Cellular Learning Automata and Its Applications

Amir Hosein Fathy Navid and
Amir Bagheri Aghababa

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/52953

1. Introduction

Cellular Automata are mathematical models for systems consisting of large number of simple identical components with local interactions. Cellular Automata 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.

Informally, a d-dimensional Cellular Automata 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 [4]. The new state of each cell depends on the previous states of a set of cells, including the cell itself, and constitutes its neighbourhood. 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 [11].

Cellular Automata provided a potential solution and is probably the most popular technique to model the dynamics of many processes, since they can predict complex global space pattern dynamic evolution using a set of simple local rules.

However, Cellular Automata is usually associated to bi-dimensional matrixes of rectangular identical cells that are not the most adequate to model and tessellate a real world geographic area.

Regular grids, or more particularly: rectangular grids are the standard grid structure that is used in previous Cellular Automata studies. Broadly, a regular grid assumes that the structure of the cell grid and the number of neighbours are homogenous for every location in the cellular space. This assumption seems highly implausible as an empirical description of the geographical or social space that underlies the processes typically studied in Cellular Automata modelling, like opinion formation or neighbourhood segregation. However, to our knowledge there are virtually no insights into how regular vs. irregular grid structures affect cellular dynamics [14].

Cellular Automata extensions using Voronoi spatial models have been previously proposed to overcome this problem. In these approaches one uses convex cells with different sizes and shapes that can provide a much more adequate terrain partition.

A different problem lies in the fact that, on regular Cellular Automata, each cell has a finite set of possible states, and transition between states is a crisp function of present cell state and neighbour cells state. Crisp data modelling and crisp transition mechanisms have known limitations when one trying to model and simulate real-world processes where uncertainty and imprecision is present and cannot simply be ignored [28].

The most prominent reason is that Cellular Automata can be seen as multi-agent system based on locality with overlapping interaction structures. In this perspective, Cellular Automata is attractive as a modelling framework that may provide a better understanding of micro/macro relations.

We will give some background specific to the study of cellular automata, and then background from other fields that are necessary for the work here [17].

We will then conclude with some potential questions that merit future investigation, and where appropriate we will discuss potential consequences of such questions.

2. Irregular Cellular Automata

Practically all social science applications of cellular modelling use a regular grid as the underlying network structure. More in particular, the standard grid structure used is a rectangular regular grid. Other regular grids could be hexagonal or triangular structures. In general, we denote grids as regular where all inner cells (i.e. cells that are not at the border of the grid) have the same number of neighbours, whatever our neighbourhood definition may be - von Neumann neighbourhood or a Moore neighbourhood of a given size. On a regular torus, this definition generalises even to border cells [7].

Regular Cellular Automata has cells with identical shape and size. Since geographic features in nature are usually not distributed uniformly, regular spatial tessellation obviously limits modelling and simulation potential of regular Cellular Automata. In order to overcome this limitation, several authors have extended the Cellular Automata model to irregular cells. The most successful approaches use the Voronoi spatial model [10].

A Cellular Automata is a system composed by several identical automata, physically organized as a 2 dimensional array of rectangular cells, where each cell is considered an automaton, A, with a set of rules, T, which gets its inputs from its own state and from neighboring cells states V:

$$A \sim (S, T, V) \tag{1}$$

Figure1 shows the regular cellular Automata with regular cells.

The Voronoi spatial model is a tessellation of space that is constructed by decomposing the entire space into a set of Voronoi regions around each spatial object. By definition, points in the Voronoi region of a spatial object are closest to the spatial object than to any other spatial object [5]. The generations of Voronoi regions can be considered as 'expanding' spatial objects at a unique rate until these areas meet each other. The mathematical expression of the Voronoi region is defined as:

$$V(p_i) = \{p \mid d(p, p_i) \le d(p, p_j), j \ne i, j = 1 \ldots n\} \tag{2}$$

In this equation, the Voronoi region of spatial object *pi*, *V*(*pi*), is the region defined by the set of locations *p* in space where the distance from *p* to the spatial object *pi*, *d*(*p*, *pi*), is less than or equal to the distance from *p* to any other spatial object *pj*. In figure 2, the Voronoi based Cellular Automata is shown.

Voronoi region boundaries are convex polygons. Points along a common boundary between Voronoi regions are equidistant to the corresponding spatial objects. Objects which share a common boundary are neighbors to each other in the Voronoi spatial model [5, 12].

In this section, Irregular Cellular Automata has been defined in context but for a better understanding, we have also explained Voronoi diagrams concept and modelling irregular grid structures using a Voronoi diagram further in this section.

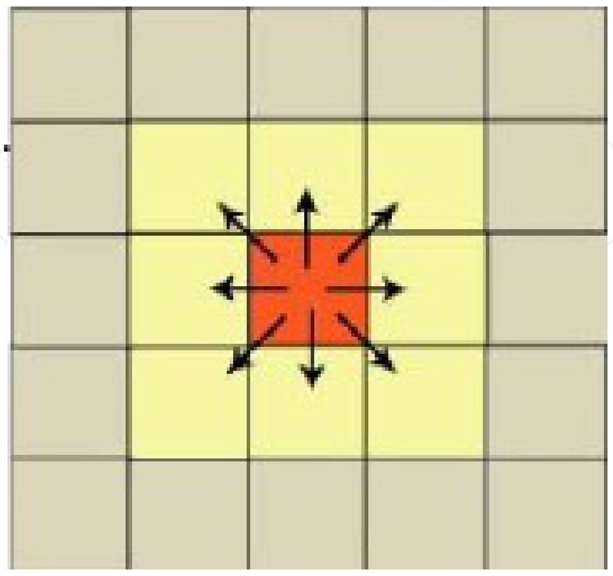

Figure 1. Regular Cellular Automata

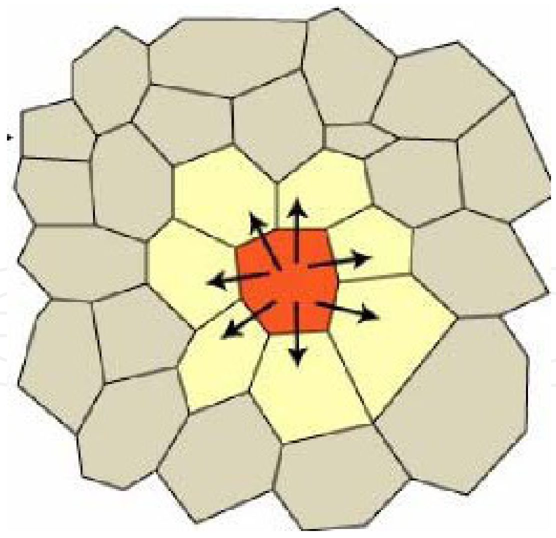

Figure 2. Voronoi Based Cellular Automata

2.1. Voronoi Diagrams

We begin with a description of elementary, though important, properties of the Voronoi diagram that will suggest some feelings for this structure. We also introduce notation used throughout this paper.

In this section, we introduce concepts of Voronoi Diagrams and describe necessary steps to create them. A naive approach to construct a Voronoi diagram is to determine the region for each point using Euclidean distance [16]. For points $p= (x_p, y_p)$ and $q= (x_q, y_q)$ in the plane, equation 3 denote their Euclidean distance.

$$d(p,q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \tag{3}$$

By $\overline{pq}$, we denote the line segment from p to q. To draw Voronoi diagram, we use perpendicular bisectors of point set on the 2D space as it is shown in figure 3.

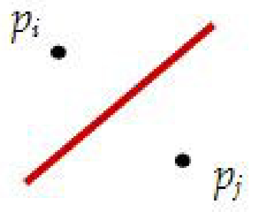

Figure 3. Divided plane with perpendicular bisector of two points

Given a set of S points $p_1, p_2, \ldots, p_n$ in the plane, a Voronoi diagram divides the plane into n Voronoi regions with the following properties:

- Each point p_i lies in exactly one region.
- If a point $q \notin S$ lies in the same region as p_i, then the Euclidian distance from p_i to q will be shorter than the Euclidian distance from p_j to q, where p_j is any other point in S.

