Ad Hoc & Sensor Wireless Networks Vol. 10, pp. 193–234
Reprints available directly from the publisher
Photocopying permitted by license only

Dynamic Point Coverage Problem in Wireless Sensor Networks: A Cellular Learning Automata Approach

MEHDI ESNAASHARI AND MOHAMMAD REZA MEYBODI

Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran, P. O. Box: 15875-4413
E-mail: Esnaashari@aut.ac.ir, mmeybodi@aut.ac.ir

Received: March 18, 2009. Accepted: September 14, 2009.

One way to prolong the lifetime of a wireless sensor network is to schedule the active times of sensor nodes, so that a node is active only when it is really needed. In the dynamic point coverage problem, which is to detect some moving target points in the area of the sensor network, a node is needed to be active only when a target point is in its sensing region. A node can be aware of such times using a predicting mechanism. In this paper, we propose a solution to the problem of dynamic point coverage using irregular cellular learning automata. In this method, learning automaton residing in each cell in cooperation with the learning automata residing in its neighboring cells predicts the existence of any target point in the vicinity of its corresponding node in the network. This prediction is then used to schedule the active times of that node. In order to show the performance of the proposed method, computer experimentations have been conducted. The results show that the proposed method outperforms the existing methods such as LEACH, GAF, PEAS and PW in terms of energy consumption.

Keywords: Wireless Sensor Network, dynamic point coverage, scheduling, learning automata, cellular learning automata.

1 INTRODUCTION

One important problem addressed in literature is the sensor coverage problem. This problem is centered on a fundamental question: "How well do the sensors observe physical space?" The coverage concept is a measure of the quality of service (QoS) of the sensing function and is subject to a wide range of interpretations due to a large variety of sensors and applications [88]. Various coverage formulations have been proposed in literature among which following three are most discussed [1]:

- Area coverage: Covering (monitoring) the whole area of the network is the main objective of area coverage problem.
- Point coverage: The objective of point coverage problem is to cover a set of stationary or moving target points using as little sensor nodes as possible.
- Barrier coverage: Barrier coverage can be considered as the coverage with the goal of minimizing the probability of undetected penetration through the barrier (sensor network).

In this paper, we focus on the problem of dynamic point coverage which is the point coverage problem with non-stationary target points. Any solution to this problem must select a subset of sensor nodes in the network to be active and monitor the target points. This can be done through a fixed or a dynamic scheduling mechanism. In fixed scheduling mechanisms [2, 3, 5, 39–43, 47–57], the set of sensors in the network is divided into disjoint sets so that every set completely covers the entire area of the network. These disjoint sets are activated successively, so at any moment in time only one set is active. Because all target points are monitored by every sensor set, the goal of this approach is to determine a maximum number of disjoint sets so that the time interval between two activations for any given sensor is longer. Besides its complexity, this approach needs the information of the network topology to be available in a central node which is not always possible. In dynamic scheduling mechanisms, nodes are locally scheduled to be active or inactive based on the movement paths of target points [6–10, 11, 59–69]. In such schemes, usually some of the nodes which have higher residual energy than other nodes are active all of the times and monitoring the whole area for detecting target points. Whenever a target point is detected, these active nodes track the target and estimate its movement path. This estimation leads to a prediction about the location of the target point in near future. Sleeping nodes in the vicinity of the predicted location are then activated by currently active nodes. This activation is performed using some sort of notification messages. This dynamic scheduling approach has two major drawbacks; one is the overhead of the notification messages and the other is that sleeping nodes should have the ability to receive messages, and hence they cannot power off their receiving antenna. This means that a sleeping node can only switch off its processing unit, but its communicating unit must be in idle state, waiting for notification messages. According to [12], energy consumption of a sensor node in receiving and idle states is nearly equal to the energy consumed during the transmission state. As the energy consumed by a processing unit is in the order of .001 of the energy consumed by a communicating unit, it is concluded that using these dynamic scheduling mechanisms, not so much energy saving can be gained in sleeping nodes.

To overcome the above drawbacks, in this paper a novel dynamic scheduling algorithm based on irregular cellular learning automata is proposed to deal with the problem of dynamic point coverage. In the proposed algorithm, the sensor network is mapped into an irregular cellular learning automata. In this mapping, each sensor node in the network is mapped into a cell in the ICLA. Each cell is equipped with a learning automaton. The learning automaton residing in each cell in cooperation with the learning automata residing in neighboring cells dynamically learns (predicts) the existence of any target points in the vicinity of its corresponding node in the network in near future. This prediction is then used to schedule the active times of that node. Instead of notification messages which are exchanged between neighboring nodes in the dynamic scheduling schemes, in the proposed method, a local base station in each neighborhood is always active and is responsible for queuing and relaying control packets between neighbor nodes during their active times. As a consequence, sleeping nodes in the proposed method can switch off their communicating units as well as their processing units, saving more energy and hence prolonging the network lifetime. Experimental results show that the proposed method outperforms the existing methods such as LEACH[1] [19], GAF[2] [29], PEAS[3] [55, 89] and PW[4] [7] in terms of energy consumption.

The rest of this paper is organized as follows. Section 2, gives a brief literature overview. The problem statement is given in Section 3. Learning automata, cellular learning automata and irregular cellular learning automata will be discussed in Section 4. In Section 5 the proposed method is presented. Simulation results are given in Section 6. Section 7 is the conclusion.

2 RELATED WORK

We classify the scheduling algorithms into 3 categories; MAC layer algorithms, routing layer algorithms and application layer algorithms. Application layer algorithms can be further classified into grid-based algorithms, coverage-related algorithms and tracking-based algorithms.

MAC Layer scheduling algorithms: In this category of algorithms, scheduling is performed in the MAC layer [21–28]. Usually, this is done by allocating a slot for one node per neighborhood uniquely. This collision-free slot is used by that node for transmissions to any or all of its neighbors. Thus

[1] Low Energy Adaptive Clustering Hierarchy.

[2] Geographical Adaptive Fidelity.

[3] Probing Environment and Adaptive Sensing.

[4] Proactive Wakeup based on PEAS.

two nodes cannot be assigned the same slot if one station is within the range of the others, or if two stations have common neighbors. The objective of these algorithms is to allow communication without interference, while maximizing the number of parallel transmissions.

Routing layer scheduling algorithms: This category of scheduling algorithms relates to the hierarchical routing [14–20]. In hierarchical routing algorithms, usually a number of clusters are formed in the network, each having one node as cluster head. The rest of the nodes in each cluster are called cluster members. Cluster members must send their readings to their cluster heads, therefore, cluster heads schedule the sending time of their members to prevent intra-cluster collisions. This way, a cluster member needs to be activated only at its scheduled sending times.

One of the famous hierarchical routing layer algorithms is LEACH [19] given by Heinzelman *et al*. In this algorithm, each node locally and based on a random probability decides to be a cluster head. A cluster head then advertises itself in the area of the network. Other nodes join one of the clusters advertised in the network based on the signal strength of the advertisement packets. When clusters are formed, each cluster head adapts a TDMA-based scheduling approach for data transmissions within its cluster. Each sensor node s_k in a cluster sends its reading to its cluster head at the time slots assigned to s_k by the cluster head. Each cluster head receives data packets from its members, aggregates them and sends them directly to the sink node.

Grid-based scheduling algorithms: In this category of scheduling algorithms [29–38], usually the network is partitioned into rectangular or hexagonal grids. Only one node is scheduled to be active in each grid at any time, to monitor that grid and to relay the data from other grids. Grid dimensions are calculated in such a way that a node resides within a grid can monitor the whole area of that grid and communicate with all nodes residing in its neighboring grids.

GAF [29] algorithm given by Xu *et al*. is one of the most known grid-based algorithms in the sensor network. In this algorithm, the area of the network is partitioned into a number of rectangular cells. The dimensions of each cell is selected so that each node resides within a cell can monitor the entire area of that cell and be able to communicate with any node resides in its adjacent cells. Each node s_k in a cell waits for a random duration to receive a packet from its neighbors within the same cell indicating that the sender of the packet will be active in the cell. If no such packet is received, s_k decides to become the active node of the cell and broadcasts a packet within its cell to indicate its decision. This packet also contains the active duration of the node s_k. This active duration is selected based on the energy level of s_k; more energy level results in longer active duration. When the active node for each cell is selected, rest of the nodes in that cell will go to sleep for the duration of the specified active duration. Selection of the active node for each cell is repeated when the active duration of the active node of that cell is over.

Coverage-related scheduling algorithms: This category of scheduling algorithms, deals with the problem of covering (monitoring) either the entire area of the network or some target points (stationary or moving) in the area of the network using as little sensor nodes as possible [1]. Usually, a minimum number of nodes are selected to be active and monitor the area or the target points while the rest of the nodes are inactive and save energy. The node selection is repeated periodically or based on a certain schedule to allow balance energy consumption of all nodes. A number of centralized [2–4, 39–43] and decentralized [5, 15, 42–58] methods are given in literature for addressing this problem. In centralized methods, by assuming that the sink node has the topology information of the network, usually the problem is solved optimally using a linear integer programming approach or sub-optimally using a heuristic approach. In distributed methods, each node locally checks whether it is necessary for it to be active or not. It is necessary for a node to be active only if the sensing region of the node cannot be covered completely by its neighbors.

One of the famous algorithms in this category is PEAS [55, 89] given by Ye *et al.* At the startup of the algorithm, each node s_k sleeps for an exponentially distributed duration. When this duration is over, s_k becomes active and probes for an active node in its neighborhood. If some nodes are active in the neighborhood of the node, s_k returns to sleep mode for a newly selected duration. This duration is computed based on the local information collected during the probing process. Otherwise s_k stays in active mode until its energy is depleted completely.

Tracking-based scheduling algorithms: The last category of scheduling algorithms is tracking-based algorithms. These scheduling algorithms are used to address the tracking problem in which a number of targets are moving throughout the network and the objective of the network is to monitor the movement path of these targets. Since at any instance of time, only a fraction of the nodes are able to monitor a moving target, the rest of the nodes can be inactive and save their energy. In these algorithms [6–10, 11, 59–69], usually the nodes which are currently monitoring the targets use some sort of prediction to predict the movement path of the targets. Based on this prediction, a subset of currently inactive nodes which are more probable to be able to monitor the moving targets in near future are scheduled to be activated. Activation is performed using some sort of notification messages. Note that these methods assume that inactive nodes have the ability to receive notification messages.

PW [7] given by Gui *et al.* is one of the most known tracking-based scheduling algorithms given in literature. In this algorithm, the operation of the network is divided into two phases; surveillance and tracking. In the surveillance phase, the network actively monitors the area to check if any target point enters the field. In the tracking phase, a target which has been entered the field is being tracked by the network. In the surveillance phase, an extension of the PEAS algorithm called PECAS is used. The major difference between the PECAS and PEAS is in that an active node in PECAS algorithm remains active only for a specified duration whereas in PEAS, an active node remains active until its energy is completely depleted. In the tracking phase, each sensor node has four working modes: Waiting, Prepare, SubTrack, and Tracking mode. A node is in Tracking working mode if it is within a circle centered around the location of the target with radius r (*tracking circle*). A node is in SubTrack working mode if it is outside the *tracking circle* but within a circle centered around the location of the target with radius $r + R$ (*subtrack circle*). A node is in Prepare working mode if it is outside the *subtrack circle*, but within a circle centered around the location of the target with radius $r + 2R$ (*prepare circle*). Rests of the nodes are in Waiting working mode. A node may change its working mode if it senses a target, if it cannot sense a target any more or if it receives a packet from a neighbor s_k indicating that a target exists in the sensing range of s_k.

3 PROBLEM STATEMENT

Consider a sensor network consists of N sensor nodes $s_1, s_2, \ldots, s_N$, M reporter nodes $r_1, r_2, \ldots, r_M$ and one sink within a $L \times L$ rectangular, completely known and accessible field (Ω). Sensor nodes, which are responsible for sensing and monitoring the field, are scattered randomly throughout the area of the network so that Ω is completely covered. Reporter nodes are placed manually in specified positions within Ω and form an infrastructure through which packets received from sensor nodes are relayed towards the sink. This manual placement of reporter nodes is feasible since we assume that Ω is completely known and accessible. Reporter nodes are powerful and rechargeable nodes which are always active. All sensor nodes have the same sensing range of R_s and communicating range of R_c. Each sensor node s_k has 4 different modes of operation [72] as follows:

- **On-duty ($CPU_A S_A C_A$)**: CPU, sensing and communicating units are switched on referred to as active mode.
- **Sensing Unit On-duty ($CPU_A S_A C_S$)**: The CPU and the sensing units are switched on, but the communicating unit is switched off.
- **Communicating Unit On-duty ($CPU_A S_S C_A$)**: The CPU and the communicating units are switched on, but the sensing unit is switched off.
- **Off-duty ($CPU_S S_S C_S$)**: CPU, sensing and communicating units are switched off referred to as sleep mode.

Note that in $CPU_A S_A C_A$, $CPU_A S_A C_S$, $CPU_A S_S C_A$ and $CPU_S S_S C_S$, index A stands for active and index S stands for sleep. For further simplicity in notation, we use index x in the above notations to refer to more than one operation mode, i.e. $CPU_A S_A C_x$ refers to both $CPU_A S_A C_A$ and $CPU_A S_A C_S$ modes, and $CPU_x S_x C_x$ refers to all 4 modes of operation. At any instance of time, a sensor node can be only in one of the above 4 operation modes. The operation mode of a sensor s_k at time instant t is denoted by$\mathrm{O}_{s_k}(t)$.

Let P be a finite set of target points residing in Ω. P is divided into two disjoint sets; *Moving objects* (P^M) and *Events* (P^E). A target point $p_i^M \in P^M$ is a moving object which has a continuous movement trajectory. On the other hand, a target point $p_i^E \in P^E$ is an event which occurs somewhere in Ω repeatedly or randomly following a Poisson distribution and lasts for a short static or random duration.

