Received: 22 August 2021 | Revised: 14 November 2021 | Accepted: 2 December 2021

DOI: 10.1049/cmu2.12323

ORIGINAL RESEARCH PAPER

Solving the target coverage problem in multilevel wireless networks capable of adjusting the sensing angle using continuous learning automata

Azam Qarehkhani[1] | **Mehdi Golsorkhtabaramiri**[1] | **Hosein Mohamadi**[2] | **Meisam Yadollahzadeh-Tabari**[1]

[1] Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol, Iran

[2] Department of Computer Engineering, Azadshahr Branch, Islamic Azad University, Azadshahr, Iran

Correspondence
Mehdi Golsorkhtabaramiri, Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol, Iran
Email: golesorkh@baboliau.ac.ir

Abstract

Today, a directional sensor network is a popular environment for solving the target coverage problem. Monitoring all targets in a DSN is a crucial challenge to scholars working in this field of study. Adjusting the angle and range of the sensors can be an efficient technique for improving the network performance. In this way, the network has the most extended lifespan and, at the same time, spends the least time to find the best cover set. In this method, each sensor dynamically adjusts its own sensing angle in order to find the targets by choosing the best range. The present study proposed a continuous learning automata-based method to choose the optimum sensing angle for the sensors in a DSN. Then, to evaluate the proposed algorithm performance, its results were compared to those of a conventional automata-based method whose algorithm worked based on continuous automata. The comparative analysis confirmed the superiority of the proposed method over the conventional automata-based method regarding the extension of the network lifespan.

1 | INTRODUCTION

In recent years, the developments of micro-electro-mechanical systems, wireless communications, and sensor technology have enabled the rapid development of sensor nodes and the production of small, low-cost sensors. The sensor technology integrates measurement, data processing, and wireless communications operations [1,2]. Today, wireless sensor networks (WSNs) are used for different purposes such as environmental monitoring, battlefield monitoring, medical equipment etc. A WSN consists of many sensors located in different geographical locations. In general, these sensors have disk-like sensing ranges [2]. However, in real-world applications, sensor nodes may have limited sensing angles; these sensors form directional sensor networks (DSNs) [3]. The problem of coverage in DSNs, which refers to target area monitoring, is one of the critical issues in estimating network lifespan referring to the network's ability to cover a specific area or event. The coverage problem is generally divided into three categories: coverage of an area, coverage of border, and coverage of targets. The latter refers to covering a set of stations or moving points in the sensing range [4].

This paper focuses on the target coverage problem to cover most targets with the minor sensor overlap. The scenario addressed in this study can be expressed in this way: Assume there are m targets with known locations, which are monitored by n sensors with adjustable sensing angles and ranges. Each sensor has a limited battery lifespan, and batteries cannot be recharged. One way of increasing the lifespan of the network is to schedule the network sensors' activities to determine the activity and inactivity of the nodes [5]. At each unit of time, every sensor in the network is either active or passive. The amount of energy that an active sensor consumes is dependent on the sensing range of the sensor. Remember that passive sensors consume no energy. The scheduling technique, which is used to reduce power consumption and increase the network lifespan with minimal overlap, schedules each sensor to be activated or deactivated. Each sensor monitors the targets through its work direction and sensing range. Sensors may collect different information as needed; information may include temperature, vibration, humidity, or other types of information such as audio or video. In directional sensor networks, sensors usually monitor targets that are in the direction of operation and their sensing range using visual, infrared, or ultrasound cameras.

1.1 | Our contribution

Despite numerous methods proposed so far to solve the target coverage problem, this paper proposes an algorithm based on continuous learning automata to determine the sensors' optimal sensing angles to find a new solution to the target coverage problem. In the current paper, a continuous action-set learning automaton[1] was used to determine the appropriate sensing angle of each sensor. The sensing angle was considered as the main parameter of the problem, for which automata were created. After defining a team of CALA, each automaton in this team randomly selects an action from the continuous action set, independent of all other automata. The selected action is first applied to the random environment; then, the random environment feeds the amplification signals back to LA. Each automaton uses the signals to update the sensing angle of the corresponding sensor. This learning process is performed iteratively until the optimal set of actions is obtained from the angle of each sensor for sensing the targets. In this study, the network lifespan, energy consumption, number of cover sets, and runtime were calculated. A comparison was made between the results of the proposed method and those of an algorithm based on conventional learning automata, which was extracted from the existing literature.

The remaining parts of the paper are organized as follow. In Section 2, the literature on the solution of the target coverage problem in sensor networks is reviewed. In Section 3, the problem of maximizing the network lifespan with adjustable sensing angles in sensors is discussed in detail. Then, Section 4 explains the algorithm of continuous action-set learning automata. In Section 5, the CALA algorithm is proposed to calculate the best starting angle to solve the target coverage problem. Section 6 describes the experiments executed in this study for the evaluation of the proposed algorithm performance. Finally, Section 7 presents the final results and concludes the whole study.

2 | LITERATURE REVIEW

Some issues such as network connections, the extension of network lifespan, energy consumption, and the number of cover sets are the main challenges to DSNs [6]. Coverage is one of the most significant performance criteria for a sensor network [7]. In general, coverage in a sensor network determines how the field of sense is controlled or tracked by sensors [8].

Cardei et al. first introduced the problem of target coverage for Omni-directional sensor networks [9]. Cardei and Du [6] proposed a greedy algorithm that worked based on the target scheduling method. This algorithm was aimed at solving the MSCD problem and also finding the disjoint set covers (DSC). In this algorithm, each cover set works until the whole energy of its sensors is completely discharged. Then, due to the lack of energy, the sensors in this cover set cannot participate in the subsequent cover sets. The set of non-disjoint sensors in the study of Cardei et al. [9] was modelled as non-disjoint covers set so that each cover set could work for a fixed period of time and consume a part of its energy proportional to *r*. Then, it can participate in forming the next cover set if the remaining energy of each sensor is not zero [3]. They proved the NP-hardness of the problem and proposed two linear programming and greedy algorithms to solve it. In addition, they showed that the complexity of computations and runtime of the greedy algorithm were less than those of the linear programming method. In addition, the network lifespan in the greedy algorithm was longer than the linear programming.

The multiple directional cover sets[2] problem [10] is defined as the organization of sensors and their working directions into non-disjoint cover sets. Each cover set can cover all targets in the network. In each cycle, the generated cover sets are activated to increase the lifespan of the network. Yang et al. [11] hypothesized that each target in a network may require different coverage, and the distance between a target and the corresponding sensor affects the quality of the coverage. Katie in [12] suggested a high energy efficiency-based algorithm to construct disjoint and non-disjoint cover sets to provide the coverage required by the targets. The algorithm conserves the number of targets between the sensors and sinks to discover the optimal path and save the energy of the sensor nodes. The researchers in [13] presented the optimal scheduling method using the ILP approximation algorithm and the Greedy-MCSS algorithm to find the best cover sets. The maximum lifespan target coverage (MLTC) algorithm calculates the upper bound by observing the least covered targets by the minimum number of sensors and designing subsets to monitor the targets repeatedly. Mao et al. [14] considered the localization techniques to solve the random placement of sensors, which is formulated as the k-cover problem. The k-cover problem considers the coverage of each target in the network by multiple sensors, which increases the complexity of the algorithm.

In [15], the network area is geographically gridded into square cells. The node with the maximum energy is selected as the aggregator in each cell. Given the remaining energy and level of coverage, the firefly algorithm works to select and preserve the most appropriate node in each cell and inactivate the other nodes. This leads to improving the lifespan and coverage of WSNs. In [16], the researchers proposed a hierarchical clustering algorithm based on selecting cluster-head

[1] CALA

[2] MDCS

and cluster members in WSNs, which was called HCABS. The HCABS algorithm resulted in increased lifespan and reduced energy consumption, latency, and overhead across the network.

