Irregular Cellular Learning Automata and Its Application to Clustering in Sensor Networks

M. Esnaashari[1], M. R. Meybodi[1,2]

[1] Soft Computing Laboratory, Computer Engineering and Information Technology Department
Amirkabir University of Technology, Tehran, Iran
(Esnaashari,mmeybodi)@aut.ac.ir

[2] Institute for Studies in Theoretical Physics and Mathematics (IPM)
School of Computer Science, Tehran, Iran

Abstract

In the first part of this paper, we propose a generalization of cellular learning automata (CLA) called irregular cellular learning automata (ICLA) which removes the restriction of rectangular grid structure in traditional CLA. This generalization is expected because there are a number of applications which cannot be adequately modeled with rectangular grids. One category of such applications is in the area of wireless sensor networks. In these networks, nodes are usually scattered randomly throughout the environment, so no regular structure can be assumed for modeling their behavior. In the second part of the paper, based on the proposed model we design a clustering algorithm for sensor networks. Simulation results show that the proposed clustering algorithm is very efficient and outperforms similar existing methods.

Index Terms— *Irregular cellular learning automata, Cellular learning automata, Sensor networks, Clustering algorithm*

1. Introduction

Cellular automata are mathematical models for systems consisting of large number of simple identical components with local interactions. CA is a non-linear dynamical system in which space and time are discrete. It is called cellular because it is made up of cells like points in a lattice or like squares of checker boards, and it is called automata because it follows a simple rule [2]. The simple components act together to produce complicated patterns of behavior. Cellular automata perform complex computations with a high degree of efficiency and robustness. They are especially suitable for modeling natural systems that can be described as massive collections of simple objects interacting locally with each other [3, 4]. Informally, a d-dimensional CA consists of an infinite d-dimensional lattice of identical cells. Each cell can assume a state from a finite set of states. The cells update their states synchronously on discrete steps according to a local rule. The new state of each cell depends on the previous states of a set of cells, including the cell itself, and constitutes its neighborhood [5]. The state of all cells in the lattice is described by a configuration. A configuration can be described as the state of the whole lattice. The rule and the

initial configuration of the CA specify the evolution of CA that tells how each configuration is changed in one step. Formally, a CA can be defined as follows:

Definition 1. A d-dimensional cellular automata is a structure $A = (Z^d, \Phi, N, F)$, where

- Z^d is a lattice of d-tuples of integer numbers. Each cell in the d-dimensional lattice Z^d is represented by a d-tuple $(z_1, z_2, ..., z_d)$.
- $\Phi = \{1, ..., m\}$ is a finite set of states.
- $N = \{\bar{x}_1, \bar{x}_2, ..., \bar{x}_m\}$ is a finite subset of Z^d called the neighborhood vector, where $\bar{x}_i \in Z^d$. The Neighborhood vector determines the relative position of the neighboring lattice cells from any given cell u in the lattice Z^d. The neighbors of a particular cell u are the set of cells $\{u + \bar{x}_i | i = 1,2,...m\}$. We assume that there exists a neighborhood function $\bar{N}(u)$ that maps a cell u to the set of its neighbors, that is $\bar{N}(u) = (u + \bar{x}_1, u + \bar{x}_2, ..., u + \bar{x}_m)$.

  For sake of simplicity, we assume that the first element of the neighborhood vector (i.e. $\bar{x}_1$) is equal to d-tuple (0, 0, ..., 0) or equivalently $u + \bar{x}_1 = u$. The neighborhood function $\bar{N}(u)$ must satisfy the following two conditions:

  - $u \in \bar{N}(u)$ for all $u \in Z^d$.
  - $u_1 \in \bar{N}(u_2) \Leftrightarrow u_2 \in \bar{N}(u_1)$ for all $u_1, u_2 \in Z^d$
- $F : \Phi^{\bar{m}} \to \Phi$ is the local rule of the cellular automata. It computes the new state for each cell from the current states of its neighbors.

On the other hand, learning automaton is a simple entity operates in an unknown random environment. In a simple form, the automaton has a finite set of actions to choose from and at each stage, its choice (action) depends upon its action probability vector. For each action chosen by the automaton, the environment gives a reinforcement signal with fixed unknown probability distribution. The automaton then updates its action probability vector depending upon the reinforcement signal at that stage, and evolves to the some final desired behavior. A class of learning automata is called variable structure learning automata and are represented by quadruple $\{\alpha, \beta, p, T\}$ in which $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ represents the action set of the automata, $\beta = \{\beta_1, \beta_2, \cdots, \beta_r\}$ represents the input set, $p = \{p_1, p_2, \cdots, p_r\}$ represents the action probability set, and finally $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ represents the learning algorithm. Let α_i be the action chosen at time *n*, then the recurrence equation for updating *p* is defined as

$$\begin{aligned} p_i(n+1) &= p_i(n) + a.(1 - p_i(n)) \\ p_j(n+1) &= p_j(n) - a.p_j(n) \qquad \forall j \;\; j \neq i \end{aligned} \tag{1}$$

for favorable responses, and

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

for unfavorable ones. In these equations, *a* and *b* are reward and penalty parameters respectively. If *a* = *b*, learning algorithm is called L_{R-P} [1], if *a* << *b*, it is called $L_{R\varepsilon P}$ [2], and if *b* = 0, it is called L_{R-I} [3].