The points $p_1, \ldots, p_n$ are called Voronoi sites. The Voronoi diagram for two sites p_i and p_j can be easily constructed by drawing the perpendicular bisector of line segment $\overline{p_i q}_j$.

Such diagrams would consist of two unbounded Voronoi regions, denoted by $V(p_i)$ and $V(p_j)$, in equation 4. In general, a Voronoi region $V(p_i)$ is defined as the intersection of $n - 1$ half-planes formed by taking the perpendicular bisector of the segment for all where $i \neq j$.

$$V(p_i) = H(p_i p_1) \cap H(p_i p_2) \cap \ldots \cap H(p_i p_n) \qquad (4)$$

In this notation, $H(p_i p_j)$ refers to the half-plane formed by taking the perpendicular bisector of $p_i p_j$ in figure 4. We know that the intersection of any number of half-planes forms a convex region bounded by a set of connected line segments. These line segments form the boundaries of Voronoi regions and are called Voronoi edges. The endpoints of these edges are called Voronoi vertices [8, 12].

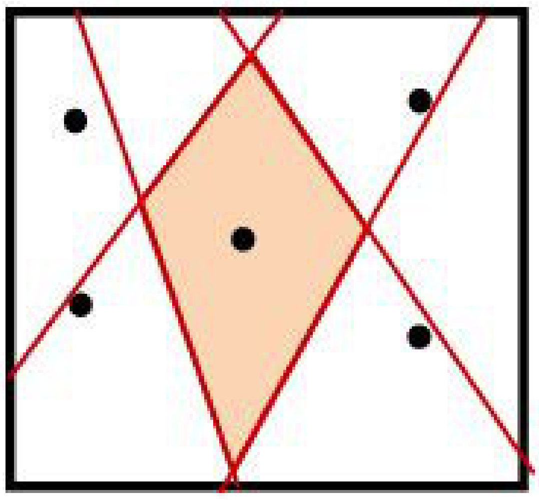

Figure 4. Create half-plane by perpendicular bisector

The points on Voronoi edges of Voronoi diagram are in equal distance of Voronoi sites p_i and p_j. You can see an example of Voronoi diagram in Figure 5.

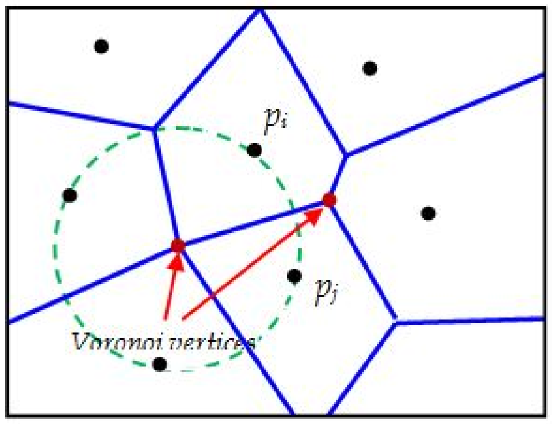

Figure 5. Voronoi diagrams of collection points on plane

Equation (5) shows the Voronoi region of p with respect to *S*, for $p,\ q \in S$.

$$V(p,S) = \bigcap_{q \in S, q \neq p} H(p,q) \tag{5}$$

Finally, the Voronoi diagram of *S* is defined by equation 6.

$$Voronoi(S) = \bigcup_{p,q \in S, p \neq q} \overline{V(p,S)} \cap \overline{V(q,S)} \tag{6}$$

By definition, each Voronoi region is the intersection of $n - 1$ open half-planes containing the site *p*.

2.2. Properties of Voronoi Diagrams

- The number of Voronoi vertices is at most $2n - 5$.
- The number of Voronoi edges is at most $3n - 6$.
- Each Voronoi vertex is the common intersection point of exactly three edges.
- If site $p_i \in S$ is the nearest neighbor of site $p_j \in S$, then the Voronoi regions $V(p_i)$ and $V(p_j)$ will share a common edge.
- Region *V(p)* is unbounded iff *p* is an extreme point of *S*. That is, *p* will be part of the convex hull of *S*.

Given a triangle Δ*abc*, the perpendicular bisector of each edge will intersect at a common point *q* called the circumcenter. The circumcenter is equi-distant from points *a*, *b*, *c* and these points all lie on a circle with *q* as its center. This circle is called the circumcircle for triangle Δ*abc* [16].

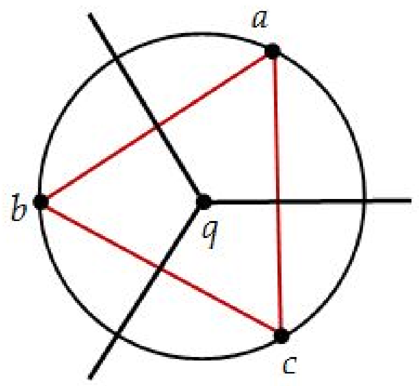

Figure 6. The circumcircle with circumcenter *q*

If a circumcircle is empty in its interior then in a Voronoi diagram:

- *a, b, c* would be Voronoi sites
- *q* would be a Voronoi vertex
- The perpendicular bisectors of Δ*abc* would be Voronoi edges.

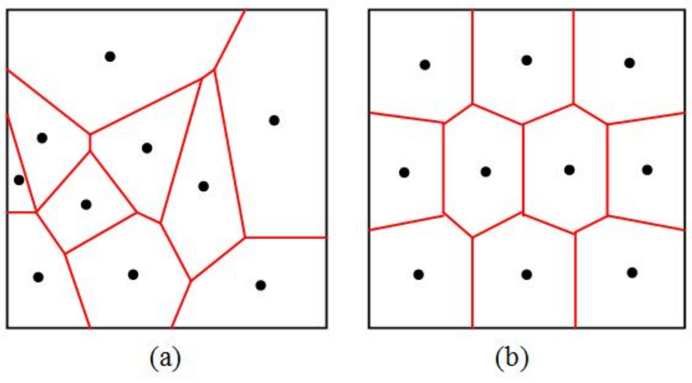

Figure 7. Voronoi Diagrams with (a) 10 random points (b) 10 simultaneous points

Figure 7 shows, on the left, the Voronoi regions corresponding to 10 randomly selected points in a square; the density function is constant. The dots are the Voronoi generators and the circles are the centroids of the corresponding Voronoi regions. Note that the generators and the centroids do not coincide. On the right, the 10 dots are simultaneously the generators for the Voronoi tessellation and the centroids of the Voronoi regions.

2.3. Constructing Voronoi Diagrams

2.3.1. Naive Approach

A naive approach to construct a Voronoi diagram is to determine the region for each site one at a time. Since each region is the intersection of $n-1$ half-planes, we can use an $O(n \log n)$ half-plane intersection algorithm to determine this region. Repeating for all n points, we have an $O(n2 \log n)$ algorithm.

2.3.2. Divide and Conquer

To construct a Voronoi diagram using the divide and conquer method, first partition the set of points S into two sets L and R based on x-coordinates. Next, construct the Voronoi diagrams for the left and right subset $V(L)$and $V(R)$. Finally, merge the two diagrams to produce $V(S)$. If the merge step can be carried out in linear time, then the construction of $V(S)$ can be accomplished in $O(n \log n)$ time [16].

2.4. Irregular Grids in a Cellular Automaton

To model irregular grid structures, we use a Voronoi diagram. The crosses in Voronoi diagram are the generators of the grid. The edges of the resulting polygons are points with equal distance to their neighboring generators [10].

