Computers and Electrical Engineering 72 (2018) 859–876

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Barrier coverage in adjustable-orientation directional sensor networks: A learning automata approach[☆]

Mohammad Khanjary[a], Masoud Sabaei[b], Mohammad Reza Meybodi[b,*]

[a] Computer and Electrical Engineering Department, Science and Research Branch, Islamic Azad University, Tehran, Iran
[b] Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 5 July 2017
Revised 8 January 2018
Accepted 9 January 2018
Available online 19 January 2018

Keywords:
Barrier coverage
Percolation
Learning automata
Directional sensor networks
Camera sensor networks

ABSTRACT

Barrier coverage is one of the main applications of wireless sensor networks. There are two kinds of barrier coverage: weak and strong. While weak barrier coverage can only guarantee detecting intruders moving along predetermined or congruent paths, strong barrier coverage guarantees detecting all intruders regardless of their crossing paths. In this paper, we first present a centralized algorithm based on distributed learning automata to find strong barrier lines in adjustable-orientation directional sensor networks in which nodes can adjust their orientation in a non-overlapping form such as camera sensor networks. Then, we will present the distributed version of the proposed algorithm to be used in practical sensor networks. Moreover, the results of extensive simulations will be presented to compare the performance of proposed algorithms against the optimal algorithm, a greedy algorithm and a previously-proposed algorithm. The results confirm that the proposed algorithms achieve near-optimal results and outcome other algorithms completely.

1. Introduction

A *wireless sensor network* (WSN) consists of several small, low-cost, low-power sensor nodes which are deployed randomly in the *region of interest* (RoI) to monitor some aspects of the environment such as humidity, light, temperature, voice and video. Generally, a sensor node includes four components: 1) *a sensing unit* to acquire data from the environment 2) *a processing unit* to process raw data 3) *a transceiver unit* to send/receive the data and 4) *a power unit* to supply the required energy. However, some additional units could be added to sensor nodes depending on application requirements. WSNs could be used in a wide range of applications and environments such as monitoring battle field, forest, undersea, border, volcano and urban [1].

Barrier coverage is one of the main applications of sensor networks and is defined as detecting the intruders attempting to cross the RoI. There are two kinds of barrier coverage: weak and strong. While weak barrier coverage can only guarantee detecting intruders moving along predetermined and congruent paths, strong barrier coverage guarantees detecting all intruders regardless of their crossing paths.

Recently, directional sensor networks have received a great deal of attention due to their wide range of applications such as target detection and tracking. A directional sensor network consists of a number of directional sensors such as ultrasound,

[☆] Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. M. H. Rehmani.
[*] Corresponding author.
E-mail addresses: khanjary@srbiau.ac.ir (M. Khanjary), sabaei@aut.ac.ir (M. Sabaei), mmeybodi@aut.ac.ir (M.R. Meybodi).

https://doi.org/10.1016/j.compeleceng.2018.01.009
0045-7906/© 2018 Elsevier Ltd. All rights reserved.

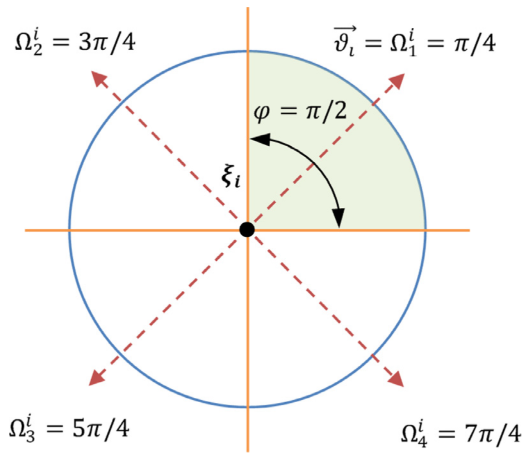

Fig. 1. The possible orientations of a typical directional sensor with $\varphi = \pi/2$ in non-overlapping form.

infrared and video sensors. The most important characteristics of directional sensors are their limitation in sensing (field-of-view) angle and sensing radius and also randomness of their orientation. Due to these unique characteristics, there are important differences between directional sensor networks and traditional omnidirectional sensor networks. The sensing region of a directional sensor is modelled by a circular sector while the sensing region of omnidirectional sensors is modeled by a disk. Moreover, rotation in some types of directional sensors allows them to adjust their orientation and cooperate with their adjacent nodes.

On the other hand, learning automata has been recognized as an efficient artificial intelligence tool that performs very well in dynamic environments such as wireless, ad hoc, and sensor networks. A cellular learning automata is defined as a graph in which each vertex represents a cell and is equipped with a learning automaton and each edge induces an adjacency relation between two cells (two automatons). The learning automaton residing in a cell determines its action according to its action probability vector and some internal rules. In recent years, cellular learning automata has been used to solve different problems of sensor networks such as congestion avoidance, data aggregation, backbone formation, barrier Coverage, deployment and scheduling [2].

In general, directional sensor networks could be classified to three classes based on type of their orientation: 1) *Aligned-orientation directional sensor networks* (ALODSNs) [3] in which orientation of sensors is identical and fixed, 2) *Fixed-orientation directional sensor networks* (FIODSNs) [4] in which orientation of sensors is fixed and distributed on $[0, 2\pi)$ uniformly and, 3) *Adjustable-orientation directional sensor networks* (ADODSNs) [5] in which the deployment of sensors is like FIODSNs but their orientation could be adjusted after deployment.

According to the above-mentioned classification, in this paper, we first present a centralized algorithm based on distributed learning automata to construct barrier lines in ADODSNs. Then, we will present the distributed version of the proposed algorithm. Afterwards, the results of extensive simulations will be presented. The simulations confirm that the proposed algorithms outcome similar algorithms completely and achieve near-optimal results.

The main contributions of the paper are two folds:

1. Introducing and modelling *adjustable-orientation directional sensor networks* (ADODSNs) with *directional barrier graph* (*DBG*) which could be used as a basis in other applications of ADODSNs.
2. Introducing two algorithms based on learning automata to find near-optimal number of strong barrier lines. Finding more barrier lines increases the network lifetime by activating one barrier in each round of the network. The simulations show that the proposed algorithms significantly improve the performance in comparison to other algorithms.

The remainder of this paper is organized as follows: Section 2 presents problem statement, Section 3 reviews the related works, Section 4 reviews the theory of automata, section 5 introduces our algorithms, Section 6 presents the simulation results and finally Section 7 concludes the paper.

2. Problem statement

In this paper, we consider a wireless sensor network with n stationary directional sensor nodes where all nodes are aware of their location. Our sensing model is similar to. Each node S_i has a directional sensor whose sensing region is a sector of a disk centered at ξ_i with a sensing radius of r and a field-of-view angle of φ. The orientation vector of a sensor is defined as the bisector of its sensing sector. In ADODSNs, a directional sensor has ability to change its orientation to any direction in a given set Ω.

Consider Fig. 1 as an example. In Fig. 1, $\varphi = \pi/2$, $\Omega = \{\pi/4, 3\pi/4, 5\pi/4, 7\pi/4\}$ and $\overrightarrow{\vartheta_i}$ is the orientation vector of the sensor. Therefore, this directional sensor has four possible orientations in non-overlapping form. The set of all possible orientations in an ADODSN with n sensor nodes could be denoted as $\Pi = \{\Omega^i_j | \ i \in \{1,\ 2, \ldots, n\},\ \ j \in \{1,\ 2, \ldots, q\}\}$ where q is number of possible orientations for each sensor ($q = 4$ for sample sensor in Fig. 1).

In order to provide barrier coverage in RoI, we need to find a subset of sensors and their orientations (in fact a set of sensing sectors) to form the barrier line in such a way that any crossing path from top to the bottom of RoI is covered. We formally define the directional sensor barrier and its corresponding coverage problem as follows.

Definition 1. *A directional sensor barrier* (DSB) is a set of sensing sectors which satisfy the following conditions: 1) the left and right boundary sensors overlap left and right boundaries of RoI respectively, 2) any two neighboring sectors overlap, and 3) it includes at most one sector from each sensor node [6].

3. Related works

A sensor network prepares barrier coverage if at least one spanning cluster of sensors exists which spans the RoI. Most of works in barrier coverage have been done on omnidirectional sensor networks.

Du et al. [7] introduced k-discrete barrier coverage model, whose goal is covering some specific discrete points of interest by deploying sensors in k barrier lines. They analyzed the problem and proposed a scheduling algorithm to maximize the network lifetime.

Likewise, Mostafaei et al. [8] considered barrier coverage by using minimum number of nodes. They modeled the barrier coverage problem by a graph and proposed a learning automata-based algorithm to achieve near optimal results. Saipulla et al. [9] considered the barrier coverage when sensors are deployed along a line e.g. dropping from an aircraft. However, the sensors would be distributed with random offsets from the line due to wind and other environmental factors.