[1] Linear Reward-Penalty
[2] Linear Reward epsilon Penalty
[3] Linear Reward Inaction

Cellular learning automata [1], which is a combination of cellular automata (CA) and learning automata (LA), is a powerful mathematical model for many decentralized problems and phenomena. The basic idea of CLA, which is a subclass of stochastic CA, is to use learning automata to adjust the state transition probability of stochastic CA. A CLA is a CA in which a learning automaton is assigned to every cell. The learning automaton residing in a particular cell determines its action (state) on the basis of its action probability vector. Like CA, there is a rule that the CLA operates under. The rule of the CLA and the actions selected by the neighboring LAs of any particular LA determine the reinforcement signal to the LA residing in a cell. The neighboring LAs of any particular LA constitute the local environment of that cell. The local environment of a cell is nonstationary because the action probability vectors of the neighboring LAs vary during evolution of the CLA. The basic idea of CLA, which is a subclass of stochastic CA, is to use learning automata to adjust the state transition probability of stochastic CA. A d-dimensional CLA is formally defined below.

Definition 2. A d-dimensional cellular learning automata is a structure $A = (Z^d, \Phi, A, N, F)$ where

- Z^d is a lattice of d-tuples of integer numbers.
- Φ is a finite set of states.
- A is the set of LA each of which is assigned to one cell of the CLA.
- $N = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_m\}$ is a finite subset of Z^d called the neighborhood vector, where $\vec{x}_i \in Z^d$.
- $F : \Phi^{\vec{m}} \to \beta$ is the local rule of the cellular learning automata, where β is the set of values that the reinforcement signal can take. It computes the reinforcement signal for each LA based on the actions selected by the neighboring LA. It computes the new state for each cell from the current states of its neighbors.

CLA has found many applications such as image processing [6, 7, 8, 9], rumor diffusion [10], modeling of commerce networks [8], channel assignment in cellular networks [11] and VLSI placement [12], to mention a few.

In the first part of this paper, we propose a generalization of cellular learning automata (CLA) called irregular cellular learning automata (ICLA) which removes the restriction of rectangular grid structure in traditional CLA. This generalization is expected because there are applications which cannot be adequately modeled with rectangular grids. One category of such applications is in the area of wireless sensor networks. In these networks, nodes are usually scattered randomly throughout the environment, so no regular structure can be assumed for modeling their behavior. In the second part of the paper, based on the proposed model, we design a clustering algorithm for sensor networks. The proposed clustering algorithm tries to maximize the probability of selecting nodes with higher energy and higher number of neighbors as cluster heads. To evaluate the performance of the proposed algorithm several experiments have been conducted using different clustering algorithms and the results are compared with the proposed algorithm Simulation results show that the proposed clustering algorithm is very efficient in terms clustering formation in the network) and outperform similar existing methods.

The rest of this paper is organized as follows. In section 2 irregular cellular learning automata is presented. Section 3 gives an overview on clustering algorithms for wireless sensor networks. The proposed ICLA based clustering algorithm is described in section 4. Simulation results are given in section 5. Section 6 is the conclusion.

2. Irregular Cellular Learning Automata

The idea of irregular cellular automata was suggested in mid 80s [13], but due to the computationally intensive operations required to search irregular neighborhood, it has been received less attention since then. In an informal way, ICA is a configuration of points in the space with no prior restriction. Each point has a number of other points as its neighbors. The few examples of ICA all use Voronoi polygons or the related Delaunay triangulation to divide space and determine the neighbors of each point [14]. Voronoi polygons divide space into regions surrounding objects such that any point in an object's polygon is closer to that object than to any other object, while Delaunay triangulation is a triangulation of the points in a Voronoi diagram where the circumcircle of each triangle is an empty triangle.

Irregular cellular learning automata (Figure 1) is a combination of irregular cellular automata (ICA) and learning automata. We define ICLA as an undirected graph in which, each vertex represents a cell which is equipped with a learning automaton. The learning automaton residing in a particular cell determines its state (action) on the basis of its action probability vector. Like CLA, there is a rule that the ICLA operate under. The rule of the CLA and the actions selected by the neighboring LAs of any particular LA determine the reinforcement signal to the LA residing in a cell. The neighboring LAs of any particular LA constitute the local environment of that cell. The local environment of a cell is non-stationary because the action probability vectors of the neighboring LAs vary during evolution of the ICLA. An ICLA is formally defined below.

Definition 3. An irregular cellular learning automata is a structure $A = (G < E, V >, \Phi, A, F)$, where

- G is an undirected graph, with V as the set of vertices and E as the set of edges.
- Φ is a finite set of states.
- A is the set of LA each of which is assigned to one cell of the ICLA.
- $F : \underline{\Phi}_j \to \underline{\beta}$ is the local rule of the irregular cellular learning automata in each vertex j, where $\underline{\Phi}_j = \{\Phi_i | (i, j) \in E\} + \{\Phi_j\}$ is the set of states of all neighbors of j and β is the set of values that the reinforcement signal can take. β computes the reinforcement signal for LA_j based on the actions selected by the neighboring LA.