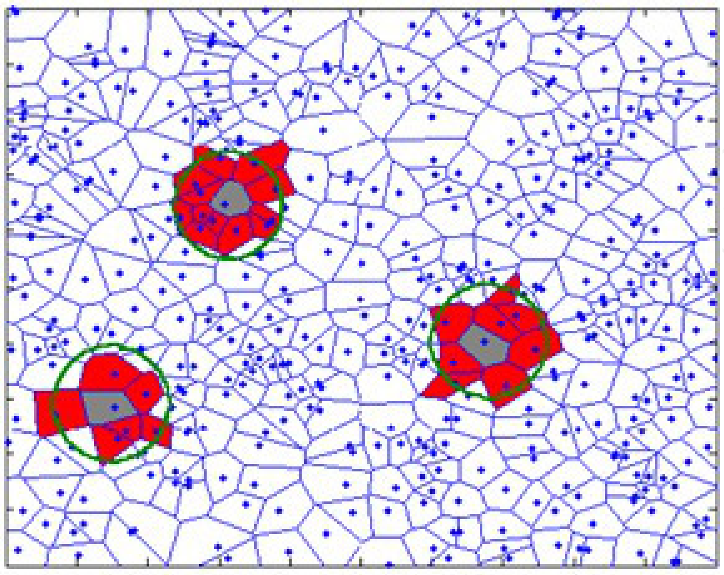

Figure 8. Neighborhoods of three different cells in an irregular field. Focal cells are gray and neighbor cells are red.

Figure 8 shows a decisive feature of irregular grids: even the cells inside the grid have different numbers of next neighbors. The figure shows locations of three different cells with 6, 8 and 12 next neighbors in an irregular grid. We define a next neighbor cell here as a cell that has a common border (not just a common edge) with the focal cell. Notice that this definition implies a von Neumann neighborhood on a rectangular grid. More in general, it has been found in simulation analyses that in a Voronoi graph, the number of neighbor cells varies between 3 and 14 [10].s

The idea of irregular cellular automata was suggested in mid 80s, but due to the computationally intensive operations required to search irregular neighborhood, it has been paid less attention to since then. In an informal way, Irregular Cellular Automata is a configuration of points in the space with no prior restriction. Each point has a number of other points as its neighbors such as figure 8. [8].

3. Learning Automata Concepts

In this section, we present the learning automata concept, cellular learning automata and irregular cellular learning automata.

3.1. Learning Automata

Learning Automaton is a simple entity which 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 on the reinforcement signal at that stage, and evolves to some final desired behavior [1].

Learning Automata is an abstract model which randomly selects one action out of its finite set of actions and performs it on a random environment. Environment, then evaluates the selected action and responses to the automata with a reinforcement signal. Based on the selected action and received signal, the automata updates its internal state and selects its next action. Figure 9 depicts the relationship between an automata and its environment.

Environment can be defined by the triple $E=\{\alpha, \beta, c\}$ where $\alpha=\{\alpha_1, \alpha_2 \ldots, \alpha_r\}$ represents a finite input set, $\beta=\{\beta_1, \beta_2, \ldots, \beta_r\}$ represents the output set, and $c=\{c_1, c_2, \ldots, c_r\}$ is a set of penalty probabilities where each element c_i of c corresponds to one input action α_i. Learning automata are classified into fixed structure stochastic, and variable structure stochastic [17, 18]. In the following, we consider only variable structure automata.

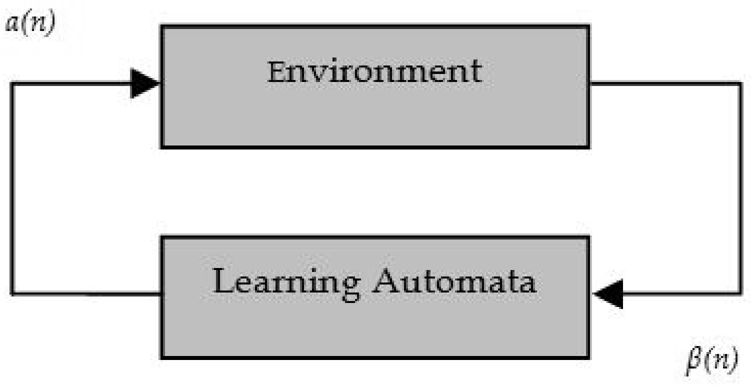

Figure 9. Relationship between learning automata and its environment

A variable structure automata is defined by the quadruple $\{\alpha, \beta, p, T\}$ in which $\alpha=\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ represents the action set of the automata, $\beta=\{\beta_1, \beta_2, \ldots, \beta_r\}$ represents the input set, $p=\{p_1, p_2, \ldots, p_r\}$ represents the action probability set, and finally $p(n+1)=T[\alpha(n), \beta(n), p(n)]$ represents the learning algorithm. This automaton operates as follows. Based on the action probability set p, automaton randomly selects an action α_i, and performs it on the environment. Having received the environment's reinforcement signal, automaton updates its action probability set based on equation (7) for favorable responses, and on equation (8) for unfavorable ones [18].

$$\begin{aligned} p_i(n+1) &= p_i(n) + a.(1-p_i(n)) \\ p_j(n+1) &= p_j(n) - a.p_j(n) \qquad \forall j \quad j \neq i \end{aligned} \tag{7}$$

$$\begin{aligned} p_i(n+1) &= (1-b).p_j(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)p_j(n) \qquad \forall j \quad j \neq i \end{aligned} \tag{8}$$

In these two equations, a and b are reward and penalty parameters, respectively. For $a = b$, learning algorithm is called $L_{R\text{-}P}$, for $a \ll b$, it is called $L_{R\varepsilon P}$, and for $b=0$ it is called $L_{R\text{-}I}$. For more information about learning automata the reader may refer to Learning automata that are, by design, "simple agents for doing simple things". The full potential of a Learning Automata is realized when multiple automata interact with each other. Interaction may assume different forms such as tree, mesh, array and etc. Depending on the problem that needs to be solved, one of these structures for interaction may be chosen. In most applications, full interaction between all Learning Automatons is not necessary and is not natural. Local interaction of Learning Automatons which can be defined in a form of graph such as tree, mesh, or array, is natural in many applications.

On the other hand, Cellular Automata are mathematical models for systems consisting of large numbers of simple identical components with local interactions. Cellular Automata and Learning Automata are combined to obtain a new model called Cellular Learning Automata (CLA). This model is superior to Cellular Automata because of its ability to learn and also is superior to single Learning Automata because it is a collection of Learning Automatons which can interact with each other.

3.2. Cellular Learning Automata

Cellular Learning Automata is a mathematical model for dynamical complex systems that consists of large number of simple components. The simple components have learning capability and act together to produce complicated behavioral patterns. A Cellular Learning Automata is a Cellular Automata in which a Learning Automata will be assigned to its every cell [4]. The learning automaton residing in each cell determines the state of the cell on the basis of its action probability vector. Like Cellular Automata, there is a rule that Cellular Learning Automata operates according to it. The rule of Cellular Learning Automata and the actions selected by the neighboring Learning Automatons of any cell determine the reinforcement signal to the Learning Automata residing in that cell. In Cellular Learning Automata, the neighboring Learning Automatons of any cell constitute its local environment. This environment is non-stationary because of the fact that it changes as action probability vectors of neighboring Learning Automatons vary [7].

The operation of cellular learning automata could be described as follows: At the first step, the internal state of every cell is specified. The state of every cell is determined on the basis of action probability vectors of the learning automata residing in that cell. The initial value of this state may be chosen on the basis of past experience or at random. In the second step, the rule of Cellular Learning Automata determines the reinforcement signal to each learning automaton residing in that cell. Finally, each learning automaton updates its action probability vector on the basis of supplied reinforcement signal and the chosen action. This process continues until the desired result is obtained (figure 10). Formally a d−dimensional Cellular Learning Automata is given below.

A d−dimensional cellular learning automata is a structure $A = (Z^d, \Phi, A, N, F)$, here

1. Z^d is a lattice of d−tuples of integer numbers.
2. Φ is a finite set of states.
3. A is the set of Learning Automatons each of which is assigned to each cell of the Cellular Automata.
4. $N = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_m\}$ is a finite subset of Z^d called neighborhood vector where $\bar{m}$ represents the number of neighboring cells and $\bar{x}_i \in Z^d$.