We denote the Euclidean distance between a sensor node s_k located at $(x(s_k), y(s_k))$ and a target point p_i located at $(x(p_k), y(p_k))$ as $d(s_k, p_i)$, i.e. $d(s_k, p_i) = \sqrt{(x(s_k) - x(p_i))^2 + (y(s_k) - y(p_i))^2}$.

Assuming the binary sensing model [13] and sensing range of R_s for all sensor nodes in the network, we say a target point $p_i \in P$ is sensed, detected or monitored by a sensor node s_k at time t if and only if $d(s_k, p_i) < R_s$ and $\mathrm{O}_{s_k}(t) = CPU_A S_A C_x$. The network detects a target point $p_i \in P$ at time t if and only if at least one of the sensor nodes of the network detects p_i at time t.

Definition 1. Network lifetime (T) is defined as the time elapsed from the network startup to the time at which the Ω is not further completely covered by the network due to node deaths.

Definition 2. Activation time of a target point $p_i \in P$ denoted by τ_{p_i} is the summation of all the times during which p_i can be detected by the network.

Definition 3. Activation time of a sensor node s_k denoted by τ_{s_k} is the summation of all the times during which the sensor is in $CPU_A S_A C_x$ operation mode; that is $\mathrm{O}_{s_k}(t) = CPU_A S_A C_x$.

Definition 4. Detection time of a target point $p_i \in P$ denoted by $\tau_{p_i}^d$ is the summation of all the times during which p_i is detected by the network.

Definition 5. Detection time of a target point $p_i \in P$ by a sensor node s_k denoted by τ_{s_k,p_i} is the summation of all the times during which p_i is detected by s_k.

Definition 6. Network detection rate denoted by η_D is the rate of the target detection in the network and is defined according to equation (1).

$$\eta_D = \frac{\sum_{p_i \in P} \tau_{p_i}^d}{\sum_{p_i \in P} \tau_{p_i}} \tag{1}$$

Definition 7. Network sleep rate denoted by η_S is defined as the ratio of the times during which nodes of the network are in sleep mode to the times during which they can be in sleep mode. η_S is defined according to equation (2).

$$\eta_S = \frac{\sum_{s_k} \left(\frac{\mathrm{T} - \tau_{s_k}}{\mathrm{T} - \sum_{p_i \in P} \tau_{s_k, p_i}} \right)}{N} \tag{2}$$

The objective of the network is to detect the target points and report their locations to the sink. We assume that the reporter nodes are placed so that each sensor node can directly communicate with at least one r_i. Sensor nodes send their packets directly to reporter nodes and reporter nodes forward received packets towards the sink.

To prolong the network lifetime, a sensor node will switch to the $CPU_A S_x C_A$ operation mode only if it wants to communicate with a reporter node; otherwise, the communicating unit of the sensor node will be switched off. The sensing unit of a sensor node has to be switched on only if a target point is in its sensing range, but since sensor nodes have no knowledge about the movement paths of target points, they cannot calculate the times for switching their sensing units on or off. As an alternative way, we assume the time to be divided into a number of very short epochs (Ep) having equal durations (τ_{Ep}). The sensing units of sensor nodes can be switched on or off only at the startup of each epoch. Therefore, at the startup of each epoch Ep_k^n, a sensor node s_k has to predict if any target points will pass through its sensing region during Ep_k^n. Based on this prediction, s_k switches its sensing unit on or off during Ep_k^n. Index k in Ep_k^n states that we assume no synchronization between sensor nodes, and hence, each sensor node has its own timing and epochs. We refer to the operation mode of a sensor node s_k during the epoch Ep_k^n as $\mathrm{O}_{s_k}(Ep_k^n)$.

Having the above definitions and assumptions, the problem is to locally predict the status of the sensing unit of each node at the startup of each epoch such that the network sleep rate (η_S) is maximized while the network detection rate (η_D) does not drop below an acceptable level.

To clarify the problem, we give an application. A trial wireless sensor network which is going to be tested in San Francisco is a sensor network that announce which of the parking spaces of the city are free at any moment [70]. This network uses a wireless sensor embedded in a 4-inch-by-4-inch piece of plastic, fastened to the pavement adjacent to each parking space. In this application, each sensor node has to monitor its parking space and reports the free times of the parking space to a local base station (a reporter node). The local base station then prepares the information of free parking spaces for drivers passing the area and requesting such information. From the viewpoint of the sensor network, cars coming into the parking spaces and getting out of them are moving target points which must be monitored. If a sensor node s_k switches periodically between $CPU_A S_A C_x$ and $CPU_x S_S C_x$ operation modes, it will have a longer lifetime than if it always remains in $CPU_A S_A C_x$ operation mode, however this makes it possible for a driver to be mistakenly guided to s_k's parking space, while it is occupied by another car. Such incorrect guidances are acceptable while their rate is below an acceptable level, and hence, a local prediction can be performed in each sensor node to predict the status of its sensing unit at the startup of each epoch.

4 CELLULAR LEARNING AUTOMATA

In this section we briefly review cellular automata, learning automata, cellular learning automata and irregular cellular learning automata.

Cellular Automata: Cellular automata (CA) is a mathematical model 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 [74]. CA performs complex computations with a high degree of efficiency and robustness. 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 state of each cell at any time instant is determined by a rule from states of neighboring cells at the previous time instant.

Learning Automata: Learning automata (LA) 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. Learning automata are classified into fixed-structure stochastic, and variable-structure stochastic. A variable-structure learning automaton is defined by the quadruple $\{\alpha, \beta, p, T\}$ in which $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ 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. After receiving the environment's reinforcement signal, automaton updates its action probability set based on equations (3) for favorable responses, and equations (4) for unfavorable ones.

$$\begin{aligned} p_i^{n+1} &= p_i^n + a.(1 - p_i^n) \\ p_j^{n+1} &= p_j^n - a.p_j^n \quad \forall j \; j \neq i \end{aligned} \tag{3}$$

$$\begin{aligned} p_i^{n+1} &= (1 - b).p_i^n \\ p_j^{n+1} &= \frac{b}{r-1} + (1 - b)p_j^n \quad \forall j \; j \neq i \end{aligned} \tag{4}$$

In these equations, a and b are reward and penalty parameters respectively. If $a = b$, learning algorithm is called L_{R-P}[5], if $b \ll a$, it is called $L_{R\varepsilon P}$[6], and if $b = 0$, it is called L_{R-I}[7]. For more information about learning automata the reader may refer to [75, 76].

Cellular Learning Automata: Cellular learning automata, which is a combination of cellular automata and learning automata, is a powerful mathematical model for many decentralized problems and phenomena. A CLA is a CA in which a learning automaton is assigned to every cell. At any instant of time, the action probability vector of the LA resides in a particular cell constitutes the state of that cell. Like CA, there is a rule that the CLA operates under. The local rule of CLA and the actions selected by the neighboring LAs of any particular LA determine the reinforcement signal to that LA. The neighboring LAs of any particular LA constitute the local environment of the cell in which that LA resides. The local environment of a cell is nonstationary because the action probability vectors of the neighboring LAs vary during the evolution of the CLA. The operation of a CLA could be described as follows: At the first step, the internal state of every cell is specified. This initial value may be chosen on the basis of past experience or at random. In the second step, each LA selects one of its actions based on its action probability vector and performs it on the environment. Next, the local rule of CLA determines the reinforcement signal to each LA. Finally, each LA updates its action probability vector on the basis of the supplied reinforcement signal and the chosen action. This process continues until the desired result is obtained. A CLA is called synchronous if all LAs are activated at the same time in parallel. A CLA is called asynchronous (ACLA) if at a given time only some LAs are activated independently from each other, rather than all together in parallel. The LAs may be activated in either time-driven or step-driven manner. In time-driven ACLA, each cell is assumed to have an internal clock which wakes up the LA associated to that cell while in step-driven ACLA; a cell is selected in fixed or random sequence. CLA has found many applications such as image processing [77–80], rumor diffusion [81], modeling of commerce networks [79], channel assignment in cellular networks [82] and VLSI placement [83], to mention a few. For more information about CLA the reader may refer to [81, 84–86].

Irregular Cellular Learning Automata: An Irregular cellular learning automata (ICLA) (Figure 1) is a cellular learning automata in which the restriction of regular grid structure is removed. This generalization is expected because there are applications such as wireless sensor networks, immune network systems, graphs, etc. that cannot be adequately modeled with regular

[5]Linear Reward-Penalty.

[6]Linear Reward epsilon Penalty.

[7]Linear Reward Inaction.

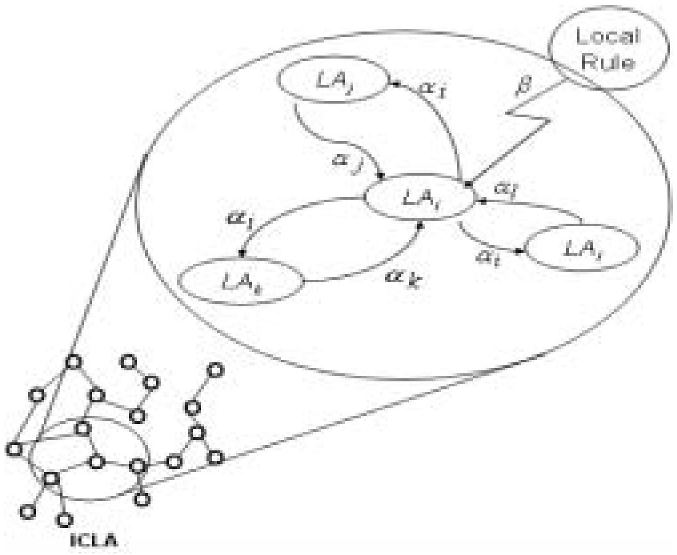

FIGURE 1
Irregular cellular learning automata.

grids. An ICLA is defined as an undirected graph in which, each vertex represents a cell which is equipped with a learning automaton. Like CLA, there is a rule that the ICLA operates under. The rule of the ICLA and the actions selected by the neighboring LAs of any particular LA determine the reinforcement signal to that LA. The neighboring LAs of any particular LA constitute the local environment of the cell in which that LA resides. The local environment of a cell is non-stationary because the action probability vectors of the neighboring LAs vary during the evolution of the ICLA. The operation of ICLA is identical to the operation of CLA. Like CLA, an ICLA can be synchronous or asynchronous and an asynchronous ICLA can be time-driven or step-driven. ICLA is recently used as a learning model in a clustering algorithm for wireless sensor networks [87].

5 THE PROPOSED METHOD

The proposed scheduling method consists of 3 major phases; Initialization, mapping and operation. During the initialization phase, each sensor node in the network finds a reporter node in its neighborhood through which it can sends its packets towards the sink. Mapping the network topology to an ICLA is done in the mapping phase. Finally, scheduling the active times of sensor nodes for different epochs is performed during the operation phase. We explain these three phases in more details in the subsequent sections.

5.1 Initialization

Each sensor node has to find a reporter node through which it can send its' packets towards the sink. We refer to the reporter node of a sensor node s_k as $r(s_k)$. During the initialization phase, all sensor nodes are in $CPU_A S_C C_A$ operation mode; that is both CPU and communicating units of all sensor nodes are switched on. Each reporter node r_i periodically broadcasts *ReporterADV* packets which contain the id and location of r_i. A reporter node r_i broadcasts *ReporterADV* packets at times $Rnd_i + \tau_{Adv}$. τ_{Adv} is the period of transmitting *ReporterADV* packets and Rnd_i is a random delay which is used to reduce the probability of collisions between neighboring reporter nodes. A sensor node s_k, upon receiving a *ReporterADV* packet from a reporter node r_i, performs one of the followings:

1. If s_k has not seen any *ReporterADV* packets before, then it sets $r(s_k)$ to r_i.
2. If $r(s_k) \neq r_i$ and the distance between r_i and s_k is less than the distance between $r(s_k)$ and s_k, then s_k sets $r(s_k)$ to r_i.
3. Otherwise s_k ignores the received packet.

Since we assume that the reporter nodes are placed so that each sensor node can directly communicate with at least one reporter node, using the above procedure, each sensor node s_k is able to find its $r(s_k)$ during the initialization phase.

5.2 Mapping

In the mapping phase, a time-driven asynchronous ICLA which is isomorphic to the sensor network topology is created. Each sensor node s_k in the sensor network corresponds to the cell c_k in ICLA. Two cells c_k and c_m in ICLA are adjacent to each other if $r(s_k)$ is equal to $r(s_m)$; that is, the reporter nodes of s_k and s_m are identical. The learning automaton in each cell c_k of ICLA, referred to as LA_k, has two actions α_0 and α_1. Action α_0 is "switch off the sensing unit of sensor node s_k" and action α_1 is "switch on the sensing unit of sensor node s_k". The probability of selecting each of these actions is initially set to 0.5.

5.3 Operation

During the operation mode, normal operation of the network is performed; that is sensor nodes sense the environment for the existence of target points and if any target is detected, send information about the detected target to the sink through the infrastructure of reporter nodes. The operation phase is divided into a number of epochs. Each epoch starts with an *evaluation* phase, followed by a *status selection* phase and ends with a *monitoring* phase. The first epoch is an exception, since it starts with the *status selection* phase and it has no *evaluation* phase.

Each cell of ICLA during the operation phase is activated asynchronously. The nth activation of cell c_k occurs at the startup of the epoch Ep_k^n. We call n the local iteration number for the cell. The operation phase starts when a cell in ICLA is activated.