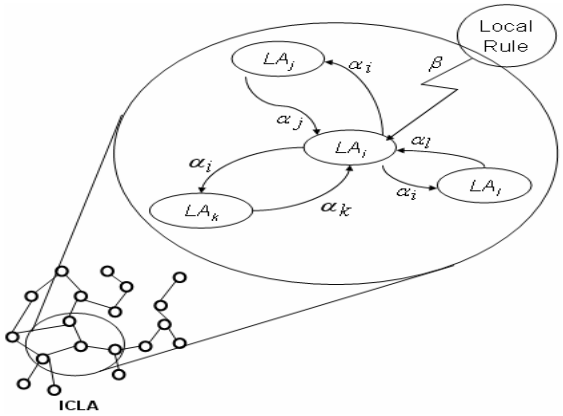

Figure 1. Irregular cellular learning automata

Note that in the definition of ICLA, no explicit definition of neighborhood of each cell is given. This is because neighborhood in ICLA is implicitly defined in definition of the graph G.

In what follows, we consider ICLA with n cells. The learning automaton A_i which has a finite action set $\underline{\alpha}_i$ is associated to cell i (for $i=1,2,,,,n$) of the ICLA. Let the cardinality of $\underline{\alpha}_i$ be m_i. The state of the ICLA represented by $p= (p_1, p_2, ..., p_n)$, where $p_i = (p_{i1}, p_{i2},..., p_{im_i})$ is the action probability vector of A_i.

The operation of the ICLA takes place as the following iterations. At iteration k, each learning automaton chooses an action. Let $\alpha_i \in \underline{\alpha}_i$ be the action chosen by A_i. Then all learning automata receive a reinforcement signal. Let $\beta_i \in \underline{\beta}$ be the reinforcement signal received by A_i. This reinforcement signal is produced by the application of local rule $F(\underline{\Phi}_i) \to \underline{\beta}$. Finally, each LA updates its action probability vector on the basis of the supplied reinforcement signal and the chosen action by the cell. This process continues until the desired result is obtained.

3. Clustering Algorithms

In this section, we will briefly overview some of the existing clustering algorithms for wireless sensor networks.

One of the first clustering algorithms introduced for sensor networks is the low energy adaptive clustering hierarchy (LEACH) algorithm [15]. The operation of LEACH is separated into two phases: the setup phase and the steady state phase. During the setup phase, a predetermined fraction of nodes p, elect themselves as cluster heads by comparing a chosen random number with a predefined threshold. After the cluster heads have been elected, they broadcast an advertisement message to the rest of the nodes in the network that they are the new cluster heads. Upon receiving this advertisement, all the noncluster head nodes decide on the cluster to which they want to belong, based on the signal strength of the advertisement. The noncluster head nodes inform the appropriate cluster heads that they will be members of the cluster.

There are algorithms which elect cluster heads based on a certain criterion such as number of neighbor nodes [16, 17], or remaining energy of the nodes [18]. In the algorithm presented in [19], each node waits for a random duration. After this period, if no message is received from a certain cluster head, the node states itself as a new cluster head. In some other methods, such as the one in [20], cluster heads are specified prior to network deployment. These specified nodes are placed in certain positions and other nodes are scattered around them. After the network deployment, these nodes try to form clusters having balanced traffic load. This means that fewer nodes are assigned to the clusters closer to the sink node and more nodes are assigned to the clusters far from the sink. In the sensor network considered in [21], two classes of nodes are available; sensor nodes and aggregator nodes. Aggregator nodes are placed in certain position and play the role of cluster heads. In this paper three different algorithms are presented for scattering sensor nodes around the aggregators.

In [22], sink node is assumed to be a mobile node which queries data from different part of the network based on its distance from different nodes. For this reason, it is required to have a dynamic clustering algorithm which can adapt itself to the changing position of the mobile sink, considering the overall energy consumption in the network.

A number of different routing algorithms for sensor networks are surveyed in [23, 24]. One major class of these algorithms is the hierarchical routing. In hierarchical routing algorithms, at

first a clustering scheme is applied to form a kind of hierarchy and afterward, the process of routing is divided into inter and intra-cluster phases. Some examples of these algorithms are "Fixed size cluster routing", TEEN[4], and APTEEN[5]. AIMRP[6] [25] assumes that the sink node is placed in the center of the network. In the setup phase, sink initiates a Tire message to the nodes in the radius of *a* around itself. These nodes constitute the first level. These nodes rebroadcast the Tier message to the nodes in the radius of *a* around themselves. This process continues until all nodes in the network find their level in the hierarchy to the sink node.

In [26] an algorithm called Hybrid Energy-Efficient Approach (HEED)[7] has been given. This algorithm has four primary goals: (i) prolonging network lifetime, (ii) terminating the clustering process within a constant number of iterations/steps, (iii) minimizing control overhead (to be linear in the number of nodes), and (iv) producing well-distributed cluster heads and compact clusters. Before a node starts executing HEED, it sets its probability of becoming a cluster head, CH_{prob} based on its residual energy. Using this probability, each node decides to become a tentative cluster head or not. Tentative cluster heads, declare themselves to their neighbors using a cluster-head message, and double their CH_{prob} probability. Once CH_{prob} in a node reaches 1, that node becomes a final cluster head. Nodes which are not cluster head (tentative or final), upon receiving a cluster-head message, join to the declared cluster. Nodes which do not receive ant cluster-head message after a while, decide to become cluster head.

