Ad Hoc & Sensor Wireless Networks, Vol. 0, pp. 1–33
Reprints available directly from the publisher
Photocopying permitted by license only

A Cellular Learning Automata-based Algorithm for Solving the Coverage and Connectivity Problem in Wireless Sensor Networks

REZA GHADERI[1], MEHDI ESNAASHARI[2] AND MOHAMMAD REZA MEYBODI[2]

[1]*Computer Engineering Department, Islamic Azad University, Arak, Iran*
E-mail: Ghaderi.re@gmail.com
[2]*Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran*
E-mail: Esnaashari@aut.ac.ir, mmeybodi@aut.ac.ir

Presence of redundant nodes is common in wireless sensor networks because of various reasons such as high probability of failures and necessity of long lifetime. When such redundancy exists, some distributed algorithms are needed for selecting minimal subset of nodes as active nodes in a manner that network area is covered entirely with the selected active nodes. In this paper, a distributed algorithm is proposed which attempts to minimize the number of active nodes in the network using cellular learning automata in such a way that the following two conditions are met: 1. network area is covered entirely, and 2. network of selected active nodes is connected. In the proposed algorithm, each node is equipped with a learning automaton which locally decides for the node to be active or not based on the remaining energy of the node and its neighbors' situations. To ensure the network connectivity, we analytically determine the radio transmission range of sensor nodes according to their sensing range so that complete coverage of the network area guarantees the connectivity of active nodes. The time and space costs of the proposed algorithm are analytically determined and compared with those of similar existing algorithms such as PEAS and PECAS. Simulation results in J-Sim simulator environment specify the efficiency of the proposed algorithm over existing algorithms such as PEAS and PECAS—especially against high ratio of unexpected failures and nodes' energy depletion.

Keywords: Wireless Sensor Networks, Cellular Learning Automata, Network Coverage, Active Nodes Connectivity, Energy Conserving

1. INTRODUCTION

One of the basic issues in wireless sensor networks is the coverage problem which specifies how the network is monitored by sensor nodes [1]. The coverage concept is a measure of the quality of service (QoS) of the sensing function and is subject to a wide range of interpretations due to a large variety of sensors and applications [2]. Various coverage formulations have been proposed in literature among which following three are most discussed [3]:

- Area coverage: Covering (monitoring) the whole area of the network is the main objective of area coverage problem.
- Point coverage: The objective of point coverage problem is to cover a set of stationary or moving target points using as little sensor nodes as possible.
- Barrier coverage: Barrier coverage can be considered as the coverage with the goal of minimizing the probability of undetected penetration through the barrier (sensor network).

The focus of this paper is on a sub-problem of area coverage problem, called the set cover problem. In set cover problem, by assuming redundancy in the number of deployed sensor nodes throughout the area of the network, the main objective is to select a subset of sensor nodes as active nodes so that the set of active nodes covers the entire area of the network. Another important objective of the set cover problem is to select the desired subset of active nodes so that the network of active nodes is connected. Connectivity of the set of active nodes is a necessity to ensure that all active nodes can transfer their sensed data to the base station (sink).

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

In this paper, a distributed density control algorithm called Geographical Density Control based on Cellular Learning Automata (GDC-CLA) for wireless sensor networks is proposed. The main purpose of this algorithm is to maintain coverage and connectivity in the network with minimum number of active nodes, so that the total consumed energy of nodes is minimized. GDC-CLA can be used in long-lived wireless sensor networks as an energy con-

serving protocol. To ensure the network connectivity, we analytically determine the radio transmission range of sensor nodes according to their sensing range in such a way that complete coverage of the network area guarantees the connectivity of active nodes. This way, we need to only focus on the coverage problem without any concern about connectivity. In GDC-CLA, each node is equipped with a learning automaton which decides for the node to be active or not based on the states (either active or not) of its neighboring sensor nodes as well as its own energy level.

We used J-Sim simulator to evaluate the performance of GDC-CLA. Simulation results specify the efficiency of GDC-CLA over existing algorithms such as PEAS [4] and PECAS [5] —especially against high ratio of unexpected failures and nodes' energy depletion.

The rest of this paper is organized as follows. In Section 2, a literature overview is presented. In Section 3, cellular automata, learning automata and cellular learning automata are briefly reviewed. The problem statement is given in Section 4. Section 5 explains the analytical determination of the relationship between coverage and connectivity. The proposed density control algorithm is described in Section 6. Analytical studies of the time and space costs are presented in Section 7. Simulation results are given in Section 8. Section 9 is the conclusion.

2. RELATED WORK

Coverage and connectivity problem is one of the basic issues in a wireless sensor network about which many distributed and centralized algorithms have been presented [4–12].

In [10], the problem of finding the maximal number of covers is paid attention to in which a cover is defined as a set of nodes that can completely cover the monitored area. This is an NP-complete problem which needs centralized heuristic solution. [11] pays attention to coverage and connectivity problem in wireless sensor networks (CCP-WSN) with a centralized manner. The solution is a hybrid solution consisting of two phases: a genetic algorithm and a local search based on two classical graph algorithms. The genetic algorithm is used for coverage problem with the minimal number of nodes needed to cover entire area of the network. Classic algorithms of Dijkstra and Prim guarantee the connectivity of active nodes.

ASCENT protocol [7] configures the topology of sensor network automatically. In ASCENT, each node, based on the number of active neighbors and data loss rate which is measured through data traffic, decides to choose for sleeping or awake state. In ASCENT, the complete coverage of monitored region is not paid attention to.

[8, 9] pay attention to the presentation of a relation between sensing range and radio communication range of nodes so that if the relation is established,

entire area coverage guarantees network connectivity. In this relation, the radio communication range is taken into consideration at least twice the sensing range. In [12] a localized algorithm based on simultaneous network for maintaining connected area coverage under various ratios of sensing and radio communication radiuses is proposed.

In [4, 6] a distributed, probing-based density control algorithm named PEAS is presented. In PEAS, a subset of nodes remains in active mode to maintain coverage and connectivity and the rest of nodes go to sleep. Each sleeping node checks the existence of its working neighbor nodes after awakening. If no working node is within its probing range, it starts to operate in the active mode; otherwise, it returns to sleeping state. PEAS guarantees the working nodes connectivity based on determining the relationship between probing range and radio transmission range but it does not guarantee complete coverage of the monitored area. In this algorithm, a working sensor node remains awake continuously until its failure or depletion of battery power. PECAS algorithm [5] operates similar to PEAS, but unlike PEAS, in PECAS a node goes to sleep mode after a duration existing in working state and one of its neighboring nodes replaces with it. Thus, there is established a balancing for energy consumption of sensor nodes.

3. CELLULAR LEARNING AUTOMATA

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

Cellular Automata: Cellular Automata (CA) is a mathematical model for systems consisting of large number of simple identical components with local interactions. CA is a non-linear dynamical system in which space and time are discrete. It is called cellular because it is made up of cells like points in a lattice or like squares of checker boards, and it is called automata because it follows a simple rule [13]. The simple components act together to produce complicated patterns of behavior. CA performs complex computations with a high degree of efficiency and robustness. It is especially suitable for modeling natural systems that can be described as massive collections of simple objects interacting locally with each other [14, 15]. Informally, a d-dimensional CA consists of an infinite d-dimensional lattice of identical cells. Each cell can assume a state from a finite set of states. The cells update their states synchronously on discrete time steps according to a local rule. The new state of each cell depends on the previous states of a set of cells, including the cell itself, and constitutes its neighborhood [16]. The state of all cells in the lattice is described by a configuration. A configuration can be described as the state of the whole lattice. The rule and the initial configuration of the CA specify the evolution of CA that tells how each configuration is changed in one time step.

Learning Automata: Learning Automata (LA) is an adaptive decision-making device that operates on unknown random environments. It has a finite set of actions to choose from and at each stage, its choice (action) depends upon its action probability vector. For each action chosen by the automata, the environment gives a reinforcement signal with fixed unknown probability distribution. The automata then updates its action probability vector depending upon the reinforcement signal at that stage, and evolves to some final desired behavior. A class of LA is called variable structure learning automata and is represented by quadruple $\{\alpha, \beta, p, T\}$ in which $\alpha = \{\alpha_0, \alpha_1, ..., \alpha_{r-1}\}$ represents the action set of the automata, $\beta = \{\beta_0, \beta_1, ..., \beta_{r-1}\}$ represents the input set, $p = \{p_0, p_1, ..., p_{r-1}\}$ represents the action probability set, and finally $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ represents the learning algorithm. Let α_i be the action chosen at time step n. Then, the recurrence equation for updating p is defined as

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

for favorable responses, and

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)p_j(n) \qquad \forall j, \quad j \neq i \end{aligned} \tag{2}$$

for unfavorable ones. In these equations, a and b are reward and penalty parameters, respectively. If $a = b$, learning algorithm is called L_{R-P}[1]; if b << a, it is called $L_{R\varepsilon P}$[2]; and if $b = 0$, it is called L_{R-I}[3]. For more information about learning automata, the reader may refer to [17, 18].

Cellular Learning Automata: Cellular learning automata, which is a combination of cellular automata and learning automata, is a powerful mathematical model for many decentralized problems and phenomena. The basic idea of CLA, which is a subclass of stochastic CA, is to utilize LA to adjust the state transition probability of stochastic CA. A CLA is a CA in which a learning automaton is assigned to every cell. The learning automaton residing in a particular cell determines its action (state) on the basis of its action probability vector. Like CA, there is a rule that the CLA operates under. The local rule of CLA and the actions selected by the neighboring LAs of any particular LA determine the reinforcement signal to the LA residing in a cell.

