation and real network applications,” in *Peer-to-Peer Computing*, Seattle, Washington, USA, 2009, pp. 87–88.

[44] L. Li, D. Alderson, W. Willinger, and J. Doyle, “A first-principles approach to understanding the internet’s router-level topology,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 3–14, 2004.

[45] P. Mahadevan, D. Krioukov, M. Fomenkov, B. Huffaker, X. Dimitropoulos, and A. Vahdat, “Lessons from three views of the internet topology,” University of California, San Diego, CA, USA, tr-2005-02, 2005.

[46] B. Huffaker, D. Plummer, D. Moore, and K. C. Claffy, “Topology discovery by active probing,” in *Proceedings of Symposium on Applications and the Internet Workshops*, Washington, DC, USA, 2002, pp. 90–96.

7. APPENDIX A

The detailed descriptions of the phases of the proposed algorithm are given in this section. Before we present the proposed algorithm in more details, we need to define the followings.

1. Eleven types of messages (*Hello, Alive, ClusteringRequest, ClusteringReply, SetRole, WarmUpCell, ReadyCell, AutomatonActivatonRequest, AutomatonActivatonReply, ReinforcementSignal,* and *NewConfig*) are used in the proposed algorithm.

2. *Meeting point (MP)* is a global cache in the network which provides an entry point to the overlay for the new peers.

3. Each peer uses function *M-Activate* that is shown in Fig. 25 to activate the cell of its corresponding cluster. Since in the proposed algorithm, the learning automata of the cells are distributed among peers of the network, and the design of function *Activate* of *CADCLA-VL* that is shown in **Fig. 3** is not appropriate for peer-to-peer networks we designated the function *M-Activate* that is a customized version of function *Activate*. The descriptions of the function *M-Activate* are given in the next paragraph.

In this function, *peer*$_i$ sends a *WarmUpCell* messages to the landmark peers of the neighboring clusters of its corresponding cluster. This message activates the cells of the neighboring clusters to compute their *restructuring signals*. After sending the *WarmUpCell* messages, *peer*$_i$ waits for gathering *ReadyCell* messages. *Peer*$_i$ saves the information of *ReadyCell* messages and calls *automaton trigger* function to find the value of *automaton trigger signal*. If the value of *automaton trigger signal* is true, *peer*$_i$ activates its learning automaton to select an action and sends *AutomatonActivationRequest* messages to other peers of its corresponding cluster to activate all learning automata of its corresponding cluster to collectively select their actions. After sending *AutomatonActivationRequest* messages, *peer$_i$* waits to gather *AutomatonActivationReply* messages. These messages provide required information for executing the *local rule*. *Peer$_i$* uses gathered information and executes the *local rule* for finding the *reinforcement signal* of learning automata. The *reinforcement signal* is used to collectively update all learning automata of the peers of the cluster of *peer$_i$*. Finally, *peer$_i$* find the configuration of peers of its cluster using the *structure updating rule* and send *NewConfig* messages to all peers of its corresponding cluster to activate them to change their connections according to the result of *structure updating rule*. Fig. 25 shows the pseudocode of the *M-Activate* function.

```
Algorithm  M-Activate()
    Notation:
            cluster$_x$ denotes the cluster that the peer belongs to at any time.
            cell denotes the cell corresponds to the cluster Cluster$_x$.
01  Begin
02                                  // preparation phase//
03     Send WarmUpCell messages to the landmark peers of the neighboring clusters of
                                                                    cluster Cluster$_x$;
       // these messages activates the neighboring cells of the cell to compute their
          restructuring signals. //
04     Compute the restructuring signal;
05     Wait for a certain duration THRESHOLD for gathering ReadyCell messages;
06     Save the information of ReadyCell messages into the peer;
       // ReadyCell messages containing restructuring signals of the neighboring cells of
          the cell.//
07     Call the automaton trigger function to determine the value of automaton trigger
                                                                               signal;
       // function automaton trigger function uses information saved in the peer.//
08                                  // structure updating phase//
09     If (the value of the restructuring signal is equal to 1) Then
10       Find the configuration of peers of cluster Cluster$_x$ using the structure updating rule;
11       Send NewConfig messages to all peers which must change their connections;
         // this message contains the information about new configuration for peers were
            determined by the structure updating rule.//
12       Call the automaton trigger function to determine the value of automaton trigger
                                                                               signal;
13     EndIf
14                                  //state updating phase//
15     If (the value of automaton trigger signal is equal to true)Then
16       Activate the learning automaton of the peer to select an action;
17       Send AutomatonActivationRequest messages to all peers of the cluster Cluster$_x$;
         // these messages activate learning automata of the cell to choose their actions.//
18       Wait for a certain duration THRESHOLD for gathering AutomatonActivationReply
                                                                             messages;
```

```
19        Save the information of AutomatonActivationReply messages into the peer;
20        Compute the reinforcement signal using the local rule; // using information
                                                        gathered in the peer//
21        Send ReinforcementSignal messages to all peers of the cluster $Cluster_x$;
22        Update the learning automaton of the peer;// reinforcement signal has been
                                                computed by the local rule.//
23      EndIf
24    End
```
Fig. 25. Pseudo code of the procedure which a peer executes for using the cell of its corresponding cluster