In [27] an enhancement on HEED algorithm in which, only nodes with residual energy higher than a specified threshold can become cluster head is presented. Also at the end of the algorithm, when some nodes do not join to any cluster yet, unlike HEED in which all these nodes become cluster head, HEED algorithm is executed again for electing cluster heads among them.

4. The Proposed Algorithm

Three main objectives of a good clustering method for sensor networks are *i*) producing well-distributed cluster heads, *ii*) maximizing the election probability of high energy level nodes as cluster heads and *iii*) minimizing the number of sparse clusters. In this section an attempt has been made to design a clustering algorithm to satisfy all three objectives. The first objective is implicitly satisfied by the use of ICLA for the clustering. This is because in each neighborhood of the ICLA, at least one cluster head will be elected and this will guarantee the well-distribution of cluster heads among the network. The other two objectives will be satisfied through a proper selection of local rule for ICLA. Figures 2 and 3 give the pseudo code for the proposed clustering algorithm. Based on the senor network topology an ICLA is created, that is an ICLA is mapped into the network in such a way that each node in the sensor network be mapped into a node in the ICLA. Any two nodes in the sensor networks that are close enough to hear each other's signal have an edge in the corresponding graph *G* of the ICLA.

[4] Threshold-Sensitive Energy-Efficient Protocol
[5] Adaptive Periodic TEEN
[6] Address-light Integrated MAC, and Reporting Protocol
[7] Hybrid Energy-Efficient Approach

```
ICLA-Clustering()
begin
//Initial phase
    if (IsSink)
        sendMessage(StartClustering);
    else
        M = receiveMessage();
        if (M is StartClustering)
            sendMessage(StartClustering);
            goto ClusteringPhase;
        end
    end

//Clustering phase
    wait(); //Until all nodes start this phase
    //Initializing probability of actions
    a[0].probability = .5; //Not_Be_Head
    a[1].probability = .5; //Be_Head

    //Clustering Iterations
    MainLoop();

    //Final Decision
    if (a[0].probability < EPSILON)
        becomesHead();
        sendMessage(FinalClusterHead);
    else
        FinalClusterHeads[] = receiveMessage();
        joinToBest(FinalClusterHeads);
    end
end
```

Figure 2. Pseudo code for the ICLA-Clustering algorithm. Each node follows this code separately

The proposed algorithm has two phases. In the initial phase, sink node initiates 'Start-Clustering' message throughout the network. Each node, upon receiving this message, rebroadcasts it to its neighbors and then enters the clustering phase which is performed using ICLA. LA in each node has two different actions $a_0=0$ and $a_1=1$. a_1 indicates that the node should declare itself as the cluster head, and a_0 indicates the opposite decision. At the start of the clustering phase, each node sets the initial probability of these two actions to .5 and enters the clustering iterations. At each iteration *k*, the LA in each node chooses one of its actions a_0 or a_1. Each node *i* then forwards the selected action of its LA to its neighbors. Afterward, node *i*, waits for a specified duration to receive its neighbors messages indicating their selected actions. At the end of this duration, each node calculates the response β based on the local rule of the ICLA. This local rule is designed in such a way that to both maximizing the election probability of high energy level nodes as cluster heads and minimizing the number of sparse clusters. The local rule for node *i* can be defined informally as follows:

```
MainLoop()
begin
    while (a[0].probability > EPSILON &&
           a[1].probability > EPSILON)
    begin
        action = selectAction();
        sendMessage(action);
        NeighborActions[] = receiveMessae(NeighborActions);

        //Following information are exchanged
        //periodically between neighbors
        MaxEnergyLevel = max(NeighborEnergyLevels[]);
        MaxNeighborNumber = max(NeighborNumbers[]);

        //Compute the response
        if (action is a[1])
            if (No i in NeighborActions[i] is a[1])
                reward(action);
            else if (isBetterForBecommingHead(MaxEnergyLevel,
                     MaxNeighborNumber))  //Using equation (4)
                reward(action);
            else
                penalize(action);
            end
        else
            if (No i in NeighborActions[i] is a[1])
                penalize(action);
            else if (isBetterForBecommingHead(MaxEnergyLevel,
                     MaxNeighborNumber))  //Using equation (4)
                penalize(action);
            else
                reward(action);
            end
        end
    end
end
```

Figure 3. Pseudo code for the main loop each node follows in the proposed clustering method

- If selected action of LA in node *i* is a_1 (node declares itself as a cluster head) then
  - If no neighbor of node *i* declares itself as a cluster head, a favorable response will be generated.
  - If some neighbors of node *i* declare themselves as cluster head, but node *i* has more residual energy and more number of neighbors among them, a favorable response will be generated.
  - Otherwise, an unfavorable response will be generated.
- If selected action of LA in node *i* is a_0 (node doesn't declare itself as a cluster head) then
  - If no neighbor of node *i* declares itself as a cluster head, au unfavorable response will be generated.
  - If some neighbors of node *i* declare themselves as cluster head, but node *i* has more residual energy and more number of neighbors among them, an unfavorable response will be generated.
  - Otherwise, a favorable response will be generated.