1. If (the activation of cell c_k of ICLA is its first activation (local iteration $n = 1$)) then
   - 1-1. *Status Selection* phase: sensor node s_k makes decision for the status of its sensing unit for epoch Ep_k^n. The operation mode of s_k during the *status selection* phase is $CPU_A S_A C_S$.
   - 1-2. *Monitoring* phase: sensor node s_k switches its sensing unit on or off based on the decision made in the previous step. If the sensing unit is active, then s_k monitors its sensing region and sends any required information about detected target points in its sensing region to $r(s_k)$. Required information about a detected target is application specific, and hence we do not specify any details for it. The operation mode of s_k during the *monitoring* phase is $CPU_A S_A C_S$ (if the sensing unit is selected to be active) or $CPU_S S_S C_S$ (if the sensing unit is selected to be sleep).
2. If (the activation of cell c_k is not its first activation (local iteration $n > 1$)) then
   - 2-1. *Evaluation* phase: sensor node s_k evaluates its selected status and sends it to $r(s_k)$. The operation mode of s_k during the *evaluation* phase is $CPU_A S_A C_A$.
   - 2-2. *Status Selection* phase: This step is similar to the step 1-1.
   - 2-3. *Monitoring* phase: This step is similar to the step 1-2.

Figure 2 gives the transition diagram of the operation modes of a sensor node in different phases of the proposed scheduling algorithm.

5.3.1 Status selection phase

As we stated before, at the startup of each epoch, every sensor node has to select the status of its sensing unit for the monitoring phase of that epoch. This selection must be done based on a short-term prediction about the movement paths of target points in the vicinity of each sensor node. Learning automaton resides in the cell c_k of the ICLA helps the sensor node s_k to select the status of its sensing unit for the monitoring phase of each epoch Ep_k^n. This is done through the following algorithm.

1. Sensor node s_k checks its sensing region for at most *CheckingDuration* to see if any target points can be detected;
2. If (any target points can be detected) Then
   - 2-1. s_k sets its operation mode for the monitoring phase of the epoch Ep_k^n to $CPU_A S_A C_S$;

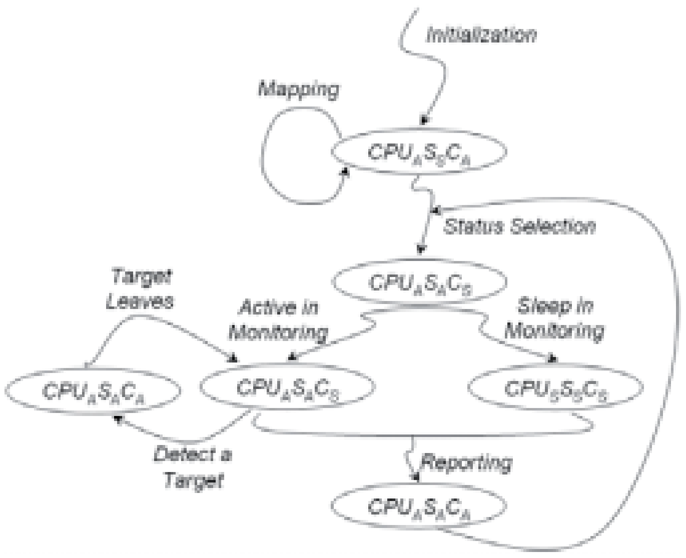

FIGURE 2
Transition diagram of the operation modes of a sensor node in different phases of the proposed scheduling algorithm.

3. Else
   3-1. LA_k decides whether to switch the sensing unit of s_k on or off for the monitoring phase of the epoch Ep_k^n; that is, the cell c_k chooses one of its actions using its action probability vector. We refer to this action by $\alpha_{i,k}^n$
4. s_k enters the monitoring phase of the epoch Ep_k^n with the specified operation mode;

In the above algorithm, *CheckingDuration* is a constant which specifies the maximum duration of the *selection status* phase.

5.3.2 Monitoring phase

During the *monitoring* phase, a sensor node s_k may be in $CPU_A S_A C_S$ or $CPU_S S_S C_S$ operation mode based on the selection done in the *status selection* phase. A sensor node s_k which is in $CPU_A S_A C_S$ operation mode, senses its sensing region and if detects a target point, switches its communicating device on and sends any required information about it to $r(s_k)$. A sensor node s_k which is in $CPU_S S_S C_S$ operation mode, does nothing during the *monitoring* phase. In other words, the short-term prediction which is performed by each sensor node s_k in the *status selection* phase is applied to the status of the sensing unit of s_k during the *monitoring* phase. Each *monitoring* phase lasts for the duration of *MonitoringDuration*. Since a short-term prediction can be valid only for a short duration of time, long values of *MonitoringDuration* results in lesser network detection rate (See experiment 5).

5.3.3 *Evaluation* phase

At the startup of each epoch Ep_k^n, each sensor node s_k has to evaluate its prediction for the status of its sensing unit during the epoch Ep_k^{n-1} (which is used as a reinforcement signal to LA_k). The evaluation phase of the epoch Ep_k^n will be performed only if in the status selection phase of epoch Ep_k^{n-1}, the status of the sensing device of sensor node s_k is selected by LA_k (step 3-1 in the status selection phase).

The evaluation process consists of two parts; computing the internal feedback (denoted by $\underline{\delta}_k^{n-1}$) and computing the external feedback (denoted by $\overline{\delta}_k^{n-1}$). For computing the internal feedback, each sensor node s_k evaluates its predicted status based on its own information about the target points passed through its sensing region whereas for computing the external feedback, the evaluation is performed using the information of the neighboring sensor nodes. The external feedback can enhance the internal feedback due to the fact that a target point which passes through the sensing region of a sensor node s_k may also pass through the sensing region of its neighboring sensor nodes with a high probability.

Depending on the action selected by LA_k in epoch Ep_k^{n-1}, the internal feedback is computed differently as follows:

- **The action selected by *LA_k* was "switch on the sensing device":** Sensor node s_k uses equation (5) for computing the internal feedback ($\underline{\delta}_k^{n-1}$).

$$\underline{\delta}_k^{n-1} = \begin{cases} 1 - \dfrac{\tau_{M,s_k}^{n-1}}{MonitoringDuration}; & \\ \qquad\qquad \dfrac{\tau_{M,s_k}^{n-1}}{MonitoringDuration} < NDR_M & \\ -\dfrac{\tau_{M,s_k}^{n-1}}{MonitoringDuration}; & Otherwise \end{cases} \tag{5}$$

  In the above equation, τ_{M,s_k}^{n-1} is a fraction of the *monitoring* phase of epoch Ep_k^{n-1} during which no target point is detected by s_k. If τ_{M,s_k}^{n-1} is a substantially small fraction of *MonitoringDuration* then it can be concluded that the selected action of LA_k (switch on the sensing device) was a proper selection for epoch Ep_k^{n-1}. In other words, if $\frac{\tau_{M,s_k}^{n-1}}{MonitoringDuration}$ is lower than a specified threshold (NDR_M), then the internal feedback of s_k for epoch $Ep_k^{n-1}(\underline{\delta}_k^{n-1})$ is positive. Otherwise, $\underline{\delta}_k^{n-1}$ is negative.

- **The action selected by *LA_k* was "switch off the sensing device":** In this case, sensor node s_k has no information about the passage of target points through its sensing region during the *monitoring* phase of epoch Ep_k^{n-1}. Therefore it cannot evaluate the action selected by its learning automaton. To compensate this lack of information, sensor node s_k

checks its sensing region for a very short duration called *Evaluation-Duration* (*EvaluationDuration* $\ll$ *MonitoringDuration*) to see if any target point can be detected. This short term checking is used by sensor node s_k for computing the internal feedback. In this case, the internal feedback of the sensor node s_k is computed using equation (6).

$$\underline{\delta}_k^{n-1} = \begin{cases} 1 - \dfrac{\tau^n_{E,s_k}}{EvaluatingDuration}; & \dfrac{\tau^n_{E,s_k}}{EvaluatingDuration} > NDR_E \\ -\dfrac{\tau^n_{E,s_k}}{EvaluatingDuration}; & Otherwise \end{cases} \tag{6}$$

In the above equation, τ^n_{E,s_k} is a fraction of the *EvaluationDuration* during which no target point is detected by s_k. If τ^n_{E,s_k} is a substantially small fraction of *EvaluationDuration* then we conclude that the action selected by LA_k (switch off the sensing device) was not a proper selection for epoch Ep_k^{n-1}. In other words, if $\frac{\tau^n_{E,s_k}}{MonitoringDuration}$ is higher than a specified threshold (NDR_E), the internal feedback of s_k for epoch $Ep_k^{n-1}(\underline{\delta}_k^{n-1})$ is positive. Otherwise, $\underline{\delta}_k^{n-1}$ is negative.

Each node s_k computes its internal feedback using equation (5) or (6) as described above and then creates an *InternalEvaluation* packet which contains $\alpha_{i,k}^{n-1}$ and $\underline{\delta}_k^{n-1}$. This packet is then sent to $r(s_k)$. $r(s_k)$ upon the reception of an *InternalEvaluation* packet from a sensor node s_k, replies to this packet with a *NeighborEvaluations* packet which contains for each neighbor s_l of s_k its $\alpha_{i,l}^{n-1}$ and $\underline{\delta}_l^{n-1}$. Using the information received by the *NeighborEvaluations* packet, each sensor node s_k computes the external feedback ($\overline{\delta}_k^{n-1}$) as follows:

$$\overline{\delta}_k^{n-1} = \begin{cases} 0; & |N(s_k)| = 0 \\ \dfrac{|N_A|}{|N(s_k)|}; & \dfrac{|N_A|}{|N(s_k)|} > LNSA \text{ and } \alpha_{i,k}^{n-1} = \alpha_1 \\ -\dfrac{|N_A|}{|N(s_k)|}; & \dfrac{|N_A|}{|N(s_k)|} > LNSA \text{ and } \alpha_{i,k}^{n-1} = \alpha_0 \\ \dfrac{|N_A|}{|N(s_k)|} - 1; & \dfrac{|N_A|}{|N(s_k)|} \le LNSA \text{ and } \alpha_{i,k}^{n-1} = \alpha_1 \\ 1 - \dfrac{|N_A|}{|N(s_k)|}; & \dfrac{|N_A|}{|N(s_k)|} \le LNSA \text{ and } \alpha_{i,k}^{n-1} = \alpha_0 \end{cases} \tag{7}$$

In the above equation, $N(s_k)$ is the set of neighbor nodes of s_k i.e. set of sensor nodes s_l for which $r(s_l)$=$r(s_k)$ and $|S|$ represents the cardinality of the set S.

N_A is a subset of $N(s_k)$ and is defined according to equation (8).

$$N_A = \{s_l \in N(s_k) | (\alpha_{i,l}^{n-1} = \alpha_1) \wedge (\underline{\delta}_l^{n-1} > 0).\} \tag{8}$$

In other words, N_A is the subset of $N(s_k)$ whose sensing devices were active in their previous *monitoring* phase and their internal evaluations are positive. $\frac{|N_A|}{|N(s_k)|}$ gives the fraction of such neighbors. If this fraction falls below a certain threshold (*LNSA*), it can be concluded that $\alpha_1$was not a proper selection for sensor node s_k in epoch Ep_k^{n-1}. According to the equation (7), only a proper selection will result in a positive external feedback.

Using the computed internal and external feedbacks, each sensor node s_k computes its feedback for the previous epoch (δ_k^{n-1}) using the equation (9).

$$\delta_k^{n-1} = \begin{cases} \psi_k^{n-1}.\underline{\delta}_k^{n-1} + (1 - \psi_k^{n-1}) \cdot \overline{\delta}_k^{n-1}; & \overline{\delta}_k^{n-1} > 0 \\ \underline{\delta}_k^{n-1}; & \textit{otherwise} \end{cases} \tag{9}$$

In the above equation, ψ_k^{n-1} is a coefficient which specifies the impact of internal and external feedbacks on the value of δ_k^{n-1}. It can be a time varying coefficient (as it is indicated in equation (9)) or a constant coefficient (See experiments 6 and 7). δ_k^{n-1} is then used for the computation of the reinforcement signal β_k^{n-1} as follows:

$$\beta_k^{n-1} = \begin{cases} 0; & \delta_k^{n-1} \geq 0 \\ 1; & \textit{Otherwise} \end{cases} \tag{10}$$

This reinforcement signal is given to LA_k. LA_k updates its action probability vector based on the selected action $\alpha_{i,k}^{n-1}$and the given β_k^{n-1} according to the following learning algorithm with time varying parameters a^n and b^n:

$$\begin{aligned} p_{i,k}^{n+1} &= p_{i,k}^n + a_k^n.(1 - p_{i,k}^n) \\ p_{j,k}^{n+1} &= 1 - p_{i,k}^{n+1} \end{aligned} \tag{11}$$

$$\begin{aligned} p_{i,k}^{n+1} &= (1 - b_k^n) \cdot p_{i,k}^n \\ p_{j,k}^{n+1} &= 1 - p_{i,k}^{n+1} \end{aligned} \tag{12}$$

Reward parameter $a_k^n$and penalty parameter b_k^n are time varying parameters which vary according to equations (13) and (14).

$$a_k^n = a \cdot \delta_k^n \tag{13}$$

$$b_k^n = -b \cdot \delta_k^n \tag{14}$$

In equations (13) and (14), a and b are two constants which control the rate of learning. Higher values of a and b results in faster learning and lesser precision.

In short, on the nth activation of each cell c_k of ICLA, if LA_k selected any action on the $(n-1)$th activation of ICLA, the reinforcement signal β_k^n is computed and the selected action is rewarded or penalized accordingly. Then, the node s_k checks if any target points exist in its sensing region and if so, it selects its operation mode as $CPU_A S_A C_S$ for the upcoming *monitoring* phase. Otherwise, LA_k decides whether to switch the sensing device of s_k on or off for the upcoming *monitoring* phase.

6 EXPERIMENTAL RESULTS

To evaluate the performance of the proposed method several experiments have been conducted and the results are compared with the results obtained for LEACH [19] from routing layer algorithms, GAF [29] from grid-based algorithms, PEAS [55, 89] from coverage-related algorithms, Proactive Wakeup based on PEAS (PW) [7] from tracking-based algorithms.

