Abstract—Redundant node deployment is a common strategy in wireless sensor networks. This redundancy can be due to various reasons such as high probability of failures, long lifetime expectation, etc. One major problem in wireless sensor networks is to use this redundancy in order to extend the network lifetime while keeping the entire area under the coverage of the network. In this problem, which is known as set cover problem, the main objective is to select a subset of sensor nodes as active nodes so that the set of active nodes covers the entire area of the network. In this paper, an scheduling algorithm is presented for solving the set cover problem using cellular learning automata. In this algorithm, each node is equipped with a learning automaton which locally decides for the node to be active or not based on the situations of its neighbors. Simulation results in J-sim simulator environment specify the efficiency of the proposed scheduling algorithm over existing algorithms such as PEAS and PECAS.

Index Terms—Area coverage, cellular learning automata, learning automata, scheduling algorithm, wireless sensor networks.

I. INTRODUCTION

One of the basic issues in wireless sensor networks is the coverage problem which specifies how the network is monitored by sensor nodes [1]. The focus of this paper is on a sub-problem of coverage problem, called the set cover problem. In set cover problem, by assuming redundancy in the number of deployed sensor nodes throughout the area of the network, the main objective is to select a subset of sensor nodes as active nodes so that the set of active nodes covers the entire area of the network.

Selecting a subset of sensor nodes as active nodes is a suitable approach for prolonging the network lifetime. This is because in most scenarios of sensor networks, sensor nodes have limited batteries, which are not rechargeable. Hence, deactivating a number of sensor nodes which lets them conserve their batteries for later times, prolongs the lifetime of the network. Furthermore, deactivating some of the sensor nodes decreases the probability of collisions, which in turn, decreases the need for retransmitting packets. In this paper, we refer to any algorithm which can select a subset of sensor nodes in such a way that the entire area of the network is covered as scheduling algorithm.

In this paper, a distributed, adaptive scheduling algorithm based on cellular learning automata for wireless sensor networks is proposed. The main purpose of this algorithm is to maintain coverage in the network with minimum number of active nodes, so that the total consumed energy of nodes is minimized. In the proposed algorithm, each node is equipped with a learning automaton which decides for the node to be active or not based on the states (either active or not) of its neighboring sensor nodes.

We used J-sim simulator to evaluate the performance of the proposed algorithm. Simulation results specify the efficiency of the proposed algorithm over existing algorithms such as PEAS [2] and PECAS [3]—especially against high ratio of unexpected failures.

The rest of this paper is organized as follows. In Section II, a literature overview is presented. In Section III, learning automata and cellular learning automata are briefly reviewed. The problem statement is given in Section IV. The proposed scheduling algorithm is described in Section V. Simulation results are given in Section VI. Section VII is the conclusion.

Manuscript received May 14, 2012; revised August 14, 2012.

R. Ghaderi is with the Computer Engineering Department, Islamic Azad University, Arak 38135 Iran (e-mail: Ghaderi.re@gmail.com).

M. Esnaashari and M. R. Meybodi are with the Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran 15914 Iran (e-mail: Esnaashari@aut.ac.ir; mmeybodi@aut.ac.ir).
it.

### III. CELLULAR LEARNING AUTOMATA

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

#### A. 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 responds 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. A class of LA is called variable structure learning automata and is represented by quadruple \( \{\alpha, \beta, p, T\} \) in which \( \alpha = \{\alpha_0, \alpha_1, \ldots, \alpha_n\} \) represents the action set of the automata, \( \beta = \{\beta_0, \beta_1, \ldots, \beta_m\} \) represents the input set, \( p = \{p_0, p_1, \ldots, p_n\} \) represents the action probability set, and finally \( p(n+1) = T(\alpha(n), \beta(n), p(n)) \) represents the learning algorithm. Let \( \alpha_i \) be the action chosen at time \( n \). Then, the recurrence equation for updating \( p \) is defined as

\[
p_i(n+1) = p_i(n) + a[1 - p_i(n)] \\
p_i(n+1) = (1 - a)p_i(n) 
\]

\( \forall j, j \neq i \) (1)

for favorable responses, and

\[
p_i(n+1) = (1 - b)p_i(n) \\
p_i(n+1) = \frac{b}{r - 1} + (1 - b)p_i(n) 
\]

\( \forall j, j \neq i \) (2)

for unfavorable ones. In these equations, \( a \) and \( b \) are reward and penalty parameters respectively. If \( a = b \), learning algorithm is called \( L_{a,b}^{-1} \); if \( b \ll a \), it is called \( L_{a,b}^{-2} \), and if \( b = 0 \), it is called \( L_{a,b}^{-3} \). For more information about learning automata, the reader may refer to [8], [9].