Silvestri et al. [10] proposed MobiBar, an autonomous deployment algorithm for barrier coverage in mobile sensors. MobiBar coordinates sensor movements to construct distinct barriers. They proved that MobiBar will be run in a finite time and will be found maximum barrier lines. Also, they proposed a self-reconfigure procedure to deal with dynamic coverage requirements and sensor failures. Moreover, Kim et al. [11] considered the problem of organizing hybrid sensor networks (which consists of a number of energy-scarce static sensors and energy-plentiful mobile sensors) to maximize the network lifetime. They proposed a heuristic algorithm for the problem and proved that the lifetime of constructed hybrid barrier will be more than similar works. Also in [12], Kim et al. considered preparing maximum fault-tolerance barrier coverage in hybrid sensor networks. They proposed an algorithm to relocate the mobile sensors and maximize fault-tolerance of the barrier coverage in hybrid sensor networks.

Recently, barrier coverage in directional sensor networks has been considered too. Shih et al. [13] proposed a distributed algorithm called CoBRA to select the minimum camera sensors in a dense deployment to establish the barrier line. Tao et al. [14] proposed a centralized algorithm for line-based deployed directional sensor networks. Also, Wang et al. [15] studied the problem of constructing a camera barrier in such a way that every point on it is full-view covered. An object is full-view covered if there is always a camera to cover it no matter the direction it faces. Likewise, Yang et al. [16] proposed a distributed algorithm for the full-view barrier coverage problem.

As the recent researches on barrier coverage in directional sensor networks, Cheng et al. [17] considered quality of monitoring in visual sensor networks. They considered three factors including importance of image, breadth of image and rotation capability in barrier coverage and proposed an algorithm to find the barrier line with β breadth. The proposed algorithm ensures that if any intruder crosses the barrier line, quality of captured image will be more than required quality. Also, Fan et al. [18] considered directional probabilistic sensing model for directional sensors. They showed that in practical conditions, the sensing radius of sensors will be decreased with the increase of field-of-view angle. Then, they proposed a distributed probabilistic barrier construction scheme based on energy efficiency ratio (EER). In each step of the algorithm, a relay sensor with largest EER will be selected until the end boundary reached. They showed that their method could form barrier line with smaller energy consumption than other methods.

4. Automata

4.1. Cellular automata (CA)

CA is a mathematical tool to model systems with a large number of identical components. These components (cells) follow simple rules and act together to produce complicated behaviors. Each cell has k possible states and updates its state based on a local rule. New state of each cell depends on its previous state and also state of some neighbor cells. CA is a suitable tool for modeling natural systems and phenomena [19].

4.2. Learning automata (LA)

A LA could be considered as an adaptive decision-making system that could improve its performance by selecting the best action from a finite set of possible actions and interacting with random environments.

In each step, an action will be selected stochastically based on *action probability vector* (APV) defined over the set of possible actions. The selected action will be considered as an input to the random environment (see Fig. 2). The environment

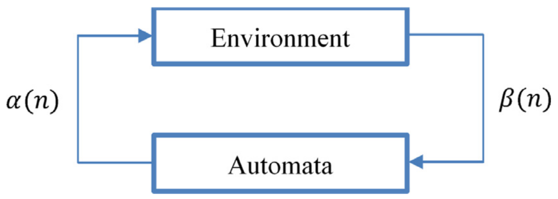

Fig. 2. The relationship between the learning automaton and the random environment.

feedbacks its response to the action by a reinforcement signal and automaton updates its APV according to the reinforcement signal. In other words, LA aims to find the optimal action in a finite set of possible actions by minimizing the penalty received from the environment. It has been shown that LA performs well in complex, dynamic, and random environments [20].

The environment could be defined as a triple $E=(\alpha,\ \beta,\ c)$, where $\alpha=\{\alpha_1,\alpha_2,\ldots,\ \alpha_r\}$ is the finite set of possible inputs, $\beta=\{\beta_1,\beta_2,\ldots,\ \beta_r\}$ is the output set, and $c=\{c_1,c_2,\ldots,\ c_r\}$ is the set of penalty associated with each input.

Also, a LA could be *fixed structure* or *variable Structure*. If APV and state transitions of the LA change during the learning, the LA is called *variable Structure* LA, otherwise it is called *fixed structure* LA. A *variable structure* LA is denoted by a quadruple $L=(\alpha,\ \beta,\ p,\ T)$ where $\alpha=\{\alpha_1,\alpha_2,\ldots,\ \alpha_r\}$, $\beta=\{\beta_1,\beta_2,\ldots,\ \beta_r\}$ and $p=\{p_1,p_2,\ldots,\ p_r\}$ denote the set of actions, set of inputs and the APV associated to actions respectively, and $p(n+1)=T(\alpha(n),\ \beta(n),\ p(n))$ is the learning algorithm. The learning algorithm will be repeated many times and in each step operates as follows.

Automaton chooses an action (α_i) randomly based on its current APV and performs it to the environment. When automaton receives the reinforcement signal from the environment, it updates its APV by Eq. (1) for rewarding responses and Eq. (2) for penalizing ones. In Eqs. (1) and (2), a and b are *reward* and *penalty* parameters, respectively. If $a=b$ the learning algorithm is called the *reward-penalty* (L_{R-P}), if $a \gg b$ the learning algorithm is called *reward ε-penalty* ($L_{R-\varepsilon P}$) and if $b=0$ it is called the *reward-inaction* (L_{R-I}) [2,20].

$$p_j\ (k\ +\ 1) = \begin{cases} p_j(k) + a\left[1 - p_j(k)\right] & j = i \\ (1\ -\ a)p_j(k) & \forall\ j \neq i \end{cases} \tag{1}$$

$$p_j(k\ +\ 1) = \begin{cases} (1\ -\ b)p_j(k) & j\ =\ i \\ \left(\frac{b}{r-1}\right) + (1\ -\ b)p_j(k) & \forall\ j \neq i \end{cases} \tag{2}$$

4.3. Cellular learning automata (CLA)

CLA is an effective combination of CA and LA and a powerful mathematical model for many decentralized and dynamic systems. CLA combines the ability of learning in LA and ability of modelling complex structures with simple identical components in CA. In other words, a CLA is a CA in which each cell is equipped with one (or more) learning automaton. In CLA, the adjacent automatons act as environment for each cell. The operation of CLA could be summarized as follows. Initially all actions of a cell has identical probability. Each cell selects one of its possible actions based on its current APV and notices the action to its neighbor cells. Neighbor cells determine the reinforcement signal based on CLA rules and feedback the response to the automaton. At the end of iteration, each learning automaton will update its APV according to the received reinforcement signals. This process will be continued until the desired result is obtained [21].

4.4. Distributed learning automata (DLA)

In order to define DLA, a quadruple $\langle A,\ E,\ T,\ A_0\rangle$ is used where $A=\{A_1,A_2,\ldots,\ A_n\}$ denotes the set of learning automatons, $E \subset A \times A$ denotes set of edges between automatons where the edge $e(i,\ j)$ represents the action α_j of the automaton A_i, T denotes the learning algorithm and A_0 denotes the root automaton of DLA.

A DLA operates as follows. The root automaton (A_0) selects an action based on its current APV. According to the chosen action, another automaton will be activated. Again, the activated automaton will select an action based on its current APV and will active another automaton. This process will be continued until the activated automaton is a leaf automaton (an automaton which interacts with the environment). The chosen actions, along the path of activated automatons will be applied to the environment. The environment evaluates the actions, produces a reinforcement signal and sends it back to the DLA. The reinforcement signal will be used by the activated automatons to update their APVs. If the average probability of activated automatons reaches to a threshold, algorithm will be terminated. A DLA has only one root automaton which always is active and at least one leaf automaton [2,20].

As an example of DLA, consider Fig. 3 including four automatons and A_i as the root automaton. If A_i selects action α_i^2, then A_k will be activated. Likewise, A_k can choose one of its actions (α_k^2 *and* α_k^3) to activate one of its connected automatons.

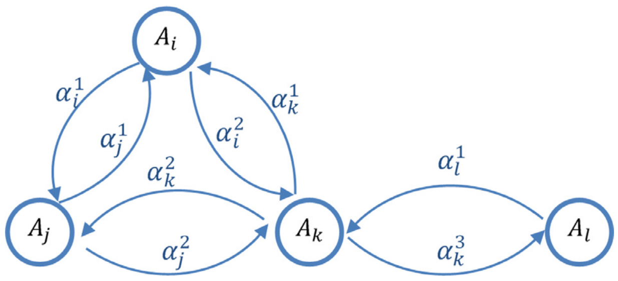

Fig. 3. A sample distributed learning automata (DLA).

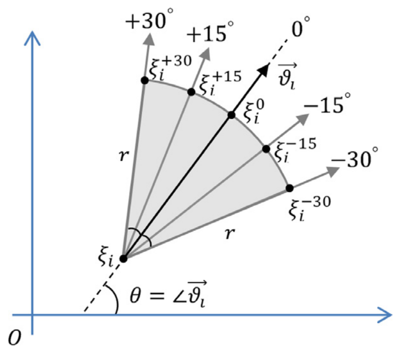

Fig. 4. Approximating the sensing region (sector) of a sensor with field-of-view angle ($\varphi = \pi/3$) by five lines.

5. Barrier coverage for ADODSNs based on DLA

5.1. Overlap detection between directional sensors

To detect overlap between sensing sector of directional sensors, we use our proposed method in [22]. In [22], we proposed a method called *algorithmic overlap detection* (AOD) that each node can locally and distributedly detect overlap between its possible sensing sectors and sensing sectors of its neighbors. The proposed method could be summarized as follows.