[1]Linear Reward-Penalty
[2]Linear Reward epsilon Penalty
[3]Linear Reward-Inaction

The neighboring LAs of any particular LA constitute the local environment of that cell. The local environment of a cell is nonstationary because the action probability vectors of the neighboring LAs vary during the evolution of the CLA. The operation of a CLA could be described as follows: At the first time step, the internal state of every cell is specified. This initial value may be chosen on the basis of past experience or at random. In the second time step, each LA selects one of its actions based on its action probability vector and performs it on the environment. Next, the local rule of CLA determines the reinforcement signal to each LA. Finally, each LA updates its action probability vector on the basis of the supplied reinforcement signal and the chosen action. This process continues until the desired result is obtained. A CLA is called synchronous if all LAs are activated at the same time in parallel. A CLA is called asynchronous (ACLA) if at a given time only some LAs are activated independently from each other, rather than all together in parallel. The LAs may be activated in either time-driven or step-driven manner. In time-driven ACLA, each cell is assumed to have an internal clock which wakes up the LA associated to that cell while in a step-driven ACLA, a cell is selected in fixed or random sequence. CLA has found many applications such as image processing [19–21], rumor diffusion [22], modeling of commerce networks [23], channel assignment in cellular networks [24], and sensor networks [25–27] to mention a few. For more information about CLA, the reader may refer to [22, 28].

Nomenclature

N	Number of sensor nodes in the network
s_i	Sensor node i ($1 \leq i \leq N$)
Ω	Rectangular network area
R_s	Sensing range of each node s_i
R_t	Transmission range of each node s_i
t	Current time
$O_{S_i}(t)$	Operation mode of each node s_i at time instant t
M	Number of target nodes in the network
tr_j	Target node j ($1 \leq j \leq M$)
c	Length of each squared cell in cellular coverage computation method
R_{cs}	Computational sensing range of each node s_i
cl_k	Squared cell k ($1 \leq k \leq$ Number of squared cells in Ω)
R_n	Neighborhood radius of each node s_i
LA_i	Learning automaton residing in each cell i of cellular learning automata
r	Number of actions in each LA_i
α_l	Action l ($0 \leq l \leq r-1$) of each LA_i
p_l	Probability of α_l ($0 \leq l \leq r-1$) of each LA_i

N_I	An upper bound on the number of sensor nodes required for completely covering Ω
τ_0	Maximum time interval between two subsequent transmissions of HELLO packets in the proposed algorithm
T	Maximum sleeping time of each node s_i in the proposed algorithm
Cn	The smallest positive integer satisfying inequality (12)
a	Reward parameter in each LA_i
b	Penalty parameter in each LA_i
Pr_0	A small predetermined probability used in inequality (12)
MS	Maximum speed of each node tr_j
N_{alive}	Number of sensor nodes which are alive during the network lifetime

4. PROBLEM STATEMENT

Consider a sensor network consists of N sensor nodes $s_1, s_2, \ldots s_N$ and one sink within a rectangular area (Ω). Sensor nodes, which are responsible for sensing and monitoring the area, are scattered randomly throughout the area of the network so that Ω is completely covered. All sensor nodes have the same sensing range of R_s and transmission range of R_t. Each sensor node s_i has 4 different modes of operation [29] as follows:

- **On-duty ($CPU_A S_A C_A$)**: CPU, sensing and communicating units are switched on referred to as active mode. A sensor node in the active mode is referred to as an active node.
- **Sensing Unit On-duty ($CPU_A S_A C_S$)**: The CPU and the sensing units are switched on, but the communicating unit is switched off.
- **Communicating Unit On-duty ($CPU_A S_S C_A$)**: The CPU and the communicating units are switched on, but the sensing unit is switched off.
- **Off-duty ($CPU_S S_S C_S$)**: CPU, sensing and communicating units are switched off referred to as sleep mode.

Note that in $CPU_A S_A C_A$, $CPU_A S_A C_S$, $CPU_A S_S C_A$ and $CPU_S S_S C_S$, index A stands for active and index S stands for sleep. At any instance of time, a sensor node can be only in one of the above 4 operation modes. The operation mode of a sensor s_i at time instant t (t is the current time) is denoted by $O_{S_i}(t)$.

As the focus of this paper is on the set cover problem, we assume that the number of sensors (N) is greater than the one required to cover the entire area of the network. Thus, a density control algorithm can be adopted to use this redundancy for prolonging the lifetime of the network. Such a controlling algorithm, selects a connected subset of sensor nodes as active nodes so that the set of active nodes fully covers Ω.

Definition 1. Network lifetime is defined as the time elapsed from the network startup to the time at which Ω is not further completely covered by the network due to node deaths.

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

Since the network lifetime is highly related to the objective of the network, in this paper, we consider a sensor network, whose objective is to track a set of targets $tr_1, tr_2, \ldots tr_M$ moving around within the area of the network according to the random waypoint mobility model [30]. We denote the Euclidean distance between a sensor node s_i located at $\left(x(s_i), y(s_i)\right)$ and a target tr_j located at $\left(x(tr_j), y(tr_j)\right)$ as $d(s_i, tr_j)$,i.e. $d(s_i, tr_j) = \sqrt{\left(x(s_i) - x(tr_j)\right)^2 + \left(y(s_i) - y(tr_j)\right)^2}$.

Assuming the binary sensing model (in which the sensing power of each sensor node is fixed within the sensing range and is zero out of the sensing range) [31] and sensing range of R_s for all sensor nodes in the network, we say a target node tr_j is sensed, detected or monitored by a sensor node s_i at time t if and only if $d(s_i, tr_j) \leq R_s$ and $O_{s_i}(t) = CPU_A S_A C_A$. The network detects a target node tr_j at time t if and only if at least one of the sensor nodes of the network detects tr_j at time t.

5. ENSURING CONNECTIVITY

In this section, we propose a constraint on the relationship between sensing and transmission ranges of a sensor node so that if satisfied, covering the network area entirely ensures network connectivity. This way, we need to only focus on the coverage problem without any concern about connectivity. To this end, we have to first specify the method which we used for computing the fraction of the area covered by a network.

5.1 Cellular Coverage Computation

In [8, 9], authors proved that complete coverage of a convex area implies connectivity among the set of working nodes if $R_t \geq 2R_s$. To compute the coverage, they divided the area into square grids. A grid is considered to be covered if the center of the grid is covered, and coverage is defined as the ratio of the number of grids that are covered by at least one sensor node to the total number of grids. But it is clear that covering the center of a grid does not necessarily imply complete coverage of that grid. Therefore, using this method for computing the coverage of the network area may result in a coverage which is more than its actual value. Regarding [8, 9], as connectivity among the working nodes depends on the area coverage, this problem becomes more serious. To remedy this problem, we propose a novel method for computing the coverage of the network area, called cellular coverage computation. In the proposed

method, to compute the fraction of the area which is covered by the network, network area is divided into a number of squared cells of the length c and the covered fraction of the network is set to the fraction of cells which are completely covered. A cell is completely covered if it is within the sensing range of at least one sensor node completely. Let R_{cs} be the computational sensing range of a sensor node which is defined according to equation (3).

$$R_{cs} = R_s - \sqrt{2}c \tag{3}$$

With the above definition of the computational sensing range, if there exists any point within a cell cl_k whose distance to a sensor node s_i is less than or equal to R_{cs}, then cell cl_k is completely covered by sensor node s_i (Figure 1).

It is straight forward that considering cellular coverage computation, resultant covered fraction of the network will be no more than the actual covered fraction. Based on the proposed cellular coverage computation method, we will prove that complete cellular coverage of a rectangular area implies connectivity of the active nodes iff $R_t \geq 2R_{cs} + 2c$.

5.2 The Proposed Constraint

Definition 2. Distance of two nodes s_i and s_j is defined according to the following equation:

$$d(s_i, s_j) = \sqrt{\left(x(s_i) - x(s_j)\right)^2 + \left(y(s_i) - y(s_j)\right)^2} \tag{4}$$

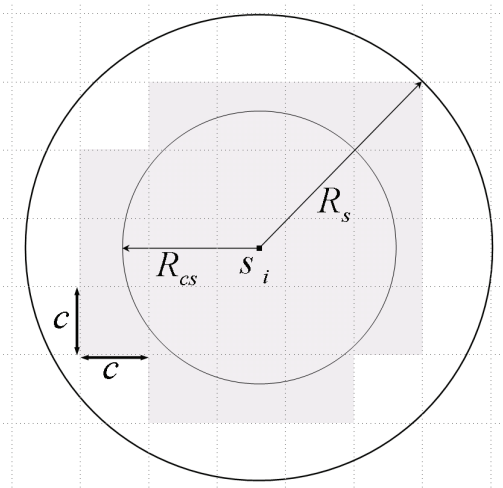


FIGURE 1
Checking whether a cell is completely covered by a sensor node s_i using the computational sensing range of the node (dark cells are covered by sensor node s_i).

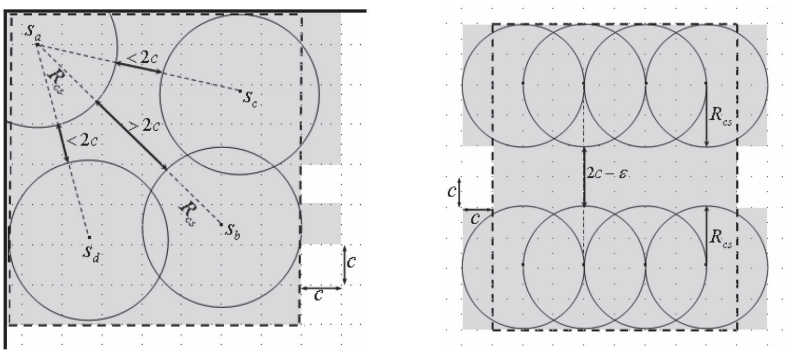
Definition 3. Distance of two disjoint non-empty sets of nodes S and S' is defined according to the following equation:

$$d(S,S') = \min_{s_i \in S, s_j \in S'} \left(d(s_i, s_j) \right) \tag{5}$$

Following lemma, gives the maximum distance between any two disjoint non-empty subsets of sensor nodes of a sensor network, which fully covers Ω.