The simulation environment is a 100(m) $\times$ 100(m) area through which 100 sensor nodes are scattered randomly. Sensing range (r) of sensor nodes is assumed to be 10(m). For placing the reporter nodes, the simulation environment is divided into a number of square cells with dimensions 14(m) $\times$ 14(m). One reporter node is placed on each boundary point of cells.

Energy consumption of nodes follows the energy model of the J-sim simulator [71]. Based on this model, the power consumption of a node during the $CPU_A S_A C_A$, $CPU_A S_A C_S$, $CPU_A S_S C_A$ and $CPU_S S_S C_S$ operation modes are 18.9(mW), 2.901(mW), 18.9(mW), 0.001(mW) respectively. Energy required to switch a node from one operation mode to another operation mode is assumed to be negligible. *CheckingDuration*, *MonitoringDuration*, *EvaluatingDuration*, τ_{Adv}, NDR_M, NDR_E and *LNSA* are set to 5(s), 100(s), 20(s), 1(s), .6, .6 and .2 respectively. ψ_k^n is assumed to be constant and is set to .4 for all sensor nodes. Learning parameters a and b are both set to .1.

In all experiments, following different types of moving target points are considered to exist in the area of the network:

- **Constant Events:** Constant events have constant start times and durations. At the startup of the simulation, number of constant events is selected uniformly at random from the range [1, 10]. The start time of each event is a constant which is uniformly and randomly selected from the range [1, 1000] and the duration of each event is a constant which is uniformly and randomly selected from the range [1, 10]. Each event has a static occurrence position which is selected uniformly at random in Ω. Events are repeated with the same start times and durations every 1000 seconds of the simulation.
- **Noisy Events:** Noisy events are like constant events except that the start times and durations of events are affected every 1000 seconds by a normally distributed random noise.

- **Poisson Events:** The occurrences of Poisson events follow the Poisson distribution. To generate such events, a number of Poisson random number generators (selected uniformly and randomly from the range [1, 10]) are used. Each Poisson random number generator separately generates events in a randomly selected location in Ω.
- **Straight Path Moving Objects:** Each straight path moving object has a start position, stop position and a velocity. A straight path moving object starts from its start position and moves directly towards its stop position using its velocity. The start and stop positions are selected uniformly at random in Ω and the velocity is selected uniformly and randomly from the range [0, *MaxVelocity*]. The number of straight path moving objects is selected randomly from the range [1, 10]. A straight path moving object restarts its movement from its start position with a newly random velocity when it reaches its stop position.
- **Complex Path Moving Objects:** These target points are like straight path moving objects except that they are not moving from their start positions directly towards their stop positions. Instead, each complex path moving object has a sorted list of points

  $$\langle (x^{start}, y^{start}), (x^1, y^1), (x^2, y^2), \ldots, (x^{Stop}, y^{Stop}) \rangle.$$

  A complex path moving object starts from its start point and moves directly towards (x^1, y^1) using its velocity. Whenever the target point reaches (x^1, y^1), it changes its direction towards (x^2, y^2) with a newly random velocity. This movement continues until the target point reaches its stop position. The number of intermediate points for each moving object is selected randomly from the range [1, 5]. A complex path moving object restarts its movement from its start position with a newly random velocity when it reaches its stop position.
- **Random Waypoint Moving Objects:** A random waypoint moving object follows the random waypoint movement model [73].

All simulations have been implemented using J-Sim simulator [71] and the results are averaged over 25 runs.

6.1 Experiment 1

In this experiment, the proposed algorithm is compared with GAF, LEACH, PEAS and PW algorithms in terms of network detection rate (η_D), network sleep rate (η_S), network redundant active rate (η_R) defined by equation (15) and the mean energy consumption of nodes.

$$\eta_R = \frac{\sum_{s_k} \left(\frac{\tau_{s_k} - \sum_{p_i \in P} \tau_{s_k, p_i}}{\mathrm{T}} \right)}{N} \tag{15}$$

Table 1 gives the parameters used for simulating the moving target points.

Target Point	Parameter	Distribution	Value
Constant Events	Number of Events	Uniform	[1, 10]
	Event Start Time	Uniform	[1, 1000]
	Event Duration	Uniform	[1, 10]
Noisy Events	Number of Events	Normal	$\mu =$ Randomly selected from the range [1, 10] $\sigma = 20$
	Event Start Time	Uniform	$\mu =$ Randomly selected from the range [1, 1000] $\sigma = 20$
	Event Duration	Uniform	$\mu =$ Randomly selected from the range [1, 10] $\sigma = 20$
Poisson Events	λ	Poisson	2.0
	Event Duration	Constant	2
Straight Path Moving Objects	*MaxVelocity*	Uniform	1
Complex Path Moving Objects	*MaxVelocity*	Uniform	1
	Number of Intermediate points	Uniform	[1, 5]

TABLE 1
Parameters used for simulating different target points in experiment 1

Figures 3 through 6 give the results of comparison in terms of η_D, η_S, η_R and mean consumed energy of all nodes of the network respectively. As it can be seen from Figure 3, network detection rate in the proposed algorithm outperforms PEAS and GAF, approaches PW, and about 4% worse than the LEACH for which the network detection rate is equal to 1.

Figure 4 shows that for the proposed algorithm the network sleep rate (η_S) is about 0.6 which is better than all the existing algorithms.

Figure 5 shows for the proposed algorithm that the fraction of the times in which nodes are in active mode while no target point is in their sensing region is about .25 on average which is better than all the other methods.

Finally, Figure 6 shows that for the proposed algorithm, the mean energy consumption is lower than the mean energy consumption of all the existing algorithms except for GAF. The mean energy consumption of GAF algorithm is nearly equal to that of the proposed algorithm.

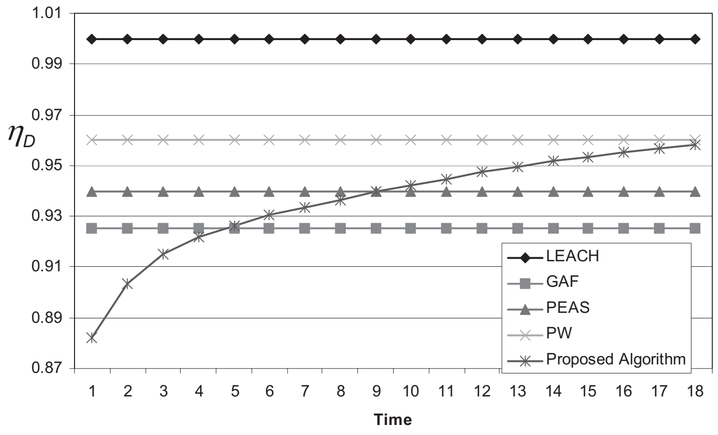

FIGURE 3
Network detection rate (η_D).

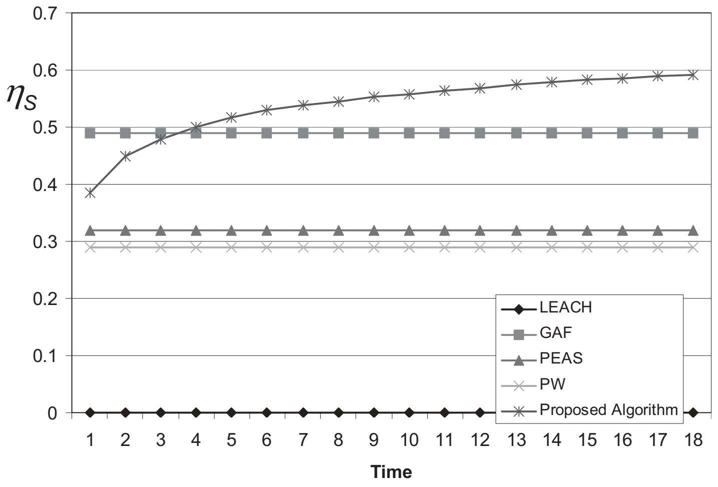

FIGURE 4
Network sleep rate (η_S).

Table 2 gives the network lifetime for the proposed method and the existing methods. We use the number of simulation round at which the network area (Ω) is no further completely covered by the network (*Definition 1*). As it is shown, the proposed method can better prolong the network lifetime.

6.2 Experiment 2

In this experiment, we study the effect of the noise level in noisy events on the performance of the proposed algorithm. For this purpose, we change the

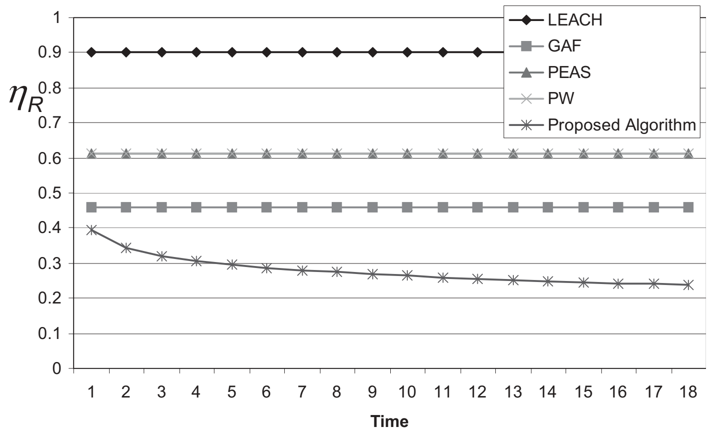

FIGURE 5
Network redundant active rate (η_R).

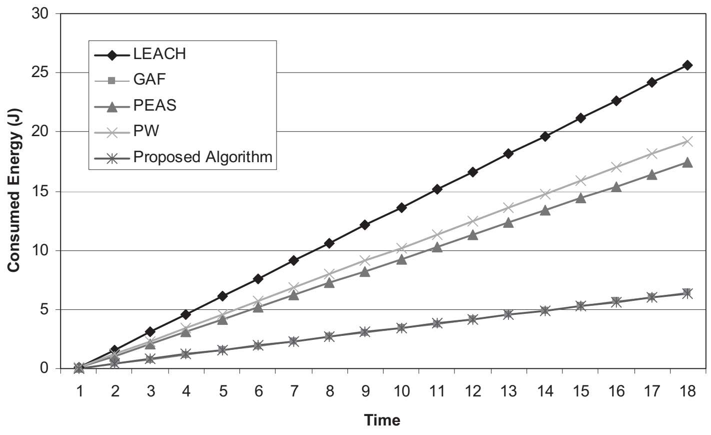

FIGURE 6
Mean Consumed Energy of all nodes of the network.

Algorithm	LEACH	PEAS	PW	GAF	Proposed Method
Network Lifetime	198	269	238	317	319

TABLE 2
Network lifetime

standard deviation of the normally distributed noise from 20 to 400. Figures 7 to 10 show the results of this study. As it is shown, increasing the noise level does not affect the performance of the proposed method very much.

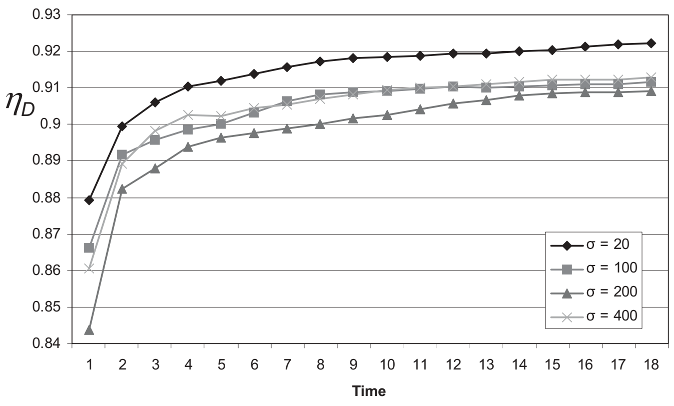

FIGURE 7
Network detection rate (η_D).

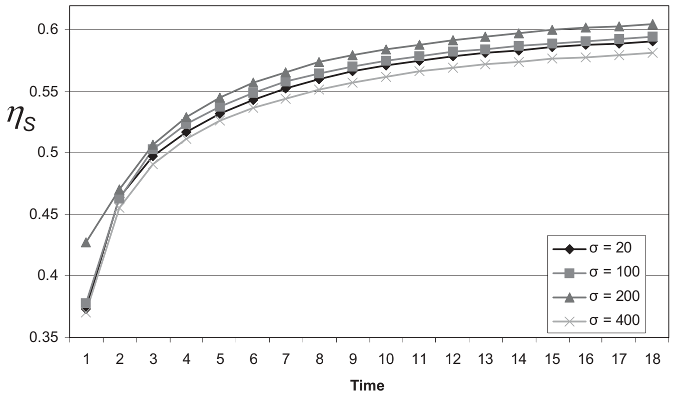

FIGURE 8
Network sleep rate (η_S).

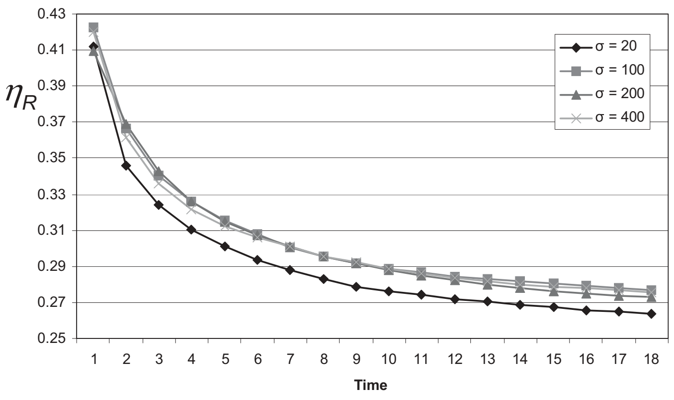

FIGURE 9
Network redundant active rate (η_R).