Because sensing region of each directional sensor is a circular sector, if distance between two sensors is more than twice the sensing radius, overlap will never occur. Therefore, the overlap detection procedure only will be done for sensors with less distance. To reduce complexity of calculations for evaluating overlap between sensing sectors, we approximate the sectors with a few line segments and evaluate the overlap between line segments instead of sectors themselves. The accuracy of overlap detection by AOD method will be more than 96% if interval between line segments is 15° ($\pi/12$).

As an instance, consider Fig. 4, in which a sensing sector with field-of-view angle $\varphi = \pi/3$ has been shown. As seen, this sector will be modeled with five line segments ($\overline{\xi_i\xi_i^{-30}}$, $\overline{\xi_i\xi_i^{-15}}$, $\overline{\xi_i\xi_i^{0}}$, $\overline{\xi_i\xi_i^{+15}}$ and $\overline{\xi_i\xi_i^{+30}}$) for 15° intervals. Likewise, the sensing sector of the neighbor sensors will be modelled by five line segments. Then, overlap between each pair of line segments of two neighbor sensors will be evaluated. Because overlap detection between two line segments could be evaluated simply by mathematics, each node can evaluate overlap between its orientations and orientations of its neighbors locally, distributedly and independently [22].

5.2. Directional barrier graph (DBG)

In this section, we introduce the directional barrier graph, $DBG(V, E)$, which will be used to model barrier coverage in ADODSNs. We model each sensor node with q vertices in DBG where q is the number of possible orientations of sensors (as introduced in section 2). Therefore, each vertex in DBG corresponds to a sector of a sensor. There is a directed edge from vertex v to vertex v' in DBG if their corresponding sectors (1) overlap and (2) belong to two different sensor nodes and (3) x coordinate of corresponding sensor to v' is greater than x coordinate of corresponding sensor to v (i.e. corresponding sensor of v' is located on right hand of corresponding sensor of v). By using the AOD method explained in previous section, each node can locally and simply evaluate possible overlap between its orientations and orientations of its neighbors. In addition, there are a virtual vertex *source* ($\mathbb{S}$) and a virtual vertex *target* ($\mathbb{T}$) that have overlap with the vertices whose corresponding sectors have overlap with *left* and *right boundaries*, respectively. Initially, DBG has cardinality of $O(nq)$ vertices and $O(n^2q^2)$ edges. The importance of the DBG lies in the fact that each simple $\mathbb{S} \rightarrow \mathbb{T}$ path in it corresponds to a DSB (see section 2). Therefore, constructing the *DBG* will be the first step of our algorithms. However, a pair of disjoint paths P and P' from $\mathbb{S}$ to $\mathbb{T}$ in *DBG* may not correspond to two disjoint DSBs because two vertices on P and P' may belong to a common sensor.

Pruning rule for *DBG*. A vertex in *DBG* could be on a barrier path if it has at least an ingoing edge and an outgoing edge. Therefore, vertices with no edge or only ingoing or outgoing edges could be removed from *DBG* initially.

5.3. Centralized barrier coverage algorithm based on DLA: CBCDLA

Given a randomly and densely deployed ADODSN represented by a *DBG*, in this section, we propose an algorithm based on DLA to construct near-optimal number of barrier lines. By using the AOD method explained in section 5.1, overlap between each pair of sectors of the neighbor sensors could be evaluated and the *DBG* graph of the sensor network could be constructed. Due to time-varying and unknown characteristics of sensor networks, we are going to use DLA in our algorithm.

In the first step of the algorithm, a network of learning automatons will be generated by equipping every directional sensor node with q learning automatons. We use the model proposed in [23] to use multiple LA in each sensor node. In other words, there is an automaton corresponding to each possible orientation of a sensor node that will be activated when that sector is selected through the barrier line construction by previous automaton.

We model the DLA by $\langle A, \alpha, \rho, A_{\mathbb{S}} \rangle$ where $A = \{A_{i,j} | 1 \le i \le n,\ 1 \le j \le q\}$ denotes the set of automatons (i.e. $A_{i,j}$ is the corresponding automaton to jth sector of sensor i) and $\alpha = \{\alpha_{i,j} | 1 \le i \le n,\ 1 \le j \le q\}$ and $\rho = \{\rho_{i,j} | 1 \le i \le n,\ 1 \le j \le q\}$ denote the set of actions and the APV of $A_{i,j}$ respectively and $A_{\mathbb{S}}$ is the root automaton corresponding to *source* ($\mathbb{S}$). It must be noticed that due to random deployment of sensor nodes, number of overlapping sectors for different sensors which will be represented by outgoing edges (i.e. number of actions of automatons) is not identical. Therefore, the set of possible actions for automaton $A_{i,j}$ and its APV could be defined as $\alpha_{i,j} = \{\alpha^1_{i,j}, \alpha^2_{i,j}, \ldots, \alpha^{r_{i,j}}_{i,j}\}$ and $\rho_{i,j}(n) = \{\rho^1_{i,j}(n), \rho^2_{i,j}(n), \ldots, \rho^{r_{i,j}}_{i,j}(n)\}$ where $r_{i,j}$ is number of actions of $A_{i,j}$ and $\rho^k_{i,j}(n)$ denotes the probability of kth action of automaton $A_{i,j}$ at iteration n.

For learning automaton $A_{i,j}$, the set of actions is defined as the set of all automatons which their corresponding sectors have overlap with corresponding sector of $A_{i,j}$ and their corresponding sensors are on the right hand. At the beginning of the algorithm, the probability of all actions of the learning automatons will be initialized equally. Through the execution of the algorithm, each learning automaton updates its APV by using L_{R-I} reinforcement scheme [20].

As a sample ADODSN consider part (a) of Fig. 5 including four directional sensor nodes with field-of-view angles equal to $\varphi = \pi/2$. Therefore, each sensor has four possible orientations ($q = 4$). As seen, sensor S_i and S_k are *left boundary* and *right boundary* sensors, respectively. First, q automatons will be assigned to each sensor node. Then, each automaton evaluates overlap between its corresponding sector and sectors of its neighbors. After evaluation, the *DBG* will be constructed as it is shown in part (b) of Fig. 5. Due to pruning rule of *DBG*, automatons with just ingoing or outgoing edges or with no edge could be removed from *DBG*. Therefore, automatons A^i_1, A^i_3, A^i_4, A^j_1, A^j_2, A^j_3, A^k_1, A^k_2, A^k_3, A^l_1 and A^l_4 will be removed.

In our algorithms, the network plays the role of the random environment. Selecting actions by a sequence of automatons will construct a barrier line. Based on the length of the barrier (number of sensors on the barrier line), the response of the environment will be produced as *favorable* or *unfavorable*. If *favorable*, the selected actions through the barrier line will be rewarded, otherwise no action will be done.

Automatons in our algorithms can be in either *active* or *passive* mode. At the beginning of each stage, all automatons will be set in their *passive* mode but the automaton corresponding to the $\mathbb{S}$ ($A_{\mathbb{S}}$) which represents the root of DLA. Also, the APV of actions of all automatons will be initialized equally for all its outgoing edges. For instance, the set of actions and APV of automaton $A_{i,2}$ in Fig. 5 will be as $\alpha_{i,2} = \{A_{j,3}, A_{j,4}\}$ and $\rho_{i,2}(0) = \{0.5, 0.5\}$.

The algorithm will be started with $A_{\mathbb{S}}$ which represents the *left boundary*. $A_{\mathbb{S}}$ selects one of its actions (one of its outgoing edges) stochastically, adds the outgoing edge to the queue of current barrier path (τ), increases the length of current barrier path ($\omega(\tau)$) and activates the corresponding automaton of the selected action. Again, the activated automaton selects one of its possible actions stochastically, adds it to the current barrier path, increases the length of barrier line and activates the corresponding automaton. This process will be repeated until automaton $A_{\mathbb{T}}$ (which represents the *right boundary*) is reached.

The algorithm will be done in several stages, and at each one, a barrier line is generated by activating several automatons in DLA. The APV of the activated automatons will be updated according to the optimality of the barrier line. If length of recent constructed barrier line is less than the dynamic threshold, it will be saved as the best barrier line in that stage and all selected actions through the barrier path will be rewarded. As the algorithm goes on, LA learns the way of selecting automatons (in fact sectors) that leads to construct barrier lines with minimum number of sensors. The process of generating barrier lines goes on until the probability of constructing barrier line exceeds $\mathbb{P}$ or number of repetitions exceeds $\mathbb{K}$. The probability of constructing a barrier line is defined as the average probability of selecting its automatons. The last constructed barrier line in each stage, will be the best barrier line due to number of sensor nodes in that stage. The pseudo code for the algorithm has been shown in Fig. 6.

