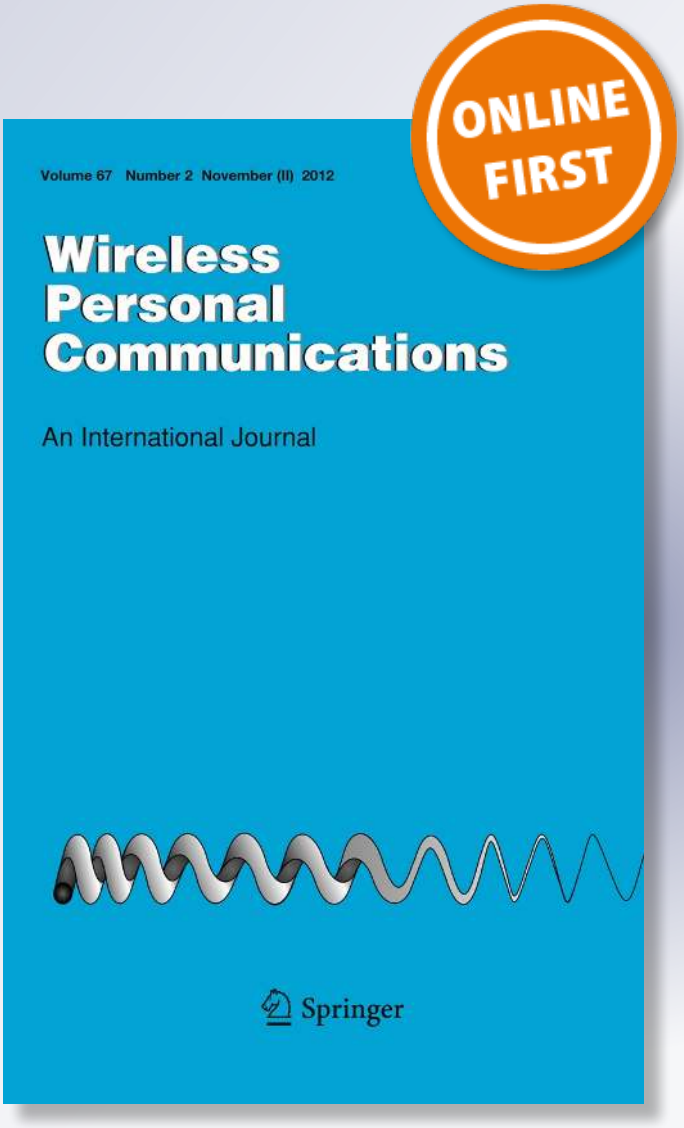
Maximizing Lifetime of Target Coverage in Wireless Sensor Networks Using Learning Automata

Habib Mostafaei & Mohammad Reza Meybodi

Wireless Personal Communications
An International Journal

ISSN 0929-6212

Wireless Pers Commun
DOI 10.1007/s11277-012-0885-y



Springer

Your article is protected by copyright and all rights are held exclusively by Springer Science +Business Media New York. This e-offprint is for personal use only and shall not be self-archived in electronic repositories. If you wish to self-archive your work, please use the accepted author's version for posting to your own website or your institution's repository. You may further deposit the accepted author's version on a funder's repository at a funder's request, provided it is not made publicly available until 12 months after publication.

Author's personal copy

Wireless Pers Commun
DOI 10.1007/s11277-012-0885-y

Maximizing Lifetime of Target Coverage in Wireless Sensor Networks Using Learning Automata

Habib Mostafaei · Mohammad Reza Meybodi

© Springer Science+Business Media New York 2012

Abstract In wireless sensor networks, when each target is covered by multiple sensors, we can schedule sensor nodes to monitor deployed targets in order to improve lifetime of network. In this paper, we propose an efficient scheduling method based on learning automata, in which each node is equipped with a learning automaton, which helps the node to select its proper state (active or sleep), at any given time. To study the performance of the proposed method, computer simulations are conducted. Results of these simulations show that the proposed scheduling method can better prolong the lifetime of the network in comparison to similar existing methods.

Keywords Wireless sensor network · Energy efficiency · Sensor scheduling · Maximum disjoint set covers · Learning automata (LA)

1 Introduction

Recent improvements in affordable and efficient integrated electronic devices have a considerable impact on advancing the state of wireless sensor networks, which constitute the platform of a broad range of applications including national security, surveillance, health care and environmental monitoring. Sensor nodes are small devices equipped with one or more sensors, one or more transceivers, processing, storage resources and possible actuators [1].

H. Mostafaei (✉)
Department of Computer Engineering, Urmia Branch,
Islamic Azad University, Urmia, Iran
e-mail: h.mostafaei@iaurmia.ac.ir

M. R. Meybodi
Computer Engineering and Information Technology Department,
Amirkabir University of Technology, Tehran, Iran
e-mail: mmeybodi@aut.ac.ir

Published online: 25 October 2012

Author's personal copy

Energy efficiency remains a critical issue in wireless sensor networks as long as sensor nodes are powered by battery. In the past few years, much research work has been done on making efficient use of battery energy toward a longer network lifetime. Among many others, energy aware routing, energy efficient data dissemination and aggregation, transmission power control and node activity scheduling are the common approaches to improve energy efficiency. A duty cycle is therefore introduced to allow each sensor to switch between active and sleep modes to save energy. On the other hand, a certain amount of active nodes should be present to ensure a desired level of coverage at all times. The method to rotate the role of each sensor to meet certain objectives is called scheduling, where nodes alternate between active and sleeping modes.

One major problem in the area of sensor networks is the coverage problem. This problem deals with the ability of the network to cover a certain area or some certain events. Coverage problem is classified into three different types [16]:

- Area coverage: covering (monitoring) the whole area of the network is the main objective of area coverage problem.
- Point coverage (target): the objective of point coverage problem is to cover a set of stationary or moving points.
- 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 target cover-age. A common definition of this problem is to cover (monitor) some stationary or moving target points in the area of sensor network using as few sensor nodes as possible [16].