#### B. Cellular Learning Automata

Cellular Learning Automata (CLA), which is a combination of Cellular Automata (CA) [10] and learning automata, is a powerful mathematical model for many decentralized problems and phenomena. The basic idea of CLA, which is a subclass of stochastic CA, is to utilize LA to adjust the state transition probability of stochastic CA. A CLA is a CA in which a learning automaton is assigned to every cell. The learning automaton residing in a particular cell determines its action (state) on the basis of its action probability vector. Like CA, there is a rule that the CLA operates under. The local rule of CLA and the actions selected by the neighboring LAs of any particular LA determine the reinforcement signal to the LA residing in a cell. CLA has found many applications such as image processing [11], rumor diffusion [12], channel assignment in cellular networks [13], and sensor networks [14], [15] to mention a few. For more information about CLA, the reader may refer to [12], [16].

### IV. PROBLEM STATEMENT

Consider a sensor network consists of \( N \) sensor nodes \( s_1, s_2, \ldots, s_N \) within an area \( \Omega \). Sensor nodes, which are responsible for sensing and monitoring the area, are scattered randomly throughout the area of the network so that \( \Omega \) is completely covered. All sensor nodes have the same sensing range of \( R \) and transmission range of \( R_t \). Each sensor node \( s_i \) has 4 different modes of operation [17] as follows:

- **On-duty** (CPU, Sensing, Communicating): CPU, sensing and communicating units are switched on referred to as active mode. A sensor node in the active mode is referred to as an active node.
- **Sensing unit on-duty** (CPU, Sensing, Communicating): The CPU and the sensing units are switched on, but the communicating unit is switched off.
- **Communicating unit on-duty** (CPU, Sensing, Communicating): The CPU and the communicating units are switched on, but the sensing unit is switched off.
- **Sleep mode** (CPU, Sensing, Communicating): CPU, sensing and communicating units are switched off referred to as sleep mode.

Note that in \( CPU, Sensing, Communicating \), and \( CPU, Sensing \), index \( A \) stands for active and index \( S \) stands for sleep. At any instance of time, a sensor node can be only in one of the above 4 operation modes.

We assume that the number of sensors (\( N \)) is greater than the one required to cover the entire area of the network. Thus, a scheduling algorithm can be adopted to use this redundancy for prolonging the lifetime of the network. Such a controlling algorithm, selects a subset of sensor nodes as active nodes so that the set of active nodes fully covers \( \Omega \).

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

The above definitions and assumptions having been considered, the problem is to design a scheduling algorithm, which tries to maximize the lifetime of the network.

### V. PROPOSED SCHEDULING ALGORITHM

The proposed scheduling algorithm must provide two requirements: 1. complete coverage of the network area, and 2. the set of the selected active nodes should consume as little power as possible so as to prolong the network lifetime. We consider the following assumptions:

- The set of sensor nodes residing throughout the network area covers the area completely.
- Sensor nodes are aware of their physical locations (using some localization techniques [18], [19]).
- \( N \) which is the number of sensor nodes within the area of the network is known by all nodes.
In what follows, we give the details of the proposed algorithm.

A. Detailed Description of the Proposed Scheduling Algorithm

Initially, the network graph is mapped into a cellular learning automaton. In this mapping, each sensor node \( s_i \) in the network is mapped into a cell \( i \) in CLA, and two nodes \( i \) and \( j \) are adjacent in CLA if their corresponding sensor nodes are located within the sensing ranges of each other, or in other words, two cells are adjacent if the distance between their corresponding sensor nodes is not more than \( R_s \) given by following equation:

\[
R_s = 2R_s - \varepsilon \equiv 2R_s
\]  

We refer to \( R_s \) as neighborhood radius hereafter.