5.4. Distributed barrier coverage algorithm based on DLA: DBCDLA

While the structure of DBCDLA is based on CBCDLA, it must be departed to some phases to be able to operate as a fully distributed algorithm. Like CBCDLA in DBCDLA, a network of leaning automatons will be formed by residing q automatons

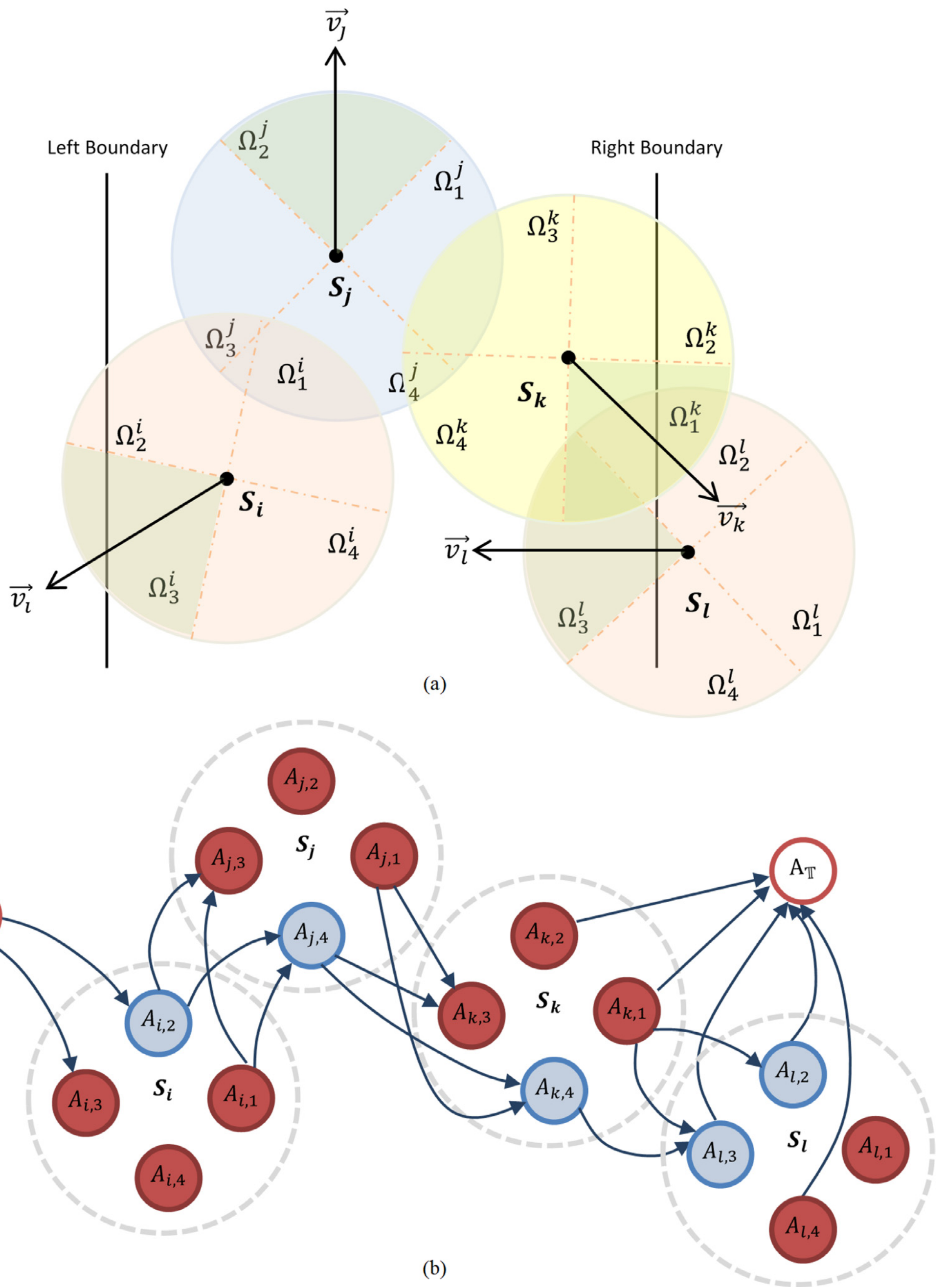

Fig. 5. (a) A sample ADODSN including four nodes with $\varphi = \pi/2$ and (b) its corresponding *DBG*.

in each sensor node. Each sensor node knows its location coordinates and has a *sensor ID* (SID), a *field-of-view angle* (φ) and *an orientation vector* ($\vec{\vartheta_i}$). Also, each automaton has an *automaton ID* (AID), a *status mode* (passive/active), a *set of actions* and an *APV*. The DBCDLA has three phases as follows.

5.4.1. Initialization phase

In this phase, each sensor node exchanges its data with its neighbors (sensors in the distance less than $2r$) by sending INITIALIZATION packets. An INITIALIZATION packet includes SID, φ, coordinates and orientation vector of the sender. By using this information, each sensor node will evaluate overlap between its sectors and sectors of the neighbors by using AOD method (see section 5.1) locally and distributedly. This operation will be done for all neighbor nodes. At the end of this phase, each sensor constructs its local *DBG*. Local *DBG* for a sensor is a subgraph of *DBG* which starts from its sectors.

```
Centralized Barrier Coverage Algorithm Based on Distributed Learning Automata (CBCDLA)
Input:
    Distributed barrier graph, DBG(V,E) corresponding to directional sensor network
    Virtual Source node $\mathbb{S}$ which has overlap with all start boundary sensors (left border)
    Virtual Target node $\mathbb{T}$ which has overlap with all end boundary sensors (right border)
    Number of sensors ($n$), Maximum number of stages ($\mathbb{K}$) and stop threshold ($P$)
Output:
    strong barrier lines with near optimal minimum number of sensors
Assumption:
    Assign a learning automaton $A$ to each vertex of DBG graph ($A_{i,j}$ corresponds to $j$th sector of sensor $i$)
    Set status of all automatons to passive state but source node ($A_{\mathbb{S}}$)
Begin
While (a barrier line could be built)
    Let $T_k$ denotes the dynamic threshold at stage $k$, initially set to $n$
    Let $k$ denotes the stage number, initially set to zero
    Repeat
        Let $\tau$ is the current barrier path and initially set to null
        Let $\omega(\tau)$ be the average length of current barrier path and initially set to zero
        First automaton ($\mathbb{S}$) is activated and denoted as $A_i$
        While ($\mathbb{T}$ is not reached and number of visited vertices is less than $n$)
            Automaton $A_i$ randomly chooses one of its actions (say $A_{i,j}$)
            Add edge $(A_i, A_j)$ to $\tau$ and increase its weight $\omega(\tau)$
            Automaton $A_j$ is activated and set as $A_i$
        End While
        If $(\omega(\tau) \le T_{k-1})$ Then
            Reward the selected actions by the activated automatons along the constructed barrier line
        End if
        $k \leftarrow k + 1$
        $T_k \leftarrow [(k-1)T_{k-1} + \omega(\tau)]/k$
    Until (the probability of barrier line formation is greater than $\mathbb{P}$ or number of stages exceeds $\mathbb{K}$)
End While
End Algorithm
```

Fig. 6. The pseudo code of centralized barrier coverage algorithm based on distributed learning automata (CBCDLA).

As an example consider sensor S_j in Fig. 5. After exchanging INITIALIZATION packets with its neighbors (S_i and S_k) and evaluating possible overlap between its sectors and sectors of the neighbors, Sensor S_j creates its local *DBG* as it is shown in Fig. 7. For sensor S_j (and its corresponding automatons) only possible actions of its automatons is important. In other words, automatons with no outgoing edges could not be used to construct the barrier line. Therefore, in its local *DBG*, only automatons with outgoing edges ($A_{j,1}$, $A_{j,4}$) will be kept and other automatons will be removed. Afterwards, the probability of outgoing edges (actions) of each automaton will be initialized equally. Also, $A_{\mathbb{S}}$ will set dynamic threshold of the algorithm to number of nodes ($T_k \leftarrow n$) initially.

The pruning rule. After finishing overlap detection procedure, each sensor removes automatons without outgoing edges.

5.4.2. Activation phase

In this phase, ACTIVATION packets will be sent by sensors in a sequence that starts from root of DLA ($A_{\mathbb{S}}$). Each ACTIVATION packet includes *sensor ID* (SID) and *automaton ID* (AID) of the sender (i.e. last activated automaton), *queue of sensors* (in

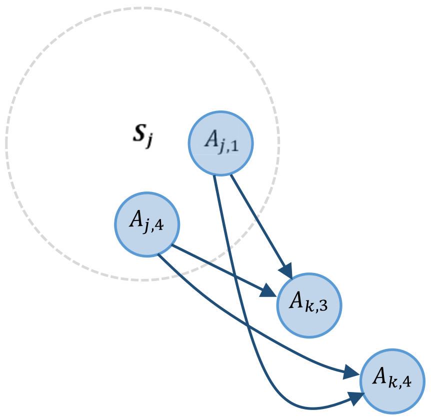

Fig. 7. The local *DBG* corresponding to sensor S_j in Fig. 5.

fact sectors of sensors) on the current barrier path so far, *length of current barrier path* (number of sensors on current barrier path), *selected action* (AID of the automaton in receiver which must be activated), *sensor ID of receiver* and the *dynamic threshold* (T_k). The dynamic threshold will be put in ACTIVATION packets by $A_{\mathbb{S}}$ and other nodes only forward it without any change.

When a sensor node receives the ACTIVATION packet, looks for selected action (automaton) in its local *DBG*. If the selected automaton does not exist and had been pruned (e.g. $A_{j,2}$ in Fig. 7), it means the barrier construction via selected actions by previous sensors is not possible. If the selected automaton exists in local *DBG* of receiver sensor, the corresponding automaton will be activated. Again, the activated automaton selects one of its possible actions stochastically, puts its SID and AID to the queue of sensors and automatons on the current barrier path, increases the length of current barrier path, sets the SID and AID of the receiver and sends the ACTIVATION packet to the receiver.