Sensor networks are usually redundantly deployed, i.e., each target is covered by more than one sensor. Sensor nodes are often scheduled to turn on and off, alternating between an active working mode and a sleep mode. The motivation for this scheme is not just to turn off redundant sensors to save energy. Research shows that if batteries are given sufficient recovering period between two intensive consumption periods, the actual battery lifetime is extended [24]. Therefore appropriate scheduling will not only improve sensor network lifetime, but also individual battery's performance.

In this paper, we propose a learning automata-based scheduling mechanism for target coverage applications. We assume that a large number of sensor nodes are dispersed randomly in close proximity of a set of targets and the objective of the scheduling mechanism is to select a subset of sensor nodes as active nodes, which can cover all of the targets. In the proposed scheduling mechanism, each sensor node is equipped with a learning automaton, which helps the node to select its proper state (active or sleep) at any time during the operation of the network.

The rest of the paper is organized as follows. In Sect. 2, we present energy efficient and coverage related work. Section 3 describes the target coverage problem. Learning automata (LA) as a basic learning strategy used in the proposed method will be discussed in Sect. 4. In Sect. 5, the proposed method is presented. Sect. 6 presents the simulation results and Sect. 7 concludes the paper.

2 Related Work

Wang [29] provided a good resource for various coverage control problems in sensor networks, a hot topic that has been intensively researched in recent years. Due to some unique characteristics of sensor networks such as energy constraint and adhoc topology, the coverage

Author's personal copy

problems in sensor networks have many new scenarios and features that entitle them an important research issue in recent years.

In sensor coverage problems, the goal is to have each location in the physical space of interest within the sensing range of at least one sensor. Cardei and Wu [8] survey recent sensor coverage problems proposed in literature and categorize them according to the following design criteria: objective of the problem: maximize network lifetime or minimize the number of sensors deployed sensor deployment method: deterministic versus random

In [12] authors categorized scheduling algorithms that 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 [16]. Usually, a minimum number of nodes are selected to be active and monitor the area or the target points while the rests of the nodes are inactive and save energy. This number of selected nodes is called cover set. The node selection is repeated periodically or based on a certain schedule to allow balance energy consumption of all nodes. A number of centralized [9,34,22,32,3,2,30,31] and decentralized [19,20,10,18,17,13,4,5] methods are given in the 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 and periodically 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. Necessity for becoming an active node can be determined using a learning scheme [19,20].

Authors proposed energy efficient centralized mechanisms by dividing the sensor nodes into disjoint sets, such that every set can individually perform the coverage tasks [6,15]. These sets are then activated successively, and while the cur-rent sensor set is active, all other nodes are in the sleep mode. The goal of this approach is to determine a maximum number of disjoint sets, as this has a direct impact on conserving sensor energy resources as well as on prolonging the network lifetime. Cardei and Du [6] address the target coverage problem where disjoint sensor sets are modeled as disjoint set covers, such that every cover completely monitors all the target points. Disjoint set coverage problem is proved to be NP-complete, and a lower approximation bound of 2 for any polynomial-time approximation algorithm is indicated. The disjoint set cover problem [6] is reduced to a maximum flow problem, which is then modeled as a mixed integer programming.

In [7] the authors abstract the objective of maximizing the network lifetime under maximum set cover (MSC) problem. MSC is proven to be NP-complete by a polynomial transformation from a well-known NP-complete problem. Authors propose two heuristics and Greedy method to compute maximum number of set covers.

Slijepcevic and Potkonjak [15] address the area coverage problem where the area is modeled as a collection of fields, where every field has the property that any enclosed point is covered by the same set of sensors. The most-constrained least-constraining algorithm [15] computes the disjoint covers successively, selecting sensors that cover the critical element (field covered by a minimal number of sensors), giving priority to sensors that: cover a high number of uncovered fields, cover sparsely covered fields and do not cover fields redundantly.

If faulty nodes exist in a sensor network, single coverage is not sufficient to satisfy the QoS requirements. In [26], the k-coverage problem is addressed, i.e., to select a minimal active set of sensor nodes to maintain a complete area k-coverage, which is defined as a minimum set cover problem. It further extends it to address the probabilistic k-coverage problem, which requires a point is covered by at least k sensors with a required probability.

Author's personal copy

Moving target detection is a different category of coverage problem. In Megerian et al. [28], defined the worst and best-case coverage problems and proposed polynomial time algorithms to compute them. The coverage calculation here is independent of paths traveled by the target, which is different from [15]. Esnaashari and Meybodi in [11] proposed a learning automata-based scheduling solution to the dynamic point coverage problem. In the proposed scheduling algorithm, the learning automaton of each node learns the sleep duration of that node based on the movement pattern of a single moving target point in the network. Optimization techniques have been used to improve the performance of communication networks since a very early stage [25] provided a collection of problems and optimization techniques for telecommunication.

3 Problem Statement

In this section, we first state the target coverage problem in wireless sensor networks and then restate this problem as a disjoint set coverage problem.

Consider a sensor network of n sensor nodes s1, s2, . . . , sn, which are randomly deployed within an LxL rectangular area Ω. Furthermore, consider m targets $r_1, r_2, \ldots r_m$ with fixed locations within Ω which must be continuously monitored (covered) by the sensor network. We assume that the number of sensors deployed in Ω is greater than that required for monitoring the target points. Thus, a scheduling mechanism can be applied to alternate the activity status of sensor nodes between active and sleep states.

Definition 1 (*Target coverage problem* (*TCP*)) Given m targets and a wireless sensor network with n sensors, randomly deployed in Ω, schedule the sensor nodes activity such that all the targets are continuously observed and network lifetime is maximized [7].

In this paper we focus on designing a node scheduling mechanism, and do not address the problem of selecting which protocol is used for data gathering or node synchronization. To efficiently transmit data from the sensors to the BS, a mechanism like LEACH [14] can be used [7].