The local rule can be defined formally as follows. Let $\underline{a}_i$ be the action selected by node *i* and $\underline{a}_j$ be the action selected by the neighbor node *j* (which can be either a_0 or a_1). Equation (3) gives the local rule of the ICLA.

$$\beta = \begin{cases} 0; & \underline{a}_i \cdot (P \cdot \sum_{j=1}^{N_i} \underline{a}_j) + (1 - \underline{a}_i)) + \\ & (1 - \underline{a}_i).(-P \cdot \sum_{j=1}^{N_i} \underline{a}_j) - (1 - \underline{a}_i) \geq 0 \\ 1; & \textit{Otherwise} \end{cases} \quad (3)$$

In (3), N_i is the number of neighbors of node i and P is a parameter which specifies the suitability of node i for becoming cluster head among its neighbors. P is defined using (4) given below.

$$P = \alpha \cdot [EnergyLevel_i - EnergyLevel_{Max}] + (1-\alpha) \cdot [N_i - N_{Max}] - K \quad (4)$$

In equation (4), $EnergyLevel_i$ is the remaining energy level of the node i. α is a coefficient which controls the effect of energy level and number of neighbors on the selected cluster heads. Higher value of α indicates cluster heads with more energy level, and lower value of it indicates cluster heads with higher number of neighbors. K is a constant which controls the suitability of node i for becoming cluster head among its neighbors. Large values of K indicate more suitability which results in few clusters. This may leave some parts of the network with no cluster. On the other hand, choosing small values of K results in too many clusters which leads to more number of sparse clusters. Finally, $EnergyLevel_{Max}$ and N_{Max} are defined using equations (5) and (6) respectively.

$$EnergyLevel_{Max} = \max_{1 \le j \le N_i} (EnergyLevel_j) \quad (5)$$

$$N_{Max} = \max_{1 \le j \le N_i} (N_j) \quad (6)$$

In these equations, $EnergyLevel_j$ is the remaining energy level of the neighbor j and N_j is the number of neighbors of the node j. This information about neighbor nodes are exchanged between nodes periodically via simple announcement messages.

For positive values of P, node i would be a better choice than its neighbors for becoming cluster head, but negative values of P indicates that some neighbors are better choices than i for this purpose.

Based on the response specified by the equation (3), LA resides in node i rewards (penalizes) its selected action $\underline{a}_i$ using equations (7)((8)), respectively.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a.(1 - p_i(n)) \\ p_j(n+1) &= 1 - p_i(n+1) \end{aligned} \quad (7)$$

$$\begin{aligned} p_i(n+1) &= (1-b) \cdot p_i(n) \\ p_j(n+1) &= 1 - p_i(n+1) \end{aligned} \quad (8)$$

In the above two equations, a is the reward rate and b is the punishment rate. In the simulations conducted, equations (9) and (10) are used to compute a and b, respectively.

$$a = \underline{a}_i \cdot [Alpha \cdot (N_i.EnergyLevel_i)] + (1-\underline{a}_i) \cdot [\sum_{j=1}^{N_i} \underline{a}_j \Big/ (N_i \cdot EnergyLevel_i)] \quad (9)$$

$$b = \underline{a}_i \cdot [\sum_{j=1}^{N_i} \underline{a}_j \Big/ (N_i \cdot EnergyLeve \quad l_i)] + (1 - \underline{a}_i) \cdot [Alpha \cdot (N_i.EnergyLeve \quad l_i)] \quad (10)$$

In equations (9) and (10), *Alpha* is a parameter which is defined through equation (11). This parameter is used to control the value of *a* and *b* so that they will be always less than 1.

$$Alpha = \min(EnergyLevel_i, 1/(N_i \cdot EnergyLevel_i)) \quad (11)$$

This process continues until the action probability of one of the actions of the LA in each node passes a certain threshold near to 1. Nodes for which, the probability of selecting action a_1 of their LA passes this threshold, become cluster heads and broadcast messages indicating that they are final cluster heads. Other nodes wait for the final cluster head messages from their neighbors and join to the one which has higher energy level and higher number of neighbors.

5. Experimental Results

To evaluate the performance of the proposed method several experiments have been conducted. In these experiments, the proposed method is compared with the basic HEED clustering method proposed in [26] and its extension given in [27]. Also, we study the convergence behavior of learning automata in different nodes of the network. All simulations have been implemented using NS2 simulator. We use IEEE 802.11 as the MAC layer protocol. Nodes of the network are placed randomly on a 2 dimensional grid. In all experiments, α is set to .3 and *K* is set to 1.2. Simulations are performed for 50, 100, 200, 300, 400, and 500 nodes. The results are averaged over 20 runs.