This process will be continued till the ACTIVATION packet reaches to $A_{\mathbb{T}}$. This means the current attempt to create barrier line has been succeeded. $A_{\mathbb{T}}$ compares the length of current barrier path with the dynamic threshold (T_k) and if it is less, selected actions on the barrier path will be rewarded. To do it, $A_{\mathbb{T}}$ creates a REWARD packet, puts the queue of SIDs and AIDs on current constructed barrier path (which was included in ACTIVATION packet) and its *length* ($\omega(\tau)$) into the packet and sends it back to the last sensor in the queue. The receiver of REWARD packet will update APV of its corresponding automaton and sends it back again to the previous sensor in the queue. This process will be continued until REWARD packet reaches to $A_{\mathbb{S}}$.

$A_{\mathbb{S}}$ updates its APV too and then save the constructed barrier path (queue of SIDs and AIDs) and its length as the best known barrier line in current stage. Also, $A_{\mathbb{S}}$ will update the dynamic threshold ($T_k \leftarrow [(k-1)T_{k-1} + \omega(\tau)]/k$) similar to CBCDLA. On the other hand, if the timer which has been set by $A_{\mathbb{S}}$ for current stage expires, it means the barrier path construction has been stopped in the middle of the way because it reached to an automaton with no outgoing edge or the length of constructed barrier line was not less than best known constructed barrier line in that stage. Due to L_{R-I} scheme of our learning algorithm, in this situations no changes will be made to APVs or the dynamic threshold (T_k). Therefore, next stage will be started by $A_{\mathbb{S}}$ by selecting one of its actions stochastically and sending ACTIVATION packet to the next sensor in the barrier path.

5.4.3. Barrier selection phase

If number of repetitions in the current stage exceeds $\mathbb{K}$ or the probability of last barrier line is greater than $\mathbb{P}$, $A_{\mathbb{S}}$ creates a BARRIER packet and includes SID of sensors on the best known barrier line and their corresponding automatons (AIDs) and sends it to the next sensor in the queue of barrier path. Upon receiving the BARRIER packet, each sensor will tune its orientation and forward it to the next sensor in the queue.

5.5. Analysis and methodology

5.5.1. Analysis of algorithm

While using centralized algorithms in sensor networks is not applicable in many scenarios, it also has significant confliction with sensor networks philosophy. Therefore, we consider the optimal solution and the CBCDLA to compare the results and the trend of the algorithms. As it is explained in related works section, while most of related works has been presented for omnidirectional sensor networks, we considered directional sensor networks with ability to adjust the orientation. Due to time-varying and dynamic characteristics of sensor networks, we used distributed learning automata in our algorithm. As the similar existent distributed algorithm we consider Zhang et al. [6] to compare the results of our distributed algorithm (DBCDLA).

In comparison to other algorithms, our proposed algorithms need to reside q automatons in each sensor node before deployment where q is the number of possible orientations of sensors (see section 2). Therefore, each node needs more space to store the automatons. Also, like Zhang et al. algorithm the algorithm needs some data exchanges mostly in phase 1 (initialization) and phase 2 (activation). However, as it will be shown in simulation section (experiment 4), our algorithm shows less message overload in comparison to Zhang's algorithm. Also, using a near-optimal algorithm such as DBCDLA helps to find more barrier lines and consequently extends the network lifetime by scheduling active sensors. In each round, a set of sensors will be activated to create the barrier line and other sensors will be set in their sleep mode.

5.5.2. *Convergence of algorithm*

In this section, we bring a theorem that proves the convergence of the proposed algorithm. The theorem 1 confirms that the APVs will be converged to the optimal solution as much as possible when all automatons use the linear reward-inaction (L_{R-I}) reinforcement scheme to update their APVs. Because our approach is similar to approach that has been used in [8] for omnidirectional sensors, we leave the theorem without prove and refer readers to it.

Theorem 1. *Let $q_i(k)$ be the probability of barrier path τ_i at stage k. If $\underline{q}(k)$ is updated according to linear reward-inaction reinforcement scheme as CDBDLA, then there exists a learning rate $a^*(\varepsilon)\,\varepsilon\,(0,\ 1)$, $\forall\,\varepsilon > 0$ so that for all $a \in (0,\ a^*)$, we have $prob[\lim_{k\to\infty} q_i(k) = 1] \geq 1-\varepsilon$.*

Proof. See [8], page 54. □

6. Simulation

In this section, we present the results of four groups of simulations that have been done to evaluate the performance of the proposed algorithms. Like similar researches, we consider number of barrier lines as the major performance parameter.

To evaluate the performance of the proposed algorithms, we have simulated them using the J-Sim platform with Sensor-Sim patch [24] on a PC with 8 GB of RAM and an Intel Core i5 3.5 GHz CPU.

In each scenario of the simulations, n homogenous sensor nodes with identical sensing radius (r) and transmission range (R) are randomly deployed in a $200 \times 150\,m^2$ rectangle region where $R = 2r$. Location of nodes is fixed and all nodes know their location. Each sensor node has an omnidirectional antenna IEEE 802.11 with CSMA/CA for its medium access control protocol. The disk of possible orientations of each sensor is divided into $q = 2\pi/\varphi$ non-overlapping sensing sectors with equal sizes, where φ is the field-of-view angle of sensors (e.g. for a sensor with $\varphi = \pi/2$, number of possible orientations will be $q = 2\pi/\varphi = 4$). Due to random deployment of sensors, orientation vector ($\vec{v}$) of each sensor will be belonged to [0, 2π). Each simulated case is repeated for 50 runs and its average is computed.

In the first group of experiments, effect of *number of nodes* on *number of barrier lines* will be evaluated, in the second group of experiments, the effect of *sensing radius of sensors* on *number of barrier lines* will be examined and in the third group of experiments, the impact of *learning rate of the algorithms* on *number of barriers* will be considered. Each group of experiments will be done for three different field-of-view angles ($\varphi \in \{\pi/2,\ \pi/3,\ \pi/4\}$). Also, in the fourth group of experiments, impact of *number of sensors* and *sensing radius of sensors* on *message overload of network* and *residual energy of active nodes* will be examined for a sample field-of-view angle ($\varphi = \pi/3$).

To evaluate the proposed algorithms, we compare the results of our algorithms with the *optimal solution*, a *greedy algorithm* and also the *distributed algorithm by Zhang* et al. [6]. The greedy algorithm simply constructs the barrier lines sequentially from left to right one by one. It must be noticed that while the proposed algorithm by Zhang et al. guarantees weak barrier coverage, our proposed algorithm guarantees strong barrier coverage.

Experiment 1. In this group of simulations, we evaluated the effect of *number of nodes* in *number of barrier lines*. Through these experiments, we changed the number of sensors from 100 to 400 in 50 intervals, while the sensing radius is fixed to 30 ($r = 30\,m$) and learning rate of algorithm for reward reinforcement is set to 0.1 ($a = 0.1$) for CBCDLA and 0.2 ($a = 0.2$) for DBCDLA. Results have been shown in Figs. 8 to 10 for three different field-of-view angles.

Results show that the number of disjoint barriers found by the CBCDLA algorithm is very close to the optimal values and outcomes the values of greedy algorithm completely. Also, however number of barrier lines constructed by DBCDLA is less than CBCDLA, it outcomes the Zhang's algorithm completely. Moreover, the number of disjoint barriers monotonically increases with the network size. This is because the more sensors a network has, more overlaps occur and this will lead to more disjoint barriers.

The charts expose that while CBCDLA has at most 12.5% difference with the optimal solution, the DBCDLA has at most 25% difference. In the other hand, charts show that CBCDLA could improve the results of greedy solution up to 50% while DBCDLA could improve the results of greedy solution up to 47% and the result of Zhang et al. solution up to 40.47%.

Experiment 2. In this group of simulations, the impact of *sensing radius of sensors* (r) on *number of barrier lines* will be considered. While the *number of sensor nodes* is fixed to 300 ($n = 300$) and *learning rate of algorithm* for reward reinforcement is set to 0.1 for CBCDLA ($a = 0.1$) and 0.2 for DBCDLA ($a = 0.2$), *sensing radius of sensors* will be increased from 15 m to 40 m in 5 m intervals. The simulation results have been shown in Figs. 11 to 13.

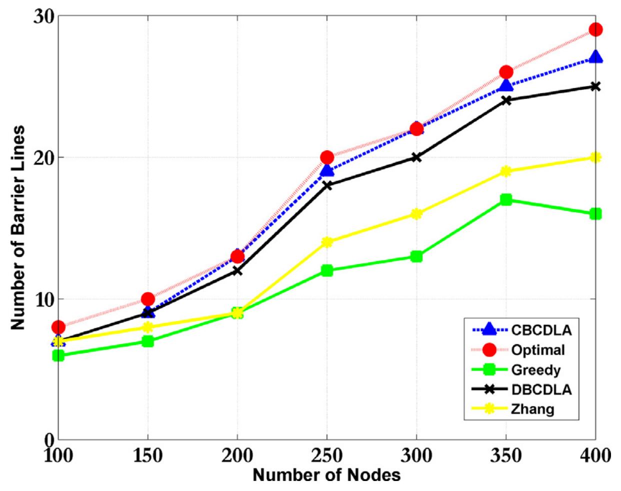

Fig. 8. Effect of number of nodes on number of barrier lines when $r = 30\ m$ and $\varphi = \pi/2$.