Lemma 1. Consider a sensor network residing in a rectangular area which consists of homogeneous sensor nodes. Assume that the network is fully covered according to the cellular coverage computation method with cells of the length c. For this network, the supremum of the distance between any two disjoint non-empty subsets of sensor nodes is $2R_{cs} + 2c$.

Before giving the proof of this lemma, we first look at two special cases of sensor placements in a fully covering sensor network in order to have a better understanding of what Lemma 1 states. These two special cases are shown in Figures 2(a) and 2(b). In both cases, the area of the network (given by the dashed rectangle) is covered completely by the nodes. Although in Figure 2(a), the distance between two nodes s_a and s_b is greater than $2R_{cs} + 2c$, the connectivity of nodes s_a and s_b can be established through either node s_c or node s_d. Since all $d(s_a, s_c)$, $d(s_a, s_d)$, $d(s_b, s_c)$ and $d(s_b, s_d)$ are less than $2R_{cs} + 2c$, we can conclude that the maximum distance between any two subsets of nodes in this figure is less than $2R_{cs} + 2c$. Sensor nodes in Figure 2(b) are placed in such a way that the possible maximum distance between two subsets of nodes, which fully cover the area of the network, can be seen.



(a) (b)

FIGURE 2
(a) Although $d(s_a, s_b) > 2R_{cs} + 2c$ nodes s_a and s_b are connected through either of the nodes s_c or s_d; thus the maximum distance between any two subsets of nodes in this network is less than $2R_{cs} + 2c$. (b) The possible maximum distance between two subsets of nodes, which fully cover the area of the network.

Proof: Let the rectangle presented in dashed lines in Figure 3(a) be the area of the network. This area is fully covered by two subsets of nodes $S = \{s_a, s_c\}$ and $S' = \{s_b, s_d\}$. In this network, $d(S, S') = d(s_a, s_b)$, that is, s_a and s_b are the closest nodes among nodes of the two subsets S and S'. According to the network topology given in Figure 3(a), cells cl_1 and cl_2 can only be covered by node s_b. Similarly, cells cl_3 and cl_4 can only be covered by node s_a. Therefore, in this topology, the distance between nodes s_a and s_b is bounded from above to $2R_{cs} + 2c$, that is:

$$d(s_a, s_b) < 2R_{cs} + 2c \tag{6}$$

We now show that $2R_{cs} + 2c$ is indeed the supremum of $d(s_a, s_b)$ in any arbitrary topology. Since node s_b must cover cells cl_1 and cl_2, it can only revolve around point P. Since the lengths of $\overline{Ps_b}$ and $\overline{s_aP}$ are constants, smaller values for $\angle s_a P s_b$ will result in smaller distance between s_a and s_b. For instance, $\angle s_a P s_{b'}$ in Figure 3(b) is greater than $\angle s_a P s_{b''}$ in Figure 3(c), therefore, $\overline{s_a s_{b'}} > \overline{s_a s_{b''}}$. As a result, the supremum of $d(s_a, s_b)$ in any arbitrary topology is $2R_{cs} + 2c$. This indicates that the supremum of the distance between any two disjoint non-empty subsets of sensor nodes is $2R_{cs} + 2c$, and hence the lemma. □

In the following lemma, we will show that the supremum distance given in Lemma 1 ($2R_{cs} + 2c$) is the least possible value for transmission range, which if the area of the network is fully covered, ensures the network connectivity.

Lemma 2. Consider a sensor network residing in a rectangular area which consists of homogeneous sensor nodes. Assume that the network is fully covered according to the cellular coverage computation method with cells of the length c. This network is connected iff $R_t \geq 2R_{cs} + 2c$.

Proof:

Necessity condition. To prove the necessity condition, we must show that if the sensor network is connected, then $R_t \geq 2R_{cs} + 2c$. Alternatively, we can prove the contra-positive proposition which is: if $R_t < 2R_{cs} + 2c$ then the network is not necessarily connected. We show this by providing a counter-example. Consider the network given in Figure 4. In this network, a sensor node with computational sensing range R_{cs} and radio transmission range $R_t < 2R_{cs} + 2c$ is located at O and a number of sensor nodes are located within a square of the side length $R_t + \varepsilon$ centered at O. Note that $R_t + \varepsilon < 2R_{cs} + 2c$ and $\varepsilon > 0$. The network area (dashed square) is completely covered by this set of nodes. Although this network is completely covered, but it is not connected considering the fact that the distance between central node (located at O) and any of the other nodes is more than R_t.

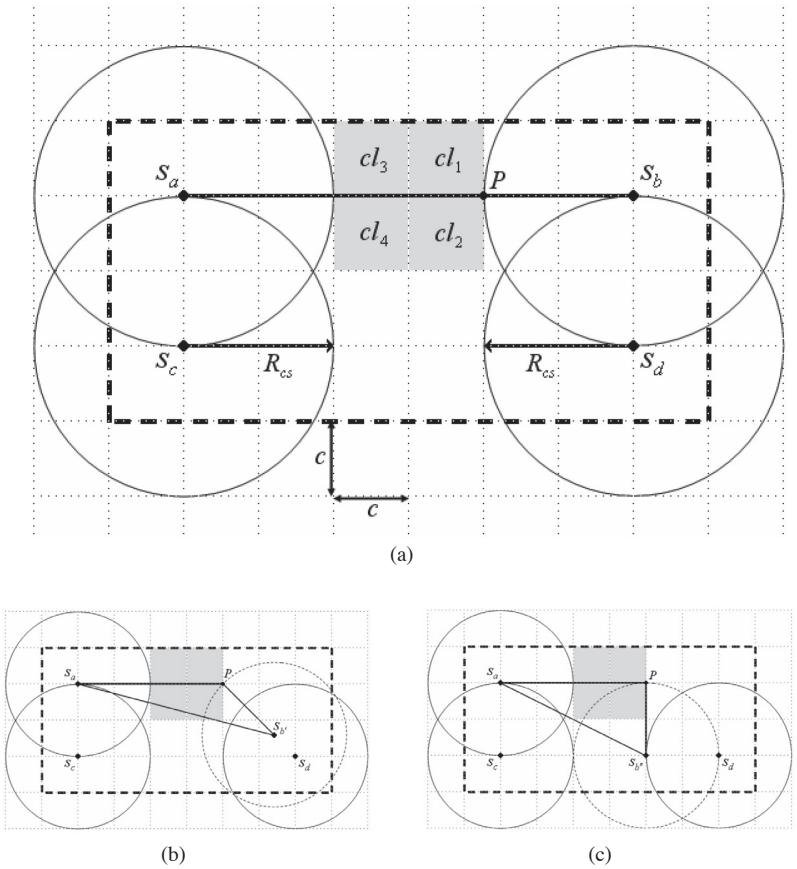


FIGURE 3
A sample network with two subsets of nodes $S = \{s_a, s_c\}$ and $S' = \{s_b, s_d\}$. The area of the network (rectangle in dashed lines) is fully covered. (a) The distance between the two subsets S and S' is at its maximum value. (b) and (c) Two sample cases which show smaller values for $\angle s_a P s_b$ will result in smaller distance between s_a and s_b .

Sufficiency condition. To prove the sufficiency condition, we must show that if $R_t \geq 2R_{cs} + 2c$ then the network is connected. We use contradiction to prove this property. Assume that $R_t \geq 2R_{cs} + 2c$, but the network is not connected. If the network is not connected, then there exists a pair of nodes between which no path exists. Assume (s_s, s_d) to be such pair of nodes with the least distance to each other among set of disconnected nodes (Figure 5).

Consider a circle centered on the line segment between s_s and s_d so that s_s is located on its perimeter. We will show via contradiction that at least one node like s_p must exist within this circle. Assume that no node exists inside the aforesaid circle. Then, the nearest node to the center of the circle is at least $R_{cs} + c$ farther from it. Let $s_{p'}$ be such a node which distance to the center of

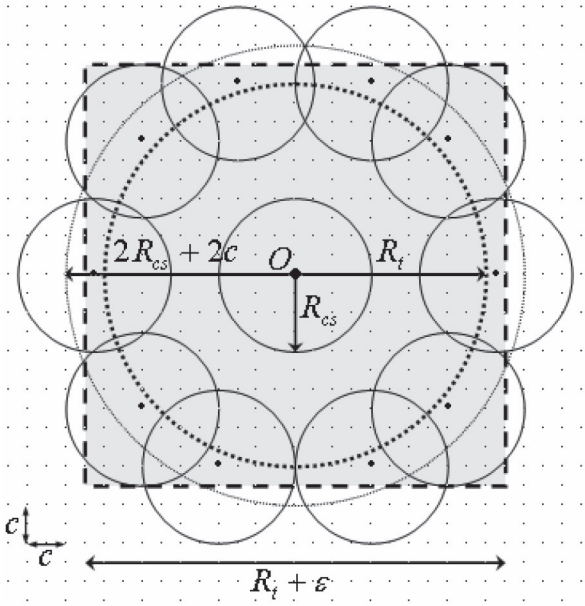


FIGURE 4
The necessity condition so that entire area coverage, guarantees the network connectivity is $R_t \geq 2R_{cs} + 2c$.