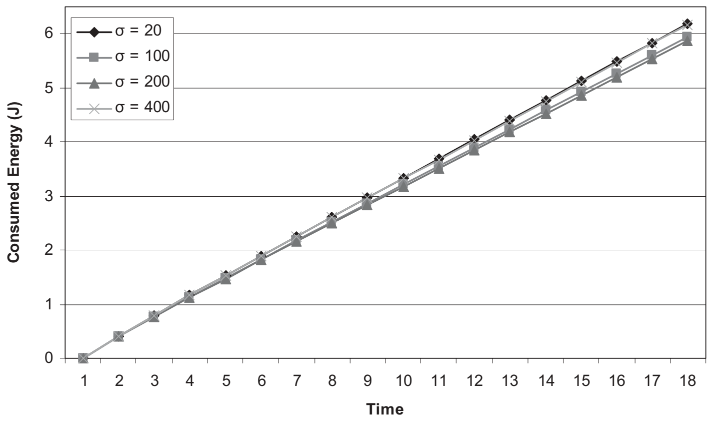

FIGURE 10
Mean Consumed Energy of all nodes of the network.

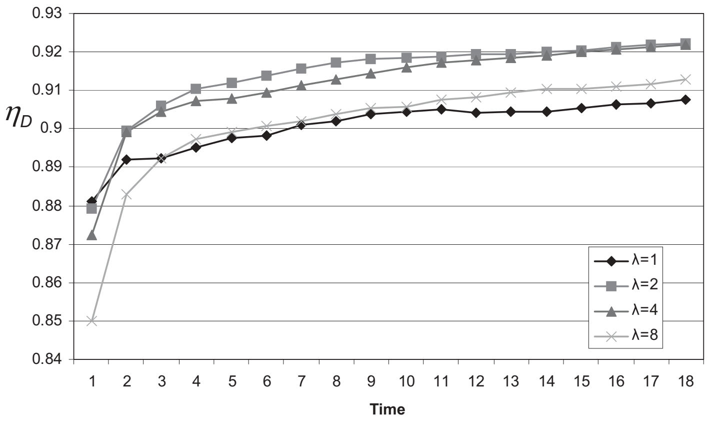

FIGURE 11
Network detection rate (η_D).

6.3 Experiment 3

In this experiment, we evaluate the performance of the proposed algorithm when the mean number of Poisson events per round (λ) varies. Figures 11 to 14 show the results of this experiment when λ varies from 1 to 8. It can be seen from these figures that the performance of the proposed algorithm does not depend very much on the value of λ.

6.4 Experiment 4

This experiment is designed to evaluate the performance of the proposed method when the *MaxVelocity* of moving objects varies. For this purpose, we change *MaxVelocity* from 0 to 1. Figures 15 to 18 show the results in terms of

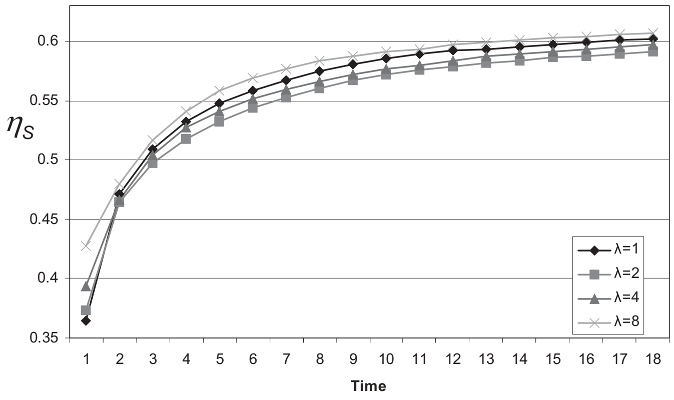

FIGURE 12
Network sleep rate (η_S).

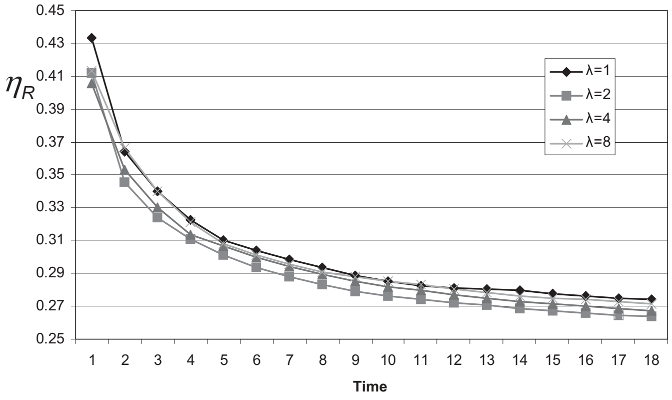

FIGURE 13
Network redundant active rate (η_R).

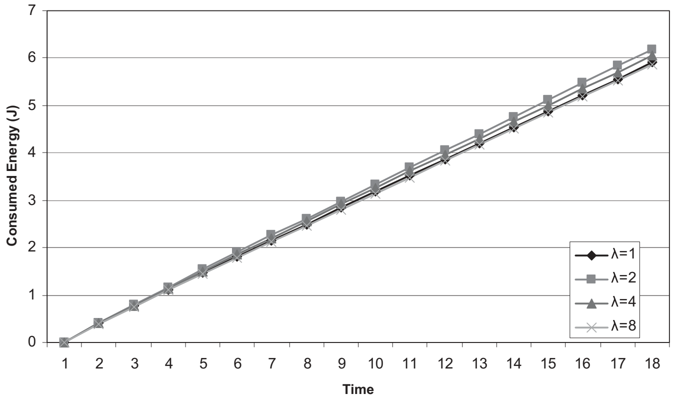

FIGURE 14
Mean Consumed Energy of all nodes of the network.

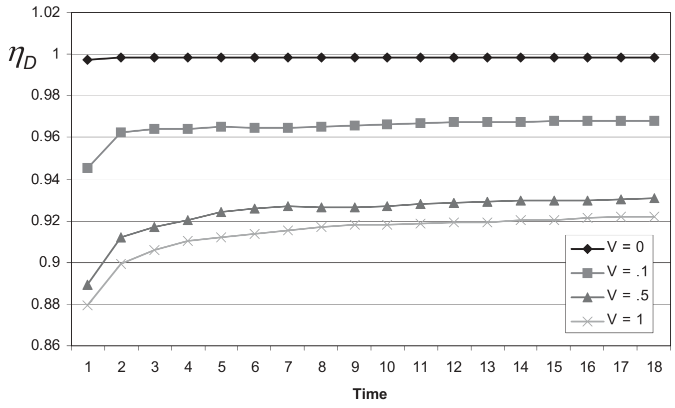

FIGURE 15
Network detection rate (η_D).

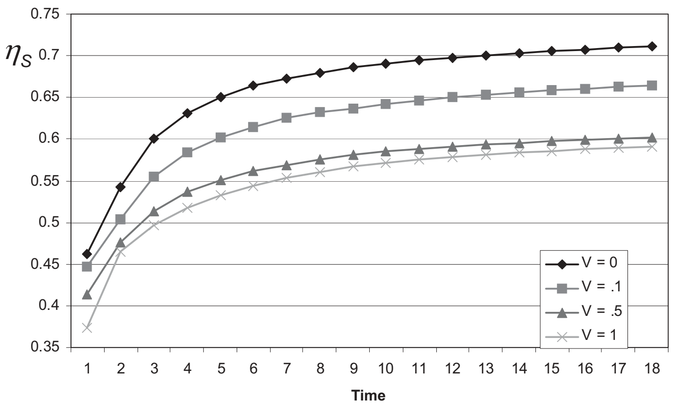

FIGURE 16
Network sleep rate (η_S).

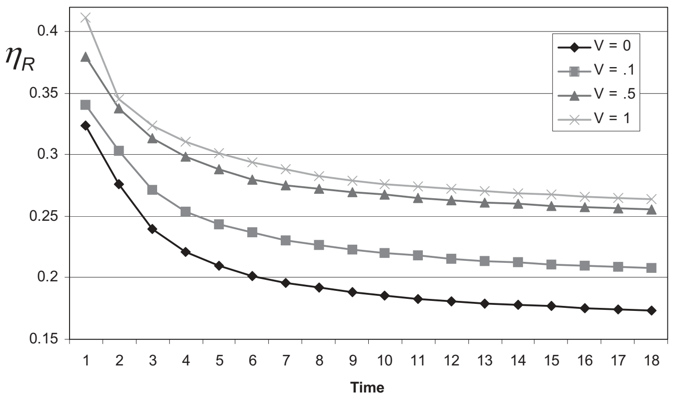

FIGURE 17
Network redundant active rate (η_R).

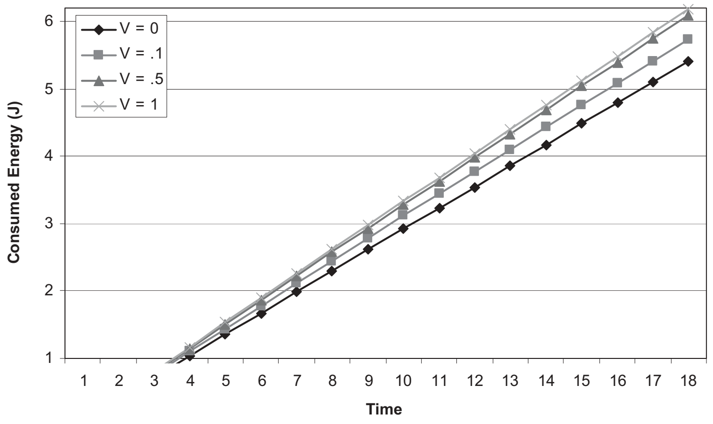

FIGURE 18
Mean Consumed Energy of all nodes of the network.

network detection rate (η_D), network sleep rate (η_S), network redundant active rate (η_R) and the mean energy consumption of nodes. As indicated by these figures, when moving objects have no movement at all ($MaxVelocity = 0$), the performance of the proposed method in terms of η_D, η_S, $\eta_R$and mean energy consumption of the nodes is very high. Increasing the value of *MaxVelocity* from 0 to .5, degrades the performance of the proposed method, but further increasing in the value of *MaxVelocity* does not affect it that much. In other words, performance of the proposed method is highly affected by the movement of target points (in comparison to the case when target points have no movement), but it is not affected too much by increasing the movement velocity.

6.5 Experiment 5

This experiment is conducted to study the effect of the *MonitoringDuration* on the performance of the proposed method. For this experiment we use the target points of experiment 1. We also let *MonitoringDuration* varies in the range [25, 200]. Figures 19 to 22 show the results of this experiment in terms of network detection rate (η_D), network sleep rate (η_S), network redundant active rate (η_R) and mean energy consumption of the nodes. From these figures we can conclude that 1. Lower values of *MonitoringDuration* results in more network detection rate at the expense of more energy consumption, more redundant active times and less sleep times. 2. Higher values of *MonitoringDuration* results in more energy saving in the network at the expense of poor network detection rate.

Figure 23 gives the mean energy consumption of nodes versus network detection rate as *MonitoringDuration* varies. As it can be seen from this figure, increasing the value of *MonitoringDuration* results in the network detection

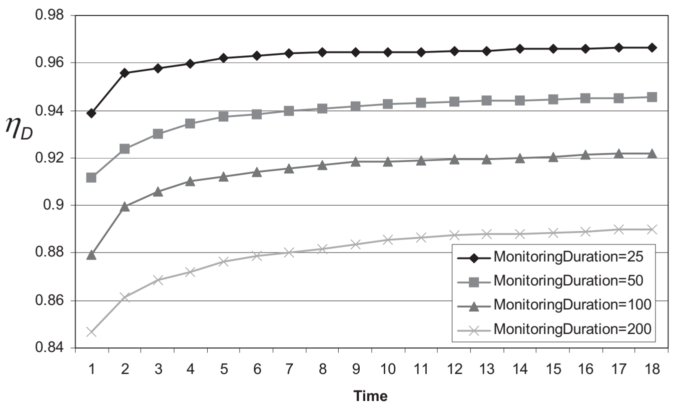

FIGURE 19
Network detection rate (η_D).

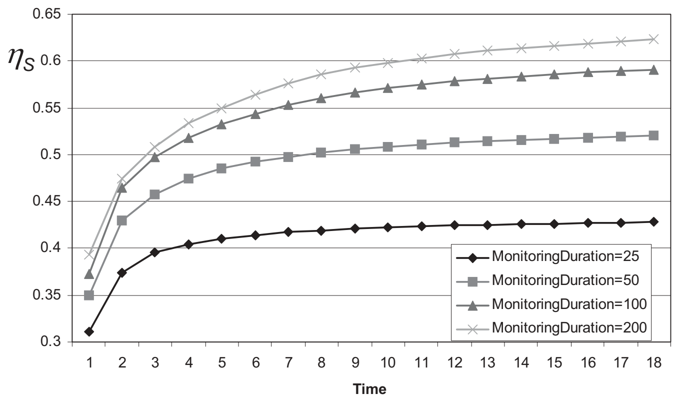

FIGURE 20
Network sleep rate (η_S).

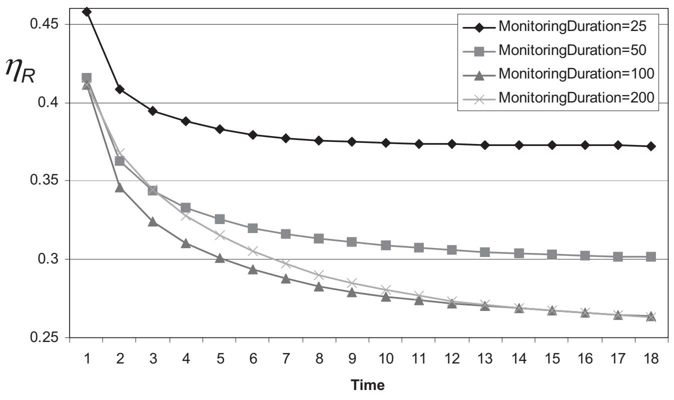

FIGURE 21
Network redundant active rate (η_R).