The neighborhood vector determines the relative position of the neighboring cells from any given cell u in the lattice Z^{d}. The neighbors of a particular cell u are set of cells $\{u+\bar{x}_i \mid i=1, 2, \ldots, \bar{m}\}$. We assume that there exists a neighborhood function $\bar{N}(u)$ mapping a cell u to the set of its neighbors according to equation (9).

$$\bar{N}(u) = \{u+\bar{x}_1, u+\bar{x}_2, \ldots, u+\bar{x}_{\bar{m}}\} \tag{9}$$

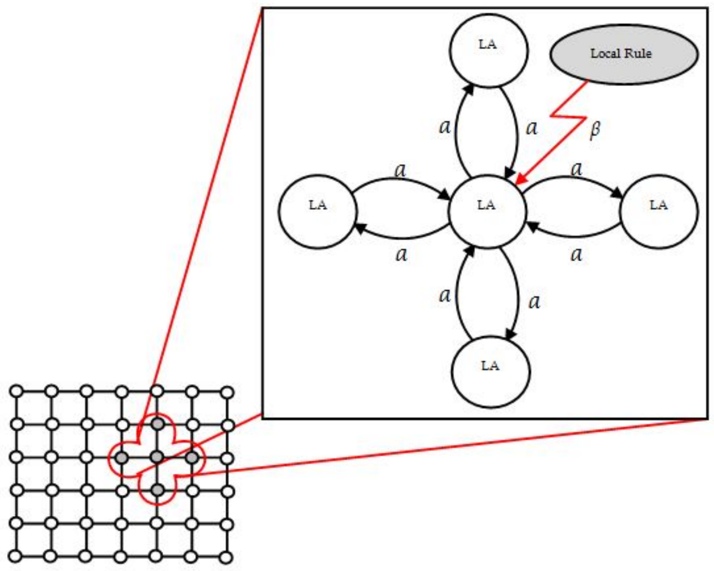

Figure 10. Cellular Learning Automata

A number of applications for Cellular Learning Automata have been developed recently such as rumor diffusion,image processing, modeling of commerce networks, fixed channel assignment in cellular networks, and VLSI Placement to mention a few (Beigy & Meybodi, 2004).

The Cellular Learning Automata can be classified into two types of *synchronous* and *asynchronous*. In synchronous Cellular Learning Automata, all cells are synchronized with a global clock and executed at the same time [10]. It is shown that the synchronous Cellular Learning Automata converges to a globally stable state for a class of rules called commutative rules. In some applications such as image processing, a type of Cellular Learning Automata in which the action of each cell in next stage of its evolution not only depends on the local environment (actions of its neighbors) but it also depends on the external environments.

3.3. Irregular Cellular Learning Automata

Irregular Cellular Learning Automata is a generalization of Cellular Learning Automata which removes the restriction of rectangular grid structure in traditional Cellular Learning Automata. This generalization is expected because there are applications which cannot be adequately modeled with rectangular grids [10].

In an informal way, Irregular Cellular Automata is a configuration of points in the space with no prior restriction. The few examples of Irregular Cellular Automata all use Voronoi polygons or the related Delaunay triangulation to divide space and determine the neighbors of each point. 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.

An Irregular Cellular Learning Automata is a combination of Irregular Cellular Automata and Learning Automata (Figure 11). We define Irregular Cellular Learning Automata as an undirected graph in which each vertex represents a cell which is equipped with a learning automaton.

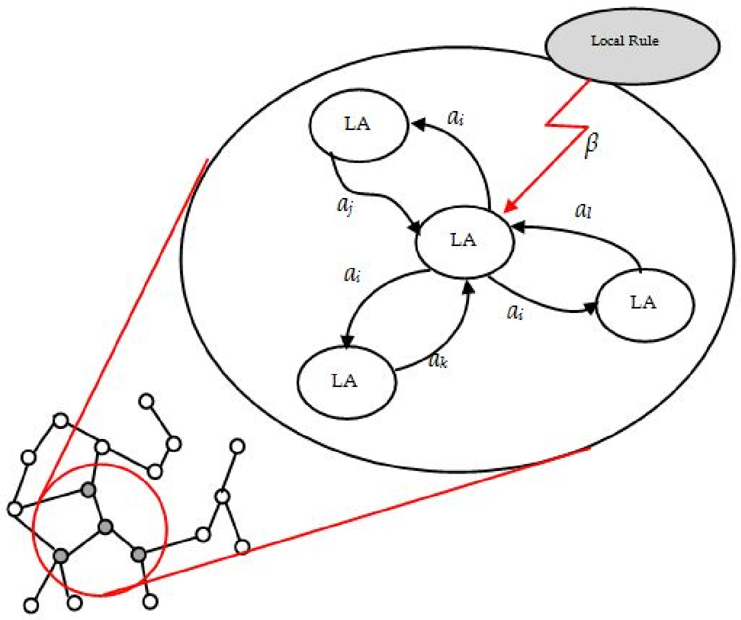

Figure 11. Irregular Cellular Learning Automata, LA means Learning Automata in each neighbor cell.

The Learning Automaton residing in a particular cell determines its state (action) on the basis of its action probability vector. Like Cellular Learning Automata, there is a rule that the Irregular Cellular Learning Automata operate according to it. The rule of the Cellular Learning Automata and the actions selected by the neighboring Learning Automatons of any particular Learning Automata determine the reinforcement signal to the Learning Automata residing in a cell. The neighboring Learning Automatons of any particular Learning Automata constitute the local environment of that cell. The local environment of a cell is non-stationary because the action probability vectors of the neighboring Learning Automatons vary during evolution of the Irregular Cellular Learning Automata.

An Irregular Cellular Learning Automata is formally defined below.

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 Learning Automata each of which is assigned to one cell of the Irregular Cellular Learning Automata.
- $F : \underline{\Phi}_j \to \underline{\beta}$ 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 Learning Automata based on the actions selected by the neighboring Learning Automata.

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

In what follows, we consider Irregular Cellular Learning Automata with n cells. The learning automaton A_i which has a finite action set α_i is associated to cell i (for $i=1, 2, \ldots, n$) of the Irregular Cellular Learning Automata. Let the cardinality of α_i be m_i. The state of the Irregular Cellular Learning Automata represented by $p = (p_1, p_2, \ldots, p_n)$, where $p_i = (p_{i1}, p_{i2}, \ldots, p_{imi})$ is the action probability vector of A_i. The operation of the Irregular Cellular Learning Automata takes place as the following iterations. At iteration k, each learning automaton chooses an action. Let $\alpha_i \in \alpha$ be the action chosen by A_i. Then all learning automata receive a reinforcement signal. Let $\beta_i \in \beta$ be the reinforcement signal received by A_i. This reinforcement signal is produced by the application of local rule $F(\Phi_i) \to \beta$. Finally, each Learning Automata 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.

There are some applications that apply Irregular Cellular Learning Automata such as Image Processing, Graph Coloring, Social Modeling, Clustering and Sensor network applications like Channel Assignment and Routing. In the following, we introduce a sensor network application.

4. Intrusion Detection in Wireless Sensor Network Using Irregular Cellular Learning Automata

4.1. Wireless Sensor Networks

A Wireless Sensor Network contains hundreds or thousands of sensor nodes. Basically, each sensor node comprises sensing, processing, transmission, mobilizer, position finding system, and power units (some of these components are optional like the mobilizer). By definition the nature of ad hoc networks is dynamically changing. Hence security is hard to achieve due to the dynamic nature of nodes. Routing protocols for WSNs are designed based on the assumption that all participating nodes are completely cooperative. In a closed MANET, all mobile nodes cooperate with each other towards a common destiny, such as emergency search/rescue or military and law enforcement operations. In an open MANET, different mobile nodes with different goals share their resources in order to ensure global connectivity [9, 15]. Lately, significant research efforts have focused on improving the security of ad hoc networks. In WSNs, nodes are both routers and terminals. Due to the lack of a routing infrastructure all the nodes have to cooperate to ensure successful communication. Clearly, cooperation means ensuring correct routing establishment mechanisms, the protection of routing information and the security of packet forwarding. One major challenge that was neglected previously is that of making wireless sensor network robust against MAC layer misbehaviors. Significant applications of WSNs include establishing survivable, efficient, dynamic communication for emergency/rescue operations, disaster relief. Security is a critical problem when implementing WSN. The fast detection of malicious nodes is vital in mobile ad hoc networks, since they rely on the cooperation of nodes for routing and forwarding. Also, cooperation of misbehaving nodes can seriously degrade the performance and jeopardize the functionality of network.