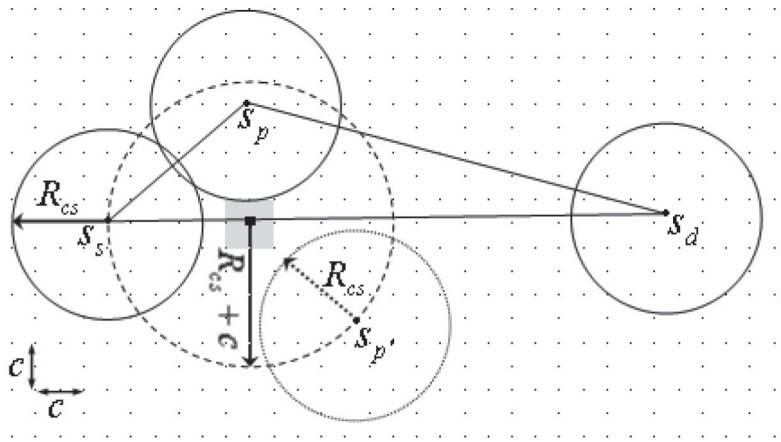


FIGURE 5
The sufficiency condition so that entire area coverage, provides the network connectivity is $R_t \geq 2R_{cs} + 2c$.

the circle is $R_{cs} + c$. $s_{p'}$ is located on the perimeter of the circle. As it is shown in Figure 5, the minimum distance between node $s_{p'}$ and any point within the cell centered on the center of the circle is more than R_{cs}. As a result, this cell

is not covered by any node. This is in contradiction with the assumption that the network is covered completely. Thus, at least one node like s_p must be located within the aforesaid circle. s_s and s_p are connected due to the fact that the distance between them is less than $2R_{cs} + 2c$ and $2R_{cs} + 2c \leq R_t$. Nodes s_p and s_d must be disconnected or otherwise s_s and s_d are connected via node s_p. Since $\angle s_s s_p s_d > \frac{\pi}{2} > \angle s_p s_s s_d$, thus length of $\overline{s_s s_d}$ is greater than length of $\overline{s_p s_d}$. This contradicts the primary assumption that nodes s_s and s_d have the minimum distance among all disconnected nodes. □

Considering Lemma 2, the constraint that we pose on the relationship between sensing and transmission radiuses of sensor nodes to ensure connectivity when network area is completely covered is $R_t \geq 2R_{cs} + 2c$.

6. PROPOSED DENSITY CONTROL ALGORITHM

GDC-CLA algorithm must provide two requirements; 1. complete coverage of the network area, and 2. network connectivity. As it is stated in Section 5, network connectivity is ensured by network coverage iff $R_t \geq 2R_{cs} + 2c$. Therefore, GDC-CLA only focuses on network coverage. We consider the following assumptions:

- The set of sensor nodes residing throughout the network area covers the area completely.
- Sensor nodes are aware of their physical locations (using some localization techniques [32–36])
- $R_t \geq 2R_{cs} + 2c$.
- N which is the number of sensor nodes within the area of the network is known by all nodes.

In what follows, we give the details of the GDC-CLA algorithm.

6.1 Detailed Description of GDC-CLA

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

$$R_n = 2R_{cs} - \varepsilon \cong 2R_{cs} \tag{7}$$

We refer to R_n as neighborhood radius hereafter.

The learning automaton in each cell i of CLA, referred to as LA_i uses linear learning algorithm given by equations (1) and (2). All learning automata are L_{R-P}. Each LA_i has two actions α_0 and α_1; α_0 is "plan to be in active mode", and α_1 is "plan to be in sleep mode". The probability of selecting each of these actions is initially computed according to equation (8).

$$p_0 = \frac{N_I}{N}$$
$$p_1 = 1 - p_0 \tag{8}$$

In this equation, N_I is a constant which is greater than the minimum number of active nodes required to cover the entire area of the network. p_0 is selected this way in order to have a suitable initial distribution (in terms of area coverage) of active nodes throughout the network area.

Neighboring nodes exchange HELLO packets with each other periodically to be aware of the states of each other. Details on how to compute the time interval between two consecutive transmissions of HELLO packets in each sensor node will be given in Subsection 6.1.1. The HELLO packet of a node contains its id and physical location. Using received HELLO packets, each sensor node s_i stores for any of its active neighboring nodes, the location and the time of the last received HELLO packet.

Sensor nodes are initially in "INITIAL" state. In this state, all nodes are in $CPU_S S_S C_S$ operation mode. Each node s_i awakes after a random duration which is selected uniformly from the range (0, τ_0). τ_0 is the maximum allowed time interval between two subsequent transmissions of HELLO packets. When a node s_i awakes, it uses its learning automaton to choose its state. If LA_i selects α_0 which is "plan to be in active mode", sensor node s_i changes its operation mode from $CPU_S S_S C_S$ to $CPU_A S_A C_A$ and its state from "INITIAL" to "WORK" and then, broadcasts a HELLO packet in its neighborhood. If LA_i selects α_1 which is "plan to be in sleep mode", s_i changes its operation mode to $CPU_A S_S C_A$ and its state to "WAIT_FOR_SLEEP", and then, waits to receive HELLO packets from its neighbors which are in "WORK" state. A node which is in "WORK" state (referred to as a working node) monitors its surrounding area for the presence of any target, informs the sink if any target is detected, and broadcasts HELLO packets to its neighboring nodes periodically.

The reinforcement signal is computed according to the following two cases:

- If s_i is in "WORK" state and its received HELLO packets within τ_0 duration guarantee that the monitored area of s_i is covered by its neighbors, then it penalizes its selected action. Otherwise, it rewards its selected action.

- If s_i is in "WAIT_FOR_SLEEP" state and its received HELLO packets within τ_0 duration, do not guarantee that the monitored area of s_i is covered by its neighbors, then it penalizes its selected action. There is an exception to this rule which will be discussed in Subsection 6.1.2. Otherwise, s_i rewards its selected action.

Each working sensor node guarantees not to change its state for a predetermined duration T starting from the time it dispatches its HELLO packet. As a result, if a node, which is in "WAIT_FOR_SLEEP" state, is ensured (through received HELLO packets) that its monitored area is covered completely by its active neighboring nodes within τ_0 duration, then it switches to "SLEEP" state. In "SLEEP" state, all nodes are in $CPU_S S_S C_S$ operation mode. Sleeping time is determined by considering the value of T and the receipt times of the last HELLO packets from active neighbors. Let t_i^R be the receipt time of the last HELLO packet from active neighbor s_i at node s_j. The sleeping time of the node s_j (T_j^{sleep}) is then computed according to the following equation:

$$T_j^{sleep} = \min_{s_i \in \text{active neighbors of } s_j} \left(T - (t - t_i^R)\right) \tag{9}$$

When the sleeping time of the node s_i is over, node s_i changes its operation mode to $CPU_A S_S C_A$ and its state to "WAIT_FOR_SLEEP".

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

- If node s_i is in "WAIT_FOR_SLEEP" state and action α_1 is penalized, next action selection time is immediately after receiving the environment response.
- If node s_i is in "WORK" state and action α_0 is penalized, next action selection time is at the dispatching time of the next HELLO packet. If the selected action of such a node is α_1, then s_i enters "WAIT_FOR_SLEEP" state, but due to the assurance of this node to its neighbors, s_i must stay in active mode and perform all of the tasks of a sensor node in "WORK" state, except for sending HELLO packets, until the end of the assurance time.
- If the selected action of node s_i is rewarded, then the next action selection time is when this action is penalized. In other words, the rewarded action is performed repeatedly until the time it receives penalty.

6.1.1 Time Interval for Transmission of HELLO Packets

As we stated before, sensor nodes which are in "WORK" state, periodically transmit HELLO packets within their neighborhood. Initially, the time inter-

val between two consecutive transmissions of HELLO packets for each sensor node s_i is set to $\tau_i = \tau_0$. Immediately after sending the kth ($k > 1$) HELLO packet, each sensor node s_i updates τ_i according to equation (10).

$$\tau_i = \begin{cases} \tau_0 & ;No_i = 0 \\ \dfrac{No_i}{N_I} \times \tau_0 & ;Otherwise \end{cases} \tag{10}$$

In the above equation, No_i is the number of working nodes from which s_i receives at least one HELLO packet within the time interval $[t - \tau_0, t]$. This way, the time interval between two consecutive transmissions of HELLO packets in each sensor node s_i is adaptively updated according to No_i. Since No_i is a measure of the traffic load in the neighborhood of s_i, equation (10) makes the time interval between two consecutive transmissions of HELLO packets in each sensor node adaptable to the local traffic load in the neighborhood of that node.

6.1.2 A Note on Energy Consumption of Nodes

In PEAS [4], a working sensor node remains active continuously until its failure or depletion of battery power. This property of PEAS algorithm causes the working nodes to die very soon. This problem, when considering high probability of node failure in the network, results in noticeable reduction of the network lifetime. To remedy this problem, in GDC-CLA whenever the remaining energy of a working node s_i falls below 50% of the initial energy, s_i reinitiates its learning automaton with action probability vector set to $[0, 1]^T$ and moves to "WAIT_FOR_INACTIVE" state. Node s_i remains in this state until the end of the assurance time to its neighbors. During this state, node s_i monitors its surrounding area for the presence of any target and informs the sink if any target is detected. At the end of the assurance time, node s_i changes its operation mode to $CPU_A S_S C_A$ and enters "WAIT_FOR_SLEEP" state.

The maximum waiting time for receiving the HELLO packets from the neighboring nodes will be changed when node s_i will return to the "WAIT_FOR_SLEEP" state in future. New value of the waiting time will be determined according to the following equation:

$$\tau' = T + Cn \times \tau_0 \tag{11}$$

where Cn is determined based on the penalty parameter b and is the least positive integer satisfying the following inequality:

$$(1-b)^{Cn} < Pr_0 \tag{12}$$

In this inequality, Pr_0 is a small predetermined probability. The reason for determining τ' this way is to increase the probability that neighboring nodes can replace node s_i. During the "WAIT_FOR_SLEEP" state, if the monitored area of sensor node s_i cannot be covered completely by the set of active nodes in its neighborhood, that is, neighboring nodes of s_i cannot be replaced by node s_i, then s_i resets its waiting time τ' to τ_0 again.

In Figure 6, transition diagram of operation mode/state of GDC-CLA algorithm in each sensor node has been shown. Figure 7 shows the pseudo code of the proposed algorithm.

6.2 Data Dispatching

Active nodes make data packets containing the information relevant to the sensed target and dispatch them to the base station whenever they sense a target. This packet contains information regarding the sensed target. Since such information is application specific, we do not give any details about it here.