The learning automaton in each cell \( i \) of CLA, referred to as \( LA_i \), uses linear learning algorithm given by (1) and (2). All learning automata are \( L_{\alpha}. \) Each \( LA_i \) has two actions \( \alpha_0 \) and \( \alpha_1. \) \( \alpha_0 \) is "plan to be in active mode", and \( \alpha_1 \) is "plan to be in sleep mode". The probability of selecting each of these actions is initially computed according to (4).

\[
p_0 = \frac{N_i}{N}\quad p_1 = 1 - p_0
\]  

In this equation, \( N_i \) is a constant which is greater than the minimum number of active nodes required to cover the entire area of the network. \( p_1 \) is selected this way in order to have a suitable initial distribution (in terms of area coverage) of active nodes throughout the network area.

Neighboring nodes exchange HELLO packets with each other periodically to be aware of the states of each other. Details on how to compute the time interval between two consecutive transmissions of HELLO packets in each sensor node will be given in Subsection V-A-1. The HELLO packet of a node contains its id and physical location. Using received HELLO packets, each sensor node \( s_i \) stores for any of its active neighboring nodes, the location and the time of the last HELLO packets, each sensor node \( s_i \) of a node contains its id and physical location. Using received HELLO packets (that its monitored area is covered) HELLO packets within \( \tau_0 \) duration, do not guarantee that the monitored area of \( s_i \) is covered by its neighbors, then it penalizes its selected action. Otherwise, it rewards its selected action.

If \( s_i \) is in "WAIT_FOR_SLEEP" state and its received HELLO packets within \( \tau_0 \) duration, do not guarantee that the monitored area of \( s_i \) is covered by its neighbors, then it penalizes its selected action. Otherwise, \( s_i \) rewards its selected action.

Each working sensor node guarantees not to change its state for a predetermined duration \( T \) starting from the time it dispatches its HELLO packet. As a result, if a node, which is in "WAIT_FOR_SLEEP" state, is ensured (through received HELLO packets) that its monitored area is covered completely by its active neighboring nodes within \( \tau_0 \) duration, then it switches to "SLEEP" state. In "SLEEP" state, all nodes are in \( CPU_sS_5C_5 \) operation mode. Sleeping time is determined by considering the value of \( T \) and the receipt times of the last HELLO packets from active neighbors. Let \( t \) be the current time of the network and \( t_i^k \) be the receipt time of the last HELLO packet from active neighbor \( s_j \) at node \( s_i \). The sleeping time of the node \( s_i \) \( (T_i^{sleep}) \) is then computed according to the following equation:

\[
T_i^{sleep} = \min_{s_j \in \text{active neighbors of } s_i} \left( T - (t - t_i^k) \right)
\]  

When the sleeping time of the node \( s_i \) is over, node \( s_i \) changes its operation mode to \( CPU_sS_5C_4 \) and its state to "WAIT_FOR_SLEEP".

Next time for action selection of the learning automaton of a node \( s_i \) is determined as follows:

- If node \( s_i \) is in "WAIT_FOR_SLEEP" state and action \( \alpha_0 \) is penalized, next action selection time is immediately after receiving the environment response.
- If node \( s_i \) is in "WORK" state and action \( \alpha_0 \) is penalized, next action selection time is at the dispatching time of the next HELLO packet. If the selected action of such a node is \( \alpha_1 \), then \( s_i \) enters "WAIT_FOR_SLEEP" state, but due to the assurance of this node to its neighbors, \( s_i \) must stay in active mode and perform all of the tasks of a sensor node in "WORK" state, except for sending HELLO packets, until the end of the assurance time.
- If the selected action of node \( s_i \) is rewarded, then the selection of the node will be given in Subsection V-A-1.
next action selection time is when this action is penalized. In other words, the rewarded action is performed repeatedly until the time it receives penalty.

In Fig. 1, the transition diagram of operation mode/state of the proposed scheduling algorithm in each sensor node has been shown. Fig. 2 shows the pseudo code of the proposed algorithm.