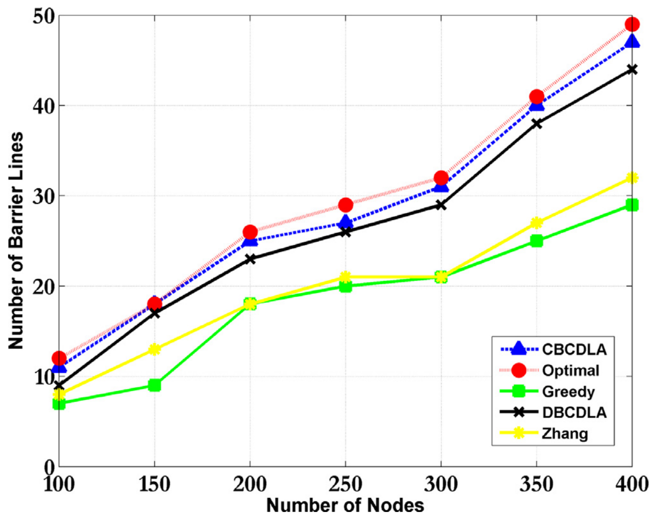

Fig. 9. Effect of number of nodes on number of barrier lines when $r = 30\ m$ and $\varphi = \pi/3$.

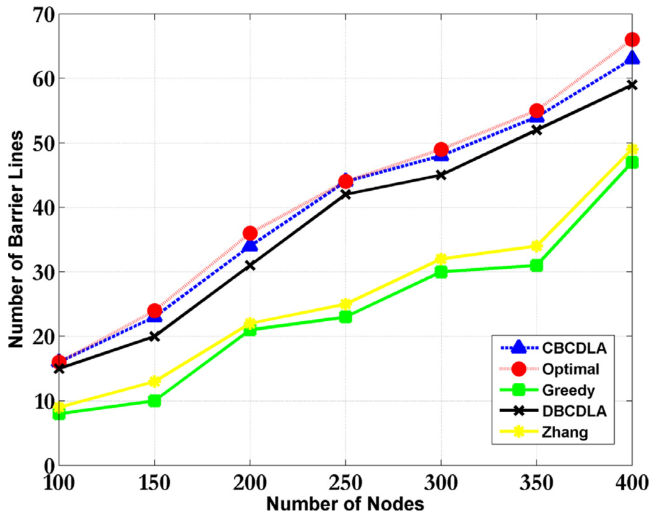

Fig. 10. Effect of number of nodes on number of barrier lines when $r = 30\ m$ and $\varphi = \pi/4$.

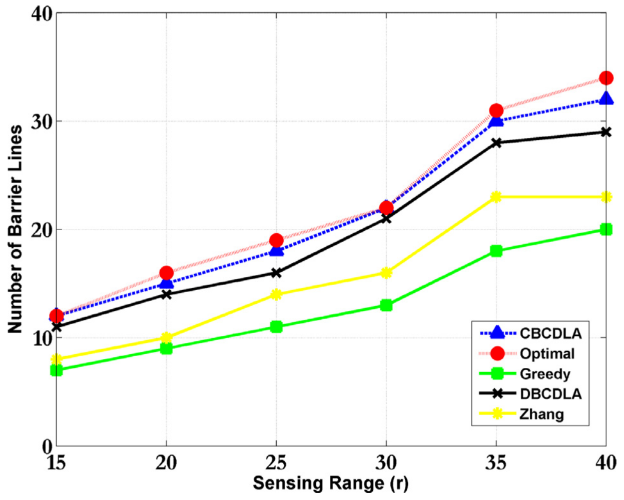

Fig. 11. Effect of sensing radius of sensors on number of barrier lines when $n = 300$ and $\varphi = \pi/2$.

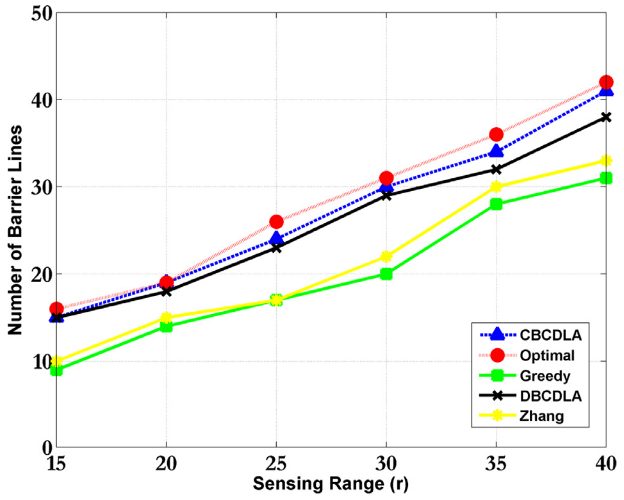

Fig. 12. Effect of sensing radius of sensors on number of barrier lines when $n = 300$ and $\varphi = \pi/3$.

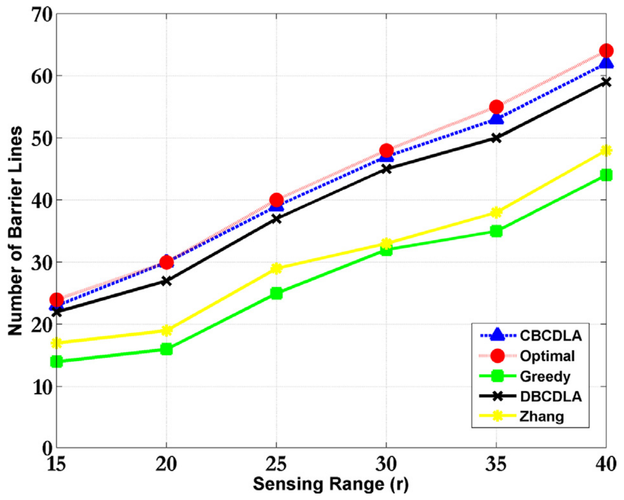

Fig. 13. Effect of sensing radius of sensors on number of barrier lines when $n = 300$ and $\varphi = \pi/4$.

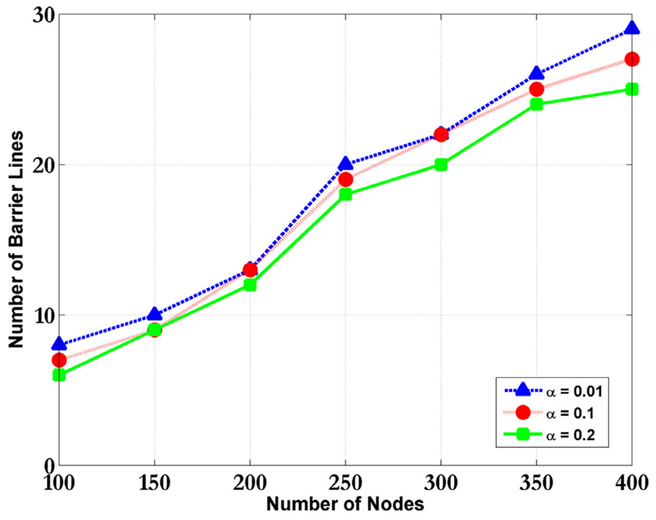

Fig. 14. Effect of learning rate on number of barrier lines when $r = 30$ m and $\varphi = \pi/2$.

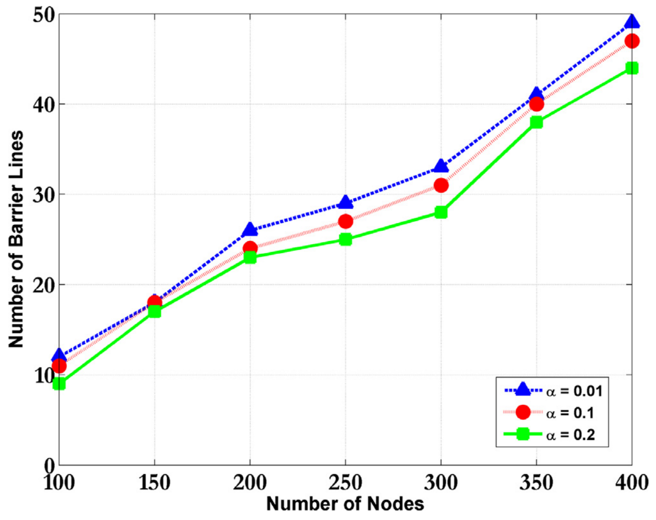

Fig. 15. Effect of learning rate on number of barrier lines when $r = 30$ m and $\varphi = \pi/3$.

The results reveal that *number of barrier lines* constructed by CBCDLA is very close to optimal ones and outcomes greedy algorithm completely. Also, while results of DBCDLA are very close to CBCDLA, they completely outcome Zhang's algorithm. Moreover, it could be concluded that the *number of barrier lines* monotonically increases with increase of sensing radius. It is because bigger *sensing radius of sensors* will lead to more overlap between adjacent sensor nodes and possibility to construct more barrier lines. The charts confirm that CBCDLA has at most 7.69% difference with the optimal solution, while the difference for DBCDLA is at most 15.78%. Also, charts show that CBCDLA could improve the results of greedy solution up to 41.66% while DBCDLA could improve the results of greedy solution up to 40% and the result of Zhang et al. solution up to 33.33%.

