Energy Efficient Target Coverage in Wireless Sensor Networks Using Adaptive Learning

Ashish Rauniyar[1,2], Jeevan Kunwar[1], Hårek Haugerud[1], Anis Yazidi[1]✉and Paal Engelstad[1,2]

[1] Autonomous System and Network Research Group,
Department of Computer Science, Oslo Metropolitan University
[2] Department of Technology Systems, University of Oslo
Oslo, Norway
✉anisy@oslomet.no

Abstract. Over the past few years, innovation in the development of Wireless Sensor Networks (WSNs) has evolved rapidly. WSNs are being used in many application fields such as target coverage, battlefield surveillance, home security, health care supervision, and many more. However, power usage in WSNs remains a challenging issue due to the low capacity of batteries and the difficulty of replacing or charging them, especially in harsh environments. Therefore, this has led to the development of various architectures and algorithms to deal with optimizing the energy usage of WSNs. In particular, extending the lifetime of the WSN in the context of target coverage problems by resorting to intelligent scheduling has received a lot of research attention. In this paper, we propose a scheduling technique for WSN based on a novel concept within the theory of Learning Automata (LA) called pursuit LA. Each sensor node in the WSN is equipped with an LA so that it can autonomously select its proper state, i.e., either sleep or active with the aim to cover all targets with the lowest energy cost. Through comprehensive experimental testing, we verify the efficiency of our algorithm and its ability to yield a near-optimal solution. The results are promising, given the low computational footprint of the algorithm.

Keywords: Wireless Sensor Network, Adaptive Learning, Learning Automata, Minimum Active Sensors, Target Coverage, Energy Efficiency

1 Introduction

Wireless Sensor Networks (WSNs) consist of a large number of identical and independent sensors being deployed either in a random manner or in a deterministic manner for effective monitoring of an area of the region of interest. The sensor nodes are the principal components of the WSNs. Usually, these sensor nodes are small, low power devices that have the ability to communicate over a short distance. The three major operations of these sensors are sensing,

This is a post-peer-review, pre-copyedit version of a book chapter published in Distributed Computing for Emerging Smart Networks; First International Workshop, DiCES-N 2019, Hammamet, Tunisia, October 30, 2019, Revised Selected Papers, which is part of the Communications in Computer and Information Science book series (CCIS, volume 1130).
The final authenticated version is available online at: https://dx.doi.org/10.1007/978-3-030-40131-3_9

processing, and communication. WSNs admit a large list of applications that cover almost any field [20]. WSN technology has it first roots in the military domain, where it was used for developing applications for effective monitoring, surveillance of the battlefield, etc. WSNs also admit other domain applications involving home monitoring, health care, temperature, disaster prevention, environmental monitoring, pollution, and so on.

An effective power management factor is one of the keys elements for enhancing the lifetime of a WSN. Batteries that are used in a sensor network are relatively small in size and therefore possess a low power storage capacity. These batteries need either a replacement or frequent recharging for continuous network operation. However, this is impractical in many real-life situations as those sensors might be deployed in areas that are difficult to access. The network lifetime is defined in the context of network coverage problems as the duration of time elapsed from the network starts functioning with full coverage from its initialization to the time instant where the coveted coverage criteria is unsatisfied [12].

There have been many research works addressing the problem of inefficient energy consumption in WSNs. More particularly, a significant amount of research has been conducted for energy-efficient data aggregation and dissemination, transmission power control and nodes activity scheduling, energy-aware routing for efficient utilization of the energy in WSNs [4].

Another important aspect of the WSNs used for monitoring purposes is the coverage area of the WSN. This area can be defined as the area within which a sensor node is able to monitor and track the specified target's activities. Intuitively, each target should be monitored by at least one of the sensor node continuously, such that there is continuity in the network operation. Generally, the network lifetime can be enhanced by scheduling the activity of each of the sensor nodes in either active state or sleep state [4]. For energy-efficient scheduling, each sensor in the network has two modes, *active* mode, and *sleep* mode. The nodes are scheduled intelligently so that they can alternate between those two modes while meeting the desired coverage requirement.

The target coverage problem by the sensor nodes in the WSN includes three families of problems which are defined as follows according to [13]:

- **Area Coverage:**
  This coverage problem is concerned with the monitoring of the targets in the entire area of the network.
- **Target Coverage:**
  This coverage problem is concerned with the monitoring of only certain targets within the specified region of the network.
- **Barrier Coverage:**
  The barrier coverage problem aims rather to minimize the probability of undetected penetration through the barrier in the network.