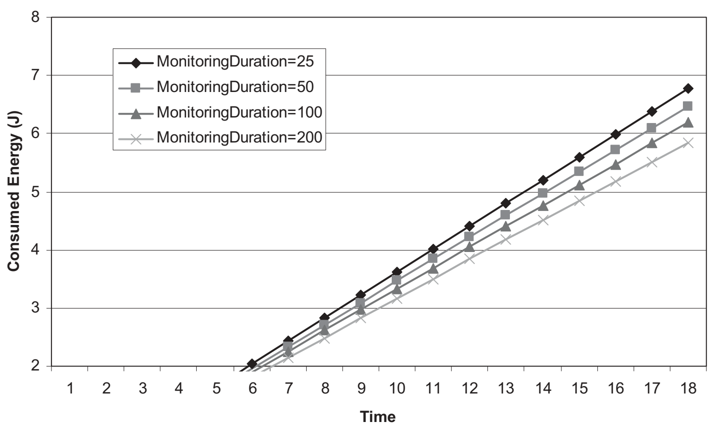

FIGURE 22
Mean Consumed Energy of all nodes of the network.

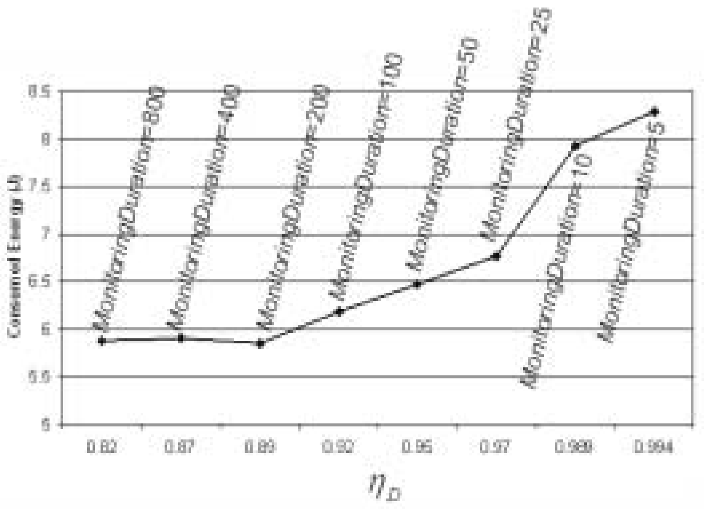

FIGURE 23
Mean energy consumption of nods vs. Network detection rate as *MonitoringDuration* varies.

rate to decrease. This can be explained as follows: By increasing the value of *MonitoringDuration* the length of the *monitoring* phase in the proposed algorithm increases. As a result, the short-term prediction of the *status selection* phase of the algorithm must apply to a longer period of time which results in the precision of the algorithm to decrease.

The figure also indicates that as *MonitoringDuration* increases, the mean energy consumption of nodes decreases to a minimum level (*MonitoringDuration* $= 200$) and then remains almost fixed. This can also be explained by the effect of a long *MonitoringDuration* on the precision of the short-term prediction performed in the *status selection* phase. A low precise prediction results in a pure chance selection, i.e. the probability of selecting for the sensing unit of the node to be off or on in each epoch becomes equal. Therefore, number of epochs in which the sensor node is in $CPU_A S_A C_S$ or $CPU_S S_S C_S$ operation mode is equal in expected sense. In addition, considering a large value for *MonitoringDuration*, the times a sensor node spends in *CheckingDuration* and *EvaluatingDuration* would be negligible. This indicates that no matter how long is the *MonitoringDuration*, each sensor node spends half of its lifetime in the $CPU_A S_A C_S$ operation mode and the other half in $CPU_S S_S C_S$ operation mode in expected sense, and hence the mean energy consumption of nodes remains almost fixed for large values of *MonitoringDuration*.

Figure 24 compares the proposed algorithm with LEACH, GAF, PEAS and PW in terms of the network detection rate and the mean energy consumption of nodes. This figure shows that there exists a value for *MonitoringDuration* (*MonitoringDuration* $=$ 10) below which the proposed algorithm outperforms GAF, PEAS and PW and approaches LEACH in terms of network detection rate and there exists another value for *MonitoringDuration* (*MonitoringDuration* $=$ 200) above which the proposed algorithm outperforms GAF, PEAS, PW and LEACH in terms of mean energy consumption of nodes.

Determination of *MonitoringDuration* for an application is very crucial and is a matter of cost versus precision. For higher network detection rate, higher price must be paid. For example, as it is indicated in Figure 24, for network detection rate to be above 98 percent, each node must consume about

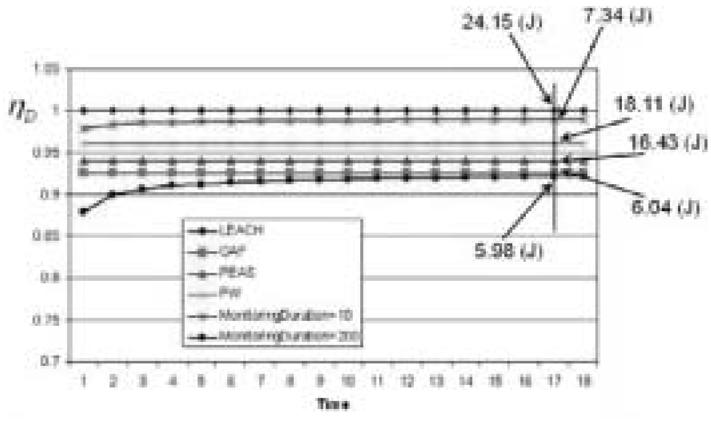

FIGURE 24
Comparison of the proposed algorithm with LEACH, GAF, PEAS and PW in terms of the network detection rate and the mean energy consumption of nodes.

7.34(J) on average during the lifetime of the network whereas to obtain a network detection rate of about 92 percent, each node must consume only about 5.98(J) on average. Reaching highest precision which can be obtained by setting *MonitoringDuration* to zero results in maximum energy consumption by the network.

6.6 Experiment 6

In this experiment, we study the effect of the internal and external feedbacks on the performance of the proposed method. To do this, we let ψ varies in the range [0, 1] ($\psi = 0$ means ignoring the effect of the internal feedback and $\psi = 1$ means ignoring the effect of the external feedback). Figures 25 to 28 show the results of this experiment in terms of network detection rate

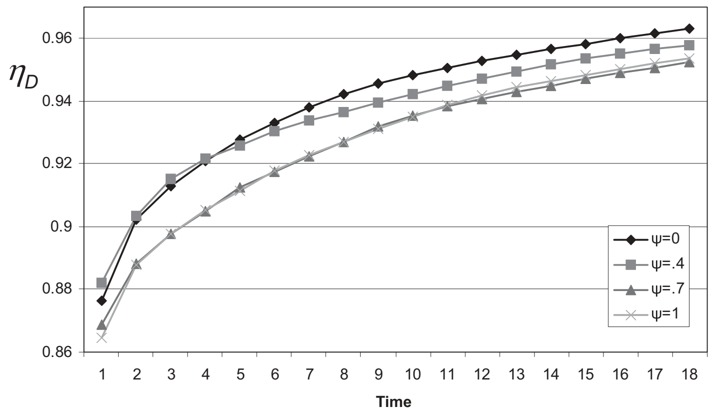

FIGURE 25
Network detection rate (η_D).

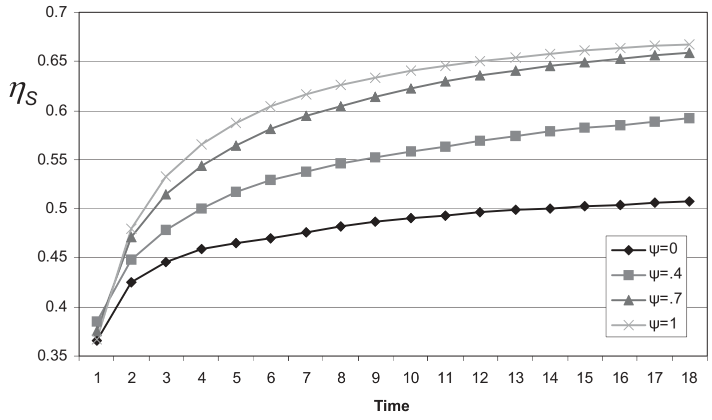

FIGURE 26
Network sleep rate (η_S).

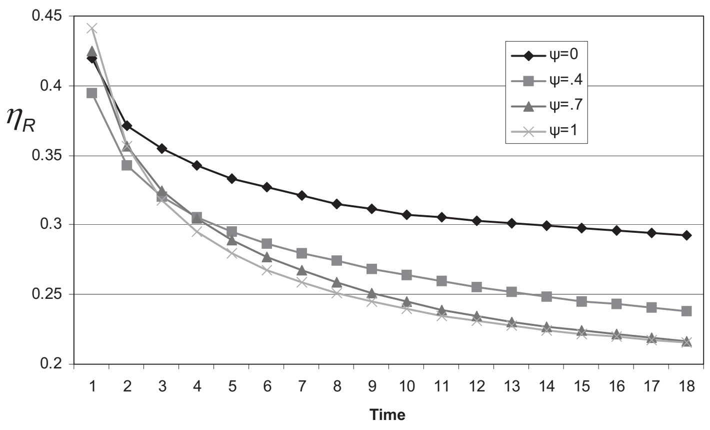

FIGURE 27
Network redundant active rate (η_R).

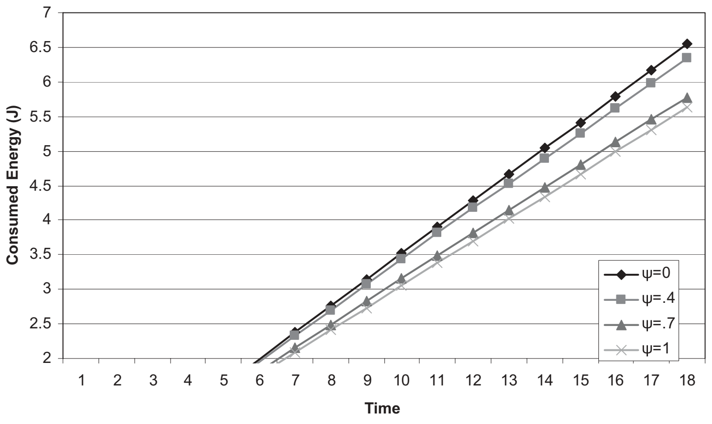

FIGURE 28
Mean Consumed Energy of all nodes of the network.

(η_D), network sleep rate (η_S), network redundant active rate (η_R) and mean energy consumption of the nodes. From these figures we can conclude that 1. decreasing the effect of the internal feedback in equation (9) by decreasing parameter ψ results in more network detection rate at the expense of more energy consumption, more redundant active times and less sleep times. 2. Increasing the effect of the internal feedback in equation (9) by increasing parameter ψresults in more energy saving in the network at the expense of lower network detection rate.

6.7 Experiment 7

Experiment 6 has shown that the internal and external feedbacks have different effects on the performance of the proposed algorithm in terms of network detection rate (η_D), network sleep rate (η_S), network redundant active rate (η_R) and mean energy consumption of the nodes. The results reported in experiment 6 are holding only if the number of sensors in the network remains unchanged. This is because of the fact that the accuracy of external feedback of sensor node s_k starts degrading as the number of its neighboring sensors starts to decrease due to malfunctioning or energy depletion. Lesser number of neighboring sensors for a sensor means that the probability of accuracy of information obtained about the surrounding environment decreases. When the accuracy of the external feedback reduces, a node has to depends on its own information about the movement pattern of target points obtained using its sensing unit. Therefore, if a node wants its detection rate not to fall too much, then it has to let its sensing unit be in active state more often, which means more energy consumption.

One way to reduce the effect of the external feedback as the number of nodes reduces is to use a time varying ψ such as the one given in equation (16). In equation 16 ψ for each sensor node is defined to be a function of the number of neighbors of the node.

$$\psi_k^n = \begin{cases} \dfrac{N(s_k) - N^n(s_k)}{N(s_k)}; & N(s_k) \neq 0 \\ 1; & N(s_k) = 0 \end{cases} \tag{16}$$

In equation (16), $N^n(s_k)$ is the number of neighbors of s_k in the nth activation of c_k of the ICLA.

In this experiment, we study the performance of the proposed algorithm when ψ varies according to the equation (16). We compare the results obtained for the proposed algorithm for a fixed value of ψ ($\psi = .4$) and a time varying ψ. Figures 29 to 32 show the results of this experiment in terms of network detection rate (η_D), network sleep rate (η_S), network redundant active rate (η_R) and mean energy consumption of the nodes. These figures show that using a time varying ψ enhances the network detection rate at the expense of more energy consumption, more redundant active times and less sleep times.

6.8 Experiment 8

This experiment is conducted to study the effect of the number of reporter nodes on the performance of the proposed method. As we stated before, for placing the reporter nodes, the simulation environment is divided into a number of square cells and one reporter node is placed on each boundary point of cells. The experiment is conducted for different sizes of the square cells: 14(m) $\times$ 14(m), 10(m) $\times$ 10(m), 7(m) $\times$ 7(m) and 5(m) $\times$ 5(m). The number of reporter nodes (M) for these cases will be 49, 100, 196 and 500 respectively.

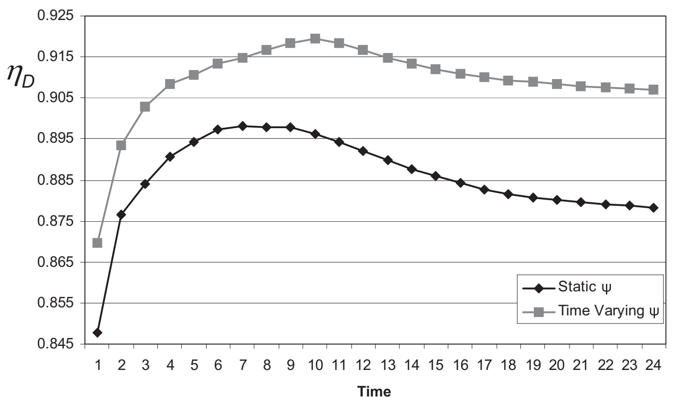

FIGURE 29
Network detection rate (η_D).