Liu and Ma [17] were the first researchers who introduced the concept of DSN and provided two schemas to increase the rate of coverage in these networks. In addition, the two schemes of Distributed Maximal Rotatable Angle[3] and Maximal Distributed Coverage First[4] were proposed by Hsu et al. [18] to maximize the number of covered targets and minimize the rotation angles of the sensors. Casta et al. [19] replaced the defective sensors with inactive sensors in DSNs, which caused an increase in the network lifespan. The authors in [20] proposed a genetic-based algorithm to solve the target coverage problem in WSNs. Their algorithm puts several sensors in each cover set to provide coverage to monitor all targets in the network. To form the best cover sets, Manju et al. chose the least number of sensors with the maximum remaining energy.

Moreover, in [21], a priority-based genetic algorithm was proposed to activate the least number of sensors in a DSN. As a solution to the coverage problem, Razaly et al. [22] suggested prioritizing the sensors to cover the targets. They also assumed that each sensor could adjust its working direction and sensing angle. They developed a greedy algorithm and an approach based on LA to solve the priority-based target coverage problem in adjustable ranges networks.

In another research, Zarei and Bag-Mohammadi [23] attempted to solve the priority-based target coverage problem in DSNs. To begin with, their proposed algorithm, that is, Cover Critical Target First[5], categorizes the targets in the network based on the number of sensors that cover them. After that, CCTF monitors critical targets to increase the coverage rate. CCTF does not have any sensitivity to the localization error and sensor orientation error. The authors in [23] supposed that if the two main ideas, that is, the critical target concept and the sensor activation sequence, are combined with each other, it can lead to target coverage with higher quality. To find an effective solution to the coverage problem in DSNs, Mohammadi et al. [24] designed a new LA-based algorithm. In their algorithm, at each phase, the optimum working direction for the coverage is chosen with this assumption that the coverage quality is not influenced by the distance between the targets and sensors within the network. The authors in [25] used a genetic algorithm to examine the coverage problem in DSNs where sensors have adjustable sensing ranges. Further, in [26], the target coverage problem and k-coverage in DSNs were discussed, and LA was used to solve the problem. In all of the above cases, the initial sensing angle of the sensors is set to a predefined value (usually zero) at the design time. In contrast, in the algorithm proposed in the present paper, the sensing angle of each sensor is defined using statistical features and continuous automata.

[3] DMRA
[4] MDCF
[5] CCTF

3 | STATEMENT OF PROBLEM

The following scenario was designed based on the studies discussed in the previous section to monitor targets with the least number of sensors, increase the network lifespan, and reduce the network runtime. Several targets were randomly scattered in a 2D Euclidean space. Several directional sensors were also deployed in the environment to monitor the targets continuously. Each directional sensor in this network had several directions, with only one active direction per unit of time, called the working direction. First, the positions of the sensors and targets were determined using GPS. Then, multi-level sensors were used since energy consumption depends on the sensing range. In addition, the optimal starting angle was obtained by calculating the angle of each sensor (θ) to the targets in the sensing range using continuous automata. Each active sensor consumes a certain amount of energy based on its sensing range [27]. The sensors were homogeneous in terms of initial battery energy and sensing range. Since the power level gradually increased the sensing range of the device, the formula $T_{(d_{i,j},b)} \subseteq T_{(d_{i,j},a)} \forall\, b \in \{1, \ldots, a-1\}$ was used for each directional sensor $d_{i,j}$ and for each level $a > 1$. For each sensor, a is the selected minimum energy level. At each stage, based on the selected sensing direction and radius, the sensor checks whether the target is being monitored or not.

The main problem is how to determine the best sensing angle for each sensor. The sensor directions are divided based on the calculated angle. If an appropriate sensing angle is selected using a continuous automaton and, based on that, an appropriate direction of work and finally a proper sensing range are selected, fewer sensors will be activated in a cover set. Reducing the number of active sensors leads to a reduction in energy consumption in a cover set. In addition, inactive sensors at this stage can be used in another cover set, which leads to an increase in the number of cover sets and ultimately this can lead to the maximization of the network lifespan and also the reduction of runtime.

Definition 1. The nature of the network for creating cover sets is dynamic nature. The sensing angle is obtained dynamically using continuous automata, and the locations of the sensors and targets are static in the simulation.

Definition 2. Remaining sensor lifespan is the energy level of each sensor at any given time.

Definition 3. Runtime is the time it takes for the cover sets to be constructed and the simulation to be completed.

Definition 4. Coverage energy is the ratio of all the targets that are being monitored by the direction of the adjusted sensor $(d_{i,j}, a)$ to the energy consumption rate Δ^a.

Definition 5. The measuring radius is defined based on the energy level a. If p is considered as the total defined energy levels, the radius of sensing is obtained using $\text{radius} = RS \times a/p$.

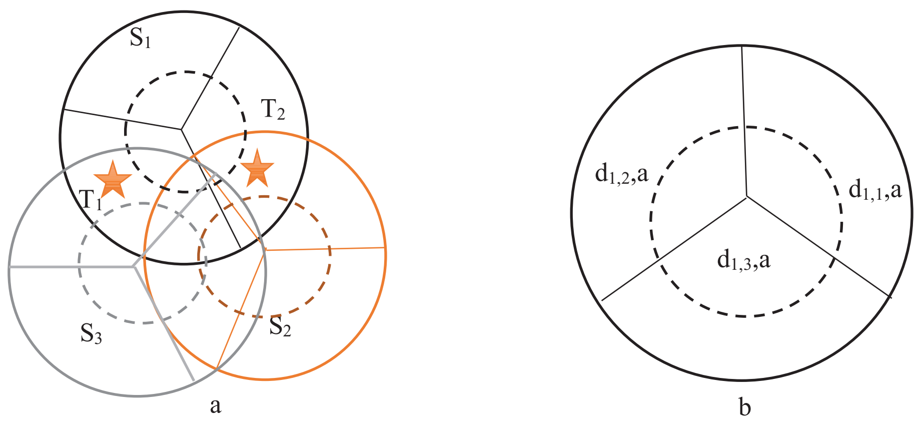

FIGURE 1 (a) A network with three directional sensors, two targets, and energy level dependent on a sensing initial angle $\theta_1 = 150°$, $\theta_2 = 0°$, $\theta_3 = 45°$. (b) A 3-direction sensor with two power levels.

TABLE 1 Scenario parameters

Notation	Meaning
M	the number of targets
N	the number of sensors
W	the number of directions per sensor
θ	initial sensing angle
a	the number of power levels, $a \geq 1$
t_m	the m-th target, $1 \leq m \leq M$
s_i	the i-th sensor, $1 \leq i \leq N$
l_{s_i}	the lifespan of sensor s_i
$d_{i,j}$	the j-th direction of the i-th sensor, $1 \leq i \leq N$, $1 \leq j \leq W$
D	the set of the directions of all the sensors,
T	$\{d_{i,j} \mid i = 1 \dots N,\ j = 1 \dots w\}$
S	the set of targets, $t_1, t_2, \dots, t_m$
$T(d_{I,j}, a)$	the set of sensors, $s_1, s_2, \dots, s_m$
$(d_{i,j}, a)$	all the targets covered by sensor direction $(d_{i,j}, a)$ when it is set at
	level aSensor direction $(d_{i,j}, a)$ that is activated at level a. This
	pair also is defined as adjusted sensor

Figure 1 shows the way of defining the direction of a sensor. Several parameters are given in Table 1 for a better comprehension of the scenario.

4 | CONTINUOUS ACTION-SET LEARNING AUTOMATA

4.1 | Preliminaries and properties of LA methods

As stated in the Related Work section, much research has been done to find proper solutions to the problem of target coverage in WSNs with the use of Omni-directional and directional sensors. In the following, the methods mentioned in that section are discussed in terms of their drawbacks.