4.2. Intrusion Detection Protocols

The security difference between wired infrastructure networks and wireless sensor networks motivated researchers to model an intrusion detection system that can handle the new security challenges such as securing routing protocols [21]. We only list here some of the existent research work that is related to our approach.

Sterne et al. proposed a dynamic intrusion detection hierarchy that is potentially scalable to large networks with using clustering [24]. This method is similar with Kachirski and Guha, but it can be structured in more than two levels. Thus, nodes on first level are cluster-heads and nodes on the second level are leaf nodes. In this model, every node has the task of monitoring, logging, analyzing, properly responding to intrusions detection if there is enough evidence, and alert or report to cluster-heads. The cluster-heads, in addition, must also perform:

1. data fusion/integration and data filtering,
2. computations of intrusion, and
3. security management.

Sumalatha and Reddy proposed an approach for misbehavior detection [25]. Detection system is implemented based on fuzzy logic concept and the DSR has been used as routing protocol. Every node implements an instance of the detection system and runs it in two phases. In the initial phase, the detecting system learns about the normal behavior of nodes with respect to the DSR protocol. Then, the node may leave the protected environment and enters the second phase where node finds some of the nodes as malicious and captures each node parameters such as number of route requests, number of route replies and number of updates at each node in the network. These parameters are used for input of the fuzzy inference system and also are fuzzified at the beginning in order to make fuzzy values. To find the crisp value of the calculated trust, trust is assigned to each node in the ad hoc network. This process mainly contains fuzzification, inference by rule base construction and defuzzification processes. The Defuzzification is the process of conversion of fuzzy output set into a single number. In this approach, the authors have used these numbers to detect the malicious nodes in the network. In one of recent works, the authors suggested learning automata-based protocol for intrusion detection (LAID) in wireless sensor networks [27]. LAID functions in a distributed manner and uses the learning automata to optimize the selection of paths in which sampling has to be performed. The system, in essence, tries to identify or approximate the location of the attacker and, thus, it catches the malicious packets sent by the attacker. LAID protocol is not energy-aware and it may not be always practically ideal for resource-constrained networks such as distributed WSN.

Further, another learning automata-based intrusion detection protocol (S-LAID) has been proposed [28]. S-LAID functions in a distributed manner with each node functioning independently without any knowledge about the adjacent nodes. S-LAID assumes that the system budget is configured prior to its installation. In this protocol, the authors considered that sampling of a packet consumes energy. In S-LAID, each node continuously samples its interface at a minimum sampling budget. According to S-LAID algorithm if malicious packets are found and the detection rate is higher than the penalty threshold, then the sampling rate is increased. The learning functions calculate the sampling rate that should be used during the next instant by the automaton. In order to maintain efficiency and increase lifetime, the authors have bound the value by the sampling rate. They also have used the rate control algorithm to moderate the increase in the sampling rate.

4.3. Irregular Cellular Learning Automata-based Intrusion Detection Protocols

In this protocol, the entire network is divided into multiple clusters. Nodes are placed into clusters with one cluster-head for each cluster (Figure 12). Each cluster-head node is aware of its cluster information. The authenticity of a node is mostly determined by the nodes that are in same cluster. Each node has an IDS agent for detecting potential abnormalities in packets forwarding process. To reduce the overhead of intrusion detection process, nodes in a cluster will cooperate to select a cluster-head node based on learning automata residing in each node for handling the detection process for the whole cluster. Data packets may traverse between different clusters. The process of misbehaving nodes detection is performed in 3 sequential phases.

- Phase 1: Detection of misbehaving nodes by cluster-head node in same cluster.
- Phase 2: Confirmation of misbehaving nodes by neighbor nodes.
- Phase 3: Reward or penalize misbehaving node by neighbor nodes.

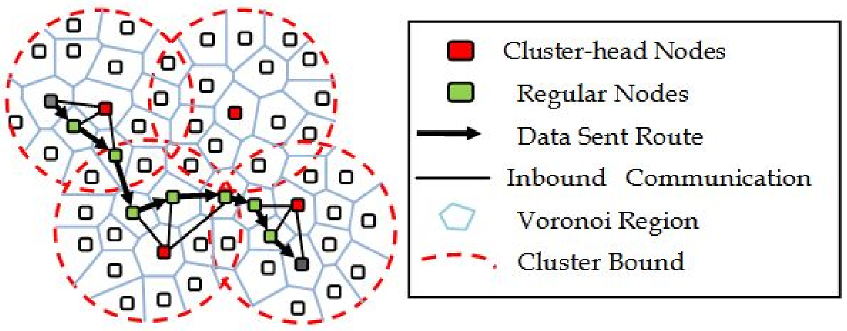

Figure 12. Network Model

To configure the routing in network, each node constructs its probability vector *{p $_1$, p $_2$,…, p $_n$ }*. Each node sends its *Id* and energy level to its cluster-head and neighbor nodes to form the clusters. Neighbor nodes construct their local routing tables upon receiving this packet. For each received packet, an entry for the node *Id* in the packet is created in routing table, and initial preference for that node is calculated as follows:

$$p(i) = \frac{EnergyLevel_i}{\sum_{j=1}^{m} EnergyLevel_j} \qquad i \neq j \quad and \quad i = 1,2,3,\ldots \tag{10}$$

Where p_i is the probability of selecting the i_{th} neighbor node, $EnergyLevel_i$ is the energy level of the i_{th} neighbor node and m is the total number of neighbor nodes. Indeed each node in the network gets the preference of all nodes that are in the same clusters and sends this preference to its cluster-heads.

1. *Phase 1*

Systematically, in our protocol, we attach an IDS agent to each mobile node. These IDS agents run independently and monitor local activities to detect abnormal behaviors. We assume the local IDS agent is tamper resistant. Several software tamper resistance techniques have been proposed that are very hard to crack and suitable for our approach. In this method, we have considered two level architecture for each node. The first layer is the internal IDS agent. IDS agent can be divided into the following components: the data collection module (DCM), the data transmission quality (DTQ) module, the cluster aggregation and fusion module (CAFM), and the intrusion response module. A diagram is given in Figure 13.

The second layer is the ICLA. This layer is a combination of the detection engine module and learning automata residing in each node.

Data Collection Module (DCM)

The functionality of the data collection module is to collect security related data via monitoring local activities and local behaviors of neighbor nodes. We define misbehaving nodes as those that have aberrations in data exchange patterns. We have used the bucket as a specific count of packets that are transmitted from one node to the other. At the end of every bucket, Data collection modules send the gathered information and statistics to CAFM. This information determines the behavior of the node and its neighbors that are sending and receiving data packets.

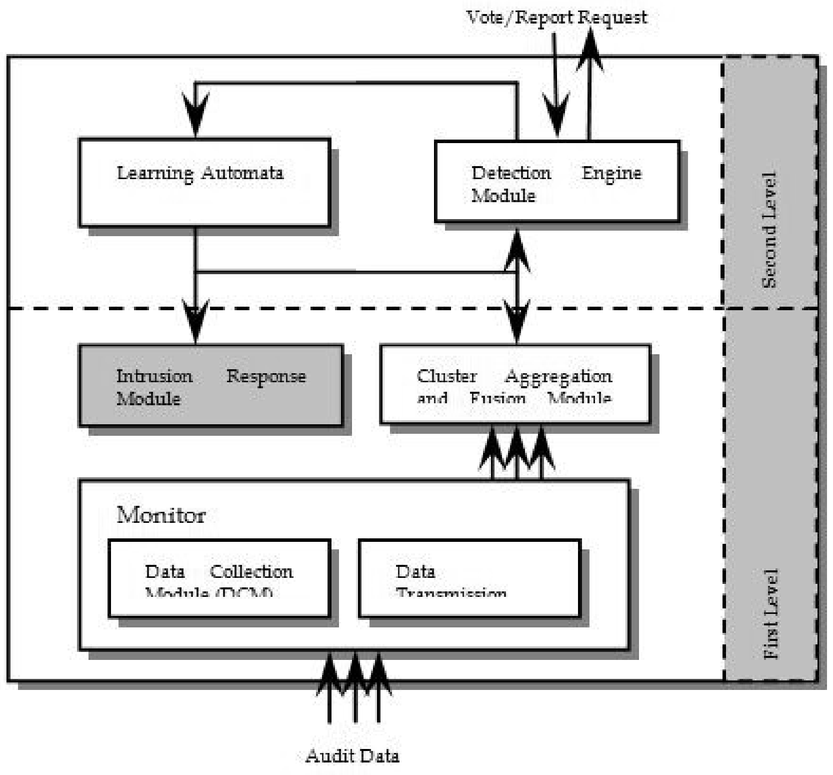

Figure 13. Internal Model for the IDS Agent

Data Transmission Quality (DTQ Function)