There is a vast amount of research on coverage problems for designing energy-efficient WSN. Many scheduling algorithms have been proposed to schedule the activity of the sensor nodes. One of the scheduling methods for energy-efficient

WSN is through Learning Automata (LA) [13,16]. This mechanism provides the sensor node to learn their state and select its appropriate state, i.e. either active or sleep mode, for the purpose of prolonging the network lifetime of the WSNs.

In order to prolong the network lifetime, this paper mainly deals with the problem of area coverage using the theory of LA. Although the theory of LA has been applied before to solve the problem of area coverage, our solution enjoys some desirable designed properties compared to [13]. In fact, we apply LA to each of the sensors to determine each state: active or sleep. Therefore, we opt to pursue the joint action of the team LA corresponding to the best solution found so far using the concept of pursuit learning [1,22,17]. Our proposed scheme is different from the work of [13] as the LA update of latter work does not track the best solution found so far. In [13], the authors implemented a reward and penalty mechanism in the following manner:

- Reward action sleep - if all targets of the sensor are covered. In other word, taking the opposite action, which is active, would not result in any benefit.
- Reward action activate - if the sensor in question covers at least one target exclusively. In other words, taking the opposite action, which is sleep, will result in at least one less target that can not be covered.

The remainder of this paper is organized as follows. In Section 2, we survey the related work and introduce some background concepts about the theory of LA, which is an essential component in our solution. Section 3 details the main approach for solving the sensor coverage problem, which is based on the theory of LA. In Section 4, we report some representative simulations results showing the efficiency of our scheme. Finally, we give some final conclusions in Section 5.

2 Background and Related Work

In this section, we shall review some basic concepts about the theory of Learning Automata (LA), which is fundamental for the understanding of our paper. Furthermore, we shall survey some pertinent related works.

2.1 Learning Automata

In the field of Automata Theory, an automaton [10,14,15,18,19] is characterized as a quintuple made out of a set of states, a set of outputs or actions, an input, a function that maps the present state and the input to the following state, and a function that maps a present state (and input) into the current output.

Definition 1: A LA is defined by a quintuple $\langle A, B, Q, F(.,.), G(.)\rangle$, where:

1. $A = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of outputs or actions that the LA must choose from. $\alpha(t)$ is the action picked by the automaton at any moment or instant t.

2. $B = \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the set of inputs or feedback to the automaton. $\beta(t)$ is the feedback at any moment t corresponding to the chosen action. The set B can be limited or unbounded. The most widely recognized LA input information is $B = \{0, 1\}$, where $\beta = 0$ represents reward or equivalently a positive feedback, and $\beta = 1$ represents penalty or equivalently a negative feedback.
3. $Q = \{q_1, q_2, \ldots, q_s\}$ is the set of finite states, where $Q(t)$ signifies the condition of the automaton at any moment t.
4. $F(.,.) : Q \times B \mapsto Q$ is a mapping as far as the state and input at the moment t, with the end goal that, $q(t+1) = F[q(t), \beta(t)]$. It is known as a *transition function*, i.e., a function that decides the condition of the automaton at any resulting time instant $t + 1$. This mapping can either be deterministic or stochastic.
5. $G(.)$: is a mapping $G : Q \mapsto A$, and is known as the *output function*. G decides the action taken by the automaton in the event that it is in a given state as: $\alpha(t) = G[q(t)]$.

If the sets Q, B and A are all finite, the automaton is said to be *finite*.

The Environment, E, ordinarily, alludes to the medium where the automaton functions. The Environment has all the outer variables that influence the activities of the automaton. Mathematically, an Environment can be preoccupied with a triple $\langle A, C, B\rangle$. A, C, and B are characterized as follows:

1. $A = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of actions.
2. $B = \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the output set of the Environment. Once more, we consider the situation when $m = 2$, i.e., with $\beta = 0$ representing a "Reward", and $\beta = 1$ representing a "Penalty".
3. $C = \{c_1, c_2, \ldots, c_r\}$ is a set of punishment or penalty probabilities, where component $c_i \in C$ relates to an input activity α_i.

The way toward learning depends on a learning loop, including the two entities: the Random Environment (RE), and the LA, as depicted in Fig. 1. In the learning procedure, the LA persistently communicates with the Environment to process reactions to its different activities (i.e., its decisions). Finally, through adequate communications, the LA endeavors to gain proficiency with the ideal action offered by the RE. The real procedure of learning is represented as a set of associations or interactions between the RE and the LA.