1. In [13], the linear programming-based methods were used to solve the problem of target coverage. In this method, the LP-based algorithms require heavy computational overhead and long running time.
2. Han et.al [28] proposed two algorithms, a Greedy algorithm and a genetic algorithm. The Greedy-based algorithm can find a faster solution than other exploratory algorithms; however, it may fail to find an optimal solution due to the local search of this algorithm.
3. Some of the features of a learning automaton that make it suitable to many applications include the following [29]:
   I. It can be used without any priori information about the underlying application.
   II. It is useful for applications with large amounts of uncertainty.
   III. It has simple structure and easy software and hardware implementation.
   IV. It requires a very low and simple feedback from the environment.
   V. The action set of LA can be a set of symbolic or numeric values.
   VI. Optimization of learning automata algorithms does not require an objective function to analyse customizable parameter functions.
   VII. A learning automaton requires little mathematical operation, which makes it useful to real-time applications.
   VIII. It has flexibility and analytical transfer tractability, which are required for most applications.
4. One of the restrictions of LA appears when it is to optimize a continuous function. LA needs to discrete the parameter space so that LA actions can obtain the possible values of the relevant parameters. With finer discretization, the number of actions increases, which causes an improvement in the accuracy of the solution. On the other hand, increasing the number of actions reduces the convergence of the learning algorithm.
5. The techniques based on the nature-inspired meta-heuristic methods (e.g. GA, Greedy etc.) use encoding methods such as chromosome encoding to find the optimum set of coverage. The complexity of these algorithms is affected by the number of chromosomes used. The main score of the CALA-based algorithms is the use of only one CALA group

(equivalent to one chromosome), which reduces the complexity of the algorithm [30].

6. In CALA, because each automaton can update its parameters independently of all other automata, calculations can be performed in parallel and with distributed methods.
7. The main difference between traditional learning methods and amplification learning methods is that amplification learning does not require information about the process of decision making.

4.2 | CALA theory

Learning Automata refers to a self-adaptive machine learning where the optimum action can be marked out from among a set of actions that are constantly interacting with the random environment [31–33]. The learning operation begins with selecting an available action based on the probability vector of the action. At each stage, the probability vector of the action is updated in interaction with the environment and receives an amplification signal. LA aims to obtain the optimum action from the set of available actions so that the minor penalty could be received from the environment. In general, learning automata can be divided into FALA[6] and CALA [34, 35].

The increase of the number of actions per automaton with finer discretization is thought to enhance the accuracy of the problem-solving process. Furthermore, an increase in the number of actions in the automaton causes a reduction in the convergence speed of the learning schema. To manage this problem, Santharam et al. [36] introduced CALA. Remember that in an automaton with a continuous action-set, the probability distribution of the action cannot be denoted by a limited vector of values as the traditional learning automata model. Therefore, the authors in [36] offered the normal probability distribution function with mean μ_k and standard deviation σ_k for the continuous automata at the moment $k > 0$. In [37], Howell et al. developed the CARLA[7] model where the action-set $[\alpha_{\min}, \alpha_{\max}]$ is defined as a continuous environment with real R values. The actions in their method are chosen with the use of a continuous probability density function. They assumed that at the beginning of the learning process, any information does not exist regarding the priority of the actions. As a result, the initial distribution of the density function can be taken into account as the probability of a uniform distribution.

A set of learning automata for function optimization is defined as f: $R^m \rightarrow R$, each of which is proposed with a continuous action-set. This function has m parameters, including $y = y^1, \dots, y^m \in R^m$, and the ith parameter is expressed by y^i. The A^i automata is defined in order to identify the i parameter of y^i where $i = 1, \dots, m$ [30]. Figure 2 shows how CALA interacts with the random environment.

Santharam et al. [36] defined the CALA algorithm to find the minimum value of f as follows. At the beginning of the learning process, it is assumed that there is no information about the priority of actions. Therefore, the probability of selecting all actions is considered to be the same. Therefore, a uniform distribution is applied to the first distribution of probability density function. Then, at instant n, each automaton selects an action based on the Gaussian distribution function $N(\mu_n^i, h[\delta_n^i])$ independently of other automata. The selected actions are given to the random environment, and amplification signals are given to automata by the environment.

The learning algorithm is presented as follows:

1. The mean and variance of the Gaussian distribution are represented by μ_0 and δ_0, respectively, which are defined by default values.
2. A random value of $\alpha_n \sim N(\mu_n^i, h[\delta_n^i]$ is defined as real value, where $N(\mu_n^i, h[\delta_n^i])$ is a function of the Gaussian distribution with mean μ_n^i and standard deviation $h\,[\delta_n^i]$. The function $h(\alpha)$ is defined at step 4.
3. $\beta_{\alpha_n}$and β_{μ_n} are the amplification signals received from the environment. Let $\beta_{\alpha_n} = f(\alpha_n)$ and $\beta_{\mu_n} = f(\mu_n)$ be the function assessment at points α_n and μ_n, respectively.
4. The variance and mean of the Gaussian distribution are updated with the formulas (1)–(5).
5. Go to step 2 until δ_n converges to δ_1.

$$\mu_{n+1}^i = \mu_n^i - \gamma f_1^i\ (\mu_n, \delta_n, \alpha_n, \beta_{\alpha_n}, \beta_{\mu_n}) \tag{1}$$

$$\delta_{n+1}^i = \delta_n^i - \gamma f_2^i\ (\mu_n, \delta_n, \alpha_n, \beta_{\alpha_n}, \beta_{\mu_n}) + c\,[\delta_l - \delta_n^i] \tag{2}$$

where

$$f_1^i\ (\mu_n, \delta_n, \alpha_n, \beta_{\alpha_n}, \beta_{\mu_n}) = \left(\frac{\beta_{\alpha_n} - \beta_{\mu_n}}{h\,[\delta_n]}\right)\left(\frac{\alpha_n^i - \mu_n^i}{h\,[\delta_n^i]}\right) \tag{3}$$

$$f_2^i\ (\mu_n, \delta_n, \alpha_n, \beta_{\alpha_n}, \beta_{\mu_n}) = \left(\frac{\beta_{\alpha_n} - \beta_{\mu_n}}{h\,[\delta_n]}\right)\left[\left(\frac{\alpha_n^i - \mu_n^i}{h\,[\delta_n^i]}\right)^2 - 1\right] \tag{4}$$

$$\text{For } \delta, \delta_l \in R\ h\ [\delta] = \begin{cases} \delta_l & for\ \delta \le \delta_l \\ \delta & for\ \delta > \delta_l \end{cases} \tag{5}$$

where $C > 0$ is a constant value and γ shows the learning rate is positive and non-zero. Further, δ_1 is selected as small as possible so that the solution could be close to the minimum. After performing the update step, the learning algorithm is repeated until the minimum value of β_{μ_n} is obtained. If $\beta_{\mu_\text{n}} = 0$, only one function can be used to evaluate the performance [29].

[6] Finite Action-set Learning Automata
[7] Continuous Action-set Reinforcement Learning Automata