Experiment 3. In this section, the effect of *learning rate of algorithm* for reward reinforcement on *number of barrier lines* will be considered. We considered three different values for learning rate of CBCDLA ($a \in \{0.01,\ 0.1,\ 0.2\}$). The number of sensors is changed from 100 to 400 in intervals of 50 and sensing radius is fixed ($r = 30$ m). The results have been shown in Figs. 14 to 16.

As it is seen, the *learning rate of algorithm* and *number of barrier lines* have reverse relation. When *learning rate of algorithm* decreases, *number of barrier lines* increases and vice versa. When learning rate is set to 0.01 the results will be almost same as the optimal solution.

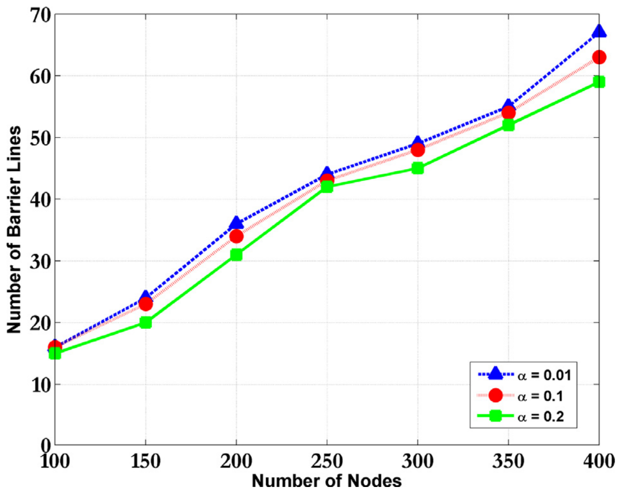

Fig. 16. Effect of learning rate on number of barrier lines when $r = 30\ m$ and $\varphi = \pi/4$.

Moreover, the charts expose that using very small learning rate as 0.01 could improve the results up to 12.5% against learning rate 0.1 and up to 25% against learning rate 0.2. Also, there is at most 18.18% difference between the results of learning rate 0.1 and 0.2.

Experiment 4. In this section, we consider *message overload* and *energy consumption* of DBCDLA and the Zhang's algorithm [6]. First, we present two charts for *message overload* against *number of sensors* and also *sensing radius of sensors*. Then, two charts for *residual energy of nodes* versus *number of nodes* and *sensing radius of sensors* will be presented. The *residual energy* is defined as the average energy level of active nodes after running the algorithms. The experiments of this section will be done on sensor nodes with sample field-of-view angle equal to $\pi/3$.

Like [20], we consider the energy model proposed by Heinzelman et al. [25] to evaluate the energy consumption of the algorithms in this experiment. In this model, each sensor node needs 50 $(\frac{nJ}{bit})$ to run the transmitter or the receiver circuit. Also, 100 $(\frac{pJ}{bit}/m^2)$ is needed by each sensor to handle the transmitter amplifier. Therefore, the required energy to receive $M\,(bit)$ of data could be estimated as

$$\mathrm{M(bit)}\left[50\left(\frac{nJ}{bit}\right)\right] = 50M\ \ (nJ) \tag{3}$$

Also, the required energy for sending $M\,(bit)$ data to a destination in a distance equal to $x\,(m)$ could be estimated by following equation

$$\mathrm{M(bit)}\left[50\left(\frac{nJ}{bit}\right)\right] + \mathrm{M(bit)}\left[100\left(\frac{pJ}{bit}/m^2\right)\right]x^2\ \left(x^2\right) = 50M\ \ (nJ) + 100Mx^2\ \ (pJ) \tag{4}$$

where in Eq. 3 and 4, nJ and pJ denote nano (10^{-9}) and pico (10^{-12}) Joule, respectively.

The maximum energy level of sensors is assumed to be $2.0\,J$ and the initial energy level of nodes will be assigned randomly and uniformly on $[1.5\,(J),\ 2.0\,(J)]$.

Fig. 17 shows the impact of *number of nodes* on *message overload*. As seen, while for all number of nodes message overload for DBCDLA is less than Zhang's algorithm, the difference between two charts increases monotonically with number of nodes. This is because, the *message overload* in DBCDLA is mostly dependent to *number of nodes* required for constructing the barrier line (in other words, sensing radius of nodes and width of RoI). While message overload in Zhang's algorithm is mostly dependent on average number of neighbors for each sensor. Therefore, when number of nodes increases, overlap between adjacent nodes and consequently message overload increase significantly.

Fig. 18 represents the impact of *sensing radius of nodes* on *message overload* when number of nodes is fixed and equal to 300. As seen for small sensing radius (less than or equal to 20 m), *message overload* of Zhang's algorithm is less than DBCDLA. While message overload of DBCDLA is almost identical for different sensing radii. As mentioned, the main parameter for message overload in DBCDLA is number of required sensors to construct the barrier line. Therefore, while bigger sensing radius leads to barrier lines with less sensors and consequently less message overload, on the other hand, higher sensing radius leads to more barrier lines and consequently higher message overload. As seen in Fig. 18, combination of these two makes almost identical message overload for different sensing radii in DBCDLA. Also, it could be found out, that message overload for Zhang's algorithm will increase exponentially for big sensing radii (more than 25 m) and number of sensors.

As our next experiment, we consider the *residual energy of active nodes* after running algorithms. Fig. 19 shows the impact of *number of nodes* on *residual energy level of active nodes*. While residual energy for DBCDLA is more than Zhang's algorithm

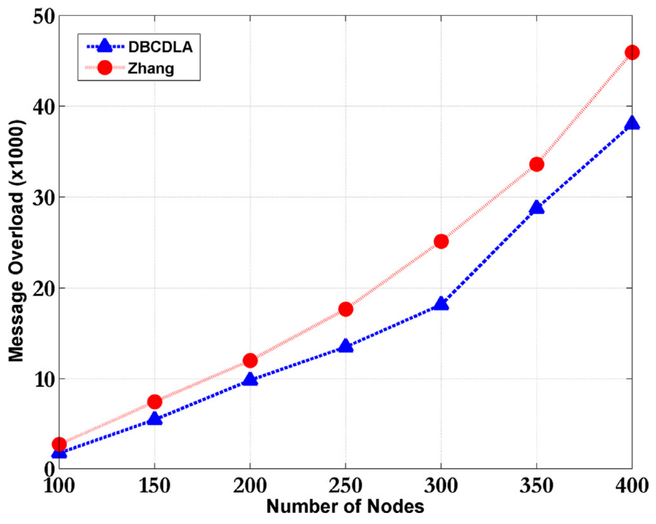

Fig. 17. Effect of number of nodes on message overload when $r = 30\ m$ and $\varphi = \pi/3$.

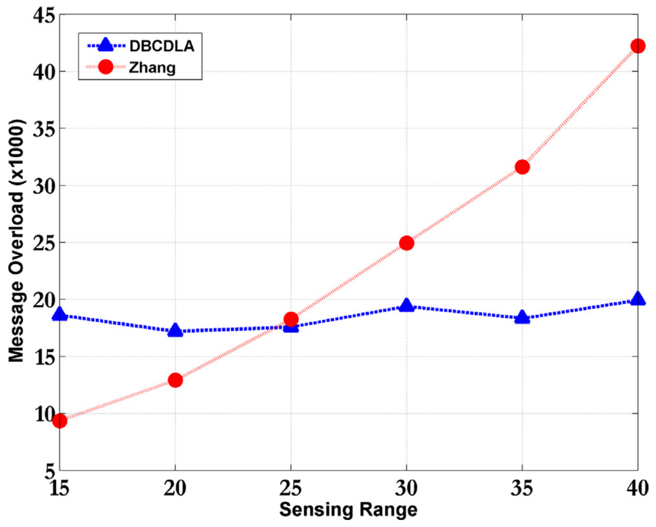

Fig. 18. Effect of sensing radius of nodes on message overload when $n = 300$ and $\varphi = \pi/3$.

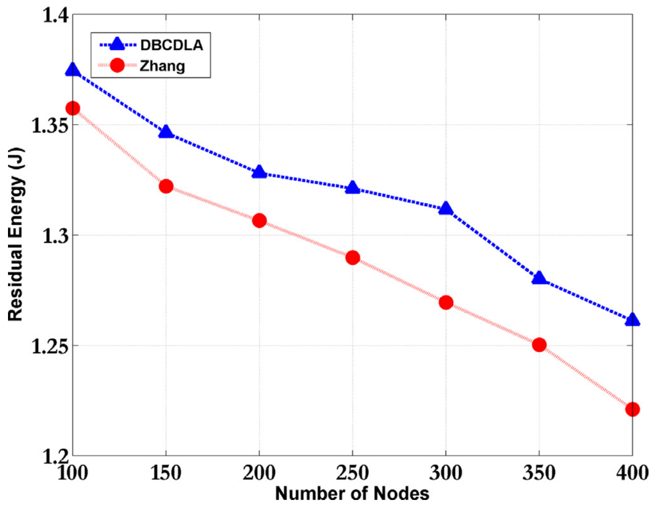

Fig. 19. Effect of number of nodes on residual energy when $r = 30$ m and $\varphi = \pi/3$.