7. ANALYTICAL STUDY

In this section, we analyze the operation of the proposed GDC-CLA algorithm and two similar existing algorithms, PEAS [4] and PECAS [5], from the two aspects of time and space costs. To do this, we first give a brief outline

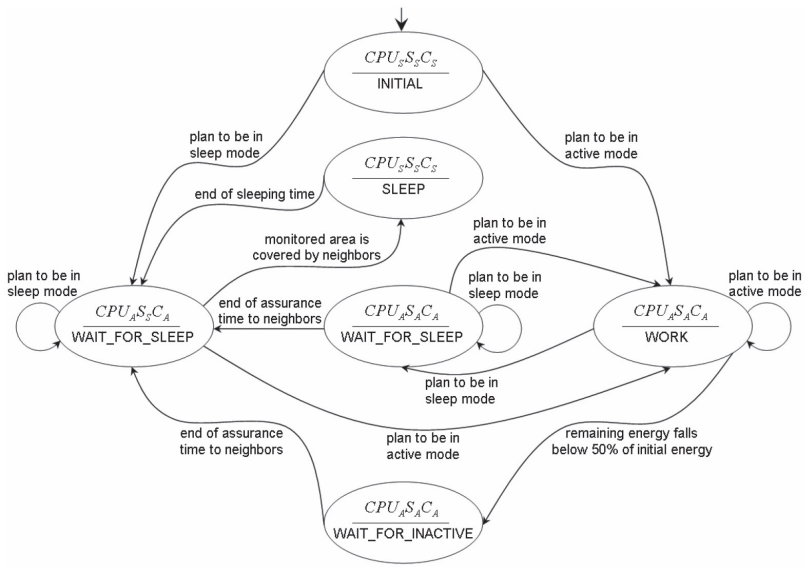


FIGURE 6
Operation mode/State transition diagram of GDC-CLA in each sensor node

```
For each node s_i corresponding with cell i in CLA do in parallel
  Initialize
  WakeupAfter(Random (0, τ_0))
  Select an action according to equation (8)
  If (the action is α_0 ) then
    ChangeOperation_Mode/StateTo(CPU_A S_A C_A / "WORK")
  else /* the action is α_1 */
    ChangeOperation_Mode/StateTo(CPU_A S_S C_A / "WAIT_FOR_SLEEP")
  End if
  While (Not SensorDead) do
    While (the node is in "WORK" State) do
      If (RemainingEnergy falls below 0.5×InitialEnergy) then
        Reinitiate LA with probability vector [0,1]^T
        ChangeOperation_Mode/StateTo(CPU_A S_A C_A / "WAIT_FOR_INACTIVE")
        Stay in "WAIT_FOR_INACTIVE" State until the end of assurance time to neighbors
        ChangeOperation_Mode/StateTo(CPU_A S_S C_A / "WAIT_FOR_SLEEP")
      else
        Broadcast a HELLO Packet
      End if
      Update LA's probability vector
    End while
    While (the node is in "WAIT_FOR_SLEEP" State) do
      If ((NOT AssuranceToNeighbors) AND (neighbors cover its monitored area)) then
        ChangeOperation_Mode/StateTo(CPU_S S_S C_S / "SLEEP")
      End if
      Update LA's probability vector
    End while
    If (the node is in "SLEEP" State) then
      WakeupAfter(SleepingTime)
      ChangeOperation_Mode/StateTo(CPU_A S_S C_A / "WAIT_FOR_SLEEP")
    End if
  End While
End for
```

FIGURE 7
Pseudo code of GDC-CLA

for each of these algorithms, to make the reader more familiar with the main procedure of these algorithms, and then present our analytical study.

7.1 Outline of the Algorithms

- **GDC-CLA:** In GDC-CLA algorithm, each node s_i acts as follows:
  - Receives the HELLO packets from its active neighbors.
  - Stores for any of its active neighbors, the location and the time of the last received HELLO packet.
  - Checks, according to the received HELLO packets, whether its monitored area is covered by its neighbors or not.
  - Updates the probability vector of its learning automaton according to equations (1) and (2) in Section 3.

- Uses its learning automaton to choose its state.
- Before going to the sleep mode, computes its own sleeping time according to equation (9) in Subsection 6.1.
- When s_i is a working node, it broadcasts HELLO packets to its neighboring nodes periodically and informs the sink if it detects any target in its monitored area.

- **PEAS and PECAS:** In PEAS and PECAS algorithms, each node s_i acts as follows:
  - Receives PROBE messages from its probing neighbor nodes.
  - Before going to the sleep mode, computes its own sleeping time. This computation differs in PEAS and PECAS algorithms.
  - When s_i is a probing node, it broadcasts PROBE messages within its probing range, and receives REPLY messages from its working neighbor nodes.
  - When s_i is a working node, it broadcasts a REPLY message for each PROBE message received from its probing neighbor nodes.

7.2 The Analysis

As we mentioned earlier, in this subsection we will analyze the operation of GDC-CLA, PEAS, and PECAS algorithms from the two aspects of time and space costs.

7.2.1 Time Cost Analysis

We consider the time cost of an algorithm from two points of view; computational complexity and learning time. The above mentioned outline having been considered, the computational complexity of these algorithms is not highly significant. Therefore, for time cost analysis, we only study the learning time.

Unlike PEAS and PECAS algorithms, which have no learning time at all, in GDC-CLA algorithm, each node s_i is equipped with a learning automaton which needs some learning time to converge to its suitable action. In what follows, we compute the learning time of GDC-CLA algorithm.

Proposition 1. Consider a node s_i which executes GDC-CLA algorithm. Let LA_i be the learning automaton residing in s_i with available actions α_l and α_m. Assume the suitable action of LA_i to be α_l. Longest learning time for LA_i will be occurred if and only if p_l is initially equal to 0 and LA_i selects its action α_m repeatedly.

***Proof*:** The proof is an immediate result of the fact that whenever α_l is being selected, the learning time is over. □

Lemma 3. Consider a node s_i which executes GDC-CLA algorithm. Let LA_i be the learning automaton residing in s_i with available actions α_l and α_m. Assume the suitable action of LA_i to be α_l. The longest learning time for LA_i having been assumed, the expected learning time of GDC-CLA algorithm in node s_i can be computed as $\frac{(1-a)}{a}\cdot\tau_0$ where a is the reward parameter and τ_0 is the maximum time interval between two subsequent transmissions of HELLO packets in GDC-CLA algorithm.

***Proof*:** Since the assumption of the lemma indicates that the longest learning time is occurred, we assume at time step $n = n'$, $p_l(n) = 0$ and $p_m(n) = 1$. According to the details of GDC-CLA algorithm, each node s_i at most waits for a duration of τ_0 (waiting time for receiving HELLO packets from the active neighbor nodes) before updating the probability vector of LA_i. Hence, the learning time of LA_i at each time step n ($\eta_i(n)$) is at most equal to τ_0. According to equation (1) in Section 3 and considering the facts that the longest learning time is occurred and the suitable action of LA_i is α_l, we can write:

$$\begin{aligned}
&p_m(n'+1) = (1-a)p_m(n') = 1-a \\
&p_m(n'+2) = (1-a)p_m(n'+1) = (1-a)^2 \\
&\qquad\vdots \\
&p_m(n'+n'') = (1-a)p_m(n'+n''-1) = (1-a)^{n''} = \varepsilon \\
&\qquad\vdots \\
&\lim_{n'''\to\infty} p_m(n'+n''+n''') = \lim_{n'''\to\infty} (1-a)^{n''+n'''} = 0
\end{aligned} \tag{13}$$

Based on the concept of the mathematical expectation, the expected learning time of LA_i ($\mathrm{E}[\eta_i]$) is computed according to the following equation:

$$\mathrm{E}[\eta_i] = \sum_{z=1}^{\infty}\left(\eta_i(n'+z)\times p_m(n'+z)\right) \tag{14}$$

To compute the above expected value when the longest learning time is occurred, we let $\eta_i(n)$ be equal to its maximum value, which is τ_0. Furthermore, we have to note that in infinity, $\eta_i(n) = 0$. Thus, equation (14) can be rewritten as:

$$\mathrm{E}[\eta_i] = \tau_0 \sum_{z=1}^{\infty}(1-a)^z = \frac{(1-a)}{a}\cdot\tau_0 \tag{15}$$

and hence the lemma. □

Lemma 4. Consider a node s_i which executes GDC-CLA algorithm. Let LA_i be the learning automaton residing in s_i with available actions α_l and α_m. Assume the suitable action of LA_i to be α_l. Furthermore, assume when $p_m < \varepsilon$, the learning process is stopped. The longest learning time for LA_i having been assumed, the maximum learning time of GDC-CLA algorithm in node s_i can be computed as $\frac{\log \varepsilon}{\log(1-a)} \cdot \tau_0$ where a is the reward parameter and τ_0 is the maximum time interval between two subsequent transmissions of HELLO packets in GDC-CLA algorithm.

***Proof*:** As shown in equation (13), at time step $n = n' + n''$, $p_m(n) = (1-a)^{n''} = \varepsilon$. Therefore, we have:

$$n'' = \frac{\log \varepsilon}{\log(1-a)} \tag{16}$$

Now, the maximum learning time of GDC-CLA algorithm in node s_i ($\max\left(\eta_i\right)$) can be computed as:

$$\max\left(\eta_i\right) = n''\tau_0 = \frac{\log \varepsilon}{\log(1-a)} \cdot \tau_0 \tag{17}$$

and hence the lemma. □

Considering the results of Lemma 3 and Lemma 4, now we show that the maximum and the expected values of the learning time of GDC-CLA algorithm are not highly significant if a, τ_0, and ε are selected appropriately. According to Table 1 (given in Subsection 8.1.1), $a = 0.75$, $\tau_0 = 1.27s$, and $\varepsilon = Pr_0 = 0.1$. Thus, the expected value of the learning time is equal to 0.42s and its maximum value is equal to 2.11s.