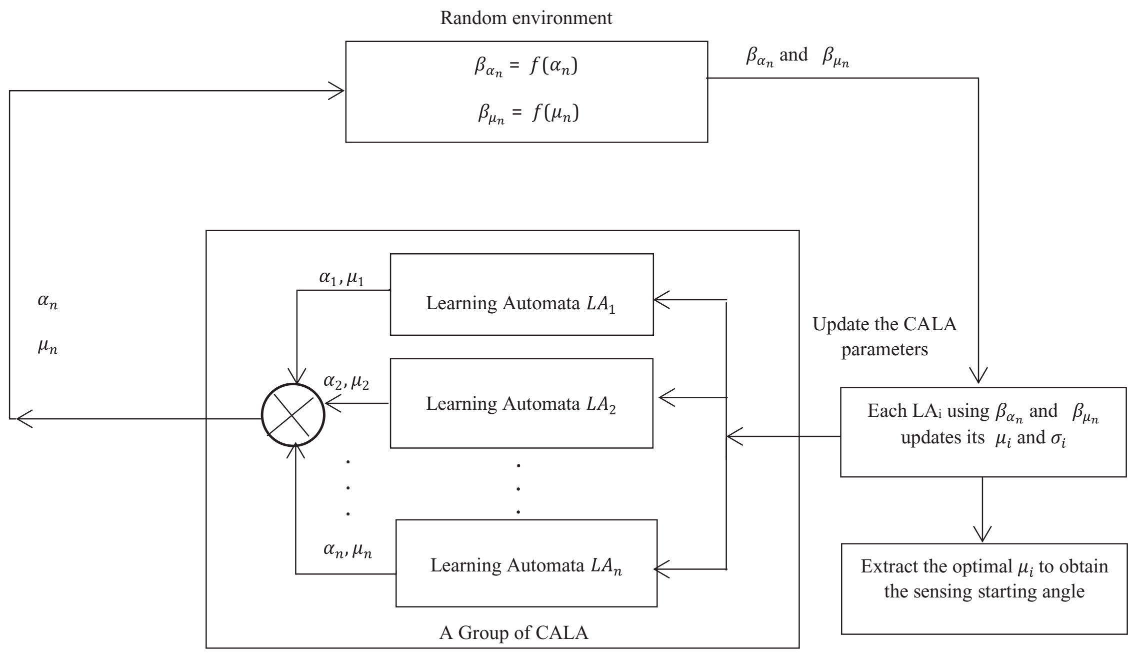
Random environment
$\beta_{\alpha_n} = f(\alpha_n)$
$\beta_{\mu_n} = f(\mu_n)$
β_{α_n} and β_{μ_n}
α_n
μ_n
α_1, μ_1
Learning Automata LA_1
α_2, μ_2
Learning Automata LA_2
α_n, μ_n
Learning Automata LA_n
A Group of CALA
Update the CALA parameters
Each LAi using β_{α_n} and β_{μ_n} updates its μ_i and σ_i
Extract the optimal μ_i to obtain the sensing starting angle

FIGURE 2 CALA interacts with the environment to obtain the optimal sensing starting angle. First, the initial sensing angle θ is obtained randomly using the uniform distribution function. The obtained θ is applied as α to a random environment. In the interaction between the random environment and CALA, the α_n factor is selected at the n-th step. This selected action is given to the Gaussian distribution function, where at the first step, μ_0 and δ_0 are given with default values, then they are updated using Equations (1)–(5), μ_n and σ_n. These steps continue until σ_n converges to σ_1.

5 | PROPOSED METHOD

A non-disjoint scheduling algorithm is presented in this section based on CALA with the aim of solving the target coverage problem. The network that operates in this algorithm is executed in several rounds, each resulting in several cover sets. Each cover set formed in this phase is capable of covering all the targets existing in the network. The proposed method first selects the starting sensing angle (θ) randomly by using a uniform distribution; then, it obtains the standard deviation and means using the Gaussian distribution function and updates θ accordingly.

Each cycle of the algorithm involves adjusting the working direction, forming a cover set, and monitoring operation. A CALA-based algorithm is initially developed to select the starting sensing angle θ, based on which the optimal sensing range and the working direction of each sensor are chosen. In the next phase, the sensor and sensing range directions are organized based on LA in several sensing sets. Finally, the cover sets are activated one after the other to perform the defined monitoring tasks. Creating cover sets is continued until all the targets in the network are completely monitored by the sensors' directions. The following subsections examine these phases.

5.1 | Adjusting the sensing angle and range

This phase proposes a CALA-based algorithm in order to determine the initial angle of measurement θ for each sensor, upon which the defined angle of the sensing range is obtained so that the maximum number of targets could be monitored. The proposed algorithm consists of two steps: The initialization of angle of working direction and adjustment of sensing range.

At the initialization step, the algorithm constructs a network of CALA, forms the action set of CALA, and adjusts the probability vector of the action.

To build a network of CALA, an LA is first created for each T_i target. The network can be constructed by a (A, α) pair, where $A = \{A_{i,j} \mid \forall t_i \in T,\ 1 \le j \le k\}$. There is a set of learning automata corresponding to the targets in the network, and $\alpha = \{\alpha_{i,j} \mid \forall A_{i,j} \in A\}$ gives a set of actions of learning automata. $\alpha_{i,j} = \{\alpha^1_{i,j}, \alpha^2_{i,j}, \ldots, \alpha^{ri}_{i,j}\}$ defines the set of actions in which the $A_{i,j}$ learning automaton can select any $\alpha_{i,j} \in \alpha$, and r_i is a coordinate of the set of action $\alpha_{i,j}$, which depends upon the minimum number of the sensor directions that cover the targets corresponding to learning automata. As mentioned earlier, each target is equipped with $A_{i,j}$ learning automata from a continuous action set. The action set of the automata is in the range of real values between zero and 360°. Before starting the learning process, the probability density function uses the uniform distribution stated in Equation (6) to create the network since there is no information about the network. Then, the parameters μ_n and δ_n are updated in feedback from the environment using the Gaussian distribution function N (μ_n, δ_n). This process continues until the selected action converges to the optimal action, for which δ_n must be small enough to converge to zero. In this case, μ^k_i converges to θ,

ALGORITHM 1 Generation of the starting angle of sensing θ using CALA

```
01. Inputs:
02. Initialize the algorithm parameters such as Sensor $S_i$, maxiter, Itr,
      $\beta_{\alpha_n}$, $\beta_{\mu_n}$
03. Output:
04. Adjusting sensing angle ($\theta$)
05. Assumptions:
06. Let $A_i$ be the continuous action-set learning automata associated with
      sensor $S_i$
07. Let $N(\mu_i^k, \sigma_i^k)$ define the Gaussian distribution over the action-set of
      $A_i$
08. Let $\beta_{\alpha_n}$, $\beta_{\mu_n}$ be the reinforcement schema generated in response to
      the environment at stage $i$
09. Begin Algorithm
10. Repeat
11.   Automaton $A_i$ selects one of its actions- $\theta$ according to $N(\mu_i^k, \sigma_i^k)$
        using Equations (1) and (2)
12.   The environment triggers a feedback $\beta_{\alpha_n}$, $\beta_{\mu_n}$
13.   Update $\mu_n^i$ and $\alpha_n^i$ with Equations (3) and (4)
14.   If ($\sigma^2 < \varepsilon$)
15.     Reward the selected actions of activated automata
16      $\theta = \mu_n$
17.   Else
18.     Penalize the selected actions of the activated automata
19.   End If
20. until (itr $\neq$ maxiter)
21. end Algorithm
22. The obtained value of $\theta$ is applied to Algorithm 2
```

where θ is the optimal sensing angle. Here, $s\ (\theta - \frac{\pi}{3}, \theta + \frac{\pi}{3})$ is the optimal sensing direction, which is selected as the first working direction. Since the sensors are multilevel, the appropriate sensing range based on the working direction of each sensor is selected. This process is repeated for each target until a network of CALAs is created. Algorithm 1 shows how θ is calculated.

$$p\ (1) = \frac{1}{2\pi} \tag{6}$$

After the creation of a CALA network, the algorithm begins the formation of its action set. To this end, each learning automaton creates the action sets by achieving its corresponding sensor's working direction and sensing range.

Suppose that $\alpha_i = \{ \alpha_i^j \mid (d_{i,j},\ a)\}$ is used for the selection of the sensor set direction within a range where α_i^j is corresponding to the selection of the sensing direction $(d_{i,j},\ a)$. When LA forms the action set, the probability vector of LA action is set. Further, a set of action vectors of the $A_{i,j}$ learning automata is given by the formula $P = \{P_{i,j} | \forall \alpha_{i,j} \in \alpha\}$, where $P_{i,j}^r$ corresponds to the probability of selecting $\alpha_{i,j}^r$.

$$P_i^j = \frac{CP\left(d_{i,j}, a\right)}{\sum CP\left(d_{i,j}, a\right)} \tag{7}$$

where $CP(d_{i,j}, a)$ is the coverage energy of the sensor based on the direction and level of energy consumed by each sensor, which means the number of targets covered by the $d_{i,j}$ direction at a level, and $\sum CP(d_{i,j}, a)$ is the total coverage energy of the sensor expressed based on all directions of the sensor that monitors the targets.