3.1 Disjoint Set Covers Problem

In this section we restate the target coverage problem as a disjoint set covers problem (DSC). Let us first consider a simple example of target coverage. As illustrated in Fig. 1a, there are six sensors and four targets in a randomly deployed network. If we consider a disk coverage model, then the targets $z1$ and $z4$ are each covered by two sensors; and the targets $z1$ and $z3$ are each covered by three sensors. We use *coverage mapping* to refer to the coverage relations among all sensors and all targets that can be represented by a *sensor-target bipartite graph*: The vertices are the sensors and targets, and an edge exists between a sensor and a target if the sensor covers the target. Figure 1b plots the sensor-target bipartite graph of the example sensor network.

In the context of target coverage, it is a common prerequisite that all targets can be covered if all sensors are activated for sensing. However, activating all the sensors at the same time is not energy efficient. If every sensor can only operate for one time unit in a continuously active state, then activating all sensors all the time results in a total network lifetime of also one time unit. Instead, we can alternatively activate sensors. For example, in the network shown in Figure 1, we can activate C1 = {s1, s3, s6} for one time unit and C2 = {s2, s4, s5} for another time unit. Since all targets are still covered by either C1 or C2, the coverage requirements

Author's personal copy

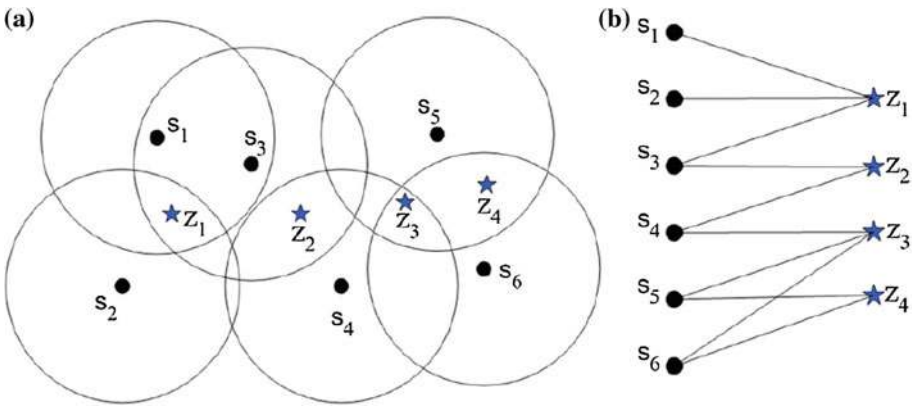


Fig. 1 Illustration of **a** a randomly deployed sensor network for covering targets and **b** the corresponding sensor-target bipartite graph

are not sacrificed. Furthermore, the target coverage lifetime can be extended to two time units. Obviously, we can have other choices of partitioning the sensors into different subsets, such as C3 = {s1, s2, s5} and C4 = {s3, s6}. The objective of the target coverage problem is to find the optimal subsets and their active intervals such that the coverage requirements can be satisfied and the total target coverage lifetime can be maximized. Note that a circular sensing area (like the one used in Fig. 1) is not a requirement here; the only requirement is that each sensor be able to identify the set of targets it can cover. It has been proved that DSC is NP-complete [7].

4 Learning Automata

An '*automaton*' is a self-operating machine or a mechanism that responds to a sequence of instructions in a certain way, so as to achieve a certain goal. The automaton either responds to a pre-determined set of rules, or adapts to the environmental dynamics in which it operates. The term '*learning*' refers to the act of acquiring knowledge and modifying one's behavior based on the experience gained. Thus, in our case, the adaptive automaton we study in this paper, adapts to the responses from the environment through a series of interactions within it. It, then, attempts to learn the best action from a set of possible actions that are offered to it by the random stationary or non-stationary environment in which it operates. The automaton, thus, acts as a decision maker to arrive at the best action.

The operation of LA can be best described through the words of the pioneers Narendra and Thathachar [21]: '…a decision maker operates in the random environment (RE) and updates its strategy for choosing actions on the basis of the elicited response. The decision maker, in such a feedback configuration of decision maker (or automaton) and environment, is referred to as the *learning automaton*. The automaton has a finite set of actions, and corresponding to each action; the response of the environment can be either favorable or unfavorable with a certain probability' (Ref. [21, p. 3]).

LA finds applications in optimization problems in which an optimal action needs to be determined from a set of actions. It should be noted that in this context, learning might be of best help only when there are high levels of *uncertainty* in the system in which the automaton

Author's personal copy

operates. In systems with low levels of uncertainty, LA-based learning may not be a suitable tool of choice [21].

A comprehensive overview of research in the field of LA can be found in the classic text by Narendra and Thathachar [21], and in the recent special issue of the *IEEE Transactions on Systems, Man, and Cybernetics, Part B* [27]. However, to ease out the understanding of the philosophy underlying our solution approach, we briefly review below some of the fundamental concepts.

4.1 The Automaton

The Automaton in our case is, typically, defined by a quintuple $\{A, B, Q, F(.,.), G(.)\}$, where [21]:

(i) $\mathrm{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of outputs or actions, and $\alpha(\mathrm{t})$ is the action chosen by the automaton at any instant t.
(ii) B is the set of inputs to the automaton,. $\{\beta_1, \beta_2, \ldots, \beta_r\}$. $\beta(t)$ is the input at any instant t, while the set B can be finite or infinite.
(iii) $\mathrm{Q} = \{q_1(t), q_2(t), \ldots, q_s(t)\}$ is the set of finite states, where $q(t)$ denotes the state of the automaton at any instant t.
(iv) $F(.,.) : Q \times B \rightarrow Q$ is a mapping in terms of the state and input at the instant t, such that, $\mathrm{q}(\mathrm{t}+1) = \mathrm{F}[\mathrm{q}(\mathrm{t}), \beta(\mathrm{t})]$. It is called a *transition function*, i.e., a function that determines the state of the automaton at any subsequent time instant $t + 1$. This mapping can either be deterministic or stochastic, depending on the environment in which the automaton operates.
(v) $G(.)$: is a mapping $G : Q \rightarrow A$, and is called the *output function*. Depending on the state at a particular instant, this function determines the output of the automaton at the same instant as: $\alpha(\mathrm{t}) = \mathrm{G}[\mathrm{q}(\mathrm{t})]$. This mapping can, again, be considered to be either deterministic or stochastic, depending on the environment in which the automaton operates [29]. Without loss of generality, G is deterministic.