The automaton is offered a set of actions, and it is obliged to picking one of them. At the point when an action is chosen among the pool of actions, the Environment gives out a response $\beta(t)$ at a time "t". The automaton is either penalized or rewarded with an obscure likelihood c_i or $1 - c_i$, separately. Based on the response $\beta(t)$, the state of the automaton $\phi(t)$ is updated and another new action is picked at (t+1). The penalty probability c_i satisfies:

$$c_i = \Pr[\beta(t) = 1 | \alpha(t) = \alpha_i] \qquad (i = 1, 2, \ldots, r). \tag{1}$$

We now present a few of the significant definitions utilized in the field. $P(t)$ is alluded to as the action probability vector, where, $P(t) =$

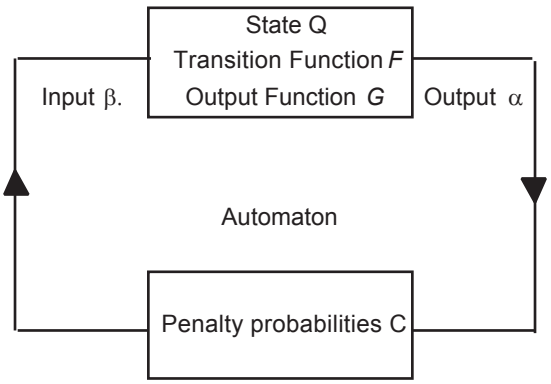

Fig. 1: Feedback Loop of LA.

$[p_1(t), p_2(t), \ldots, p_r(t)]^T$, in which every component of the vector:

$$p_i(t) = \Pr[\alpha(t) = \alpha_i],\ i = 1, \ldots, r,\ \text{such that}\ \sum_{i=1}^{r} p_i(t) = 1\ \ \forall t. \tag{2}$$

Given an action probability vector, $P(t)$ at time t, the *average penalty* is:

$$\begin{aligned} M(t) &= E[\beta(t)|P(t)] = \Pr[\beta(t) = 1|P(t)] \\ &= \sum_{i=1}^{r} \Pr[\beta(t) = 1|\alpha(t) = \alpha_i]\ Pr[\alpha(t) = \alpha_i] \\ &= \sum_{i=1}^{r} c_i p_i(t). \end{aligned} \tag{3}$$

The average penalty for the "pure-chance" automaton is given by:

$$M_0 = \frac{1}{r} \sum_{i=1}^{r} c_i. \tag{4}$$

As $t \mapsto \infty$, if the average penalty $M(t) < M_0$, in any event asymptotically, the automaton is commonly viewed as superior to the pure-chance automaton. $E[M(t)]$ is given by:

$$E[M(t)] = E\{E[\beta(t)|P(t)]\} = E[\beta(t)]. \tag{5}$$

A LA that performs better than by pure-chance is said to be *expedient*.

Definition 2: A LA is considered *expedient* if:

$$\lim_{t \mapsto \infty} E[M(t)] < M_0.$$

Definition 3: A LA is said to be *absolutely expedient* if $E[M(t+1)|P(t)] < M(t)$, implying that $E[M(t+1)] < E[M(t)]$.

Definition 4: A LA is considered *optimal* if $\lim_{t\mapsto\infty} E[M(t)] = c_l$,where $c_l = \min_i\{c_i\}$.

It ought to be noticed that no ideal LA exist. Marginally sub-optimal performance, also termed above as ϵ-optimal execution, is what that LA researchers endeavor to accomplish.

Definition 5: A LA is considered ϵ-*optimal* if:

$$\lim_{n\mapsto\infty} E[M(t)] < c_l + \epsilon, \tag{6}$$

where $\epsilon > 0$, and can be arbitrarily small, by a reasonable choice of some parameter of the LA.

2.2 Related Works

The authors in [6] use a probabilistic coverage model that takes the distance parameter for the target coverage. This algorithm is based on the modified weighted set, which helps to organize sensors into disjoint and non-disjoint set covers. In the study reported in [23], the authors introduce the concept of coverage-centric nodes. Coverage-centric nodes are the nodes that ensure larger coverage than the other nodes. In this regard, a novel algorithm called the Coverage-Centric Active Nodes Selection (CCANS) algorithm was devised. This algorithm depends on the formation of the Connected Dominating Set (CDS). The active nodes of the network structure the CDS. This provides the backbone to other nodes for sensing and communication purposes such that the data communication is processed through this route.