For more explanation, see the example given in Figure 3. The first time the sensors are placed in the network environment, the starting angle is selected randomly with a uniform distribution. Suppose for the first time, $\theta_{S_1} = 20°$, $\theta_{S_2} = 80°$, $\theta_{S_3} = 170°$, $\theta_{S_4} = 60°$ are selected (Figure 3a). Then, using the continuous automaton and the Gaussian distribution function, the starting angle of the sensor is updated based on the targets in the R range ($R = 100$). Here, $\theta = \mu$ is considered. In this example, the starting sensing angle of each sensor is updated as $\theta_{S_1} = 245°$, $\theta_{S_2} = 140°$, $\theta_{S_3} = 160°$, $\theta_{S_4} = 40°$ (Figure 3b); the initial sensing angle range for each sensor is set using ($\theta - \frac{\pi}{3}$, $\theta + \frac{\pi}{3}$) as follows:

$$\text{angle}_{S_1} = (185,\ 305)\ ,\ \text{angle}_{S_2} = (80,\ 200)\ ,$$

$$\text{angle}_{S_3} = (100,\ 220)\ ,\ \text{angle}_{S_4} = (340,\ 100)$$

It should be noted that if the starting point in a range is larger than the end point, the value of 360 should be reduced from the starting point, here $340 - 360 = -20$; thus, the sensing angle range S_4 is $\text{angle}_{S_4} = (-20,\ 100)$. If the sensors have 120° sensing angle range, then each sensor has three directions.

5.2 | Selection of working direction and sensing range

The proposed algorithm in this section identifies the best working direction based on the θ angle. Algorithm 2 displays the pseudo-code of the proposed algorithm. The operation of this algorithm is done through a number of steps. During these steps, the algorithm chooses the corresponding automata associated with each operating sensor from the set of actions considering the probability vector of the action. This means that the learning automata obtain the best angle θ based on the Gaussian distribution function. The probability vector of the action then updates each automaton based on the selected action, upon which the optimal working direction $(\theta - \frac{\pi}{3}, \theta + \frac{\pi}{3})$ is selected as the initial direction. Then, the sensing range is obtained based on a, which is also the level of energy consumption. Equation (8) is used to obtain the

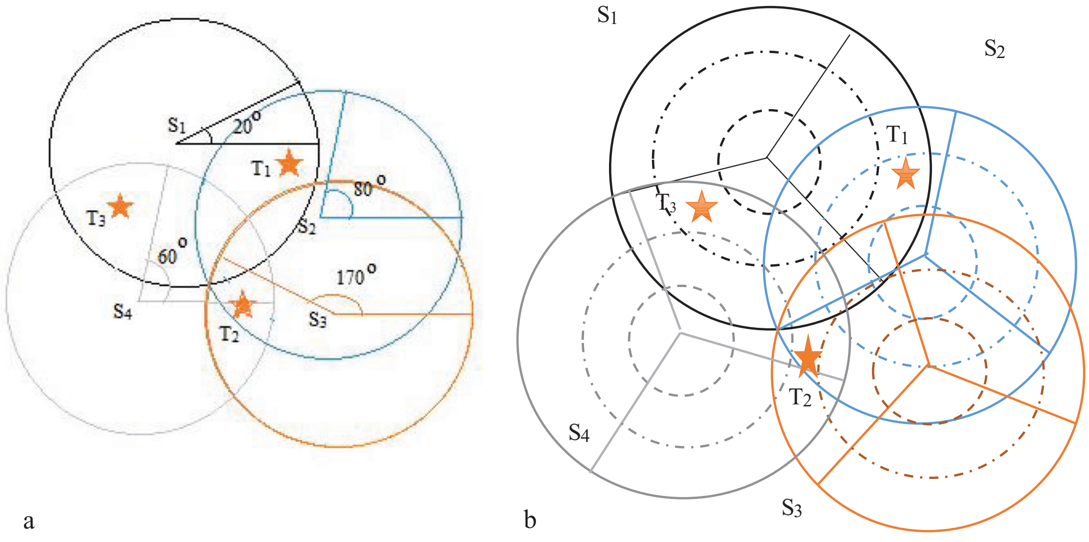

FIGURE 3 (a) The starting angle is selected randomly with a uniform distribution; (b) the starting angle of the sensor is updated based on Algorithm-1

ALGORITHM 2 The CALA-based scheduling algorithm for selecting sensing direction and range

```
01. Input: Directional sensor network Net = $(S,T)$, $S = \{s_1, s_2, \ldots, s_n\}$
02. Sensing angle $\theta$ obtained from Algorithm 1
03. Output: Appropriate work direction and sensing range
04. For each sensor, define the range of the first working
      direction $(\theta - \frac{\pi}{3}, \theta + \frac{\pi}{3})$ and count the other working directions of
      each sensor based on it.
05. for (All network nodes $s_n \in S$) do
06. $l_n \leftarrow 1$
07. end for
08. Sol $\leftarrow \phi$
09. LF $\leftarrow 0$
10. Repeat
11. $C_{cur} \leftarrow \phi$ ($C_{cur}$ denotes the cover set made)
12. $WT_l$ = min remaining activation time for $C_{cur}$
13. $LF \leftarrow LF + WT_l$
14. For each (di,j, a) $\in C_{cur}$ do
15.       $l_n \leftarrow l_n - \Delta^a/p$
16.     If $l_n = 0$ then
17. $s \leftarrow s - s_n$
18.     End if
19. End for
20. $Sol \leftarrow Sol \cup \{(C_{cur}, WT_l)\}$
21. Until (There is a target that has not been monitored)
22. Return (Sol, LF)
```

coverage energy of the directional sensor $(d_{i,j}, a)$:

$$CP\left(d_{i,j}, a\right) = \frac{\left|\left(d_{i,j},\ a\right) \cap\ T_{cur}\right|}{\Delta^a} \tag{8}$$

where $|(d_{i,j},\ a) \cap\ T_{cur}|$ shows the total number of unmonitored targets that can be monitored by $(d_{i,j},\ a)$, and Δ^a is the expressed energy consumption rate (Equation (9)):

$$\Delta^a = \frac{level\ a \text{ consumes Energy}}{level\ 1 \text{ consumes Energy}} \tag{9}$$

The maximum energy consumption is calculated by considering the working direction and the selected sensing range. The maximum energy consumption is compared with the dynamic threshold. Initially, this threshold equals the number of sensors. In case the energy consumption is lower than the dynamic threshold and all of the targets are monitored by the selected direction and range, the selected actions are rewarded; otherwise, they are penalized.

The dynamic threshold at each stage is substituted by the quantity of energy used by the formerly-rewarded cover sets. This way, the automata will learn the way to select the right direction and sensing range. The maximum network lifespan of a DSN is equal to the sum of the sensors' residual energy that covers all the targets.

5.2.1 | Forming a cover set

After setting the sensing range and working direction in the second phase, a CALA-based scheduling algorithm places a set of directional sensors in several cover sets. The proposed algorithm involves two steps: initialization and selection of the sensing nodes.

At the initialization step, a network of CALA is constructed, and the action set is set along with the probability vector of action. Comparable to the preceding algorithm, a CALA network is created by the provision of learning automata for each sensor. In each $A_{i,j}$ automaton, the action set is formed by achieving one action in each of the previously-described groups. The value of an index action of a group expresses the direction and range selected by a sensor. α_i^j corresponds to the jth group. This selection is made by learning automata $A_{i,j}$. At the beginning, the algorithm of probability vector from learning automata $A_{i,j}$ is configured as follows:

$$P_i^j\ (k) = \frac{1}{|\alpha_i\ (k)|}\ ;\ \forall\ \alpha_i^j \in \alpha_i \text{ and } k = 0 \qquad (10)$$

where $|\alpha_i(k)|$ expresses the coordinates of the action set at step k, which equals the number of previously-described groups. Using the above equation, the probability vector of the action can express that all the factors have the same probability to be selected.