4.2 The Environment

The environment, E, typically, refers to the medium in which the automaton functions. The environment possesses all the external factors that affect the actions of an automaton. Mathematically, an environment can be abstracted by a triple $\{A, C, B\}$. A, B, and C are defined as follows [21].

(i) $\mathrm{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents a finite input set,
(ii) $\mathrm{B} = \{\beta_1, \beta_2, \ldots, \beta_r\}$ is the output set of the environment, and
(iii) $\mathrm{C} = \{c_1, c_2, \ldots, c_r\}$ is a set of penalty probabilities, where element $c_i \in \mathrm{C}$ corresponds to an input action α_i.

The process of learning is based on a learning loop involving the two entities: the RE and the LA, as described in Fig. 2. In the process of learning, the LA continuously interacts with the environment to process responses to its various actions. Finally, through sufficient interactions, the LA attempts to learn the optimal action offered by the RE. The actual process of learning is represented as a set of interactions between the RE and the LA.

The RE offers the automaton with a set of possible actions $\{\alpha_1, \ldots, \alpha_r\}$ to choose from. The automaton chooses one of those actions, say α_i, which serves as an input to the RE. Since the RE is aware of the underlying penalty probability distribution of the system, depending

Author's personal copy

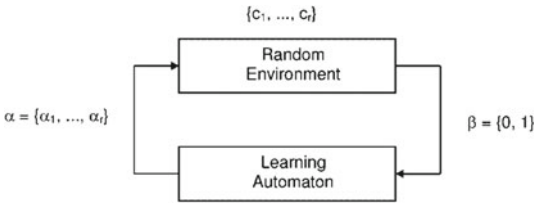


Fig. 2 The Automaton-environment feedback loop [8]

on the *penalty probability* c_i corresponding to α_i, it 'prompts' the LA with a *reward* (typically denoted by the value '0'), or a *penalty* (typically denoted by the value '1'). The reward/penalty information (corresponding to the action) provided to the LA helps it to choose the subsequent action. By repeating the above process, through a series of Environment-Automaton interactions, the LA finally attempts to learn the *optimal* action from the environment [27].

We now provide a few important definitions used in the field of LA. Given an action probability vector P(t) at time 't', the *average penalty* is defined as [21]

$$M(t) = E[\beta(t)|p(t)] = \sum_{i=1}^{r} c_i\, p_i(t) \tag{1}$$

The average penalty for the 'pure-chance' automaton is given by Narendra and Thathachar [21]:

$$M_0 = \frac{1}{r} \sum_{i=1}^{r} c_i\, p_i(n). \tag{2}$$

As t $\rightarrow \infty$, if the average penalty $M(t) < M0$, at least asymptotically, the automaton is generally considered to be better than the pure-chance automaton. $E[M(t)]$ is given by Narendra and Thathachar [21]

$$\mathrm{E[M(t)]} = \mathrm{E\{E[}\beta(\mathrm{t})|\mathrm{P(t)]\}} = \mathrm{E}[\beta(\mathrm{t})] \tag{3}$$

4.3 Action Probability Updating

In our work, we deal with the *variable structure stochastic automata* (VSSA). VSSA are the ones in which the state transition probabilities are not fixed. In such automata, the state transitions or the action probabilities themselves are updated at every time instant using a suitable scheme. The transition probabilities and the output function in the corresponding Markov chain vary with time, and the action probabilities are updated on the basis of the input. VSSA depend on random number generators for their implementation. The action chosen is dependent on the action probability distribution vector, which is, in turn, updated based on the reward/penalty input that the automaton A variable-structure automaton is defined by the quadruple $\{\alpha, \beta, P, T\}$ in which $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ represents the action set of the automata, $\beta = \{\beta_1, \ldots, \beta_n\}$ represents the input set, $P = \{P_1, \ldots, P_n\}$ represents the action probability set, and finally $p\,(n+1) = T\,[\alpha\,(n)\,, \beta\,(n)\,, p\,(n)]$ represents the learning algorithm. 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 Eqs.

Author's personal copy

(3, 4) for favorable responses, and Eq. (5) for unfavorable ones.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) &= (1-a)P_j(n) \quad \forall j, \quad j \neq i \end{aligned} \tag{4}$$

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)Pj(n) \quad \forall j, \quad j \neq i \end{aligned} \tag{5}$$

In these two equations, a and b are reward and penalty parameters respectively. For $a = b$, learning algorithm is called L_{R-P},[1] for $b \ll a$, it is called $L_{R\varepsilon P}$.[2], and for $b = 0$, it is called L_{R-I}.[3] For more information the reader may refer to Narendra and Thathachar [21].

5 Proposed Method

The proposed target monitoring algorithm consists of three phases: Initial phase, learning phase, and target monitoring phase. In the initial phase which is performed when the network starts operating, all nodes of the network participate. At the end of this phase all node in network know its neighbors and monitored targets. In learning phase which is performed in different rounds, learning automat of each node helps to node to select suitable state among its states. At the end of this phase the action probability vector of each node is valued. Finally, in target monitoring phase, learning automata of each node selects its best action. At the end of this phase each node operates based on best action which means to be active or asleep. Figure 3 shows pseudo code of our proposed method.

5.1 Initial Phase

Each node s_i in the network is equipped with a learning automaton LA_i which helps the node in determining its suitable state; whether to be active or not. Learning automaton of each node has two actions; *ACTIVE* or *ASLEEP*. At the beginning of the algorithm, *ACTIVE* and *ASLEEP* actions have the same probability equal to 0.5.

At the beginning of the algorithm, each node locally determines the targets it can cover. Then each node broadcasts an advertisement packet in its neighborhood containing its ID, position and targets it can cover. The node then listens to receive advertisement packets from its neighbors. Network operation is divided into rounds. Each round begins with a learning phase, followed by a target monitoring phase.