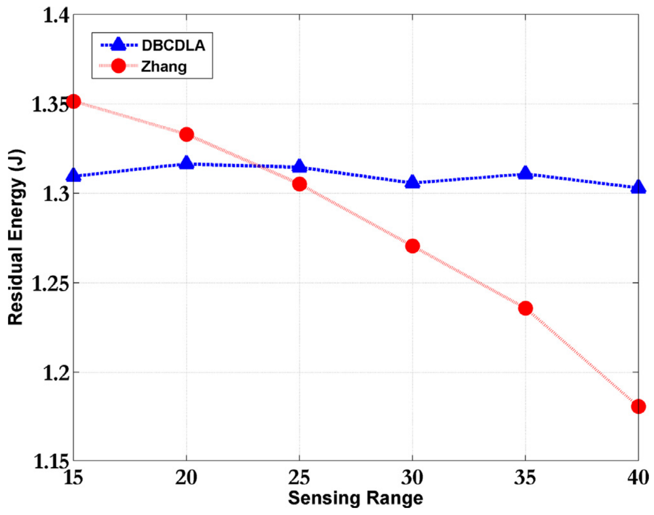

Fig. 20. Effect of sensing radius of nodes on energy residual when $n = 300$ and $\varphi = \pi/3$.

for different number of sensors, this difference increases significantly with number of nodes. Also, Fig. 20 represents the impact of *sensing radius of nodes* on *residual energy level of active nodes* when number of nodes is fixed and equal to 300. As it is expected (due to Fig 18), *residual energy* of Zhang's algorithm for sensing radius less than 20 *m* is higher than DBCDLA in this scenario, while for sensing radii bigger than 25 *m* the residual energy for Zhang's algorithm decreases exponentially. As it is explained, this trend is due to nature of the Zhang's algorithm which is dependent to number of neighbors and overlaps. The energy level of active nodes for DBCDLA is almost identical because it is dependent to number of nodes required for constructing the barrier line.

Regardless to hegemony of DBCDLA versus Zhang et al. according to results of *message overload* and *residual energy level of nodes*, it must be noticed that the main parameter for all barrier line algorithms is *number of barriers* constructed by them. In each round of sensor network, one set of sensors that is created by the barrier line algorithm will be activated and other nodes will be set on their sleep mode. Therefore, higher number of barrier lines created by an algorithm leads to more rounds for sensor network and consequently extends lifetime of the sensor network. As mentioned, DBCDLA could increase number of barrier lines up to 40.47% (according to experiment 1) and up to 33.33% (according to experiment 2) versus Zhang's algorithm.

7. Conclusion

In this paper, we presented two algorithms based on distributed learning automata to construct strong barrier lines in adjustable-orientation directional sensor networks. In this kind of networks, nodes have ability to adjust their orientation in a non-overlapping form such as camera sensor networks. First, we introduced directional barrier graph to model the network. While we used the graph to find the global barrier lines in the network, it also could be used as a basis in other algorithms.

Afterwards, we proposed a centralized algorithm based on directional barrier graph and distributed learning automata to create barrier lines. Because distributed algorithms are more suitable to be used in sensor networks, we also presented a distributed version of the algorithm by breaking the centralized algorithm to three phases and explaining the required operations in each phase.

Extensive simulations have been conducted to compare the performance of the proposed algorithms against the optimal solution, a greedy solution and a previously-proposed distributed algorithm. The results confirmed that the proposed algorithms outcome other algorithms while have negligible differences with optimal results. The simulations showed that our centralized and distributed algorithms have at most 12.5% and 25% differences with the optimal solution, respectively. Also, the distributed algorithm could improve the results of the previously-proposed algorithm up to 40.47%.

Possible future works include finding similar algorithms to construct barrier lines in heterogeneous and hybrid sensor networks and also three-dimensional sensor networks.

References

[1] Rashid B, Rehmani MH. Applications of wireless sensor networks for urban areas: a survey. J Netw Comput Appl 2016;60:192–219.
[2] Mostafaei H, Meybodi MR. An energy efficient barrier coverage algorithm for wireless sensor networks. Wireless Personal Commun 2014:1–17.
[3] Khanjary M, Sabaei M, Meybodi MR. Critical density for coverage and connectivity in two-dimensional aligned-orientation directional sensor networks using continuum percolation. IEEE Sensors J 2014;14(8):2856–63.

[4] Khanjary M, Sabaei M, Meybodi MR. Critical density for coverage and connectivity in two-dimensional fixed-orientation directional sensor networks using continuum percolation. J Netw Comput Appl 2015;57:169–81.
[5] Khanjary M, Sabaei M, Meybodi MR. Critical density in adjustable-orientation directional sensor networks using continuum percolation. Procedia Computer Sci 2017;116:548–55.
[6] Zhang L, Tang J, Zhang W. Strong barrier coverage with directional sensors. In: Proc. IEEE GLOBECOM; 2009. p. 1–6.
[7] Du J, Wang K, Liu H, Guo D. Maximizing the lifetime of k-discrete barrier coverage using mobile sensors. IEEE Sens J 2013;13(12):4690–701.
[8] Mostafaei H. Stochastic barrier coverage in wireless sensor networks based on distributed learning automata. Comput Commun 2015;55:51–61.
[9] Saipulla A, Westphal C, Liu B, Wang J. Barrier coverage of line-based deployed wireless sensor networks. In: Proc. IEEE INFOCOM; 2009. p. 127–35.
[10] Silvestri S, Goss K. MobiBar: An autonomous deployment algorithm for barrier coverage with mobile sensors. Ad Hoc Netw 2017;54:111–29.
[11] Kim D, Wang W, Son J, Wu W, Lee W, Tokuta AO. Maximum lifetime combined barrier-coverage of weak static sensors and strong mobile sensors. IEEE Trans Mobile Comput 2017;16(7):1956–66.
[12] Kim D, Kim Y, Li D, Seo J. A new maximum fault-tolerance barrier-coverage problem in hybrid sensor network and its polynomial time exact algorithm. Ad Hoc Netw 2017;63:14–19.
[13] Shih KP, Chou CM, Liu IH, Li CC. On barrier coverage in wireless camera sensor networks. In: Proc. IEEE AINA; 2010. p. 873–9.
[14] D. Tao, S. Tang, H. Zhang, X. Mao, H. Ma, Strong barrier coverage in directional sensor networks, Computer Communications, 35(8), 895–905, 2012.
[15] Wang Y, Cao G. Barrier coverage in camera sensor networks. In: Proc. ACM MobiHoc 2011 Art. ID 12.
[16] Yang R, Gao X, Wu F, Chen G. Distributed algorithm for full-view barrier coverage with rotatable camera sensors. In: Proc. IEEE GLOBECOM; 2015. p. 1–6.
[17] Cheng CF, Tsai KT. Encircled belt-barrier coverage in wireless visual sensor networks. Pervasive Mobile Comput 2017;38(1):233–56.
[18] Fan X, Chen Q, Che Z, Hao X. Energy-efficient probabilistic barrier construction in directional sensor networks. IEEE Sensors J 2017;17(3):897–908.
[19] Packar NH, Wolfram S. Two-dimensional cellular automata. J Stat Phys 1985;38(5):901–46.
[20] Torkestani JA, Meybodi MR. clustering the wireless ad hoc networks: a distributed learning automata approach. J Parallel Distrib Comput 2010;70(4):394–405.
[21] Beigy H, Meybodi MR. A mathematical framework for cellular learning automata. Adv Complex Syst 2004;7(3–4):295–320.
[22] Khanjary M, Sabaei M, Meybodi MR. A percolation algorithm for directional sensor networks. In: Proc. IEEE EIT; 2015. p. 478–83.
[23] Beigy H, Meybodi MR. Cellular learning automata with multiple learning automata in each cell and its applications. IEEE Trans Syst, Man, Cybern, Part B 2010;40(1):54–65.
[24] J-Sim official web site http://sites.google.com/site/jsimofficial. Accessed September 7th, 2017.
[25] Heinzelman WR, Chandrakasan A, Balakrishnan H. Energy-efficient communication protocol for wireless microsensor networks. In: Proc. IEEE HICSS; 2000. p. 1–10.

Mohammad Khanjary received his B.*Sc*. degree from Amirkabir University of Technology, Tehran, Iran, his M.*Sc*. degree from Qazvin Branch, Islamic Azad University, Qazvin, Iran and his Ph.D. from Science and Research Branch, Islamic Azad University, Tehran, Iran all in the field of Computer Engineering in 2004, 2009 and 2017, respectively. His current research interests include wireless networks and learning automata.

Masoud Sabaei received his M.*Sc*. and Ph.D. degrees from Amirkabir University of Technology, Tehran, Iran in the field of Computer Engineering in 1995 and 2000, respectively. Currently, he is faculty member of Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran and his current research interests include wireless networks, software defined networks, and telecommunication network management.

Mohammad Reza Meybodi received the M.*Sc*. and Ph.D. degrees from Oklahoma University, USA, in 1980 and 1983, respectively in Computer Science. Currently, he is a full professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran and his current research interests include channel management in cellular networks, learning systems, parallel algorithms, soft computing and software development.