In selecting the sensing node, the process of choosing the sensor directions and the sensing range for the formation of the cover set is elaborated. There are several steps in the proposed algorithm; at each step, all CALAs are initially in the inactive state, which helps their corresponding sensor nodes to choose the group whose selected sensing direction and range can monitor all the targets. The probability vector of CALA action is then optimally updated based on their corresponding set covers, which is equal to the number of generated set covers. The iterative process of generating cover sets and updating probability vector of action continues until a near-to-optimal solution with the maximum number of cover sets is found. Algorithm 3 provides the pseudo-code of the suggested algorithm.

5.3 | Monitoring

After forming the cover set, additional sensors will be eliminated from the generated cover set. Using this method, the removed sensors can be reused in new cover sets. This process aims to increase the number of cover sets and, this way, extend the network lifespan. A sensor is added from the cover set if all targets are still fully covered after removing it. After the elimination of the extra sensors, the updated cover set is activated sequentially to cover all the targets that exist in the network. A cover set is activated during its working time, and the amount of this working time is added to the total network lifespan. The remaining energy of the sensors in the generated cover set is updated, and the sensor without any energy is deleted from the set of available sensors. The proposed algorithm ends in two states: either the algorithm reaches the most number of cover sets, or the network can cover a specific set of targets.

5.4 | Time complexity

This section describes the algorithm and its complexity. In the proposed CALA-based algorithm, the main operation is to determine the initial sensing angle of each sensor and then select the appropriate working direction and sensing radius based on the selected angle. Here are the parameters $|S|$ and $|T|$, which denote the number of sensors and the number of targets, respectively. In the first stage, the initial sensing angle of the sensors that can sense the targets (S_{sence}) is calculated using a CALA-based algorithm (Algorithm 1) whose complexity is $|S_{\text{sence}}| \times |T|$. Algorithm 2 is then run in several rounds; the number of rounds depends on the sensor selection strategy, and the output of each round is a cover set ($|C_{cur}|$) that is capable of covering all the targets. The time complexity of this algorithm is $|C_{cur}| \times |S|$. Algorithm 3 is used to generate the cover set in each round. In Algorithm 3, until there are uncovered targets ($|T_{cur}|$), an inactive automaton is selected and labeled as active. Then, in each cycle, if a sensor can cover a target, it is placed in the

ALGORITHM 3 Algorithm for forming a cover set

```
01. Input: Directional sensor network Net = $(S,T)$, $p = 3$ is the number of
        energy levels, which is defined as the default for each sensor, and $a$ is
        the selected energy level for each sensor, that can observe the target.
02. Output: A cover set with optimal adjusted sensors
03. Assumption:
04. Assign learning Automaton $A_{i,j}$ to target node $t_i$
05. Assumption$\alpha_{i,j}$ represents the set of continuous learning
        automata $A_{i,j}$
06. Start algorithm
07. Let $T_g$ denote the dynamic threshold at stage $k$
08. Let $k$ denote the stage number initially set to $k = 0$
09. While (Itr $\neq$ maxItr)
10. $T_{cur} \leftarrow T$ ($T_{cur}$: uncovered targets)
11. $C_{cur} \leftarrow \phi$ ($C_{cur}$: covered-set)
12. $D_{cur} \leftarrow \phi$ ($D_{cur}$: list of dominated automata)
13. While $T_{cur} \neq \phi$ do
14. Find and activate the inactive automaton and name it $A_{i,j}$ .
15. If (($d_{i,j}$, $a$) monitors the target already in the cover set (Then
16. Prune($d_{i,j}$, $a$) and select another set of that sensor
17. $C_{cur} = C_{cur} \cup (d_{i,j},\ a)$
18. $T_{cur} = T_{cur} - T(d_{i,j},\ a)$
19. $D_{cur} =$
        $D_{cur} \cup (A_{i,j},$ the automata related to covered targets by $(d_{i,j},\ a))$
20. End while
21. For each $(d_{i,j}, a) \in C_{cur}$ do
22.     $Sum_E = Sum_E + (a \times (\frac{EI}{P}))$
23. End for
24. If $Sum_E \leq T_k$ then
25. Reward the selected actions of activated automata
26. $T_k \leftarrow Sum_E$
27. Else
28. Penalize the selected actions of activated automata
29. End if
30. $k \leftarrow k + 1$
31. Until ($g =$ maxitr or $T_{cov} =$ all Target)
32. End While
33. End algorithm
```

TABLE 2 Time complexity

Algorithm	Time complexity
CALA	$\lvert S\rvert \times (\lvert T_{cur}\rvert \times \lvert D_{cur}\rvert \times \lvert C_{cur}\rvert + \lvert C_{cur}\rvert + \lvert T\rvert)$
LA K-coverage [26]	$(\lvert S\rvert - \lvert T\rvert \times K) \times (\lvert T_{cur}\rvert \times \lvert D_{cur}\rvert \times \lvert C_{cur}\rvert + \lvert C_{cur}\rvert)$

coverage set, and then the list of unmonitored targets is updated by deleting the monitored target. The activated automaton and the inactivated automaton corresponding to the targets covered by the selected sensor are then placed in the set of deleted automata ($\lvert D_{cur}\rvert$). The process of activating the inactive automaton and selecting an action continues until all targets are covered. The complexity of Algorithm 3 is $\lvert S\rvert \times \lvert T_{cur}\rvert \times \lvert D_{cur}\rvert \times \lvert C_{cur}\rvert$, and finally the complexity of the whole algorithm is equal to $\lvert S\rvert * (\lvert T_{cur}\rvert \times \lvert D_{cur}\rvert \times \lvert C_{cur}\rvert + \lvert C_{cur}\rvert + \lvert T\rvert)$. Table 2 compares the complexity of the proposed algorithm (CALA) and the K-coverage automata-based algorithm (LA K-coverage) proposed in [26].

6 | SIMULATION RESULTS

In this paper, several experiments were carried out to evaluate the performance of the proposed algorithm by examining the impacts of various parameters upon the network lifespan and runtime of the proposed algorithm. In this paper, it is assumed that the sensors are to be located in places with different geographical features; the geographical locations may be such that it is difficult to access (impossible points). For this reason, we have chosen a random deployment in which the sensors are assumed to be propagated by plane and the initial positions of the sensors are identified using GPS. To this end, the sensors and targets were scattered randomly within an area of 500 m × 500 m, which was an effort to model a DSN. The initial assumption was that all the targets would be covered by at least one sensor. In all the experiments, by default, each directional sensor possessed three directions and one unit of energy. If a sensor run out of its energy, it was considered dead. Each simulation scenario was iterated for 20 times; then, the average network lifespan and runtime were computed for each scenario. In addition, a comparison was made between the results of the proposed algorithm and those of an LA-based algorithm previously proposed in [26] where each target was assumed to be covered by k directional sensors (i.e. k-cover) (the k value = 1). The simulation parameters used in the experiments are presented in Table 3.

Since the performance of the CALA-based algorithm depends on the amount of learning, selecting the correct learning rate greatly impacts the computational cost and accuracy of the algorithm [38]. Therefore, this value should be adjusted well to achieve acceptable results during an appropriate period of time. For the algorithm proposed in the current paper, the learning rate was set to 0.1. In addition, the L_{R-I} learning schema was utilized to update the probability vector.