Experiment 1: In this experiment, we study the cluster formation behavior of the proposed method in comparison with two other methods mentioned earlier. For this purpose, we make use of the following metrics: total number of clusters formed in the network, number of sparse clusters (which is defined as the clusters having less than 3 nodes) and mean number of nodes in each cluster. Figure 4 depicts the number of clusters formed in the network, number of sparse clusters is depicted in figure 5 and figure 6 depicts the mean number of nodes in each cluster. These figures show that the cluster formation in the proposed method is better than the other two methods. This is because it forms less number of clusters and more nodes in each cluster on average. Also, number of sparse clusters in the proposed method is less than that in two other methods. This means that the proposed method can better distribute cluster heads among the sensor network.

Experiment 2: In this experiment, mean energy level of cluster heads at the end of clustering algorithm and prior to any data transition is studied. Initial energy level of each node is selected randomly between 60% and 100% of the maximum energy level (full charged). For estimating the energy consumption of each node, we assume that sending a single packet will consume .005% of the maximum energy level and receiving a single packet will consume .001% of the maximum energy level. Energy consumption in idle state is assumed to be 0. Figure 7 depicts the results for the proposed method in comparison to other two methods. This figure shows that the proposed method outperforms HEED and its extension in terms of energy level of cluster heads with the factor of approximately 1.13.

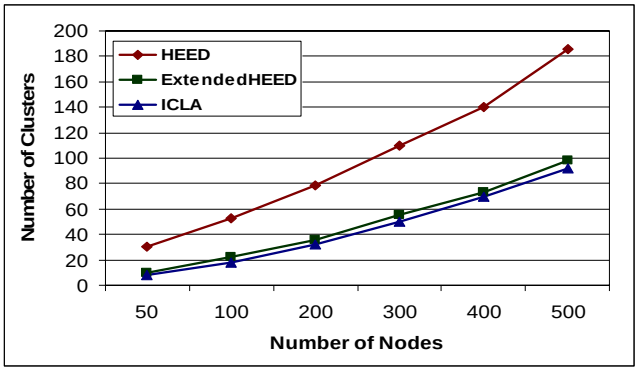

Figure 4. Total number of clusters formed in the network for the proposed method and other methods

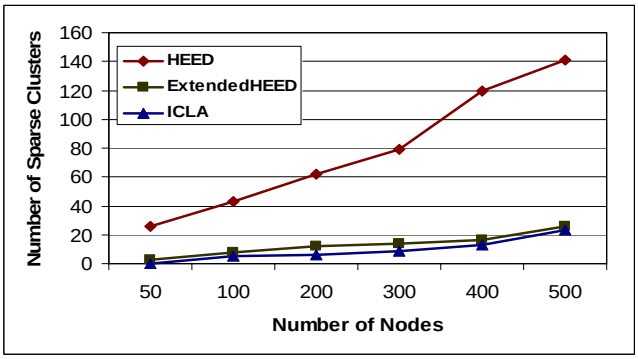

Figure 5. Number of sparse clusters formed in the network for the proposed method and other methods

Experiment 3: This experiment is conducted to better understand the convergence behavior of learning automata residing in the nodes of the network. The sensor network used in this experiment is composed of 50 nodes which are scattered randomly throughout a 2 dimensional grid of 1000x1000 meter. Figure 8 depicts this network and figure 9 depicts clusters formed in it using the proposed algorithm.

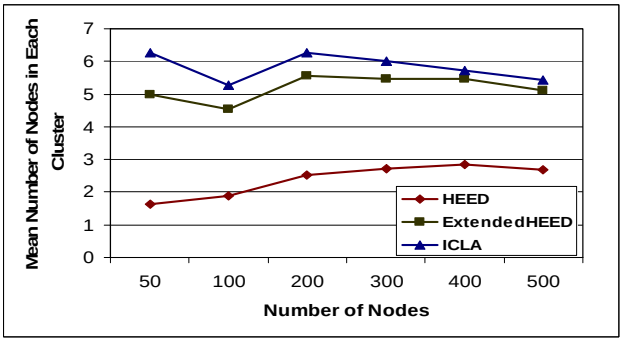

Figure 6. Mean number of nodes in each cluster for the proposed method and other methods

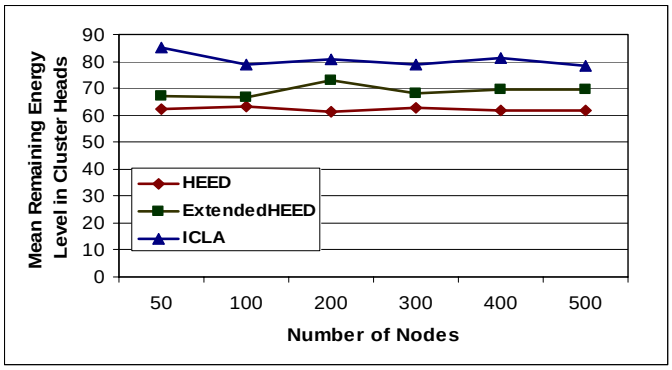

Figure 7. Mean remaining energy level in cluster heads for the proposed method and other methods