The work in [21] investigated the application of the sleep and awake schedule to the low duty cycle WSNs. In particular, the author of the latter study has taken into consideration of the explicit effect of synchronization error for designing the sleep and awake schedule. The proposed scheme in this work is divided into two main parts. The first part of the work provides a sleep and awake schedule by the use of an efficient search method for optimizing the number of sensors to ensure target coverage. In the second part, the authors focus on optimizing the quality of service of the network.

The work in [9] provides a heuristic and artificial bee colony algorithm as a scheduling technique. Through their experiments, the authors concluded that their methods help in improving the network lifetime of the sensor networks.

There is a vast majority of the literature that can be found about the target coverage problem of the WSNs. In [11], the authors have discussed the target coverage along with the data collection problem in WSNs. The authors investigate the use of polynomial-time approximation and polynomial-time constant approximation methods to analyze the complexity of different approaches for target coverage problems.

In [3], the sensor nodes are organized into several maximal set covers. These set covers are activated to monitor the targets, while the other set of nodes remains in sleep mode to save the energy. The main goal of [3] was to find the disjoint set of sensor nodes for energy conservation to increase the network lifetime. The authors have used the heuristic approach for computing the sets through the use of linear programming. The result shows that there is an increase in the lifetime with an increase in the target and sensing range with a specified number of targets. A heuristic method for organizing sensor nodes into disjoint set covers is carried out in [8,2]. The sensor set that is in the active state can only monitor the targets, and the other sensor set goes into low energy sleep mode. Also, the authors in [8] have used greedy Connected Set Coverage (CSC) heuristics algorithm to increase the network operation lifetime.

In the case of the mobile target, it is difficult to find the exact coverage and the position of the targets in a large-scale WSNs. In a practical scenario, sensor sensing follows a probabilistic sensing mode. In [5], the authors have proposed a probabilistic sensing model and circular graph for detecting the mobile targets. They have formulated a probabilistic trap coverage with maximum network lifetime problem. The authors also discussed circular coverage graph problem for determining whether a given sensor network can provide the probabilistic trap coverage or not.

3 Proposed Adaptive Learning Algorithm

3.1 Problem Formulation

Let us consider that there are a set of M targets denoted by $T = \{T_1, T_2, ...T_M\}$ which are being monitored by set of N sensor nodes S denoted by $S = \{S_1, S_2, ...S_N\}$. These two sets are deployed in a $X \times X$ area. All sensor nodes have a fixed sensing range "R". Also, we assume that there is the number of sensor nodes exceeds the number of targets. In order to increase the lifetime of the network, a scheduling algorithm has been proposed in this paper. A target point T_j within the range $1 \leq j \leq M$, is said to be covered by a sensor node if it falls inside this range of one of the sensor nodes $1 \leq i \leq N$ [13].

Fig. 2 shows the Venn diagram of the sensors and the targets with their bipartite graph. There are four sensors S1, S2, S3, and S4 with their respective targets T1, T2, T3, and T4. The circle with different colors represents the coverage of each of the sensor nodes. The bipartite graph shows the relationship between the sensors and the number of covered targets by them. We can observe that by only activating two sensors: S2 and S3, we can cover all the targets. The complexity of the problem increases exponentially as we increase the number of sensors since the number of possible configurations of N sensors in active and sleep modes is 2^N.

3.2 Adaptive Learning Algorithm

We have used the LA algorithm for scheduling the sensor nodes. We shall now delineate the details of our algorithm, which helps in finding the best active set