7.2.2 Space Cost Analysis

In this subsection, we first give some definitions and then propose two lemmas to indicate the space complexity of GDC-CLA, and PEAS and PECAS algorithms.

Definition 4. $N_{active}^{cnf}(t)$ is defined as the number of active nodes at time instant t resulted from the execution of GDC-CLA algorithm in a collision- and noise-free environment.

Definition 5. Number of active nodes at time instant t resulted from the execution of GDC-CLA algorithm in the realistic environment of a wireless sensor network, referred to as $N_{active}(t)$, can be defined as follows:

$$N_{active}(t) = N_{active}^{cnf}(t) + \delta(t)N \tag{18}$$

In the above equation, $0 < \delta(t) << 1$ is a small coefficient which depends on the traffic load of the network and specifies the effect of the noises and collisions in the network environment on the number of active nodes. Noises and collisions may prevent some HELLO packets from receiving by neighbors of an active node. Consequently, some of the nodes may temporarily operate in active mode while they are not needed, which results in increasing the average number of active nodes during the operation of the network. Clearly, in a fixed network area, when the number of sensor nodes (N) increases, the probability of collision increases as well, which in consequence affects the operation of possibly more sensor nodes.

Definition 6. Maximum number of active nodes resulted from the execution of GDC-CLA algorithm during the operation of the network, referred to as $N_{active}^{\max}$, is defined as equation (19).

$$N_{active}^{\max} = \max_{t}\left(N_{active}(t)\right) \tag{19}$$

Lemma 5. Let the network area (Ω) be of the size $X \times Y$. Assume that N sensor nodes are scattered uniformly at random throughout Ω. The space complexity of GDC-CLA algorithm is $O\left(\frac{\pi R_n^2}{X \times Y} \cdot N_{active}^{\max}\right)$ where R_n is the neighborhood radius of each node s_i.

***Proof*:** In GDC-CLA algorithm, each node s_i only stores the information of its active neighbors. During the execution of GDC-CLA algorithm, the maximum number of active nodes in the network is equal to $N_{active}^{\max}$. By the assumption of the lemma, sensor nodes are scattered uniformly at random throughout the environment of the network. Thus, the detailed description of GDC-CLA algorithm having been considered, it is straightforward that $N_{active}^{\max}$ active nodes are also distributed uniformly throughout Ω. This means that the number of active nodes in the neighborhood radius of any sensor node is at most equal to $\frac{\pi R_n^2}{X \times Y} \cdot N_{active}^{\max}$. Therefore, the space complexity of GDC-CLA algorithm is $O\left(\frac{\pi R_n^2}{X \times Y} \cdot N_{active}^{\max}\right)$, and hence the lemma. □

Lemma 6. Assume the processing time for a single packet is considerably lower than the receiving time of that packet. Then, the space complexity of PEAS and PECAS algorithms is $O(1)$.

***Proof*:** In PEAS and PECAS algorithms, a node s_i can process each received packet individually without any need to have the information of other received packets. This indicates that node s_i does not need to store the received packets' information for performing its operation. By the assumption of the lemma

(the processing time for a single packet is considerably lower than the receiving time of that packet), no new packet can be received before the processing of the previous packet is completed. Thus, node s_i does not need to queue the received packets. Therefore, the space complexity of PEAS and PECAS algorithms is $O(1)$, and hence the lemma. □

8. SIMULATION

8.1 Simulation Setup

To evaluate the performance of the proposed GDC-CLA algorithm, we conduct a number of experiments. The results obtained from the proposed algorithm are compared with the results obtained from PEAS [4] and PECAS [5]. Experiments are simulated using J-Sim simulator [37]. We use a network area of $50 \times 50\,m^2$ through which a number of sensor nodes are scattered uniformly at random. Base station (sink) is in the center of the area. We consider one moving target tr_1 being exist in the network area, whose speed is at most *MS*.

Energy model given in [9] is used in which the energy consumption ratios for transmission, reception (idle) and sleep modes are 20:4:0.01 respectively. The initial energy level of each sensor node is chosen randomly from the range of $28 \sim 35$ Jules. Sensing range of each sensor node is 17m. We set the side length of square cell to 5 m; therefore, computational sensing radius according to equation (3) and neighborhood radius according to equation (7) are 10m and 20m respectively. According to Lemma 2, the radio transmission range must be at least 30m. Like PEAS [4], each node has a raw wireless communication capacity of 20Kbps. Simulation time for the first two experiments is assumed to be 1000s. Results are averaged over 25 runs of simulations.

8.1.1 Parameters of GDC-CLA Algorithm

We consider the value of parameter *T* as 10s and rate of reward and penalty parameters as 0.75 so that the sensor nodes have rapid reactions to topology changes of the network (due to unexpected failures, energy exhaustion of sensor nodes, ...). Penalty parameter $b = 0.75$ having been considered, parameter *Cn* will be 2 according to inequality (12). The values for the parameters in GDC-CLA are listed in Table 1.

8.1.2 Evaluation Metrics

To evaluate the performance of GDC-CLA, we use the following metrics: (i) number of active nodes; (ii) percentage of area under the coverage of the network; (iii) total consumed energy of sensor nodes; (iv) network lifetime; and (v) number of sensor nodes, which are alive (the nodes which have not been dead yet due to the depletion of battery power or failure) during the network lifetime referred to as N_{alive}.

TABLE 1
Parameter values used in GDC-CLA

R_s	c	R_t	N_I	Initial energy of each sensor node	Channel capacity	T	τ_0	a,b	Pr_0	MS
17m	5m	30m	25	28 ~ 35 Jules	20Kbps	10s	1.27s	0.75	0.1	2m/s

Another metric, which can be considered for performance evaluation, is the *percentage of delivery of sensing target packets to sink*. According to the simulation results, this metric in PEAS, PECAS, and GDC-CLA algorithms is almost 100%, and thus, we do not report it in this section.

8.2 Simulation Results

8.2.1 Experiment 1

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

Figure 8 shows the results of the comparison for the 'number of active nodes' metric. It can be seen from this figure that the number of active nodes in GDC-CLA algorithm, unlike PEAS and PECAS algorithms, does not depend heavily on the number of sensor nodes in the network.

Figure 9 presents the results of the comparison between GDC-CLA, PEAS, and PECAS in terms of percentage of network area under coverage. As it can be seen from the figure, GDC-CLA fully covers the network area even in the networks of small sizes (networks in which the number of sensor nodes is below 100). Since the conditions of Lemma 2 are met, this indicates that GDC-CLA guarantees the network connectivity using its set of active nodes.

Figure 10 compares GDC-CLA with PEAS and PECAS algorithms in terms of the total energy consumption of sensor nodes. It can be seen from this figure that GDC-CLA consumes less than 70% and 45% of the energy consumed by PEAS and PECAS, respectively. The reason for this is that in GDC-CLA, the number of active sensor nodes is significantly lower than that of PEAS and PECAS algorithms.

8.2.2 Experiment 2

In this experiment, we study the tolerability of GDC-CLA algorithm against unexpected failures of sensor nodes in comparison to PEAS and PECAS algorithms. To simulate unexpected failures of sensor nodes, in this experiment, 50% of the sensor nodes experience failures in random times during the