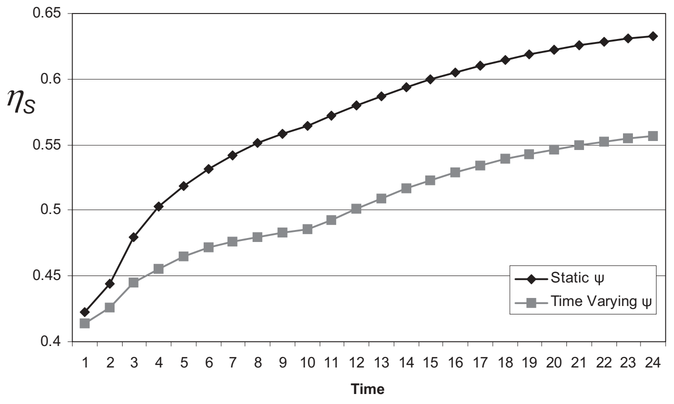

FIGURE 30
Network sleep rate (η_S).

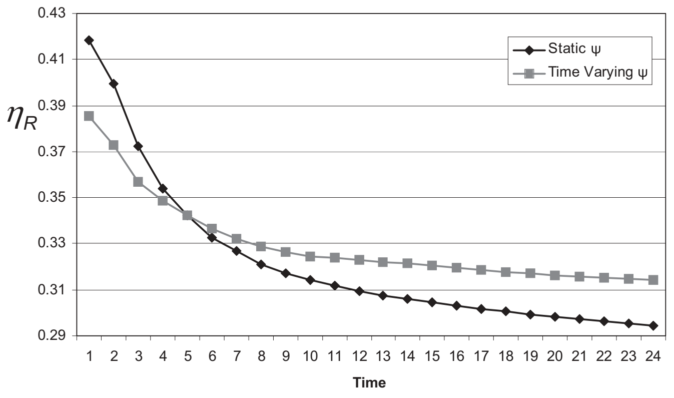

FIGURE 31
Network redundant active rate (η_R).

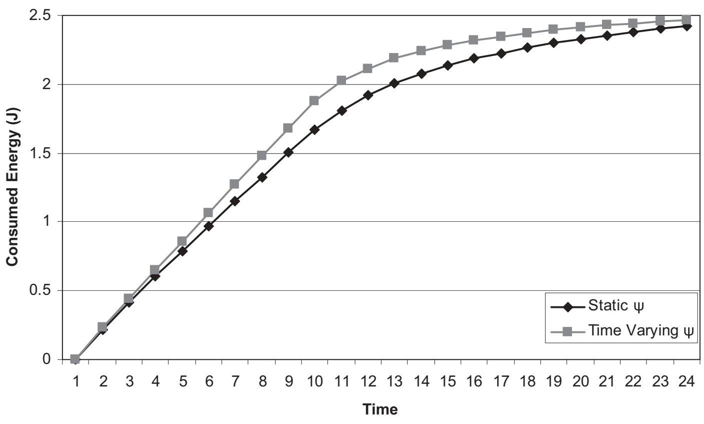

FIGURE 32
Mean Consumed Energy of all nodes of the network.

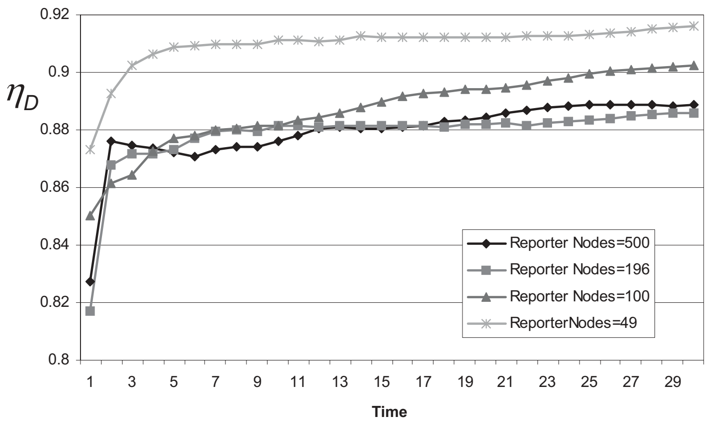

FIGURE 33
Network detection rate (η_D).

Figures 33 through 36 show the results of this experiment in terms of network detection rate (η_D), network sleep rate (η_S), network redundant active rate (η_R) and mean energy consumption of the nodes. These figures show that as the number of reporter nodes (M) increases from 49 to 196, $\eta_D$and $\eta_R$decrease and $\eta_S$and mean energy consumption of the nodes increase. Further increase in M beyond 196, has no significant effect on η_D, η_R, η_S and mean energy consumption of the nodes. The reason for this is that by increasing the number of reporter nodes from 49 to 196, the number of neighbors of each sensor node decreases which according to the results of experiment 7, reduces the accuracy of the external feedbacks of sensor nodes. When $M = 196$, no sensor node in the network has a neighbor and for this reason increasing M beyond 195

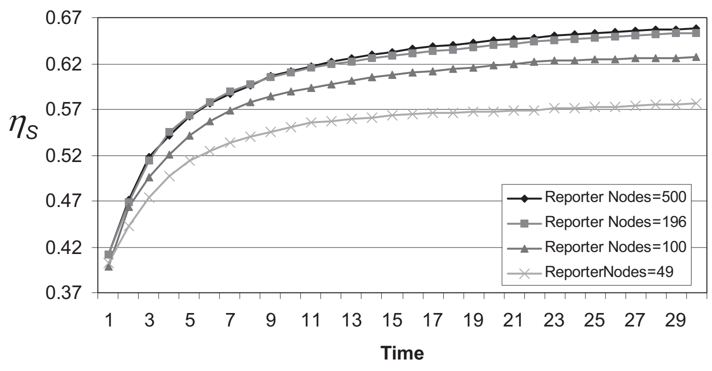

FIGURE 34
Network sleep rate (η_S).

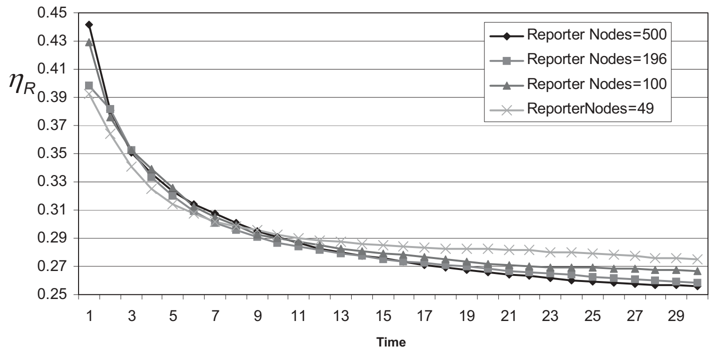

FIGURE 35
Network redundant active rate (η_R).

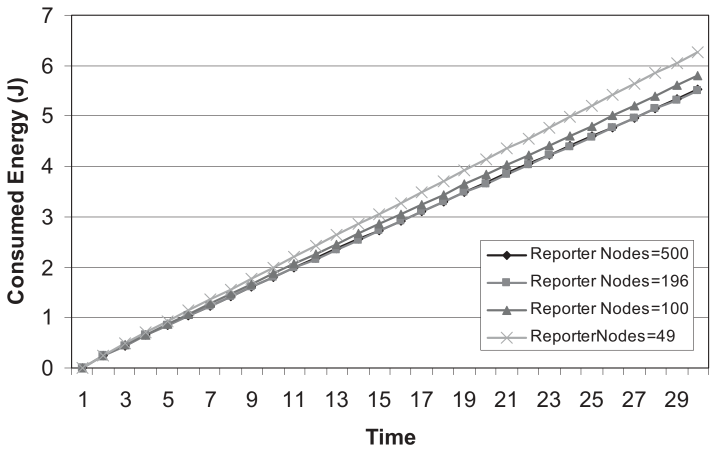

FIGURE 36
Mean Consumed Energy of all nodes of the network.

has no effect on the number of neighbors of sensor nodes. This implies that increasing M beyond 196 will have no significant effect on η_D, η_R, η_S and mean energy consumption of the nodes.

7 CONCLUSION

In this paper, an algorithm based on irregular cellular learning automata (ICLA) for the problem of detecting and monitoring a number of moving target points in an area by sensor networks was proposed. The sensor network was mapped into an ICLA. The learning automaton residing in each cell of the ICLA in cooperation with the learning automata residing in the neighboring cells dynamically predicts the existence of any target points in the vicinity of its corresponding node in the network in near future. This prediction is then used to schedule the active times of that node. The experimental results showed that there exists a parameter in the proposed method which can be tuned so that one can save more energy in the network at the expense of lesser precision or has more precision at the expense of lower network lifetime. In fact determination of this parameter for an application is very crucial and is a matter of cost versus precision. Furthermore, the experimental results showed that the proposed algorithm is robust against changes in the parameters of the target points' movement patterns such as the velocity of moving targets and the frequency of occurrences of events.

REFERENCES

[1] M. Ilyas and I. Mahgoub. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC Press, London, Washington, D.C., 2005.

[2] H. Chen, H. Wu and N.-F. Tzeng. Grid-based Approach for Working Node Selection in Wireless Sensor Networks. *Proc. of IEEE Intl. Conf. on Communications 2004 (ICC 2004),* 2004.

[3] Y. Zou and K. Chakrabarty. A Distributed Coverage- and Connectivity-Centric Technique for Selecting Active Nodes in Wireless Sensor Networks. *IEEE Transactions on Computers* **54**(8), 978–991, August 2005.

[4] F. Pedraza, A. García and A. L. Medaglia. Efficient Coverage Algorithms for Wireless Sensor Networks. *in Proc. of the 2006 Systems and Information Engineering Design Symposium,* 2006.

[5] Zh. Dingxing, X. Ming, Ch. Yingwen and W. Shulin. Probabilistic Coverage Configuration for Wireless Sensor Networks. *2nd Intl. Conf. on Wireless Communications, Networking and Mobile Computing (WiCOM 2006),* Wuhan, September 2006.

[6] R. Gupta and S. R. Das. Tracking Moving Targets in a Smart Sensor Network. *Proc. IEEE VTC* **5**, 3035–3039, October 2003.

[7] Ch. Gui and P. Mohapatra. Power Conservation and Quality of Surveillance in Target Tracking Sensor Networks. *Proc. of the 10th Annual Intl. Conf. on Mobile Computing and Networking (MOBICOM 2004),* Philadelphia, PA, USA, September–October, 2004.

[8] G. He and J. C. Hou. Tracking Targets with Quality in Wireless Sensor Networks. *13th IEEE Intl. Conf. on Network Protocols (ICNP 2005)*, Boston, MA, USA, November 2005.

[9] M. K. Watfa and S. Commuri. A Reduced Cover Approach to Energy Efficient Tracking using Wireless Sensor Networks. *World Congress in Computer Sceince, Computer Engineering and Applied Computing,* Las Vegas, Nevada, USA, June 2006.

[10] J. Jeong, S. Sharafkandi and D. H. C. Du. Energy-aware scheduling with quality of surveillance guarantee in wireless sensor networks. *Intl. Conf. on Mobile Computing and Networking,* Los Angeles, CA, USA, 2006.

[11] J. Jeong, T. Hwang, T. He and D. Du. MCTA: Target Tracking Algorithm based on Minimal Contour in Wireless Sensor Networks. *In IEEE Infocom2007 Minisymposia*, August 2007.

[12] V. Raghunathan, C. Schurgers, S. Park and M. B. Srivastava. Energy-Aware Wireless sensor networks. *IEEE Singal Processing Magazine*, 2002–03.

[13] K. Chakrabarty, S. S. Iyengar, H. Qi and E. Cho. Grid coverage for surveillance and target Location in distributed sensor networks. *IEEE Transactions on Computers* **51**, 1448–1453, 2002.

[14] G. Bontempi and Y. Le Borgne. An adaptive modular approach to the mining of sensor network data. *Workshop on Data Mining in Sensor Networks*, SIAM SDM, Newport Beach, CA, USA, April 2005.

[15] C. Liu, K. Wu and J. Pei. A Dynamic Clustering and Scheduling Approach to Energy Saving in Data Collection from Wireless Sensor Networks. *In Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'05),* Santa Clara, California, USA, September, 2005.

[16] O. Younis and S. Fahmy. An Experimental Study of Routing and Data Aggregation in Sensor Networks. *In Proceedings of the International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks (LOCAN), held in conjunction with The 2nd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS-2005),* November 2005.

[17] R. Virrankoski and A. Savvides. TASC: Topology Adaptive Spatial Clustering for Sensor Networks. *Second IEEE Intl. Conf. on Mobile Ad Hoc and Sensor Systems*, Washington, DC, November, 2005.

[18] S. Soro and W. Heinzelman. Prolonging the Lifetime of Wireless Sensor Networks via Unequal Clustering. *Proceedings of the 5th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (IEEE WMAN '05),* April 2005.

[19] W. Heinzelman, A. Chandrakasan and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. *Proc. of 33rd Hawaii Intl. Conf. on System Science (HICSS '00),* January 2000.

[20] O. Younis and S. Fahmy. Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach. In *Proc. of IEEE INFOCOM* **1**, 629–640, March 2004.

[21] S. P. Chaudhuri and D. B. Johnson. An Adaptive Scheduling Protocol for Multi-scale Sensor Network Architecture. *IEEE Intl. Conf. on Distributed Computing in Sensor Systems (DCOSS),* Santa Fe, New Mexico, June 2007.