Now we describe the algorithm that a *peer*$_i$ executes. The pseudocode of this algorithm is given in Fig. 26. This algorithm has three phases which are described in the rest of this section.

Construction **phase**: During the *construction* phase performed by *peer*$_i$, the peer tries to find an appropriate cluster. In the *construction* phase, *peer*$_i$ uses the *Meeting point* to find clusters of the network. *Peer*$_i$ randomly selects one of the clusters of the network, sends *Hello* message to the landmark peer of the selected cluster and the landmark peers of clusters adjacent to the selected clusters. Finally, *peer*$_i$ connects to the landmark peer of the selected cluster, sets its role to *Ordinary*, and goes to the *organization* phase. If *peer*$_i$ couldn't find any cluster, it sets its role to *Landmark*, registers its identifier into the *Meeting point* as new landmark peer, and goes to *maintenance* phase.

Organization **phase**: During the *organization* phase performed by *peer*$_i$, *peer*$_i$ and the peers that they have the same cluster as the cluster of *peer*$_i$ try to find an appropriate landmark peer for their cluster using the *CADCLA-VL*. In the *organization* phase, *peer*$_i$ activates the cell of its corresponding cluster by executing function *M-Activate()*. Then *peer*$_i$ sends *ClusteringRequest* messages to all peers of its corresponding cluster to inform them that their landmark peer is not valid. After sending the *ClusteringRequest* message, *peer*$_i$ waits for a certain threshold *THRESHOLD* and gathers *ClusteringReply* messages which consist of information about learning automata of other peers of the cluster. *peer*$_i$ selects one of peers (including *peer*$_i$) which the probability of selection of "change the role to landmark peer" action of its learning automaton is maximum. Finally, *peer*$_i$ sets its role and sends *SetRole* messages containing the selected landmark peer to the peers of its corresponding cluster. Note that, if the selected landmark peer is *peer*$_i$, *peer*$_i$ sets its role to *Landmark* and sets its role to *Ordinary*, otherwise. At the end of *organization* phase, *peer*$_i$ goes the *maintenance* phase.

Maintenance **phase**: In this phase, if the role of *peer*$_i$ is *Landmark, peer*$_i$ sends *Alive* messages to all peers of its corresponding cluster. *Alive* message contains information about new peers of the cluster of *peer*$_i$. *Peer*$_i$ waits for a certain duration *THRESHOLD* to receive one of the *ClusteringRequest, Hello, Alive, SetRole, WarmUpCell, AutomatonActivationRequest,* and *NewConfig* messages.

Peer$_i$, upon receiving a *ClusteringRequest* message from a *peer*$_j$, saves the information of *ClusteringRequest* message. Then, *peer*$_i$ produces a *ClusteringReply* message containing the action probability of selecting the "change the role to landmark peer" action of its learning automaton. After sending the *ClusteringReply* message to *peer*$_j$, *peer*$_i$ restarts the *maintenance* phase.

Peer$_i$, upon receiving a *Hello* message or an *Alive* message from a *peer*$_j$, saves information of *peer*$_j$, connects to *peer*$_j$, and restarts the *maintenance* phase.