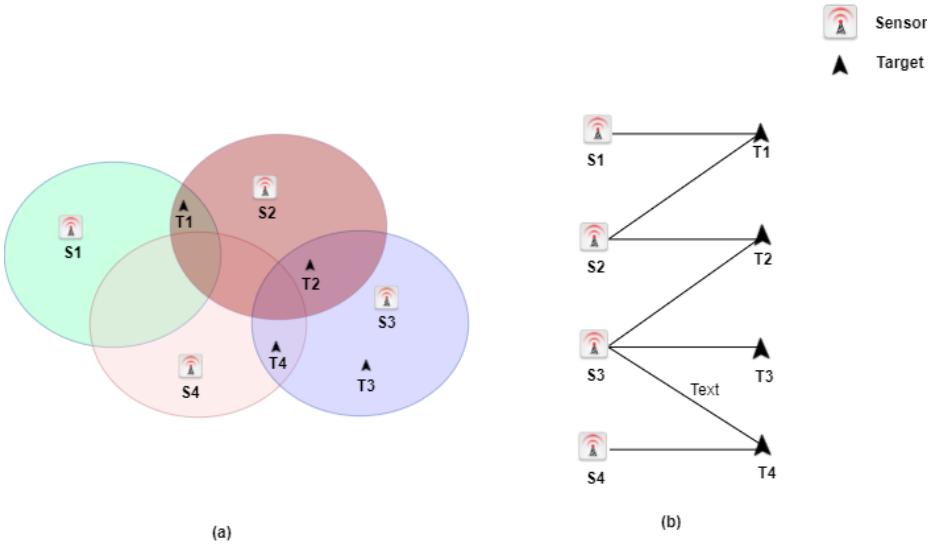

Fig. 2: (a) Sensors coverage with their targets and (b) Bipartite graph of sensors and targets

of sensors that are monitoring the maximum number of targets at any given instant. As in [13], the actual flow of the algorithm is divided into three phases, which include an initial stage, a learning phase, and the target monitoring phase. We shall now focus on the initial phase and the learning phase. The monitoring phase is identical to the one presented in [13], and therefore, it is omitted here for the sake of brevity.

Initial Phase: In this phase, each sensor node in the network are provided with LA, which helps the sensor node to select its state either to active or sleep. In the initial stage, both states are equally probable, i.e., the probability of selecting either of the active or sleep state is initialized to be 0.5. Here, sensor nodes are endowed with a certain level of autonomy permitting to establish communication, transfer messages, including their ID, position, and list of covered targets with their neighbor node autonomously. This phase is followed by the learning phase and the target monitoring phase.

Learning Phase: In the learning phase, each of the sensor nodes is equipped with LA. At first, the node is selected randomly. Using LA, each node selects its state. Then it broadcasts the message packet, including its all information to the rest of the sensor nodes.

We attach to each of the N sensors a LA. For instance, let us consider the senor S_i. The automaton's state probability vector at the node i at time t is $P_i(t) = [p_{(i,1)}(t), p_{(i,2)}(t)]$.

Algorithm 1 Learning Phase

```
Best coverage set= ∅
For each LA action set initial probability = 0.5
for Every sensor node in the network do
    Choose random action for sensor node
end for
for iteration=0 to max iterations do
    for Every sensor node in the network do
        Node= choose an action according to LA
        Update Current LA actions
        if Node state = Active then
            Current coverage set= Current coverage ⋃ node
        end if
    end for
    if |Current coverage set| > | Best Coverage set| then
        Best coverage set=Current coverage set
        Best LA actions=Current LA actions
    end if
    for Every node in the network do
        if Best LA action of node is sleep then
            Decrease the probability to be active
        else
            Increase the probability of the node to be active
        end if
    end for
end for
```

Thus, $p_{(i,j)}(t)$ is the probability at time instant t to select an action j. In our settings, we have two actions: sleep or active. For the sake of notation, let us denote 0 as the action sleep, and 1 denote the action active.

The feedback function is a binary function which yields a reward whenever the coverage has been improved. This is denoted by |Current coverage set| > |Best coverage set| in Algorithm 1. In more simple terms, if the aggregate state of the sensors chosen by the team of N LA yields an improvement in the coverage, which means covering more number of targets, the joint action of the LA that yielded that solution is rewarded by increasing the probability of the actions which formed that particular solution.

Let $J = \{j_1(t), j_2(t), ..., j_N(t)\}$ denote the action taken by the team of LA. Let $J^* = \{j_1^*(t), j_2^*(t), ..., j_N^*(t)\}$ be the best aggregate action of the team of LA so far yielding the highest coverage.

Thus, the idea of pursuit here is to reward the LA whose aggregate action is the highest possible so far, i.e., till the time instant t.

We consider the LA update equations at node i. The update is given by:

$$p_{(i,j)}(t+1) = (1-\lambda)\delta_{(i,j)} + \lambda p_{(i,j)}(t) \tag{7}$$

$$\delta_{(i,j)} = \begin{cases} 1 & \text{if } j = j_i^*(t) \\ 0 & \text{else} \end{cases} \tag{8}$$

λ is the update parameter and is time-independent.