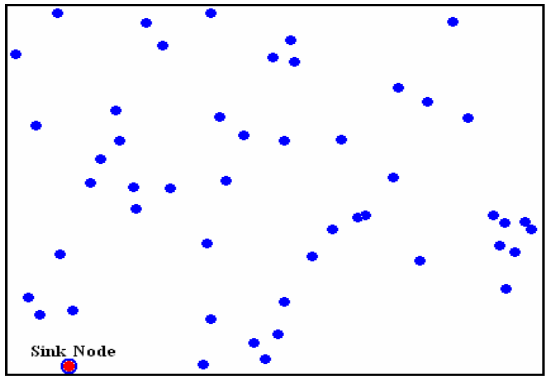

Figure 8. Sensor network of 50 nodes in a grid of 1000x1000 meter

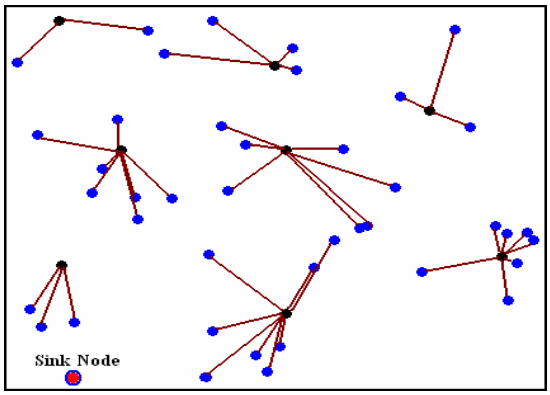

Figure 9. Sensor network of figure 8 which is clustered using proposed method

We study the action probabilities of two automata residing in two nodes; one is a cluster head and the other is a cluster member. Figure 10 shows convergence of the action probability of action "Be Head" to 1 in the learning automaton residing in the cluster head node to the action "Be Head". Figure 11 shows convergence of the action probability of action "Be Member" to 1 in the learning automaton residing in the other node.

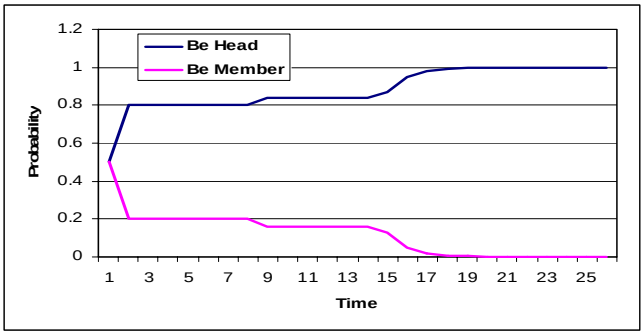

Figure 10. Action probabilities of the learning automaton in the cluster head node

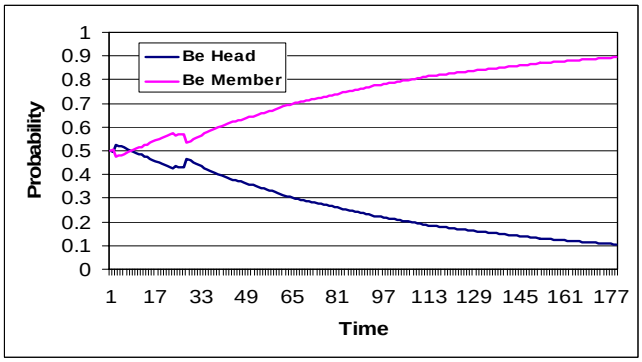

Figure 11. Action probabilities of the learning automaton in the cluster member node

6. Conclusion

In this paper we propose irregular cellular learning automata as a generalization to the cellular learning automata which removes the restriction of rectangular grid structure in traditional CLA. There are a number of problems and applications which cannot be adequately modeled with rectangular grids, one of which is the area of wireless sensor networks. In these networks, nodes are usually scattered randomly throughout the environment, so no regular structure can be assumed for modeling their behavior. Based on the proposed ICLA model, we present a novel clustering algorithm for sensor networks in which, each node in the network is a cell in the ICLA, and hence equipped with an LA. Any two nodes that are close enough together to hear each other's signal have a common edge in the graph *G* of the ICLA. LA in each node has two actions one of which indicates that the node should declare itself as a cluster head and the other one indicates the opposite decision. Using local rule of the ICLA, action probability of one of these two actions converges to 1 in each of these LA. Nodes for which probability of action "Be Head" of their LA converges to 1, becomes cluster head and other nodes become cluster members. It has been shown through simulations that the proposed clustering algorithm outperforms other similar methods in terms of cluster formation as well as remaining energy level of cluster heads at the end of the clustering.

References

[1] H. Beygi, M. R. Meybodi, "A Mathematical Framework for Cellular Learning Automata", *Advances in Complex Systems*, Vol. 7, Nos. 3 & 4, pp. 295-319, September & December 2004.