Peer$_i$, upon receiving a *SetRole* message from a *peer*$_j$ resided in its corresponding cluster, sets the identifier of its cluster to the identifier of the new landmark peer reported by the *SetRole* message. If *peer*$_i$ has been selected as a new landmark peer, then *peer*$_i$ sets its role to *Landmark*, otherwise *peer*$_i$ sets its role to *Ordinary* and sends a *Hello* message to the new landmark peer. Finally, *peer*$_i$ restarts the *maintenance* phase.

Peer$_i$, upon receiving a *WarmUpCell* message from a *peer*$_j$ which *peer*$_j$ belongs to neighboring clusters of the cluster of *peer*$_i$, computes the restructuring signal using the restructuring function and sends a *ReadyCell* message to the *peer*$_j$. The *ReadyCell* message contains the computed restructuring signal. Finally, *peer*$_i$ restarts the *maintenance* phase.

Peer$_i$, upon receiving an *AutomatonActivatonRequest* message from a *peer*$_j$, activates its learning automaton and sends *AutomatonActivatonReply* message to *peer*$_j$. This message contains the action selected by the learning automaton of *peer*$_i$, information about delays between *peer*$_i$ and other peers of the cluster of *peer*$_i$, and information about delays between *peer*$_i$ and the landmark peers of the clusters adjacent to cluster of *peer*$_i$. Then, *peer*$_i$ waits until receives a *ReinforcementSignal* message from *peer*$_j$. After receiving the *ReinforcementSignal* message, *peer*$_i$ updates its learning automaton using *reinforcement signal* reported by the *ReinforcementSignal* message and restarts the *maintenance* phase.

Peer$_i$, upon receiving a *NewConfig* message from a *peer*$_j$, *changes* its connections to its neighbors using information reported by the *NewConfig* message. Then, *peer*$_i$ restarts the *maintenance* phase.

If *peer*$_i$ does not receive any message during the specified period of *THRESHOLD* or the connections of the *peer*$_i$ are changed, *peer*$_i$ checks its role. If the role of *peer*$_i$ is *Ordinary*, *peer*$_i$ goes to the *organization* phase, and restarts the *maintenance* phase otherwise.

Algorithm *xOverlay* ()