TABLE 3 Simulation parameters

Simulation parameters	
Deployment	Random
Monitored region	500 m × 500 m
Number of directional sensors	60-100
Default power levels based on radius segmentation	3
Number of targets	6–10
Sensing radius	80–120 m
Field of view (FoV)	120 (deg)
Running Time (*S*)	(measurement parameter is seconds)
Energy consumption (*J*)	measurement parameter is joule)

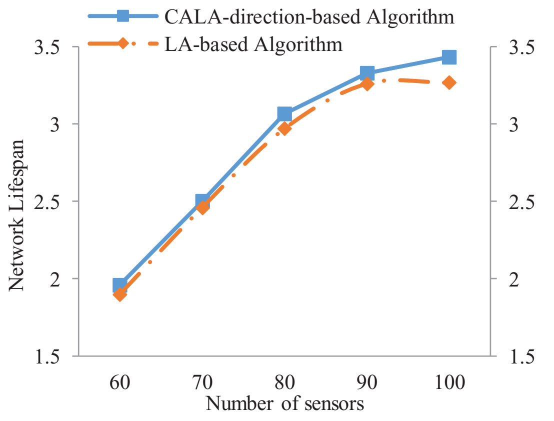

FIGURE 4 Effect of the number of sensors on the lifespan of the network

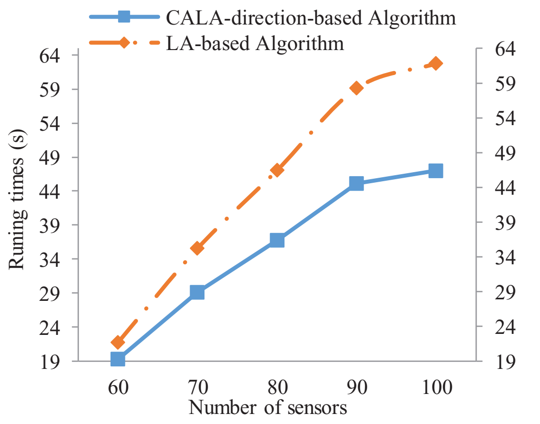

FIGURE 5 Effect of the number of sensors on the runtime

6.1 | Experiment 1

This experiment was conducted to examine how the number of sensors affects the network lifespan and runtime of the algorithm. For this purpose, the number of sensors varied between 60 and 100, with incremental step of 10. The coverage area of

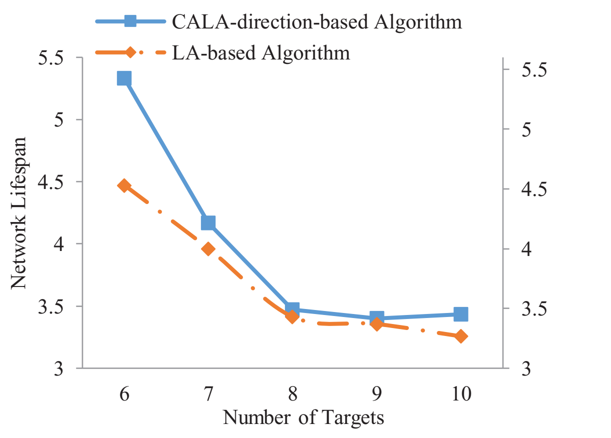

FIGURE 6 Effect of the number of targets on the lifespan of the network

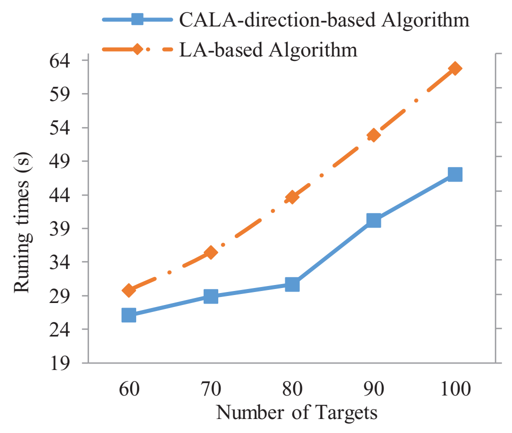

FIGURE 7 Effect of the number of targets on the runtime

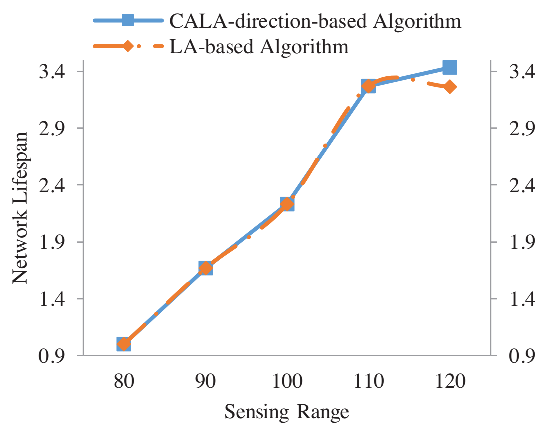

FIGURE 8 Effect of sensing range on the lifespan of the network

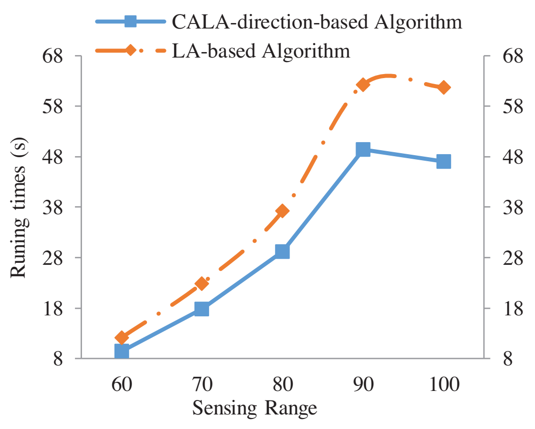

FIGURE 9 Effect of sensing range on the runtime

each sensor was based on the starting value of the sensing angle. The obtained results presented in Figure 4 show that the network lifespan increased by increasing the number of sensors. Further, the CALA-based algorithm had a longer lifespan than LA. In addition, as shown in Figure 5, the proposed algorithm had less runtime.

6.2 | Experiment 2

This experiment was aimed at examining the impact of the number of targets upon the network lifespan and runtime. To do this, the number of targets was set to 6–10, with incremental step of 1. Based on the results given in Figure 6, the network lifespan decreased with an increase in the number of targets. This was because in this condition, more sensors were required to monitor the targets. In addition, as shown in Figure 7, the proposed algorithm had less runtime.

6.3 | Experiment 3

This experiment was carried out to examine how the measurement radius affects the network lifespan. To this end, the sensing radius was changed between 80 and 120 m, with incremental step of 10 m. Based on the results presented in Figures 8 and 9, the network lifespan and runtime increased by increasing the radius since, in this condition, each sensor can monitor more targets. More cover sets are obtained by increasing the radius of the sensors, which finally leads to an increase in the network lifespan. As shown in Tables 4–6, the number of cover sets in CALA-angle is higher than that of LA, which led to a total increase in CALA-angle energy consumption compared to LA. At the same time, the runtime of the CALA-angle decreased significantly.

7 | CONCLUSION

The present study was carried out in order to investigate the target coverage problem in sensor networks. In this paper, the sensors were considered with different energy levels. The present study aimed to adjust the sensors' sensing angles using continuous learning automata to increase the network lifespan and reduce its runtime. In this regard, the proposed algorithm obtained the best sensing angle continuously based on the Gaussian distribution function, which resulted in determining the best sensing direction. The proposed algorithm had a

TABLE 4 Effect of the number of sensors on energy consumption and the number of cover sets formed

	Energy consumption		The number of cover sets					
			LA			CALA		
	LA	CALA	Minimum	Maximum	Average	Minimum	Maximum	Average
N = 100	51.9	56.62	8	11	9.5	8	11	9.9
N = 90	51.86	55.3	8	11	8.6	9	11	9.7
N = 80	48.1	51.43	8	10	8.6	8	10	9
N = 70	38.4	42.46	6	8	7.2	6	8	7.7
N = 60	25.8	32.93	4	6	5.4	5	7	6

TABLE 5 Effect of the number of targets on the energy consumption of the cover set

	Energy consumption		Cover set					
			LA			CALA		
	LA	CALA	Minimum	Maximum	Average	Minimum	Maximum	Average
T = 10	51.9	56.62	8	11	9.5	8	11	9.9
T = 9	44.86	46.16	7	11	9.5	8	11	9.4
T = 8	38.33	37.43	8	11	9.6	8	10	9
T = 7	32.8	36.6	9	11	9.7	10	12	10.4
T = 6	33.36	38.57	9	13	11	9	14	11.8

TABLE 6 Effect of sensing radius on the energy consumption of the cover sets

	Energy consumption		Cover set					
			LA			CALA		
	LA	CALA	Minimum	Maximum	Average	Minimum	Maximum	Average
RS = 120	51.9	56.62	8	11	9.5	8	11	9.9
RS = 110	53.63	56.06	8	10	9.3	8	11	9.6
RS = 100	37.36	38.6	5	7	6.63	6	7	6.72
RS = 90	29.03	29.4	4	5	4.9	5	5	5
RS = 80	19.53	19.3	3	3	3	3	3	3

relatively longer lifespan and less runtime than the LA-based algorithm based on the results. In future work, the angle and radius of the sensing can be adjusted to cover the targets using continuous automata to investigate the effects of the number of targets and sensors and the sensors' sensing range on the network lifespan.