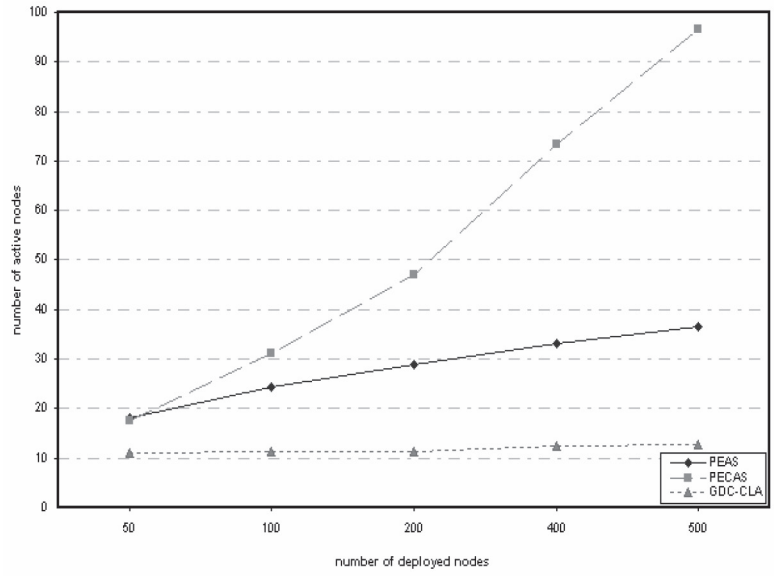


FIGURE 8
Number of active nodes versus number of sensor nodes

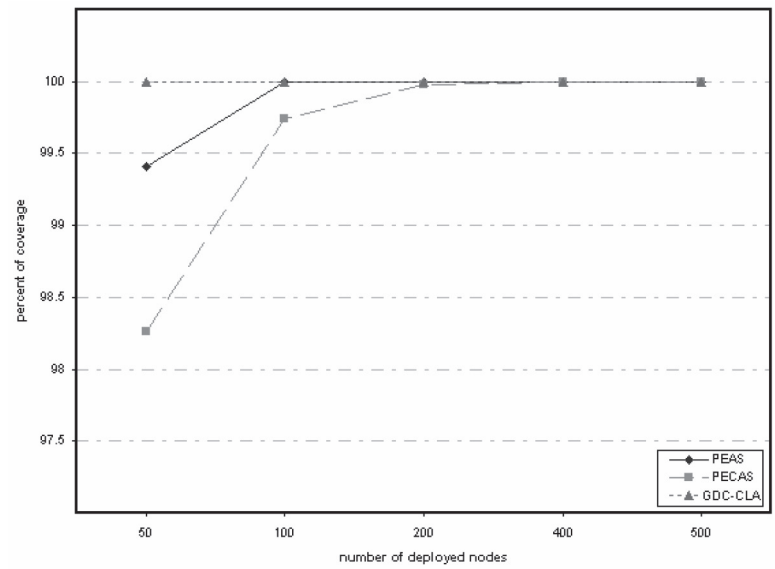


FIGURE 9
Percent of coverage versus number of sensor nodes

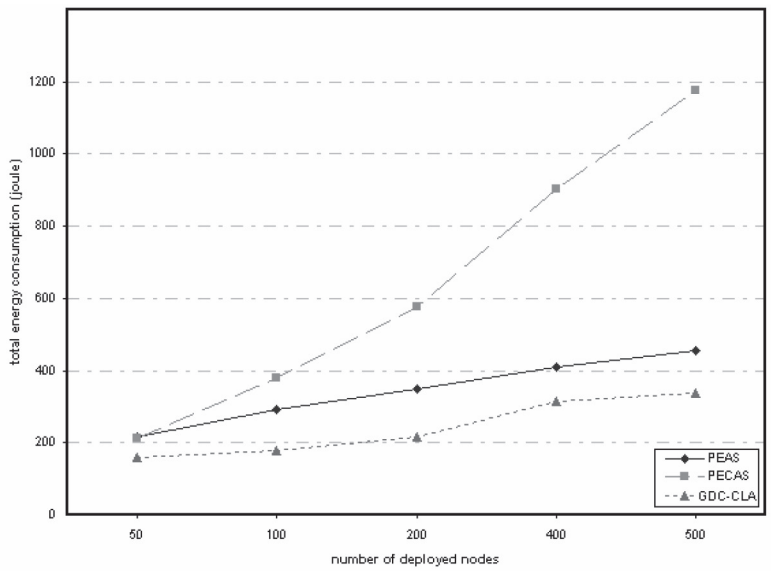


FIGURE 10
Total energy consumption versus number of sensor nodes

operation of the network. Similar to Experiment 1, networks with 50, 100, 200, 400, and 500 sensor nodes are considered for this experiment.

Figure 11, which compares GDC-CLA with PEAS and PECAS in terms of average number of active nodes, indicates that the presence of failures does not affect the performance of GDC-CLA algorithm in terms of this metric significantly. Number of active nodes in GDC-CLA algorithm is less than 49% and 40% of the number of active nodes in PEAS and PECAS algorithms respectively.

According to Figure 12, which gives the comparison of the mentioned algorithms in terms of percentage of network area under coverage, GDC-CLA provides full coverage of network area in all of mentioned network sizes even in presence of unexpected node failures.

Finally, Figure 13 gives the comparison of the mentioned algorithms in terms of the total energy consumption of sensor nodes. This figure shows that in the presence of unexpected failures, the energy consumption of GDC-CLA is less than 69% and 54% of the energy consumption of PEAS and PECAS, respectively. Similar to Experiment 1, this is due to the fact that in GDC-CLA, the number of active nodes is significantly lower than that of PEAS and PECAS algorithms.

8.2.3 Experiment 3

This experiment is conducted to study the performance of GDC-CLA algorithm in terms of network lifetime. For this experiment, number of sensor

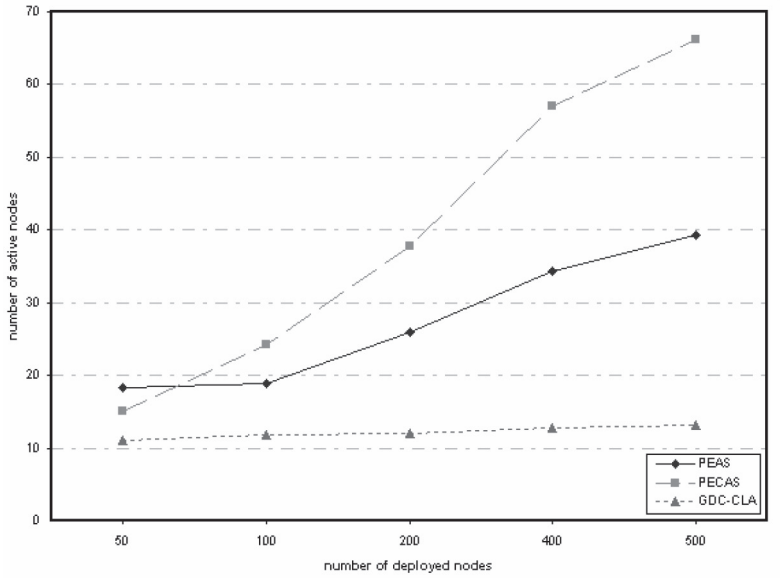


FIGURE 11
Number of active nodes versus number of sensor nodes with failures

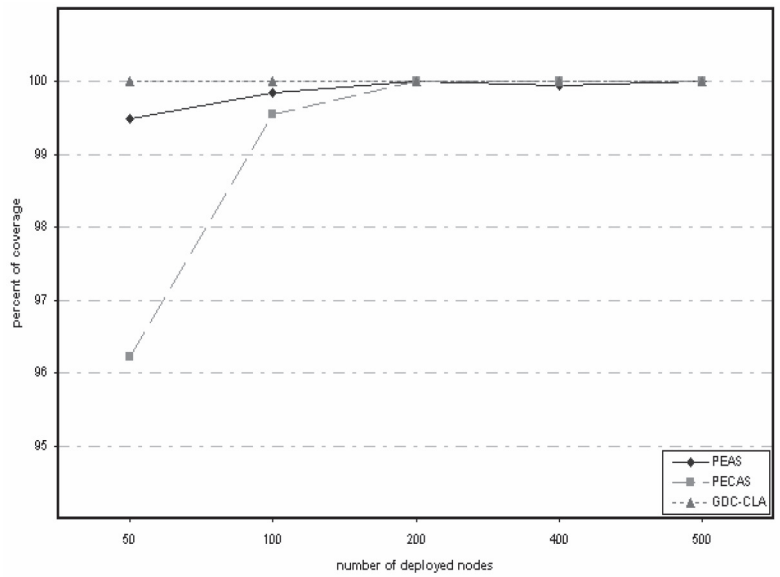


FIGURE 12
Percent of coverage versus number of sensor nodes with failures

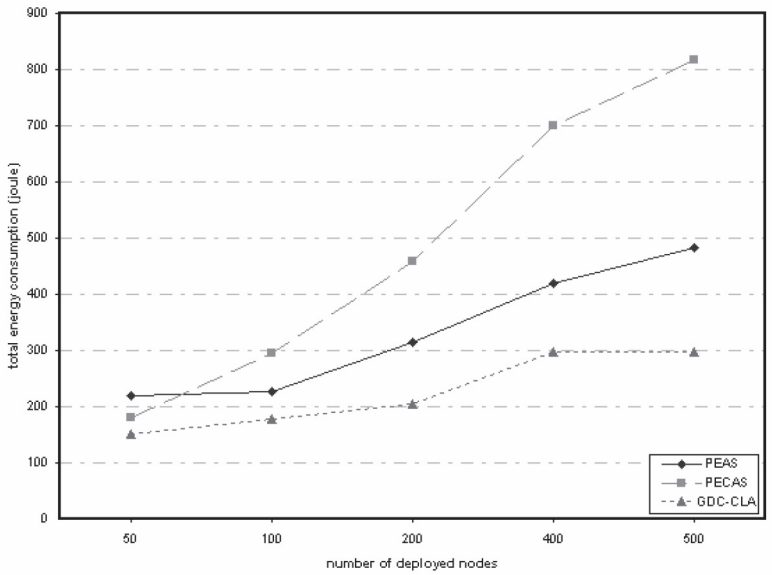


FIGURE 13
Total energy consumption versus number of sensor nodes with failures

nodes is assumed to be 250. The injection of unexpected failures of sensor nodes is performed with a rate of 4.17 failures in every 1000s. The results of simulations show that the network lifetime for PEAS, PECAS and GDC-CLA is 9000s, 12000s and 22000s, respectively. This indicates that the proposed algorithm better prolongs the lifetime of the network in comparison to existing algorithms. The results given in previous experiments having been considered, the main reason for this is that in the proposed algorithm, the number of active sensor nodes is significantly lower than that of PEAS and PECAS algorithms.

8.2.4 Experiment 4

In this experiment, we study the performance of GDC-CLA algorithm in comparison to PEAS and PECAS algorithms in terms of the number of sensor nodes, which are alive during the network lifetime. For this experiment, number of sensor nodes is assumed to be 250. The injection of unexpected failures of sensor nodes is performed with a rate of 4.17 failures in every 1000s.

Figure 14 shows N_{alive} during the network lifetime for different algorithms. As it can be seen from this figure, in GDC-CLA, each sensor node is alive for longer duration on average compared with other existing algorithms.

Note that at any time during the operation of the network, number of sensor nodes (N) is equal to the sum of alive and dead nodes. Sensor nodes' deaths can be due to battery energy depletion or unexpected failure. As the injection rate of failures is the same in all three algorithms, the main reason

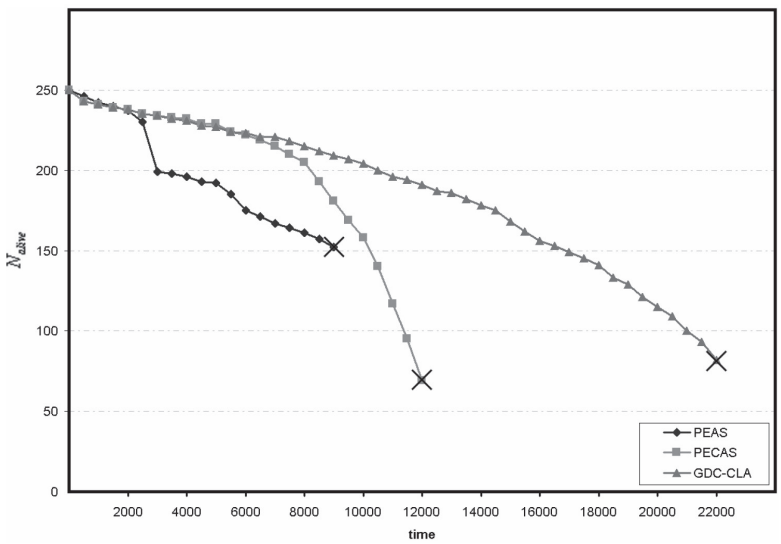


FIGURE 14
Number of sensor nodes, which are alive during the lifetime of the network; cross signs show the end of network lifetime for each algorithm.

for the difference between N_{alive} in different algorithms is due to the different number of nodes with depleted energy. For instance, the number of nodes with depleted energy at the end of the network lifetime of PEAS algorithm (9000th second of simulation time) is 57, 37, and 9 for PEAS, PECAS, and GDC-CLA respectively.