Notation:

```
        Meeting point denotes the identifier of a well-known peer which saves
                  the identifiers of the landmark peers of the network.
        cluster_x denotes the cluster that the peer belongs to at any time.
        cell denotes the cell corresponds to the cluster Cluster_x.
01  Begin
02                      //Construction phase//
03    Select a random cluster Cluster_x from Meeting point;
04    If(Cluster_x is null)Then
05      Set the role of the peer to Landmark;
```

```
06      Register the peer in the Meeting point as new landmark peer;// each landmark
                                                                        peer manages
                                                                        a cluster//
07      Goto Maintenance phase; // goto line 21//
08    Else
09      Connect to the landmark peer of cluster $Cluster_x$;
10      Send Hello message to the landmark peer of the cluster $Cluster_x$ and the
                    landmark peers of clusters adjacent to the cluster $Cluster_x$;
11      Set the role of the peer to Ordinary;
12    EndIf
13                              //Organization phase//
14    Call function M-Activate to activate the cell; // the pseudo code of this function is
                                                       given in Fig. 25//
15    Send ClusteringRequest messages to all peers of cluster $Cluster_x$;
16    Waits for a certain threshold THRESHOLD for gathering ClusteringReply
                                                                    messages;
17    Save information of ClusteringReply messages into the peer;
18    Select an landmark peer;
      // select one of the peers (including the peer) which its probability of selection of
        “change the role to landmark peer” action is higher than other peers.//

19    Set the role of the peer; // if the peer has been selected as landmark peer the peer
                                 sets its role to Landmark and sets its role to Ordinary,
                                 otherwise.//
20    Send  SetRole message to all the peers of the cluster $Cluster_x$;
       // this message contains the identifier of the selected landmark peer.
21                              //Maintenance phase//
22    MaintainInterrupt ←"False";
23    While (maintainInterrupt="False") Do
24      If (the role of the peer is Landmark) Then
25          Send Alive message to all peers of cluster $Cluster_x$;
           // this message contains information about new peers of cluster $Cluster_x$.//
26      EndIf
27      Wait for a certain duration THRESHOLD until receiving Alive,
         ClusteringRequest, Hello, SetRole, WarmUpCell, AutomatonActivatonRequest
         , and NewConfig  messages;
28       If (no message has been received or the connections of the peer has been
            changed) Then
29          If (the role of the peer is Ordinary) Then
30            MaintainInterrupt ←"True";
31          EndIf
32       EndIf
33       If (a ClusteringRequest message has been received from $peer_j$) Then
34          Save the information of ClusteringRequest message into the peer;
35          Send a ClusteringReply message to $peer_j$;
           //this message contains the action probability of selecting the "change the
             role to landmark peer" action of the learning automaton of the peer.//
36       EndIf
```

```
37        If (a Hello message or an Alive message has been received from $peer_j$) Then
38          Save the information of the received message into the peer;
39          Connect to $peer_j$;
40        EndIf
41        If (a SetRole message has been received from $peer_j$) Then
42          Set the identifier of its cluster to the identifier of landmark peer reported by
                                                                   the SetRole message;
          // If the peer has been selected as a new landmark peer, then the peer sets its
            role to Landmark, otherwise the peer sets its role to Ordinary.//
43          Send a Hello message to the landmark peer;
44        EndIf
45        If (a WarmUpCell message has been received from $peer_j$) Then
46          If (the $peer_j$ belongs to neighboring cluster of cluster $Cluster_x$) Then
              Compute the restructuring signal;
47            Send a ReadyCell message to $peer_j$;// this message contains the computed
                                                          restructuring signal.//
48          EndIf
49        EndIf
50        If (a AutomatonActivatonRequest message has been received from $peer_j$) Then
51          Activate the learning automaton of the peer to choose an action;
52          Send a AutomatonActivatonReply message to $peer_j$;
          // this message contains the action selected by the learning automaton of the
            peer, information about delays between the peer and other peers of the
            cluster $Cluster_x$, and information about delays between the peer and the
            landmark peers of the clusters adjacent to cluster $Cluster_x$.//
53          Receive ReinforcementSignal message from $peer_j$;
54          Update the learning automaton of the peer;
          // the reinforcement signal for updating the learning automaton has been
            reported by the ReinforcementSignal message.//
55        EndIf
56        If (a NewConfig message has been received from $peer_j$) Then
57          Change the neighbors of the peer using information reported by the
                                                              NewConfig message;
58        EndIf
59      EndWhile
60      Goto Organization phase;//goto line 13//
61    End
```
Fig. 26. Pseudo code of the proposed algorithm

8. APPENDIX B

Acronyms	**Definition**
CA	Cellular Automata
CMO	Control Message Overhead
CLA	Cellular Learning Automata
CADCLA	Closed Asynchronous Dynamic Cellular Learning Automata
CADCLA-VL	A Closed Asynchronous Dynamic Cellular Learning Automata with multiple learning automata in each cell.
DCLA	Dynamic Cellular Learning Automata
DICLA	Dynamic Irregular Cellular Learning Automata
HDICLA	Heterogeneous Dynamic Irregular Cellular Learning Automata
hOverlay	A version of the proposed algorithm in which the learning automata of *CADCLA-VL* are deactivated.
kOverlay(p)	A version of the proposed algorithm which its *CADCLA-VL* is tuned with an algorithm described as follows. Upon activating a cell of *CADCLA-VL* the information about each peer of its corresponding cluster will be used with probability p.
LA	Learning Automata
lOverlay	An extension of the *mOverlay* algorithm.
MRT	Mean Round-trip Time
M-Activate	A function that executes the cell activation process of the *CLA* in distributed manner.
MP	*MP* is a global cache in the network which provides an entry point to the overlay for the new peers.
mOverlay	A well-known landmark clustering algorithm.
nOverlay	A version of the proposed algorithm in which *CADCLA-VL* is replaced with a pure-chance *CADCLA-VL*.
OADCLA	Open Asynchronous Dynamic Cellular Learning Automata
PE	Potential Energy
SCLA	Static Cellular Learning Automata
TCD	Total Communication Delay
xOverlay	The proposed *CLA* based algorithm