5.2 Learning Phase

In the learning phase each node in networks works as follow: During the learning phase, learning automaton of each node s_i randomly selects one of its actions (ACTIVE or ASLEEP). Node s_i broadcasts an ACTION packet, which contains its selected action, in its neighborhood. Node s_i then waits for certain duration to receive the ACTION packets of its neighbors. When node s_i receives ACTION packets from all of its neighbors, it operates as follows:

If the selected action of LA_i was *ACTIVE* then:

[1] Linear Reward-Penalty.

[2] Linear Reward epsilon Penalty

[3] Linear Reward Inaction.

Author's personal copy

The LADSC algorithm

```
Input:  (i) Given a set S of N sensor node and a set T of M targets and sensing range.
        (ii) iters=total number of iterations
        (iii) α, β learning parameter
Output:
        A converged network`s targets that has monitor all targets
BEGIN
        //INITIAL PHASE
         obtainANetworkInstance; //Obtain a snapshot of the given network.
        For (each node n in Nework())
                senseEnvironmetn()
                determinteTargetsandNeigbhors()
        End-For
        For (each node n in Nework())
                For (each node n in NodeNeigbhors)
                SendActionPacket
                End-For
        End-For

 While(AllTargetsCouldCover)

        //LEARNING PHASE
        For (each node n in Nework())
                        For (each action a in node)
                                InitialProbability=0.5; //Initialize action probabilities
                        End-For
        End-For

        For (i =0 to iters) //Execute for all iterations
                For (each node n in Nework())
                        Node=getRandomAction(); //Randomly choose an action.
                        sendActionPacketToNodeNeigbhors()
                        receiveActionPacketFromAllNodeNeigbhors()

                        If (NodeAction=ASLEEP and NeigbhorsCouldCoverTargetsOfCurrentNode)
                                Reward this action using learning automata
                        ELSE
                                Penalize this action using learning automata
                        End IF
                        If (NodeAction=ACTIVE and NeigbhorsCouldCoverTargetsOfCurrentNode)
                                Penalize this action using learning automata
                        ELSE
                                Reward this action using learning automata
                        End IF
        END FOR
END FOR
        //TARGET MONITORING PHASE
                For (each node n in Nework())
                        SelectBestAction()
                        If (BestAction=ACTIVE)
                                NodeStatus=active
                                CurrentCoverSet=CurrentCoverSet U Node
                        ELSE
                                NodeStatus=sleep
                        End IF
                END FOR
                Monitor targets until end of current round.
      End While
    END
```

Fig. 3 Pseudo code of proposed algorithm

If all of the targets under the coverage of the node s_i are covered by those neighbors whose selected actions are *ACTIVE*, then node s_i penalizes its learning automaton. Otherwise, node s_i rewards its learning automaton.

Author's personal copy

If the selected action of LA_i was *ASLEEP* then:

If all of the targets under the coverage of the node s_i are covered by those neighbors whose selected actions are ACTIVE, then node s_i rewards its learning automaton. Otherwise, node s_i penalizes its learning automaton.

Each node s_i separately stops its *learning* phase if one of the following conditions occurred:

1) Action probability of one of the actions of LA_i exceeds a specified threshold.
2) Number of action selections by LA_i exceeds *MaxActionSelection*.

5.3 Target Monitoring Phase

When learning phase is over, and at the startup of a new monitoring phase, each node selects its state for the whole duration of the current monitoring phase according to the action with higher probability in the action probability vector of its learning automaton; that is, if for a node i, the probability of ACTIVE action is higher than the probability of ASLEEP action, then node i selects its state as ACTIVE and vice versa. In the proposed method, target monitoring phase lasts until all nodes with active state lose their residual energy. The next round will be started when the current monitoring phase is over.

5.4 Computing Network Lifetime

During the target monitoring phase, the lifetime of each ACTIVE sensor will be updated. We represent the activation time of each target monitoring phase is 1 and it shows that in every phase all selected sensors consume their energy. Therefore, every phase add one unit of lifetime to network total lifetime.

5.5 Finding Redundancy

In this section we describe how to get redundant node number that has selected by our proposed learning automata based approach. To find redundant nodes that have selected by proposed method, we fist offer the following definitions:

Definition 2 Redundant Node Number (γ): Redundant Node Number is the number of nodes that can go to sleep state after implying the proposed algorithm.

Definition 3 Coverage Redundancy: Coverage Redundancy is defined according to the following equation:

$$\text{Coverage Redundancy} = \gamma / n_r \tag{6}$$

where n_r is number of required nodes for covering targets. Optimal value for coverage redundancy is 1 because in this state we don't have any redundant node. We use RemoveRedundancy function to find redundant nodes that have been selected by our proposed method. We apply this function after learning phase and this function returns RN that shows number of redundant nodes. Figure 4 shows pseudo code of remove redundancy function. In this function C_P is the current cover set that has been selected by our algorithm.

6 Experimental Results

In this section, we evaluate the performance of the proposed scheduling mechanism, referred to as LADSC hereafter, by conducting a number of computer simulations. In these simula-

Author's personal copy

Fig. 4 Pseudo code of RemoveRedundancy

```
RemoveRedundancy(Cp)
    do for each sensor s in cover Cp
        do if Cp - {s} is still a complete cover
            then remove s from Cp
            RN++
            break
```

tions, a fixed sensor network is considered, in which all sensor nodes are randomly deployed within a 500 m × 500 m area. A number of fixed targets are also deployed randomly within this area. Sensing ranges of all sensor nodes are equal. Parameters of the conducted simulations are as follows:

N, the number of sensor nodes. We vary the number of sensor nodes in the range [100, 300] to study the effect of node density on the performance of LADSC.

T, the number of targets to be covered. We vary the number of targets in the range [5, 55].

r, the sensing range. We vary the sensing range of sensor nodes in the range 100–600 m.

Energy consumption of sensor nodes for communication tasks follows the first order energy model given in [14]. Energy required to switch a node from sleep to active mode is assumed to be negligible. Simulations are performed in wireless sensor network simulator given in [15]. Results are averaged over 20 runs.