CONFLICT OF INTEREST

The authors declared that they have no conflicts of interest to this work.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

ORCID

Mehdi Golsorkhtabaramiri https://orcid.org/0000-0002-9932-2477
Meisam Yadollahzadeh-Tabari https://orcid.org/0000-0002-5231-7611

REFERENCES

1. Akyildiz, I.F, et al.: A survey on sensor networks. IEEE Commun. Mag. 40(8), 102–114 (2002)
2. Yick, J., Mukherjee, B., Ghosal, D.: Wireless sensor network survey. Comput. Networks 52(12), 2292–2330 (2008)
3. Ai, J., Abouzeid, A.A.: Coverage by directional sensors in randomly deployed wireless sensor networks. J. Comb. Optim. 11(1), 21–41 (2006)
4. Torkestani, J.A: An adaptive energy-efficient area coverage algorithm for wireless sensor network. Ad Hoc Networks 11(6), 1655–1666 (2013)

5. Cheng, C.T., Tse, C.K., Lau, F.C.M.: A scheduling scheme for wireless sensor networks based on social insect colonies. IET Commun. 3(5), 714–722 (2009)
6. Cardei, M., Du, D.-Z: Improving wireless sensor network lifetime through power aware organization. 11(3), 333–340 (2005)
7. Fei, Z., Xing, C.: A survey of multi-objective optimization in wireless sensor networks: Metrics, algorithms and open problems. IEEE Commun. Surv. Tutorials 19(1), 550–586 (2016)
8. Junbin, L., Ming, L., Xiaoyan, K.: A survey of coverage problems in wireless sensor networks. Sens. Transducers 163(1), 240–246 (2014)
9. Cardei, M., et al.: Energy-efficient target coverage in wireless sensor networks. In: Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Miami, FL, USA, pp. 1976–1984 (2005)
10. Cai, Y., Lou, W., Li, M.: Energy efficient target-oriented scheduling in directional sensor networks. IEEE Trans. Comp 58(9), 1259–1274 (2009)
11. Yang, H., Li, D., Chen, H.: Coverage quality based target-oriented scheduling in directional sensor networks. In: Proceedings of the IEEE International Conference on Communications, Cape Town South Africa, pp. 1–5 (2010)
12. Katti, A.: Target coverage in random wireless sensor networks using cover sets. J. King Saud Univ. Comput. Inf. Sci. pp. 1–13, May (2019). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157818312412
13. Luo, C., et al.: Maximizing network lifetime using coverage sets scheduling in wireless sensor networks. Ad Hoc Networks 98, 102037 (2020)
14. Mao, G., Fidan, B., Anderson, B.D.: Wireless sensor network localization techniques. Comput. Networks 51(10), 118–133 (2007)
15. Nilsaz Dezfuli, N., Barati, H.: Distributed energy efficient algorithm for ensuring coverage of wireless sensor networks. IET Commun., 13, 578–584 (2019)
16. Golsorkhtabaramiri, M., et al.: HCABS: The hierarchical clustering algorithm based on soft threshold and cluster member bounds for wireless sensor networks. IEICE Electron. Express 9, 685–690 (2012)
17. Ma, H., Liu, Y.: On coverage problems of directional sensor networks, Mobile Ad-hoc and Sensor Networks, pp. 721–731, Springer, Berlin (2005)
18. Hsu, Y. C., Chen, Y. T., Liang, C. K.: Distributed coverage-enhancing algorithms in directional sensor networks with rotatable sensors. In International Conference on Distributed Computing and Networking, pp. 201–213, Springer, Berlin, Heidelberg (2012)
19. Costa, D., Silva, I., Guedes, L.A.: Optimal sensing redundancy formultiple perspectives of targets in wireless visual sensor networks. In: Proceedings of IEEE 13th International Conference on Industrial Informatics (INDIN), Cambridge, UK 2015
20. Manju, et al.: Genetic algorithm-based heuristic for solving target coverage problem in wireless sensor networks, Advanced Computing and Communication Technologies, pp. 257–264, Springer, Singapore (2018)
21. Wang, J., Niu, C., Shen, R.: Priority-based target coverage in directional sensor networks using a genetic algorithm. J. Comput. Math. Appl. 57(11), 1915–1922 (2009)
22. Razali, M.N, Shaharuddin, S., Mohamadi, H.: Solving priority-based target coverage problem in directional sensor networks with adjustable sensing ranges. Wirel. Pers. Commun. 95(2), 11–26 (2016)
23. Zarei, Z., Bag-Mohammadi, M.: Priority-based target coverage in directional sensor networks. IET Networks 7(6), 414–421 (2018)
24. Mohamadi, H., Ismail, A.S.B.H., Salleh, S.: A learning automata-based algorithm for solving coverage problem in directional sensor networks. Springer, Computing, 95(1), 1–24 (2013)
25. Alibeiki, A., Motameni, H., Mohamadi, H.: A new genetic-based approach for maximizing network A new genetic-based approach for maximizing network sensing ranges. Pervasive Mob. Comput. 52, 1–12 (2019)
26. Javan Bakht, A., Motameni, H., Mohamadi, H.: A learning automata-based algorithm for solving the target K-coverage problem in directional sensor networks with adjustable sensing ranges. J. Phys. Commun. 42, 101156 (2020)
27. Cerulli, R., De Donato, R., Raiconi, A.: Exact and heuristic methods to maximize network lifetime in wireless sensor networks with adjustable sensing ranges. Eur. J. Oper. Res 220, 58–66 (2012)
28. Han, Y.-H, Kim, C.-M., Gil, J.-M.: A greedy algorithm for target coverage scheduling in directional sensor networks. J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl. 1(2/3), 96–106 (2006)
29. Rezvanian, A., et al.: Introduction to learning automata models. Learning Automata Approach for Social Networks, pp. 1–49, Springer, Berlin (2019)
30. Anari, B., Torkestani, J.A., Rahmani, A.M.: Automatic data clustering. J. Appl. Soft Comput. 51, 253–265 (2017)
31. Najim, K., Poznyak, A.S.: Learning Automata: Theory and Applications. Printice-Hall, New York 1994
32. Torkestani, J.A., Meybodi, M.R.: A learning automata-based heuristic algorithm for solving the minimum spanning tree problem in stochastic graphs. J. Supercomput. 59(2), 1035–1054 (2012)
33. Esnaashari, M., Meybodi, M.R.: A learning automata based scheduling solution to the dynamic point coverage problem in wireless sensor networks. Comput. Networks 54(14), 2410–2438 (2010)
34. Thathachar, M.A.L., Harita, B.R.: Learning automata with changing number of actions. IEEE Trans. Syst. Man Cybern. 17, 1095–1100 (1987)
35. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice-Hall, New York 2012
36. Santharam, G., Sastry, P.S., Thathachar, M.A.: Continuous action set learning automata for stochastic optimization. J. Franklin Inst. B 331(5), 607–628 (1994)
37. Howell, M.N., et al.: Continuous action reinforcement learning applied to vehicle suspension control. Mechatronics 7(3), 263–276 (1997)
38. Torkestani, J.A.: An adaptive learning to rank algorithm: learning automata approach. Decis. Support Syst. 54(1), 574–583 (2012)

How to cite this article: Qarehkhani, A., Golsorkhtabaramiri, M., Mohamadi, H., Yadollahzadeh-Tabari, M.: Solving the target coverage problem in multilevel wireless networks capable of adjusting the sensing angle using continuous learning automata. IET Commun. 16, 151–163 (2022). https://doi.org/10.1049/cmu2.12323