The informed reader would observe that the above update scheme corresponds to the Linear-Reward Inaction LA update [16]. The pursuit paradigm we apply in this paper is an adaptation of the PolyLA-Pursuit scheme [7] recently proposed by Goodwin and Yazidi in the context of Machine Learning classification problems.

If $j \neq j_i^*(t)$ then $p_{(i,j)}(t+1)$ is reduced by multiplying by λ, which is less than 1.

$$p_{(i,j)}(t+1) = \lambda p_{(i,j)}(t) \tag{9}$$

However, if $j = j_i^*(t)$, then $p_{(i,j)}(t+1)$ is increased by:

$$\begin{aligned} p_{(i,j)}(t+1) - p_{(i,j)}(t) &= [(1-\lambda) + \lambda p_{(i,j)}(t)] - p_{(i,j)}(t) && (10) \\ &= (1-\lambda) + p_{(i,j)}(t)(\lambda - 1) && (11) \\ &= (1-\lambda)(1 - p_{(i,j)}(t)) \geq 0 && (12) \end{aligned}$$

The update scheme is called pursuit-LA [7] and has rules that obey the rules of the so-called Linear Reward-Inaction (LRI) LA. The idea is to always reward the transition probabilities along with the best solution obtained so far.

4 Experiments

In this section, we present our experimental results that demonstrate the effectiveness of our approach. Although we have conducted several experiments, we will report a few representative experiments for the sake of brevity. The simulations were performed using a customized simulation environment implemented in Python.

4.1 Impact of Sensing Range and Sensor Density

The primary goal of this particular experiment is to investigate the impact of sensing range and sensor density in a vast network. At first, the experiment is performed by varying the range of the sensor between 150m and 600m with 70 sensors and 50 targets. Then, the next test is conducted by increasing the density of the sensors to evaluate the algorithm performance to obtain the minimum number of active sensors. Table 1 and Table 2 show the obtained results of the experiment. Due to the stochastic nature of our algorithm, we report the average of an ensemble of 1000 experiments.

From Table 1, we can observe that as we increase the sensing range of the sensors from 150 to 600 to cover the same number of 50 targets, the average

number of active sensors drops from 12 to 6. Similarly, from Table 2, we can observe that as we increase the number of sensors nodes from 70 to 80 with a sensing range of 300 m to cover the same number of 50 targets, the average number of active sensors drops from 2 to 1.

Table 1 Results obtained for 50 targets and 70 sensors with sensor sensing range from 150m to 600m

Range of Sensors	Average Number of Active Sensors
150	11.752
200	8.869
250	7.557
300	7.054
350	6.694
400	6.523
450	6.512
500	6.510
550	6.465
600	6.241

Table 2 Results obtained for increasing the sensor number from 70 to 80 with 50 targets and sensing range 300m

Number of Sensors	Average Number of Active Sensors
70	2.145
71	2.140
72	2.135
73	2.120
74	2.110
75	1.967
76	1.830
77	1.549
78	1.420
79	1.347
80	1.102

Fig. 3 illustrates the results from the Table 1. From the plot, we see a decline in the number of active sensors as the range increases. This is because with the increasing sensing range, the coverage area of the sensors also increases.

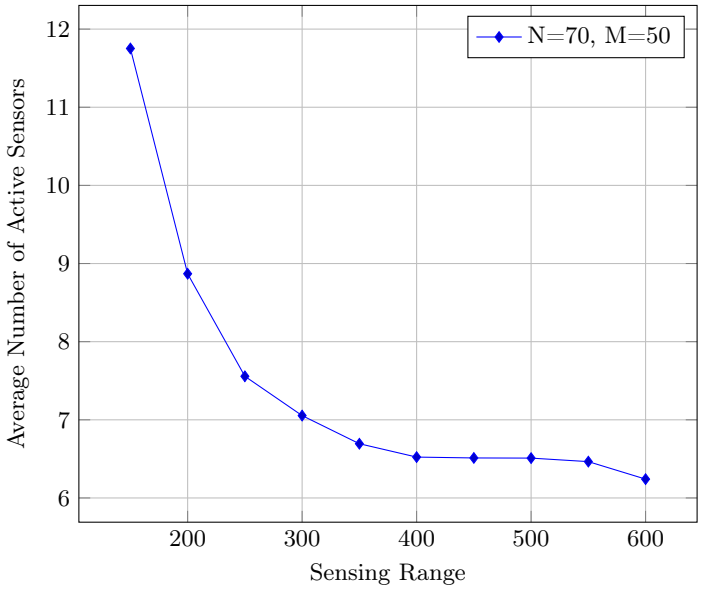

Fig. 3: A network consisting of 70 sensors and 50 targets with varying sensing range from 150m to 600 m

4.2 Effect of the Learning Parameter

The main goal of this particular experiment is to examine the impact of the choice of the learning parameter λ on the quality of the final solution and on the convergence speed.