Experiment 1

We first study how much longer lifetime we can achieve by increasing nodes. Figure 5a shows for 30 sensors and 15 targets, increasing the sensing range results in increasing network lifetime. The lifetime is not sensitive to the number of targets in this experiment since targets follow a random uniform distribution. Doubling the number of targets decrease the lifetime rarely.

Figure 5b shows for 20 targets and sensing range 300, increasing the number of sensors will get more network lifetime. When the sensing range decreases to 250, the lifetime noticeably drops. On the curve for $T = 50$ and $R = 250$, the average number of sensors covering each target increases approximately from 1 to 2, and the lifetime shows the same trend.

Experiment 2

For large networks, we apply our learning automata based method to increase lifetime. Apparently large networks show the same trend as in small networks, and lifetime increases as the number of sensors per target increases. As we can see in the Fig. 6a, in proportion as we increase the sensing range, longer lifetime gain. We compared Fig. 6a with b and observed that even the slopes of the curves are very close.

Experiment 3

This experiment is conducted to study the effect of the number of sensor nodes deployed within the area on the lifetime of the network. For This experiment, we set the number of

Author's personal copy

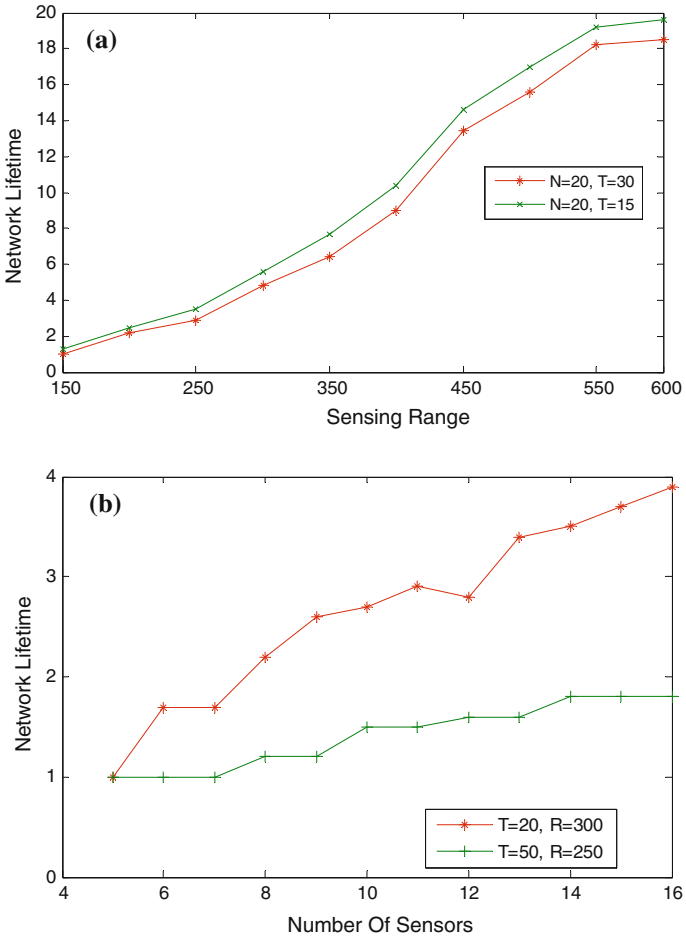


Fig. 5 **a** Increasing sensing range from 150 to 600 with $N = 20$, $T = 15$ and 30,respectively; **b** deploying more sensors, with $N = 5 - 17$, $T = 20$, range $R = 300$ and $T = 50$, range $R = 250$, respectively

targets to 10. Number of sensor nodes is set to 90. In Table 1 we consider the measurements for 90 sensor nodes and 10 targets and compare the results produced by *MC-MIP* and the heuristic proposed by Slijepcevic and Potkonjak in [27]. Our learning automata based approach produces consistently more covers; therefore we can achieve better energy savings.

Experiment 4

In this experiment, we compare the lifetime of the network when LADSC, MC-MIP [7], and Slijepcevic and Potkonjak [27] scheduling mechanisms are used. For this experiment, we set the number of targets vary in the range [15, 55] with step of 5 targets, let the number of sensor nodes are 90, and set sensing range r to 250 m. In Table 2, we present the maximum, average and minimum number of covers computed by LADSC, *MC-MIP* and the heuristic in [7]. The table shows that *LADSC* has superior result in comparison to the existing methods. This is due to the fact that in *LADSC*, the chance of a node, which covers more targets, to become an ACTIVE node is higher than a node, which covers fewer cells.