For each node $s_i$ corresponding with cell $i$ in CLA do in parallel
Initialize
WakeupAfter(Random(0,$\tau_i$))
Select an action according to (4)
If (the action is $\alpha_i$) then
  ChangeOperation_Mode/StateTo( $CPU_i$,$S_i$,$C_i$ / “WORK”)
else /* the action is $\alpha_i$ */
  ChangeOperation_Mode/StateTo( $CPU_i$,$S_i$,$C_i$ / “WAIT_FOR_SLEEP”)
End if
While (Not SensorDead) do
  While (the node is in “WAIT_FOR_SLEEP” State) do
    If (MON Assured by Neighbors) then
      If (monitored area of the node is covered by neighbors) then
        ChangeOperation_Mode/StateTo( $CPU_i$,$S_i$,$C_i$ / “SLEEP”)
      End if
    End if
  End while
  While (the node is in “SLEEP” State) do
    Broadcast HELLO Packet
    Wait for HELLO packets from neighbors
    If (the node receives at least one HELLO packet within the time interval $[t-\tau_i,t]$) then
      No $s_i$ = No $s_i$ + 1
      End if
    Update LA’s probability vector
  End while
End while
End For

Fig. 2. Pseudo code of the proposed algorithm

1) Time Interval for Transmission of HELLO Packets
As we stated before, sensor nodes which are in “WORK” state, periodically broadcast HELLO packets. Initially, the time interval between two consecutive transmissions of HELLO packets for each sensor node $s_i$ is set to $\tau_i = \tau_0$. Immediately after sending the $k$th ($k > 1$) HELLO packet, each node $s_i$ updates $\tau_i$ according to (6).

$$\tau_i = \begin{cases} 
\tau_0 ; & \text{No}_i = 0 \\
\frac{\text{No}_i}{N} \times \tau_0 ; & \text{Otherwise}
\end{cases} \quad (6)$$

In the above equation, $\text{No}_i$ is the number of working nodes from which $s_i$ receives at least one HELLO packet within the time interval $[t-\tau_i,t]$. Since $\text{No}_i$ is a measure of the traffic load in the neighborhood of $s_i$, (6) makes the time interval between two consecutive transmissions of HELLO packets in each sensor node adaptable to the local traffic load in the neighborhood of that node.

VI. PERFORMANCE EVALUATION

A. Simulation Setup
To evaluate the performance of the proposed scheduling algorithm, we conduct a number of experiments. The results obtained from the proposed algorithm are compared with the results obtained from PEAS [2] and PECAS [3]. Experiments are simulated using J-sim simulator [20]. We use a network area of 50m×50m through which a number of sensor nodes are scattered uniformly at random.

Energy model given in [7] is used in which the energy consumption ratios for transmission, reception (idle) and sleep modes are 20:4:0.01 respectively. The initial energy level of each sensor node is chosen randomly from the range of 28 to 35 Jules. Sensing and transmission ranges of each sensor node are 17m and 30m, respectively. Therefore, neighborhood radius according to (3) is 34m. Like PEAS [2], each node has a raw wireless communication capacity of 20Kbps. Simulation time for the first two experiments is assumed to be 1000s. Results are averaged over 25 runs of simulations.

1) Parameters of the Proposed Algorithm
We consider the value of parameter $T$ as 10s and rate of reward and penalty parameters as 0.75 so that the sensor nodes have rapid reactions to topology changes of the network (due to unexpected failures, energy exhaustion of sensor nodes, ...). The values for the parameters in the proposed algorithm are listed in Table I.

| TABLE I: PARAMETER VALUES USED IN THE PROPOSED ALGORITHM |
|---|---|---|---|---|
| $R_s$ | $R_t$ | $N_i$ | Initial energy of each sensor node | Channel capacity |
| 17m | 30m | 25 | 28 to 35 Jules | 20Kbps |
| $T$ | $\tau_s$ | $a, b$ |
| 10s | 1.27s | 0.75 |

2) Evaluation Metrics
To evaluate the performance of the proposed algorithm, we use the following metrics: (i) number of active nodes; (ii) percentage of area under the coverage of the network; (iii) total consumed energy of sensor nodes; and (iv) network lifetime.

B. Simulation Results

1) Experiment 1
This experiment is conducted to study the performance of the proposed algorithm in terms of number of active nodes, percentage of area under the coverage of the network, and
total consumed energy of sensor nodes in comparison to PEAS and PECAS algorithms. Experiment is repeated for 50, 100, 200, 400, and 500 sensor nodes.

Fig. 3 shows the results of the comparison for the 'number of active nodes' metric. It can be seen from this figure that the number of active nodes in the proposed algorithm, unlike PEAS and PECAS algorithms, does not depend heavily on the number of sensor nodes in the network.