We vary the number of sensors between 40 and 80 while fixing the number of targets to 30. Every sensor is provided with a sensing range of 400m. The experiment is carried out by taking different values of the learning parameter lambda, "λ". Here, the value of lambda ranges from "λ= 0.9", "λ=0.99", "λ= 0.999" to "λ=0.9999". The results are shown in Table 3.

From Table 3, we observe that the average of the minimum number of active sensors decreases as the value of the learning parameter increases. In other terms, the quality of the obtained solution improves as the learning parameter increases. However, this comes at the cost of the convergence speed measured in terms of the number of iterations. In fact, as we increase the learning parameter, we observe the required number of iterations for reaching convergence increases too.

The results of this experiment are depicted in Fig. 4, which shows the bar-graph plotting of different numbers of sensors nodes at different learning parameters. We can observe that, if the number of sensors is increased in the deployed environment, then the complexity of the problems also increases, and therefore one needs a larger value of learning parameter to achieve the optimum result.

Table 3 Results obtained with varying learning parameter lambda "λ" with sensors between 40 and 80 including 30 targets and sensing range 400m

Number of Sensors	λ= 0.9 Average Number of Active Sensors	λ= 0.99 Average Number of Active Sensors	λ= 0.999 Average Number of Active Sensors	λ= 0.9999 Average Number of Active Sensors
40	9.844	2.912	1.735	1.345 ☑
50	13.189	3.726	1.851	1.436 ☑
60	16.926	5.082	1.967	1.483 ☑
70	20.599	6.635	2.176	1.508☑
80	24.049	7.576	2.078	1.526 ☑

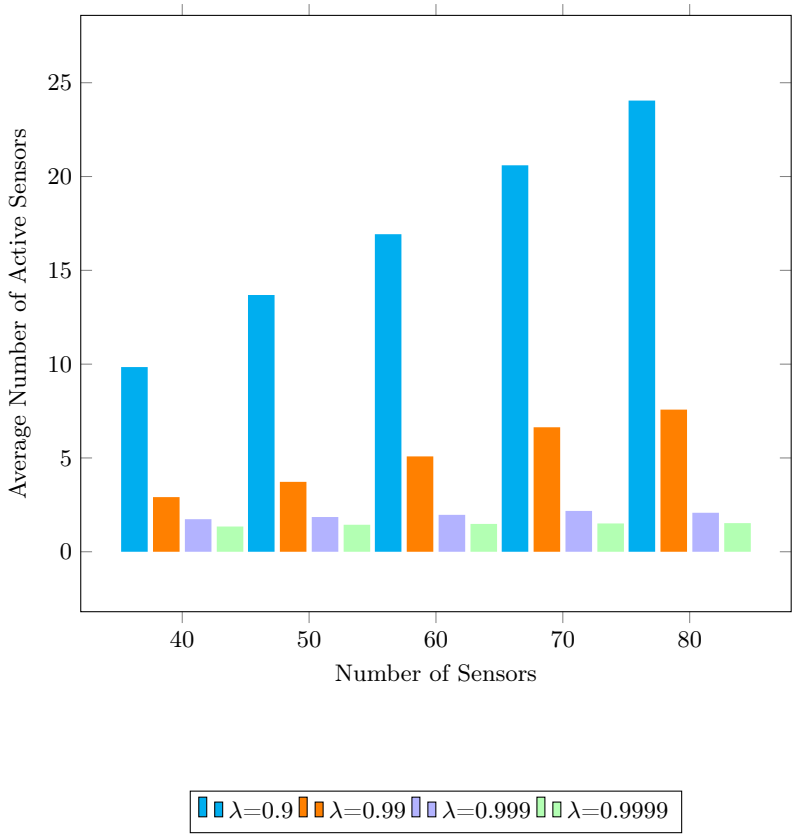

Fig. 4: Bar plot of the data obtained in Table 3 showing the effect of increasing value of learning parameter

5 Conclusion

This paper focused on solving the target coverage problem in WSN. The sensors can select their state to be either active or sleep autonomously using the concept of pursuit LA. Comprehensive experiments were carried out to evaluate the performance of our designed algorithm. The proposed algorithm provided a methodology to find the minimum number of active sensors to cover the targets and thus addressed the issue of energy-efficient target covering in the WSN.