This module has a function to measure the quality of a communication node. In our Method, the DTQ function measures changes in the environment and sends a probability to a higher layer. This probability is calculated as follow:

$$p(i) = \lambda_1 \times \frac{STB_i(\ )}{Er(\ )} + \lambda_2 \frac{EnergyLevel_i}{IEnergy - N \times SEnergy} \qquad (11)$$
$$i = 1, 2, 3, \ldots$$

In this function, *EnergyLevel* $_i$ is the level of the energy of the i $_{th}$ neighbor node. *IEnergy* is the initial energy in each node, and *SEnergy* is the required energy to transmit data. *N* is the number of Data packets or the bucket size. *Er()* is probability of error in the channel. λ_1 and λ_2 declare the effect of nodal behavior and node's energy respectively. Also *STB* $_i$ *()* is the stability of the nodal behavior. This quantity is measured as follows:

$$STB_i = \frac{\sum_i numberoffcrwardedpackets}{\sum_i numberoffcrwardedpackets + \sum_i numberoffrecievedpackets} \qquad (12)$$

Every node measures the number of received acknowledgments from the neighbor nodes it has tried to transmit to. This statistic is *STB()*.

2. *Phase 2*

Cluster Aggregation and Fusion Module (CAFM)

If a node is inter-cluster node or normal node, it sends the gathered data from the neighboring nodes to detection engine on second level. It can send the alarms and reports to its cluster-head based on voting request. While if the node is a cluster-head node, then the CAFM module receives the alarms and reports from inter-cluster nodes. Also, CAFM module of the cluster-head node allows the voting or prevents it by aggregation and fusion of the received alarms and reports. When the CAFM module of each cluster-head node receives the vote request packet, it votes for the suspect node. Voting process is performed base on the results that are calculated by the detection engine of inter-cluster nodes. At the end of the voting process, CAFM module of the cluster-head node sends the number of votes (V $_m$) to its detection engine.

Detection Engine Module

The detection engine identifies the misbehaving nodes according to the received information from CAFM module. The detection engine set a threshold (τ) according to equation (13). This threshold is determined based on the behavior and energy level of all nodes that are participating in voting process. In this equation, we use STB and energy level of each node because these values show the quality of node behavior properly.

$$\tau = \gamma_1 \times \frac{STB_i(\ )}{\sum_{j-1}^{M} STB_j(\ )} + \gamma_2 \times \frac{EnergyLevel_i}{\sum_{j-1}^{M} EnergyLevel_j} \tag{13}$$
$$i = 1, 2, \ldots$$
$$i \neq j$$

In equation (6), $STB_i\ ()$ is the stability of the nodal behavior which can be calculated by equation (6). $EnergyLevel_i$ is the level of the energy of the i_{th} neighbor node. M is the total number of nodes that participate in the voting. γ_1 and γ_2 are numbers between zero and one. According to the information of CAFM module, if the detection engine finds one or more values of *STB* in the table that are less than the threshold (τ), then it realizes that there may be one or more misbehaving nodes in its cluster. So it sends a vote request message about the suspect nodes to the CAFM module. In addition, the detection engine module makes a decision in cooperating with ICLA based on the number of vote response messages gathered by CAFM module. According to the results of voting, the node *M* is a well-behaving one and should be rewarded or it is a misbehaving one and should be punished.

3. *Phase 3*

CAFM module of the cluster-head will gather all the vote responses about suspect node *M*. CAFM module then sends the number of gathered vote response messages (V_m) to the detection engine module. According to the number of voting for the suspect node's authenticity, a decision is made as follow:

- If more than 80 percent of the participating nodes in the voting give a positive vote to suspect node *M*, then this node will be exclude from participating in the routing. Moreover, neighbor nodes of *M* add node *M* to their black lists.
- If less than 80 percent and more than 50 percent of the participating nodes in the voting give a positive vote to suspect node *M*, then this action will be penalized by learning automata residing in the node *N* with *b=0.2* according to $L_{R\text{-}P}$ learning algorithm.
- If less than 50 percent and more than 30 percent of the participating nodes in the voting give a positive vote to suspect node *M*, then this action will be penalized by learning automata residing in the node *N* with *b=0.4* according to $L_{R\text{-}P}$ learning algorithm.
- If less than 30 percent of the participating nodes in the voting give a positive vote to suspect node *M*, then this action will be rewarded by learning automata residing in the node *N* with *a=0.6* according to $L_{R\text{-}P}$ learning algorithm.

Intrusion Response Module

The Intrusion Response Module efficiently penalizes misbehaving nodes based on updated statistics which are created and sent to intrusion response module by learning automata. The intrusion response module performs the following actions according to received statistics. First, it receives the updated *STB* values (equation 5) from the second layer and saves them in *STB* table. Second, if the specific node is a cluster-head node, the intrusion response module sends the order of penalization or dismissal of the suspected node to all cluster nodes. Therefore, misbehaving nodes won't be permitted to participate in routing process.

4.4. Evaluation the Proposed Protocol

In this section, we have implemented the proposed protocol by MATLAB and Glomosim simulator, a scalable discrete event simulator developed by UCLA.

Simulation settings

The network area size is 2000*2000 (in m2). The mobility model is the random waypoint model. The minimum speed is 5 m/s, and the maximum speed is 15 m/s. We have used the IEEE 802.11 for distributed wireless sensor networks as the MAC layer protocol. The number of nodes varies from 1000 to 3000 nodes. Radio bandwidth is 250000(in bps). Initial energy level of each node is 5(mW) and radio transmit power is 10 (in dBm). The size of all data packets is set to 512 bytes. The duration of each simulation is 1800 seconds. The values of γ_1 and γ_2 are considered 0.5 in our simulations.

Simulated Attacks

In our simulation, we have implemented and used the following attacks:

- Black-hole attack: In this attack, a misbehaving node uses the routing protocol to advertise itself as having the shortest path to the node whose packets it wants to intercept. The attacker will then receive the traffic which is destined for other nodes, and then it can drop or modify the packets.
- Denial of Service: A node prevents itself from receiving and forwarding data packets to their destinations.
- Malicious Flooding: In this attack, the misbehaving node pumps a great deal of useless and garbage packets to the network. In this way, it corrodes the resources of the network such as bandwidth and energy.
- Packet dropping: A node conditionally or randomly drops data packets which are supposed to be forward.

Simulation Results

The results of simulation in figure 14 show the percentage of detection rate with variation of misbehaving nodes' percentage. In this simulation, the number of nodes is 100 and the number of clusters is 10. Obviously, at first, the percentage of detection rate in all attacks has decreased, and afterwards with increase of misbehaving nodes' percentage the detection rate has increased either. The important reason for this behavior is the application of ICLA. In voting process, gathered information of neighbor nodes is increased which have participated in voting. In fact, the system learns abnormal behavior by increase of gathered information of learning automata from its environment. Therefore, the misbehaving nodes will be detected accurately. Because of using the energy and behavior factors for detecting the malicious nodes in black-hole attack, the results for black-hole attack are detected accurately. Consequently these results are better than that of other attacks with higher population of misbehaving nodes.