Author's personal copy

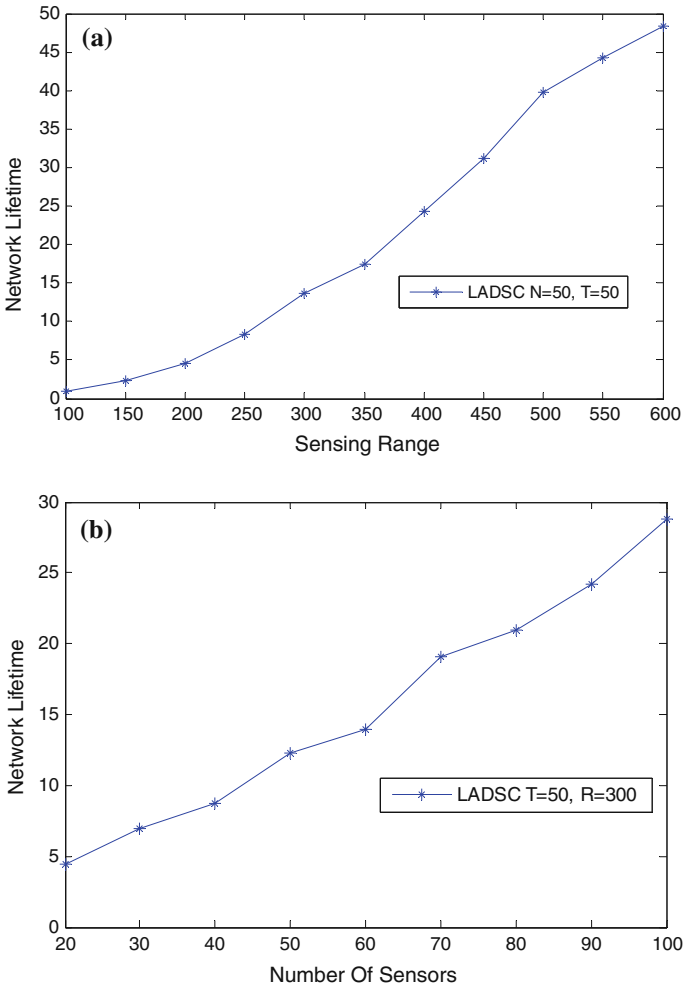


Fig. 6 **a** Increasing sensing range from 100 to 600 with 50 sensors and 50 targets; **b** varying network size from 20 to 100 sensors with fixed range $R = 300$, $T = 50$

Table 1 Measurements for 90 sensors and 10 targets randomly distributed

Sensors range	LA-DSC			MC-MIP			Slijepcevic		
	MIN	AVG	MAX	MIN	AVG	MAX	MIN	AVG	MAX
100	2	3.7	8	0	2.4	4	0	2.4	4
120	3	5.7	10	3	5.4	7	3	5	7
140	4	7.7	10	4	6.6	8	4	6	8
160	5	9.43	16	8	8.6	11	6	7.6	9
180	7	12.6	22	6	11.6	15	6	10.2	13
200	7	16.9	27	13	15	17	11	12.6	15
220	14	20.7	25	16	18.4	21	13	16.8	21
240	15	21.4	29	13	19.6	23	13	18.2	21

Author's personal copy

Table 1 continued

Sensors range	LA-DSC			MC-MIP			Slijepcevic		
	MIN	AVG	MAX	MIN	AVG	MAX	MIN	AVG	MAX
260	25	27.8	34	15	22.2	26	15	20.4	23
280	23	30.3	36	21	27	30	21	24.4	27
300	26	35.6	44	27	31.4	33	27	29.2	31

Table 2 Measurements for 90 sensors with sensing range of 250 m

Number of targets	LA-DSC			MC-MIP			Slijepcevic		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
15	17	26.95	37	17	22.8	27	16	20.8	23
20	14	22.95	32	17	19.4	22	16	18.2	21
25	17	22.8	29	18	20.4	23	16	18.2	19
30	19	23.4	30	18	21.2	24	16	18	19
35	13	21.8	30	11	19	23	11	16.6	19
40	15	21.35	29	16	19.4	22	16	16.8	18
45	16	20.35	28	17	18.4	20	15	16	17
50	15	20.35	27	18	20.6	23	15	17.2	20
55	16	21.05	26	14	17	21	14	16	18

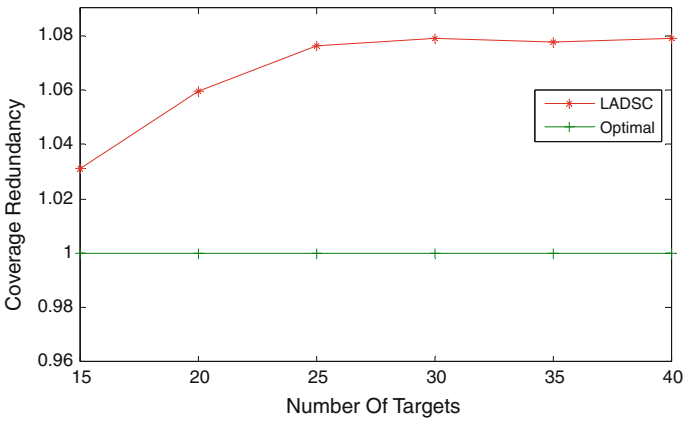


Fig. 7 Coverage redundancy for 90 sensors with sensing range of 250 m

Experiment 5

In this experiment, LADSC method is compared with the optimal method in terms of coverage redundancy. We let the number of nodes in the network to be 90 for this experiment and set the radio transmission range to 250 m. Figure 7 shows that our protocol has coverage redundancies that is about 3–8 % from the optimal value (which is 1 as we stated before). This indicates that our algorithm is able to compute near-minimal covers.

Author's personal copy

7 Conclusion

In this paper, we addressed the target coverage problem in wireless sensor networks and modeled this problem as a maximum disjoint set cover problem. Then, we proposed a learning automata-based scheduling mechanism for this problem in which each node is equipped with a learning automaton. LA help each node to decide if it is redundant or not. Redundant nodes then go to sleep mode and save their energies for later times. It was shown through computer simulations that the proposed method outperforms the existing methods in terms of the lifetime of the network.

Acknowledgments This work is supported by Islamic Azad university Urmia branch young researchers' club.