References

1. Agache, M., Oommen, B.J.: Generalized pursuit learning schemes: New families of continuous and discretized learning automata. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **32**(6), 738–749 (2002)
2. Cardei, M., Du, D.Z.: Improving wireless sensor network lifetime through power aware organization. Wireless networks **11**(3), 333–340 (2005)
3. Cardei, M., Thai, M.T., Li, Y., Wu, W.: Energy-efficient target coverage in wireless sensor networks. In: Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. vol. 3, pp. 1976–1984. IEEE (2005)
4. Chand, S., Kumar, B., et al.: Target coverage heuristic based on learning automata in wireless sensor networks. IET Wireless Sensor Systems **8**(3), 109–115 (2018)
5. Chen, J., Li, J., Lai, T.H.: Trapping mobile targets in wireless sensor networks: An energy-efficient perspective. IEEE Transactions on Vehicular Technology **62**(7), 3287–3300 (2013)
6. Diop, B., Diongue, D., Thiare, O.: Target coverage management in wireless sensor networks. In: 2014 IEEE Conference on Wireless Sensors (ICWiSE). pp. 25–30. IEEE (2014)
7. Goodwin, M., Yazidi, A.: Distributed learning automata-based scheme for classification using novel pursuit scheme. To appear in Applied Intelligence (2019)
8. Jamali, M.A., Bakhshivand, N., Easmaeilpour, M., Salami, D.: An energy-efficient algorithm for connected target coverage problem in wireless sensor networks. In: 2010 3rd International Conference on Computer Science and Information Technology. vol. 9, pp. 249–254. IEEE (2010)
9. Kittur, R., Jadhav, A.: Enhancement in network lifetime and minimization of target coverage problem in wsn. In: 2017 2nd International Conference for Convergence in Technology (I2CT). pp. 1150–1157. IEEE (2017)
10. Lakshmivarahan, S.: Learning Algorithms Theory and Applications. Springer-Verlag (1981)
11. Lu, Z., Li, W.W., Pan, M.: Maximum lifetime scheduling for target coverage and data collection in wireless sensor networks. IEEE Transactions on vehicular technology **64**(2), 714–727 (2014)
12. Mini, S., Udgata, S.K., Sabat, S.L.: Sensor deployment and scheduling for target coverage problem in wireless sensor networks. IEEE sensors journal **14**(3), 636–644 (2013)
13. Mostafaei, H., Meybodi, M.R.: Maximizing lifetime of target coverage in wireless sensor networks using learning automata. Wireless Personal Communications **71**(2), 1461–1477 (2013)
14. Najim, K., Poznyak, A.S.: Learning Automata: Theory and Applications. Pergamon Press, Oxford (1994)

15. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice-Hall, Inc. (1989)
16. Narendra, K.S., Thathachar, M.A.: Learning automata-a survey. IEEE Transactions on systems, man, and cybernetics **SMC-4**(4), 323–334 (1974)
17. Oommen, B.J., Lanctôt, J.K.: Discretized pursuit learning automata. IEEE Transactions on systems, man, and cybernetics **20**(4), 931–938 (1990)
18. Poznyak, A.S., Najim, K.: Learning Automata and Stochastic Optimization. Springer-Verlag, Berlin (1997)
19. Thathachar, M.A.L., Sastry, P.S.: Networks of Learning Automata: Techniques for Online Stochastic Optimization. Kluwer Academic, Boston (2003)
20. Tubaishat, M., Madria, S.: Sensor networks: an overview. IEEE potentials **22**(2), 20–23 (2003)
21. Wu, Y., Fahmy, S., Shroff, N.B.: Optimal qos-aware sleep/wake scheduling for time-synchronized sensor networks. In: 2006 40th Annual Conference on Information Sciences and Systems. pp. 924–930. IEEE (2006)
22. Zhang, X., Granmo, O.C., Oommen, B.J.: On incorporating the paradigms of discretization and bayesian estimation to create a new family of pursuit learning automata. Applied intelligence **39**(4), 782–792 (2013)
23. Zou, Y., Chakrabarty, K.: A distributed coverage-and connectivity-centric technique for selecting active nodes in wireless sensor networks. IEEE Transactions on Computers **54**(8), 978–991 (2005)