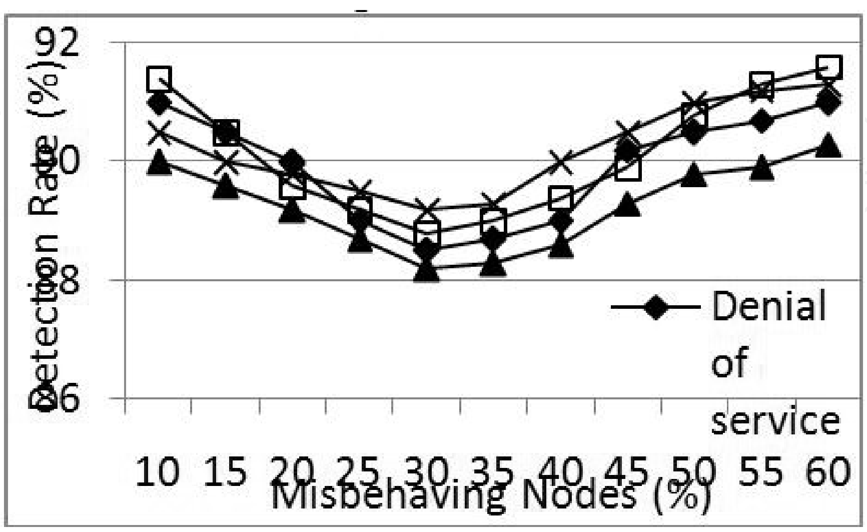

Figure 14. Detection rate vs. percentage of misbehaving nodes

Figure 15 shows the false positive rate with variation in percentage of the misbehaving nodes. In this simulation, the number of nodes is 100 and the number of clusters is 10. In this figure, first, the percentage of false positive rate has increased and then the false positive rate has decreased with increasing the percentage of misbehaving nodes. The learning automaton in each node gathers the information from the environment and this causes detection of misbehaving nodes to be performed properly. So the percentage of false positive rate has decreased after obvious quantity of 40%.

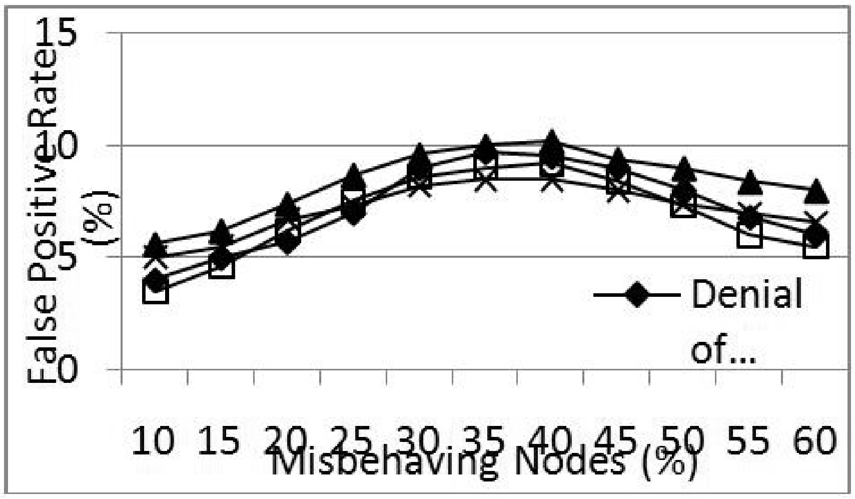

Figure 15. False positive rate vs. percentage of misbehaving nodes

In figure 16, we have shown the results of simulation and have discussed the detection rate with variation in number of clusters. In this simulation, detection rate will be decreased with increasing the number of clusters. Because of the constant total number of nodes in the network and the increase of clusters' number, the number of nodes in each cluster will be decreased. Therefore, the action probability vector in each cluster-head will be decreased, and this causes the detection rate to decrease in each cluster. As it is shown in this figure, after a specific number of clusters (10), the learning rate has increased, and, thus the detection rate has increased too. In this state, ICLA performs very well.

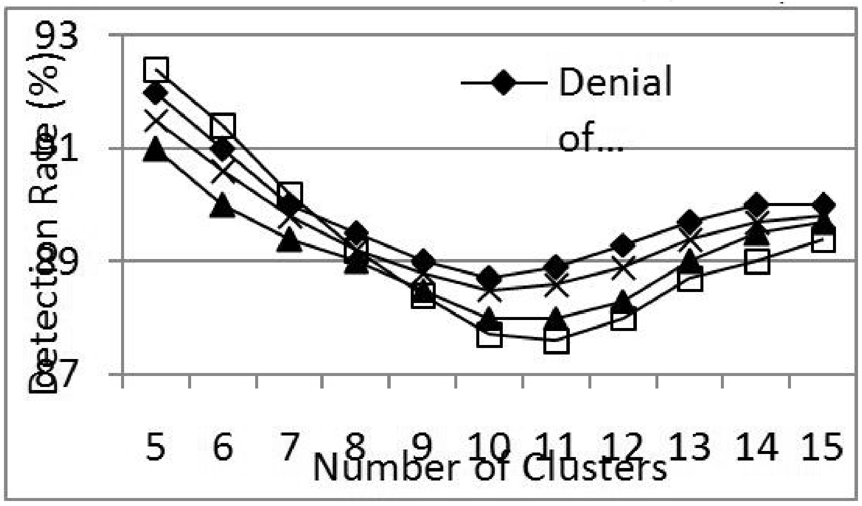

Figure 16. Detection rate vs. number of cluster

In figure 17, the simulation results of false positive rate in variation with number of clusters are illustrated. At first, false positive rate increases with increase of clusters' number, because as number of clusters increases, number of nodes inside each clusters decreases. Therefore, the length of action probability vector in cluster-heads' learning automata decreases, but after specific number of clusters (10) the false positive rate decreases due to increasing of learning rate increases. This decrease for denial of service attack has been more noticeable than the other attacks because of ICLA application in detecting attacks and nodal behavior.

In the next simulation, we have evaluated the effects of mentioned attacks on energy consumption of network in our method. Figure 18 shows the average energy consumption with variation in number of nodes. For all attacks, energy consumption has increased with increase of nodes' number. Moreover, average energy consumption for malicious forwarding attack is lower than black-hole attack, and the average energy consumption of black-hole attack is lower than other attacks. These results were predictable, because the proposed method uses the energy level of each node for detecting malicious flooding and black-hole attacks. In addition, the proposed protocol uses each node's behavior for black-hole attack and this causes the energy consumption to increase.

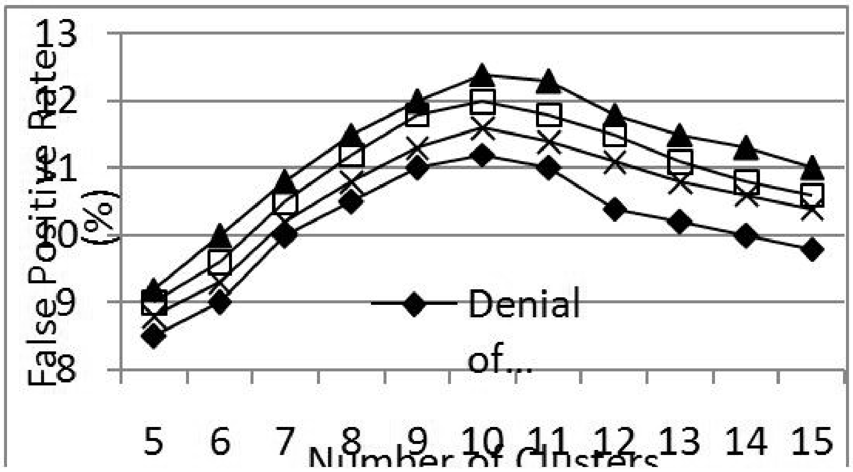

Figure 17. False positive rate vs. number of clusters

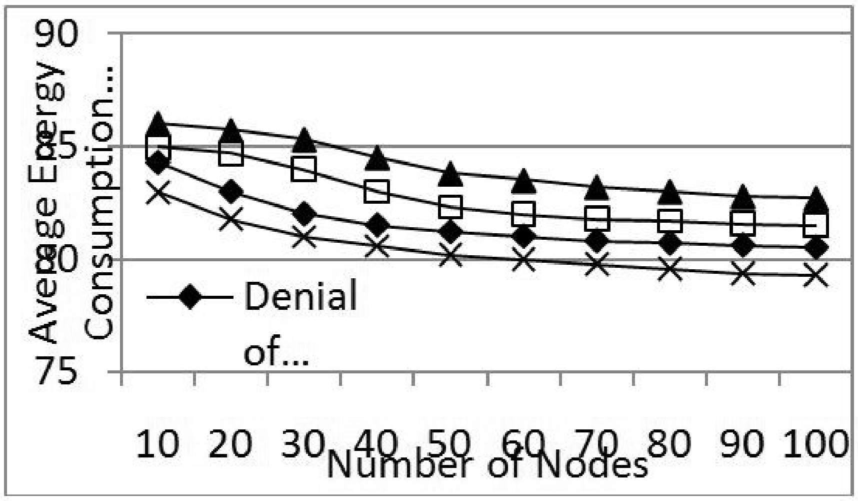

Figure 18. Average energy consumption in the network vs. number of nodes

5. Conclusion

In this Chapter, we have disscussed Cellular Learning Automata which has standard assumption of a rectangular grid structure. Then we have represented Irregular Cellular Learning Automata which are the models cause to develop tools that allow us use both rectangular and even irregular grids within one and the same Cellular Automata modeling framework. Of course, we are aware that the regularity of the grid structure is but one of a number of idealizations used in Cellular Automata modeling. For example, it has been discussed controversially whether and under what conditions a simple influence dynamics