References

1. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayircl, E. (2002). A survey on sensor networks. In *Proceedings of the IEEE communication magazine* (pp. 102–114), August, 2002.
2. Bagheri, M., Hefeeda, M., & Ahmadi, H. (2006). *A near optimal k-coverage algorithm for large-scale sensor networks*, Technical report TR 2006–2010, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, May.
3. Bagheri, M., & Hefeeda, M. (2007). Randomized k-coverage algorithms for dense sensor networks. In: *Proceedings of the IEEE INFOCOM 2007 minisymposium* (pp. 2376–2380), Anchorage, AK. May, 2007.
4. Branch, J. W., Chen, G. G., & Szymanski, B. K. (2005). ESCORT: Energy-efficient sensor network communal routing topology using signal quality metrics. In: *Proceedings of the 4th IEEE international conference on networking (IEEE ICN'05)*, Reunion Island, France, April, 2005. (vol. Part 1, pp. 438–448).
5. Cai, Y., Li, M., Shu, W., & Wu, M.-Y. (2006). ACOS: An area-based collaborative sleeping protocol for wireless sensor networks. *International journal of ad hoc and sensor wireless networks*.
6. Cardei, M., & Du, D.-Z (2005). Improving wireless sensor network lifetime through power aware organization. In *ACM wireless networks* (vol. 11, pp. 333–340).
7. Cardei, M., & Wu, J. (2006). Energy-efficient coverage problems in wireless ad hoc sensor networks. *Computer Communications, 29*(4), 413–420.
8. Cardei, M., Thai, M. T., Li, Y., & Wu, W. (2005). Energy-efficient target coverage in wireless sensor networks. In *Paper presented at the IEEE INFOCOM 2005*.
9. Chen, H., Wu, H., & Tzeng, N.-F. (2004). Grid-based approach for working node selection in wireless sensor networks. In: *Proceedings of the IEEE international conference on communications (ICC 2004)*.
10. Dingxing, Zh., Ming, X., Yingwen, C., & Shulin, W. (2006). Probabilistic coverage configuration for wireless sensor networks. In: *Second international conference on wireless communications, Networking and mobile computing (WiCOM 2006)*, Wuhan, September, 2006.
11. Esnaashari, M., & Meybodi, M. R. (2008). Dynamic point coverage in wireless sensor networks: A learning automata approach. In *Proceedings of 13th international CSI computer conference of Iran*, Springer Verlag. Kish Island, Iran, March 9–11, 2008.
12. Esnaashari, M., & Meybodi, M. R. (2010). A learning automata based scheduling solution to the dynamic point coverage problem in wireless sensor networks. *Computer Networks* doi:10.1016/j.comnet.2010.03.014.
13. Frolik, J. (2004). QoS control for random access wireless sensor networks. In: *Wireless communications and networking conference (WCNC04)*, Atlanta, March, 2004.
14. Heinzelman, W., Chandrakasan, A., & Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii international conference on system sciences*, Hawaii, USA, (pp. 1–10).
15. http://www.djstein.com/projects/index.html.
16. Ilyas, M., & Mahgoub, I. (2005). *Handbook of sensor networks: Compact wireless and wired sensing systems*. London, Washington, DC: CRC Press.

Author's personal copy

17. Liang, B., Frolik, J., & Wang, X. S. (2005). A predictive QoS control strategy for wireless sensor networks. In *The 1st workshop on resource provisioning and management in sensor networks (PRMSN 05) in conjunction with the 2nd IEEE MASS*, Washington DC, November, 2005.
18. Liu, C., Wu, K., & Pei, J. (2005). 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.
19. Mostafaei, H., Meybodi, M., & Esnaashari, M. (2010). A learning automata based area coverage algorithm for wireless sensor networks. *Journal of Electronic Science and Technology, 8*(3), 200–205.
20. Mostafaei, H., Meybodi, M., & Esnaashari M. (2010). EEMLA: Energy efficient monitoring of wireless sensor network with learning automata. In *International conference on signal acquisition and Processing*, Bangalore, India, (pp. 107–111). doi:10.1109/ICSAP.2010.14.
21. Narendra, K. S., & Thathachar, M. A. L. (1989). *Learning automata: An introduction*. Englewood Cliffs: Prentice Hall.
22. Pedraza, F., García, A., & Medaglia, A. L. (2006). Efficient coverage algorithms for wireless sensor networks. In: *Proceedings of the systems and information engineering design symposium*.
23. Raghunathan, V., Schurgers, C., Park, S., & Srivastava, M.B. (2002). Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine, 19*, 40–50.
24. Rakhmatov, D., & Vrudhula, S. (2003). Energy management for battery-powered embedded systems. *Translated Embedded Computing Systems, 2*(3), 277–324.
25. Resende, M. G. C., & Paradalos, P. M. (Eds.). (2006). *Handbook of optimization in telecommunications*. Berlin: Spinger.
26. Sheu, J. P., & Lin, H. F. (2007). Probabilistic coverage preserving protocol with energy efficiency in wireless sensor networks. In: *IEEE WCNC* (pp. 2631–2636).
27. Slijepcevic, S., & Potkonjak, M. (2001). Power Efficient Organization of Wireless Sensor Networks. In *Paper presented at the ICC*, Helsinki, Finland.
28. Thathachar, M. A. L., & Sastry P. S. (2002). Varieties of learning automata: An overview. In *IEEE transaction on systems, man and cybernetics-Part B: Cybernetics*, pp. 711–722.
29. Wang, B. (2010). *Coverage control in sensor networks*. Berlin: Springer.
30. Wang, B., Chua, K. C., Srinivasan, V., & Wang, W. (2006). Scheduling sensor activity for point information coverage in wireless sensor networks. In: *Proceedings of the WiOpt*.
31. Wang, B., Chua, K. C., Srinivasan, V., & Wang, W. (2006). Sensor density for complete information coverage in wireless sensor networks. In *Proceedings of the EWSN 2006*, Zurich, Switzerland, February.
32. Wu, Y., Fahmy, S., & Shroff, N. B. (2006). Optimal QoS-aware sleep/wake scheduling for time-synchronized sensor networks. In: *Invited sessions on optimization of communication networks 40th annual conference on information sciences and systems (CISS)*, Princeton, NJ.
33. Xu, X., & Sahni, S. (2006). *Approximation algorithms for wireless sensor deployment*. April 21, 2006.
34. Zou, Y., & Chakrabarty, K. (2005). A distributed coverage-and connectivity-centric technique for selecting active nodes in wireless sensor networks. *IEEE Transactions on Computers, 54*(8), 978–991.

Author Biographies

Habib Mostafaei received the B.S. degree from the Islamic Azad University Khoy branch, in 2006 and the M.S. degree from the Islamic Azad University Arak branch, in 2009, both in software engineering. His research interests include learning systems, sensor networks.

Author's personal copy

Mohammad Reza Meybodi received his B.S. and M.S. degrees in Economics from Shahid Beheshti University in Iran during 1973 and 1977, respectively. He also received his M.S. and Ph.D. degrees from Oklahoma University, USA during 1980 and 1983, respectively in Computer Science. Presently, he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to the present 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 wireless networks, fault tolerant systems, learning systems, parallel algorithms, soft computing and software development.