Fig. 3. Number of active nodes versus number of sensor nodes

Fig. 4 presents the results of the comparison between the proposed algorithm, PEAS, and PECAS in terms of percentage of network area under coverage. As it can be seen from the figure, the proposed algorithm fully covers the network area even in the networks of small sizes (networks in which the number of sensor nodes is below 100). Fig. 5 compares the proposed algorithm with PEAS and PECAS algorithms in terms of the total energy consumption of sensor nodes. It can be seen from this figure that the proposed algorithm consumes less than 70% and 45% of the energy consumed by PEAS and PECAS, respectively.

2) Experiment 2

In this experiment, we study the tolerability of the proposed algorithm against unexpected failures of sensor nodes in comparison to PEAS and PECAS algorithms. To simulate unexpected failures of sensor nodes, in this experiment, 50% of the sensor nodes experience failures in random times during the operation of the network. Similar to Experiment 1, networks with 50, 100, 200, 400, and 500 sensor nodes are considered for this experiment.

Fig. 5. Total energy consumption versus number of sensor nodes

Fig. 6, which compares the proposed algorithm with PEAS and PECAS in terms of average number of active nodes, indicates that the presence of failures does not affect the performance of the proposed algorithm in terms of this metric significantly. Number of active nodes in the proposed algorithm is less than 49% and 40% of the number of active nodes in PEAS and PECAS algorithms respectively. According to Fig. 7, which gives the comparison of the mentioned algorithms in terms of percentage of network area under coverage, the proposed algorithm provides full coverage of network area in all of mentioned network sizes even in presence of unexpected node failures.

Finally, Fig. 8 gives the comparison of the mentioned algorithms in terms of the total energy consumption of sensor nodes. This figure shows that in the presence of unexpected failures, the energy consumption of the proposed algorithm is less than 69% and 54% of the energy consumption of PEAS and PECAS, respectively.

3) Experiment 3

This experiment is conducted to study the performance of the proposed algorithm in terms of network lifetime. For this experiment, number of sensor nodes is assumed to be 250. The injection of unexpected failures of sensor nodes is
performed with a rate of 4.17 failures in every 1000s. The results of simulations show that the network lifetime for PEAS, PECAS and the proposed algorithm is 9000s, 12000s and 22000s, respectively. This indicates that the proposed algorithm better prolongs the lifetime of the network in comparison to existing algorithms.

![Image](https://example.com/image.png)

**Fig. 7. Percent of coverage versus number of sensor nodes with failures**

**Fig. 8. Total energy consumption versus number of sensor nodes with failures**

VI. CONCLUSION

In this paper, a novel scheduling algorithm based on cellular learning automata for solving the set cover problem in a wireless sensor network was presented. This algorithm tries to select a minimum number of active nodes, which can fully cover the entire area of the network, so that the energy consumption of sensor nodes on average is minimized, and consequently the network lifetime is maximized.

The results of simulations proved the superiority of the proposed algorithm to similar existing algorithms like PEAS and PECAS in terms of number of active nodes, energy consumption of sensor nodes, percentage of network area under coverage, and network lifetime. The results of experiments also showed that the proposed algorithm outperforms PEAS and PECAS in terms of the above mentioned metrics even in the presence of unexpected failures in the sensor nodes of the network.

REFERENCES


Reza Ghaderi received the B.S. degree in Computer Engineering from Payam-e-Noor University, Iran in 2005, and the M.S. degree in Computer Engineering from Islamic Azad University, Iran in 2009. He is a member of IACSIT. His research interests include computer networks, learning systems, soft computing, and parallel algorithms.
Mehdi Esnaashari received the B.S., M.S., and Ph.D. degrees in Computer Engineering from Amirkabir University of Technology, Iran in 2002, 2005, and 2011, respectively. His research interests include computer networks, learning systems, and soft computing.

Mohammad Reza Meybodi received the B.S. and M.S. degrees in Economics from Shahid Beheshti University, Iran in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degrees from Oklahoma University, USA in 1980 and 1983, respectively, in Computer Science. Currently, he is a Full Professor in Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at Western Michigan University, and from 1985 to 1991 as an Associate Professor at Ohio University, USA.

His research interests include channel management in cellular networks, learning systems, parallel algorithms, soft computing, and software development.