9. CONCLUSION

In this paper, a novel approach based on cellular learning automata for solving the coverage and connectivity problem in a wireless sensor network with minimum number of active nodes was presented. This approach tries to select a minimum number of active nodes, which can fully cover the entire area of the network while they are connected, so that the energy consumption of sensor nodes on average is minimized, and consequently the network lifetime is maximized. At first, the radio transmission range of sensor nodes was analytically determined according to their sensing range in such a way that complete coverage of the network area guarantees the connectivity of active nodes. As a consequence of this determination, the problem of maintaining coverage and connectivity of the network was reduced to the simpler problem of maintaining only the coverage of the network. Then, GDC-CLA algorithm was presented which tries to select a set of active nodes among all of the sensor nodes in the network so that the selected nodes can fully cover the entire

area of the network. To have a better understanding of the complexity of the proposed GDC-CLA algorithm, we analytically determined its time and space costs and compared them with those of similar existing algorithms such as PEAS and PECAS.

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

REFERENCES

[1] C. F. Huang, Y. C. Tseng, "The Coverage Problem in a Wireless Sensor Network", *Mobile Networks and Applications*, Vol. 10, No. 4, pp. 519–528, January 2005.

[2] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. B. Srivastava, "Coverage Problems in Wireless Ad-Hoc Sensor Networks", *In Proc. of 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, Vol. 3, pp. 1380–1387, Anchorage, AK, USA, April 2001.

[3] M. Ilyas, I. Mahgoub, "Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems", *CRC Press*, London, Washington, D.C., 2005.

[4] F. Ye, G. Zhong, S. Lu, L. Zhang, "PEAS: A Robust Energy Conserving Protocol for Long-Lived Sensor Networks", *In the 23rd International Conference on Distributed Computing Systems (ICDCS)*, 2003.

[5] Ch. Gui, P. Mohapatra, "Power Conservation and Quality of Surveillance in Target Tracking Sensor Networks", *In Proc. of the 10th Annual Intl. Conf. on Mobile Computing and Networking (MOBICOM 2004)*, Philadelphia, PA, USA, September–October 2004.

[6] F. Ye, G. Zhong, S. Lu, L. Zhang, "Energy Efficient Robust Sensing Coverage in Large Sensor Networks", *Technical report*, UCLA, 2002.

[7] A. Cerpa, D. Estrin, "ASCENT: Adaptive Self-Configuring Sensor Networks Topologies", *IEEE Transactions on Mobile Computing*, Vol. 3, No. 3, pp. 272–285, July 2004.

[8] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, C. Gill, "Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks", *In the First ACM Conference on Embedded Networked Sensor Systems*, November 2003.

[9] H. Zhang, J. C. Hou, "Maintaining Sensing Coverage and Connectivity in Large Sensor Networks", *Ad Hoc & Sensor Wireless Networks*, Vol. 1, pp. 89–124, March 2005.

[10] S. Slijepcevic, M. Potkonjak, "Power Efficient Organization of Wireless Sensor Networks", *In Proc. of IEEE International Conference on Communications (ICC 2001)*, pp. 472–476, Helsinki, Finland, June 2001.

[11] F. P. Quintao, F. G. Nakamura, G. R. Mateus, "A Hybrid Approach to Solve the Coverage and Connectivity Problem in Wireless Sensor Networks", *University of Minas Gerais Belo Horizonte*, MG, Brazil, 2005.

[12] A. Gallais, J. Carle, D. Simplot-Ryl, I. Stojmenovic, "Localized Sensor Area Coverage with Low Communication Overhead", *In Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2006)*, pp. 328–337, Pisa, Italy, March 2006; *IEEE Transactions on Mobile Computing*, Vol. 7, No. 5, pp. 661–672, May 2008.

[13] E. Fredkin, "Digital Machine: A Informational Process based on Reversible Cellular Automata", *Physica D45*, pp. 245–270, 1990.
[14] M. Mitchell, "Computation in Cellular Automata: A Selected Review", *Technical report,* Santa Fe Institute, Santa Fe, NM, USA, September 1996.
[15] N. H. Packard, S. Wolfram, "Two Dimensional Cellular Automata", *Journal of Stat. Phys. 38*, pp. 901–946, 1985.
[16] J. Kari, "Reversibility of 2d Cellular Automata is Undecidable", *Physica D45,* pp. 379–385, 1990.
[17] M. A. L. Thathachar, P. S. Sastry, "Varieties of Learning Automata: An Overview", *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics,* Vol. 32, No. 6, pp. 711–722, 2002.
[18] K. S. Narendra, M. A. L. Thathachar, "Learning Automata: An Introduction", *Prentice Hall*, 1989.
[19] M. R. Kharazmi, M. R. Meybodi, "Image Segmentation Using Cellular Learning Automata", *In Proc. of 10th Iranian Conf. on Electrical Engineering (ICEE-96)*, Tabriz, Iran, May 2001.
[20] M. R. Meybodi, M. R. Kharazmi, "Image Restoration Using Cellular Learning Automata", *In Proceedings of the Second Iranian Conference on Machine Vision, Image Processing and Applications*, pp. 261–270, Tehran, Iran, 2003.
[21] M. R. Meybodi, H. Beigy, M. Taherkhani, "Cellular Learning Automata and Its Applications", *Journal of Science and Technology*, No. 25, pp. 54–77, University of Sharif, Tehran, Iran, 2004.
[22] M. R. Meybodi, M. Taherkhani, "Application of Cellular Learning Automata to Modeling of Rumor Diffusion", *In Proc. of 9th Conf. on Electrical Engineering, Power and Water Institute of Technology*, pp. 102–110, Tehran, Iran, May 2001.
[23] M. R. Meybodi, M. R. Khojaste, "Application of Cellular Learning Automata in Modeling of Commerce Networks", *In Proc. of 6th Annual Intl. Computer Society of Iran Computer Conference (CSICC2001)*, pp. 284–295, Isfahan, Iran, February 2001.
[24] H. Beigy, M. R. Meybodi, "A Self-Organizing Channel Assignment Algorithm: A Cellular Learning Automata Approach", *Springer-Verlag Lecture Notes in Computer Science*, Vol. 2690, pp. 119–126, 2003.
[25] M. Esnaashari, M. R. Meybodi, "A Cellular Learning Automata based Clustering Algorithm for Wireless Sensor Networks", *Sensor Letters,* Vol. 6, No. 5, pp. 723–735, December 2008.
[26] M. Esnaashari, M. R. Meybodi, "Dynamic Point Coverage Problem in Wireless Sensor Networks: A Cellular Learning Automata Approach", *Journal of Ad Hoc & Sensor Wireless Networks*, Vol. 10, No. 2–3, pp. 193–234, February 2010.
[27] M. Esnaashari, M. R. Meybodi, "A Cellular Learning Automata-based Deployment Strategy for Mobile Wireless Sensor Networks", *Journal of Parallel and Distributed Computing*, Vol. 71, No. 7, pp. 988–1001, July 2011.
[28] H. Beigy, M. R. Meybodi, "Cellular Learning Automata with Multiple Learning Automata in Each Cell and Its Applications", *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 40, No. 1, pp. 54–66, February 2010.
[29] L. Wang, Y. Ziao, "A Survey of Energy-Efficient Scheduling Mechanisms in Sensor Networks", *Mobile Networks and Applications*, Vol. 11, No. 5, pp. 723–740, 2006.
[30] "J-Sim Sensor Network Framework", http://www.J-Sim.org/v1.3/sensor/JSim.pdf
[31] K. Chakrabarty, S. S. Iyengar, H. Qi, E. Cho, "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks", *IEEE Transactions on Computers*, Vol. 51, pp. 1448–1453, 2002.
[32] C. Savarese, J. Rabaey, K. Langendoen, "Robust Positioning Algorithms for Distributed Ad Hoc Wireless Sensor Network", *In the Proceedings of the USENIX Technical Annual Conference*, pp. 317–327, Monterey, California, June 2002.
[33] T. He, C. Huang, B. M. Blum, J. A. Stankovic, T. Abdelzaher, "Range-Free Localization Schemes for Large Scale Sensor Networks", *In Proceedings of the 9th Annual Interna-*

tional Conference on Mobile Computing and Networking (MobiCom 2003), pp. 81–95, San Diego, California, September 2003.

[34] L. Doherty, L. E .Ghaoui, K. S. J. Pister, "Convex Position Estimation in Wireless Sensor Networks", *In Proc. of 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, Vol. 3, pp. 1655–1663, Anchorage, AK, USA, April 2001.

[35] J-P. Sheu, P-C. Chen, C-S. Hsu, "A Distributed Localization Scheme for Wireless Sensor Networks with Improved Grid-Scan and Vector-based Refinement", *IEEE Transactions on Mobile Computing*, Vol. 7, No. 9, pp. 1110–1123, September 2008.

[36] J. Nie, "Sum of Squares Method for Sensor Network Localization", *Computational Optimization and Applications*, Vol. 43, No. 2, pp. 151–179, 2009.

[37] "J-Sim, Java Based Network Simulator", http://www.J-Sim.org