[22] A. M. Chou and V. Li. Slot Allocation Strategies for TDMA Protocols in Multihop Packet Radio Networks. *In Proceedings of INFOCOM*, pp. 710–716, 1992.

[23] I. Chlamtac and A. Farago. Making Transmission Schedules Immune to Topology Changes in Multi-hop Packet Radio Networks. *IEEE/ACM Transactions on Networks* **2**, 23–29, 1994.

[24] V. Rajendran, K. Obraczka and J. J. Garcia-Luna-Aceves. Energy-efficient Collision-Free Medium Access Control for Wireless Sensor Networks. *In Proceedings of the First Intl. Conf. on Embedded Networked Sensor Systems (Sensys'03),* New York, pp. 181–192, 2003.

[25] L. Bao and J. J. Garcia-Luna-Aceves. A New Approach to Channel Access Scheduling for Ad hoc Networks. *In Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom'01),* New York, pp. 210–221, 2001.

[26] M. Sichitiu. Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks. In *Proc. of IEEE INFOCOM*, volume 1, March 2004.

[27] Y. Nam, T. Kwon, H. Lee, H. Jung and Y. Choi. Guaranteeing the network lifetime in wireless sensor networks: A MAC Layer Approach. *Elsevier Computer Communications* **30**, 2532–2545, June 2007.

[28] Sh. Liu, K. W. Fan and P. Sinha. Dynamic Sleep Scheduling using Online Experimentation for Wireless Sensor Networks. *In Proceeding of SenMetrics,* San Diego, July 2005.

[29] Y. Xu, J. S. Heidemann and D. Estrin. Geography-informed energy conservation for ad hoc routing. *In Proceedings of Mobile Computing and Networking,* pp. 70–84, 2001.

[30] J. Salzmann, S. Kubisch, F. Reichenbach and D. Timmermann. Energy and Coverage Aware Routing Algorithm in Self Organized Sensor Networks. *In Proceedings of Networked Sensing Systems (INSS'07)*, June 2007.

[31] C. Schurgers, V. Tsiatsis and M. Srivastava. STEM: Topology management for energy efficient sensor networks. *In Proceedings of the IEEE Aerospace Conference*, March 2002.

[32] R. Akl and U. Sawant. Grid-based Coordinated Routing in Wireless Sensor Networks. *In Proceedings of 4th IEEE Conf. on Computer Communications and Networking*, January 2007.

[33] G. Wang, G. Cao, T. La Porta and W. Zhang. Sensor Relocation in Mobile Sensor Networks. *In Proceedings of IEEE INFOCOM,* March 2005.

[34] J. N. Al-Karaki, R. Ul-Mustafa and A. E. Kamal. Data aggregation in wireless sensor networks – exact and approximate algorithms. *In Proceedings of IEEE Workshop on High Performance Switching and Routing*, pp. 241–245, April 2004.

[35] F. Araújo and L. Rodrigues. On the Monitoring Period for Fault-Tolerant Sensor Networks. *In Proceedings of Latin-American Symposium on Dependable Computing,* pp. 174–190, 2005.

[36] Z. Jiang, J. Wu, A. Agah and B. Lu. Topology Control for Secured Coverage in Wireless Sensor Networks. *In Proceedings of 3rd IEEE Intl. Workshop on Wireless and Sensor Networks Security (WSNS'07),* October 2007.

[37] R. P. Liu, G. Rogers, S. Zhou and J. Zic. Topology control with Hexagonal Tessellation. *Intl. Journal of Sensor Networks* **2**(1/2), 91–98, 2007.

[38] R. P. Liu, G. Rogers and S. Zhou. Honeycomb Architecture for Energy Conservation in Wireless Sensor Networks. In *Proceedings of IEEE Globecom*, San Francisco, November 2006.

[39] Y. Wu, S. Fahmy and N. B. Shroff. Optimal QoS-Aware Sleep/Wake Scheduling for Time-Synchronized Sensor Networks. *Invited Sessions on Optimization of Communication Networks 40th Annual Conf. on Information Sciences and Systems (CISS),* Princeton, NJ, 2006.

[40] M. Bagheri and M. Hefeeda. Randomized k-Coverage Algorithms For Dense Sensor Networks. *in Proc. of IEEE INFOCOM 2007 Minisymposium,* Anchorage, AK, pp. 2376–2380, May 2007.

[41] M. Bagheri, M. Hefeeda and H. Ahmadi. A Near Optimal K-Coverage Algorithm for Large-scale Sensor Networks. Technical Report TR 2006-10, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, May 2006.

[42] B. Wang, K.C. Chua, V. Srinivasan and W. Wang. Scheduling Sensor Activity for Point Information Coverage in Wireless Sensor Networks. *In Proc. of WiOpt,* 2006.

[43] B. Wang, K.C. Chua, V. Srinivasan and W. Wang. Sensor Density for Complete Information Coverage in Wireless Sensor Networks. *In Proc. of EWSN 2006,* Zurich, Switzerland, February 2006.

[44] B. Liang, J. Frolik and X. S. Wang. A Predictive QoS Control Strategy for Wireless Sensor Networks. *The 1st Workshop on Resource Provisioning and Management in Sensor Networks (PRMSN 05) in Conjunction with the 2nd IEEE MASS,* Washington DC, November 2005.

[45] J. Frolik. QoS Control for Random Access Wireless Sensor Networks. *Wireless Communications and Networking Conf. (WCNC04),* Atlanta, March 2004.

[46] J. W. Branch, G. G. Chen and B. K. Szymanski. ESCORT: Energy-efficient Sensor Network Communal Routing Topology Using Signal Quality Metrics. *in Proc. of the 4th IEEE Intl. Conf. on Networking (IEEE ICN'05)*, Reunion Island, France, April 2005.

[47] Y. Cai, M. Li, W. Shu and M-Y. Wu. ACOS: An Area-based Collaborative Sleeping Protocol for Wireless Sensor Networks. *Intl. Journal of Ad Hoc & Sensor Wireless Networks*, 2006.

[48] Ch-f. Hsin and M. Liu. Network Coverage Using Low Duty-Cycled Sensors: Random & Coordinated Sleep Algorithms. *In Proc. of the 3rd Intl. Symposium on Information Processing in Sensor Networks (IPSN 2004)*, Berkeley, California, USA, April 26–27, 2004.

[49] J. Wu and Sh. Yang. Coverage Issue in Sensor Networks with Adjustable Ranges. *In Proc. of 33rd Intl. Conf. on Parallel Processing Workshops (ICPP 2004 Workshops)*, Montreal, Quebec, Canada, August 2004.

[50] M. Sigalas and G. Vouros. Poster Abstract: Energy Efficient Area Coverage in Arbitrary Sensor Networks. *In Proc. of EWSN 2006*, Zurich, Switzerland, February 2006.

[51] R. Zheng, G. He and X. Liu. Location-free Coverage Maintenance in Wireless Sensor Networks. Technical report UH-CS-05-15, Dept. of Computer Science, University of Houston, 2005.

[52] A. Giusti, A. L. Murphy and G. P. Picco. Decentralized Scattering of Wake-up Times in Wireless Sensor Networks. *In Proc. of the 4th European Conf. on Wireless Sensor Networks (EWSN)*, Delft, The Netherlands, Lecture Notes on Computer Science, Volume 4373. Springer, pp. 245–260, January 2007.

[53] H. Zhang and J. C. Hou. Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. *Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Network*, Chapter 28, pp. 453–475, Auerbach Publications, 2006.

[54] S. K. Das and W. Choi. Coverage-Adaptive Random Sensor Selection with Latency Control for Application-Specific. *In Proc. of the IEEE Intl. Conf. on Control and Automation (ICCA2005)*, Budapest, Hungray, June 2005.

[55] F. Ye, G. Zhong, J. Cheng, S. Lu and L. Zhang. PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks. In *Proceedings of the 23rd Intl. Conf. on Distributed Computing Systems (ICDCS'03)*, pp. 28–37, 2003.

[56] N. Bisnik and N. Jaggi. Randomized Scheduling Algorithms for Wireless Sensor Networks. Work in Progress, 2006.

[57] O. H. Sanli, H. Cam and X. Cheng. EQoS: An Energy Efficient QoS Protocol for Wireless Sensor Networks. *In Proc. of 2004 Western Simulation MultiConference*, San Diego, CA, January 2004.

[58] T. S. Rappaport. *Wireless Communications, Principles and Practice*. Prentice Hall, New Jersey, 1996.

[59] Kh. Th. Soe. Increasing Lifetime of Target Tracking Wireless Sensor Networks. *In Proceedings of World Academy of Science, Engineering and Technology*, pp. 2070–3740, August 2008.

[60] B. Jiang, B. Ravindran and H. Cho. Energy Efficient Sleep Scheduling in Sensor Networks for Multiple Target Tracking. *Lecture Notes in Computer Science*, pp. 498–509, 2008.

[61] L. Arienzo and M. Longo. An Energy-Efficient Strategy for Target Tracking through Wireless Sensor Networks. *Gruppo Telecomunicazioni Teoria dell'Informazione Conf.*, Florence, Italy, June 2008.

[62] X. Wang, J. J. Ma, L. Ding and D. W. Bi. Robust Forecasting for Energy Efficiency of Wireless Multimedia Sensor Networks. *Sensors Journals* **7**, 2779–2807, 2007.

[63] X. Wang, J. J. Ma, Sh. Wang and D. W. Bi. Cluster-based Dynamic Energy Management for Collaborative Target Tracking in Wireless Sensor Networks. *Sensors Journals* **7**, 1193–1215, 2007.

[64] Y. M. A. Khalifa, E. Okoene and M. B. Al-Mourad. Autonomous Intelligent Agent-based Tracking Systems. *ICGST-ACSE Journal* **7**(1), 21–31, May 2007.

[65] M. K. Watfa. An Energy Efficient Approach to Dynamic Coverage in Wireless Sensor Networks. *Journal of Networks* **1**(4), 10–20, August 2006.

[66] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu and G. Zhou. Achieving Real-Time Target Tracking Using Wireless Sensor Networks. *In Proc. of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*, April 2006.

[67] H. Yang and B. Sidkar. Lightweight Target Tracking Protocol Using Ad hoc Sensor Networks. *In Proceedings of IEEE VTC*, Stockholm, Sweden, May 2005.

[68] S. Bhattacharya, G. Xing, Ch. Lu, G.C. Roman, O. Chipara and B. Harris. Dynamic Wake-up and Topology Maintenance Protocol with Spatiotemporal Guarantees. *In Proceedings of 4th International Symposium on Information Processing in Sensor Networks*, pp. 28–34, 2005.

[69] W. Zhang and G. Cao. Optimizing Tree Reconfiguration for Mobile Target Tracking in Sensor Networks. *In Proceedings of Twenty-third Annual Joint Conf. of INFOCOM*, vol. 4, pp. 2434–2445, 2004.

[70] http://www.nytimes.com/2008/07/12/business/12newpark.html

[71] A. Sobeih, W. P. Chen, J. C. Hou, L. C. Kung, N. Li, H. Lim, H. Y. Tyan and H. Zhang. J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks. *IEEE Wireless Communications* **13**(4), 104–119, August 2006.

[72] L. Wang and Y. Ziao. A Survey of Energy-Efficient Scheduling Mechanisms in Sensor Networks. *Mobile Networks and Applications* **11**(5), 723–740, 2006.

[73] C. Bettstetter, H. Hartenstein and X. Pérez-Costa. Stochastic Properties of the Random Waypoint Mobility Model. *ACM/Kluwer Wireless Networks: Special Issue on Modeling and Analysis of Mobile Networks* **10**(5), 555–567, September 2004.

[74] E. Fredkin. Digital machine: An informational process based on reversible cellular automata. *Physica* **D45**, 245–270, 1990.

[75] M. A. L. Thathachar and P. S. Sastry. Varieties of Learning Automata: An Overview. *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics* **32**(6), 711–722, 2002.

[76] K. S. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction*. Prentice-Hall Inc, 1989.

[77] M. R. Kharazmi and M. R. Meybodi. Image Segmentation Using Cellular Learning Automata. *In Proc. of 10th Iranian Conf. on Electrical Engineering, ICEE-96*, Tabriz, Iran, May 2001.

[78] M. R. Kharazmi and 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.

[79] M. R. Meybodi, H. Beygi and M. Taherkhani. Cellular Learning Automata and its Applications. *Journal of Science Tech.*, Sharif (Sharif University of Technology, Tehran, Iran), pp. 54–77, 2004.

[80] M. R. Meybodi and 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.

[81] M. R. Meybodi and 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.

[82] H. Beigy and 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.

[83] H. Beigy and 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.

[84] H. Beigy and M. R. Meybodi. A Mathematical Framework for Cellular Learning Automata. *Advances in Complex Systems***7**(3 & 4), 295–319, September & December 2004.

[85] H. Beigy and M. R. Meybodi. Open Synchronous Cellular Learning Automata. *Advances in Complex Systems* **10**(4), 1–30, December 2007.

[86] H. Beigy and M. R. Meybodi. Asynchronous Cellular Learning Automata. *Automatica* **44**(5), 1350–1357, May 2008.

[87] M. Esnaashari and M. R. Meybodi. A Cellular Learning Automata based Clustering Algorithm for Wireless Sensor Networks. *Sensor Letters* **6**(5), 723–735, December 2008.

[88] S. Meguerdichian, F. Koushanfar, M. Potkonjak and M. B. Srivastava. Coverage Problems in Wireless Ad-Hoc Sensor Networks. *In Proc. of the IEEE INFOCOM '01, the Conference on Computer Communications*, vol. 3, pp. 1380–1387, April 2001.

[89] F. Ye, S. Lu and L. Zhang. A Randomized Energy-Conservation Protocol for Resilient Sensor Networks. *Wireless Networks* **12**(5), 637–652, October 2006.