http://dx.doi.org/10.5772/52953

similar to our model of spatial collective action can be robust with respect to variation in simultaneous vs.

We applied these tools to intrusion detection protocols which is an application from sensor networks. This is a novel approach which used of Irregular Cellular Learning Automata to detect suspect nodes by using and analyzing nodes' behavior during routing process and nodes' energy level. It also implements Irregular Cellular Learning Automata to detect abnormal behaviors. Afterwards, our method starts its voting process in which it decides to reward or penalize suspect node based on learning automata reports. The simulations show that our proposed method not only has a proper detection rate but also is an energy-aware protocol in detecting malicious nodes.

Author details

Amir Hosein Fathy Navid[1*] and Amir Bagheri Aghababa[2]

*Address all correspondence to: Amir.Fathy.n@qiau.ac.ir

1 Islamic Azad University, Hamedan Beranch, Bahar, Hamedan, Iran

2 Islamic Azad University East Tehran Branch, Tehran, Iran

References

[1] Abolhasani, S. M., Meybodi, M. R., & Esnaashari, M. (2007). LABER: A Learning Automata Based Energy-aware Routing Protocol for Sensor Networks. *IKT conference, Tehran, Iran.*

[2] Al-Karaki, J. N., & Kamal, A. E. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 6-28, 11.

[3] Ankit, M., Arpit, M., Deepak, T. J., Venkateswarlu, R., & Janakiram, D. (2006). TinyLAP: A Scalable learning automata-based energy aware routing protocol for sensor networks. *Communicated to IEEE Wireless and Communications and Networking Conference, Las Vegas, NV USA*.

[4] Beigy, H., & Meybodi, M. R. (2004). A mathematical framework for cellular learning automata. Advances on Complex Systems Nos. 3-4, September/December, New Jersey,., 7, 295-320.

[5] Carvalho, J. P., Carola, M., & Tomé, A. B. (2002). Using Rule-Based Fuzzy Cognitive Maps to Model Dynamic Cell Behavior in Voronoi Based Cellular Automata. *FCT-Portuguese Foundation for Science and Technology, Lisboa, Portugal*.

[6] Chang, J. H., & Tassiulas, L. (2004). Maximum lifetime routing in wireless sensor networks. IEEE/ACM Trans. on Networking August., 12(4), 609-619.

[7] Esnaashari, M., & Meybodi, M. R. (2007). Irregular Cellular Learning Automata and Its Application to Clustering in Sensor Networks. Proceedings of 15th Conference on Electrical Engineering (15th ICEE), Tehran, Iran May., 15-17.

[8] Fathy Navid, A. H. (2010). SELARP: Scalable and Energy-aware Learning Automata-based Routing Protocols for Wireless Sensor Networks. *Proceedings of SENSOR-COMM2010, IEEE, Venis, Italy.*

[9] Fathy, Navid. A. H., & Seyyed, Javadi. S. H. (2009). Energy Aware Routing Protocol for WSN Using Irregular Cellular Learning Automata. IEEE Symposium on Industrial Electronics and Applications (ISIEA2009), Kuala Lumpur, Malaysia, October , 4-6.

[10] Flache, A., & Hegselmann, R. (2002). Do Irregular Grids make a Difference? Relaxing the Spatial Regularity Assumption in Cellular Models of Social Dynamics. JASSS, Journal of Artificial Societies and Social Simulation of 26), 11/4/2002., 4(4)

[11] James, C., & Kingsbery Jr, . (2006). Excluded Blocks in Cellular Automata. *WILLIAMS COLLEGE Williamstown, Massachusetts*.

[12] Klein, R., & Aurenhammer, F. (2001). Voronoi diagrams. D. Forschungsgemein schaft Editor.

[13] Papadimitriou, I., & Georgiadis, L. (2005). Energy-aware routing to maximize lifetime in wireless sensor networks with mobile sink. 13th International Conference on Software, Telecommunications and Computer Networks, SoftCOM2005 September., 120-126.

[14] Schiff, J.L. (2005). Introduction to cellular automata. http://psoup.math.wise.edu/491October2008.

[15] Shah, R., & Rabaey, J. (2002). Energy aware routing for low energy ad hoc sensor networks. Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC) Orlando, FL, March., 143-150.

[16] Souvaine, D., Horn, M., & Weber, J. (2005). Voronoi diagrams, Computational Geometry. *Tufts University Editor, Spring.*

[17] Narendra, K. S., & Thathachar, M. A. L. (1989). Learning automata: an introduction. *Prentice Hall*.

[18] Thathachar, M. A. L., & Sastry, P. S. (2002). Varieties of learning automata: an overview. *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(6).

[19] Zeng, X., Bagrodia, R., & Gerla, M. (1998). GloMoSim: a library for parallel simulation of large-scale wireless networks. *in: PADS*.

[20] Esnaashari, M., Meybodi, M. R., & Sabaei, M. (2007). A novel method for QoS support in sensor networks. CSICC2007, Tehran, I. R. Iran,., 740-747.

[21] Mitrokotsa, A., Mavropodi, R., & Douligeris, C. (2006). Intrusion Detection of Packet Dropping Attacks in Mobile Ad Hoc Networks. *Proceedings of the International Conference on Intelligent Systems And Computing: Theory And Applications, Ayia Napa, Cyprus*, 111-118.

[22] Otrok, H., Debbabi, M., Assi, C., & Bhattacharya, P. (2007). A Cooperative Approach for Analyzing Intrusions in Mobile Ad hoc Networks. *Proceedings of the 27th International Conference on Distributed Computing Systems Workshops, (ICDCSW2007)*.

[23] Mishra, A., Nadkarni, K., & Patcha, A. (2004). Intrusion Detection in Wireless Ad Hoc Networks. *IEEE Wireless Communications, IEEE press.*, 48-60.

[24] Sterne, D., Balasubramanyam, P., et al. (2005). A General Cooperative Intrusion Detection Architecture for MANETs. *Proceedings of the 3rd IEEE International Workshop on Information Assurance (IWIA2005)*, 57-70.

[25] Sumalatha, V., & Reddy, P. C. (2009). A Novel Approach for Misbehavior Detection in Ad hoc Networks. International Journal of Cryptography and Security January, 17-24., 2(1)

[26] Kachirski, O., & Guha, R. (2002). Intrusion Detection Using Mobile agents in wireless Ad hoc Networks. *Proceedings of the IEEE workshop on Knowledge Media Networking*, 153-158.

[27] Misra, S., Krishna, P. V., & Abraham, K. I. (2011). A simple learning automata-based solution for intrusion detection in wireless sensor networks. *Wireless Communication and Mobile Computing* [11], 426-441.

[28] Misra, S., Abraham, K. I., et al. (2009). LAID: a learning automata-based approach for intrusion detection in wireless sensor networks. *Security and Communication Networks*, 2(2), 105-115.

[29] Kari, J. (2004). Theory of cellular automata: a survey. FIN-20014, Turku, Finland. Elsevier, 1 October.