[2] E. Fredkin, "Digital machine: A informational process based on reversible cellular automata", *Physica D45*, pp. 245-270, 1990.
[3] M. Mitchell, "Computation in cellular automata: A selected review", *Technical report,* Santa Fe Institute, Santa Fe, NM, USA, September 1996.
[4] N. H. Packard, S. Wolfram, "Two-dimensional cellular automata", *Journal of Stat. Phys. 38*, pp. 901-946, 1985.
[5] J. Kari, "Reversibility of 2D cellular automata is undecidable", *Physica D45*, pp. 379-385, 1990.
[6] M. R. Kharazmi, M. R. Meybodi, "Image Segmentation Using Cellular Learning Automata", *in Proc. of 10th Iranian Conf. on Electrical Engineering, ICEE-96,* Tabriz, Iran, May 2001.
[7] M. R. Kharazmi, M. R. Meybodi, "Image Restoration Using Cellular Learning Automata", *in Proc. of 2nd Iranian Conf. on Machine Vision, Image Processing and Applications,* Tehran, Iran, pp. 261-270, 2003.
[8] M. R. Meybodi, H. Beygi, M. Taherkhani, "Cellular Learning Automata and its Applications", *Journal of Science Tech.*, Sharif (Sharif University of Technology, Tehran, Iran), pp. 54-77, 2004.
[9] M. R. Meybodi, M. R. Khojaste, "Application of Cellular Learning Automata in Modeling of Commerce Networks", *in Proc. of 6th Annual Intl. Computer Society of Iran Computer Conf., CSICC-2001,* Isfehan, Iran, pp. 284-295, February 2001.
[10] M. R. Meybodi, M. Taherkhani, "Application of Cellular Learning Automata in Modeling of Rumor Diffusion", *in Proc. of 9th Conf. on Electrical Engineering,* Power and Water Institute of Technology, Tehran, Iran, pp. 102-110, May 2001.
[11] H. Beygi, M. R. Meybodi, "A Self-Organizing Channel Assignment Algorithm: A Cellular Learning Automata Approach", *Springer-Verlag Lecture Notes in Computer Science,* Vol. 2690, pp. 119-126, 2003.
[12] H. Beygi, M. R. Meybodi, "Call Admission in Cellular Networks: A Learning Automata Approach", *Springer-Verlag Lecture Notes in Computer Science,* Vol. 2510, pp. 450-457, 2002.
[13] H. Couclelis, "Cellular Worlds - a framework for modeling micro-macro dynamics", *Environment and Planning*, pp. 585-596, 1985.
[14] D. Stevens, "Integration of an Irregular Cellular Automata Approach and Geographic Information Systems for High-Resolution Modeling of Urban Growth", Msc. Thesis, Department of Geography, University of Toronto, 2003.
[15] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, “Energy Efficient Communication Protocol for Wireless Microsensor Networks”, *Intl. Conf. on System Sciences*, Hawaii, January 2000.
[16] C. Y. Wen, W. A. Sethares, “Automatic Decentralized Clustering for Wireless Sensor Networks”, *EURASIP Journal on Wireless Communications and Networking, pp. 686-697,* March 2005.
[17] K. Shin, A. Abraham, S. Y. Han, “Self Organizing Sensor Networks Using Intelligent Clustering”, *Intl. Conf. on Computational Science and Applications, UK, M. Gavrilova et al. (Eds.), Lecture Notes in Computer Science (LNCS 3983), Springer Verlag, Germany, pp. 40 – 49,* 2006.
[18] M. Ye, Ch. Li, G. Chen, J. Wu, “EECS: An Energy Efficient Clustering Scheme in Wireless Sensor Networks”, In *Proc. of the IEEE Intl. Workshop on Strategies for Energy Efficiency in Ad Hoc and Sensor Networks* (IWSEEASN'05), April 2004.
[19] V. Mittal, M. Demirbas, A. Arora, “LOCI: Local Clustering Service for Large Scale Wireless Sensor Networks”, *Technical Report OSU-CISRC-2/03-TR07, Ohio State University*, 2003.
[20] S. Soro, W. B. Heinzelman, “Prolonging the Lifetime of Wireless Sensor Networks via Unequal Clustering”, *5th Intl. IEEE Workshop on Algorithms for Wireless, Mobile, Ad Hoc, and Sensor Networks (WMAN05),* April 2005.
[21] Y. Guo, J. MeNair, “Cost Efficient Cluster Formation for Wireless Sensor Networks”, In *Proc. of the Intl. Conf. on Cybernetics and Information Technologies, Systems, and Applications (CITSA),* Orlando, Florida, July 2004.
[22] M. Lotfinezhad, B. Liang, “Energy Efficient Clustering in Sensor Networks with Mobile Agents”, In *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC),* New Orleans, Louisiana, March 2005.

[23] K. Akkaya, M. Younis, "A survey on routing protocols for wireless sensor networks", *Elsevier Ad Hoc Network Journal, pp. 325-349,* 2005.
[24] M. Ilyas, I. Mahgoub, "Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems", *CRC Press,* London, Washington, D.C., 2005.
[25] S. Kulkarni, A. Iyer, C. Rosenberg, "An Address-Light, Integrated MAC and Routing Protocol for Wireless Sensor Networks", *to appear in IEEE/ACM Transactions on Networking,* accepted August 2005.
[26] O. Younis and S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach", In *Proc. of IEEE INFOCOM, volume 1, pp. 629-640,* March 2004.
[27] H. Huang, J. Wu, "A Probabilistic Clustering Algorithm in Wireless Sensor Networks", In *Proc. of IEEE 62nd Semiannual Vehicular Technology Conference* (VTC), Sept. 2005.