A Cellular Learning Automata-based Deployment Strategy for Mobile Wireless Sensor Networks

M. Esnaashari[1], M. R. Meybodi[2]

[1] *Soft Computing Laboratory, Computer Engineering and Information Technology Department*
Amirkabir University of Technology, Tehran, Iran
Esnaashari@aut.ac.ir

[2] *Soft Computing Laboratory, Computer Engineering and Information Technology Department*
Amirkabir University of Technology, Tehran, Iran
mmeybodi@aut.ac.ir

Abstract: One important problem which may arise in designing a deployment strategy for a wireless sensor network is how to deploy a specific number of sensor nodes throughout an unknown network area so that the covered section of the area is maximized. In a mobile sensor network, this problem can be addressed by first deploying sensor nodes randomly in some initial positions within the area of the network, and then letting sensor nodes to move around and find their best positions according to the positions of their neighboring nodes. The problem becomes more complicated if sensor nodes have no information about their positions or even their relative distances to each other. In this paper, we propose a cellular learning automata-based deployment strategy which guides the movements of sensor nodes within the area of the network without any sensor to know its position or its relative distance to other sensors. In the proposed algorithm, the learning automaton in each node in cooperation with the learning automata in the neighboring nodes controls the movements of the node in order to attain high coverage. Experimental results have shown that in noise free environments, the proposed algorithm can compete with the existing algorithms such as PF, DSSA, IDCA, and VEC in terms of network coverage. It has also been shown that in noisy environments, where utilized location estimation techniques such as GPS-based devices and localization algorithms experience inaccuracies in their measurements, or the movements of sensor nodes are not perfect and follow a probabilistic motion model, the proposed algorithm outperforms the existing algorithms in terms of network coverage.

Keywords: mobile sensor network, cellular learning automata, self-regulated deployment

1. Introduction

Sensor deployment is a fundamental issue for wireless sensor networks [1]. The main objective of a sensor deployment strategy is to achieve desirable coverage of the network area. In general, sensor deployment strategies can be classified into the following four categories [2][3]:

- Predetermined: This strategy is useful only if the network area is completely known [2][4][5][6][7][8].
- Randomly undetermined: In this strategy, sensor nodes are spread uniformly throughout the network area [3][9][10][11][12][13].
- Biased distribution: In some contexts, the uniform deployment of sensor nodes may not always satisfy the design requirements and biased deployment can then be a viable option [14].
- Self-regulated: In this strategy which is useful only in mobile sensor networks, sensor nodes are deployed randomly in some initial positions within the area of the network. After this initial placement, sensor nodes move around and find their best positions according to the positions of their neighboring nodes [15][16][17][18][19][20].

One important problem which may arise in designing a deployment strategy for a wireless sensor network is how to deploy a specific number of sensor nodes so that the covered section of the area is maximized. Since this is an optimization problem, random based deployment strategies (Randomly

undetermined and biased distribution) are not suitable approaches for solving it. Predetermined deployment strategies which are commonly centralized are not well suited to this problem as well. This is because a counterpart of this problem in computational geometry domain, which is to maximize the guarded interior of an art gallery by a specific number of vertex guards, is proved to be APX-hard [21]. This indicates that finding an approximation close enough to the optimal solution of the problem is impossible in polynomial time. Self-regulated deployment strategies on the other hand are well suited to this problem.

To the best of our knowledge, all of the self-regulated deployment strategies given in literature require for every node to be aware of either its geographical position within the area of the network or its relative distance to all of its neighbors. If none of this information is available to the nodes of a network, the problem becomes more complicated. In this paper, we propose a self-regulated deployment strategy based on cellular learning automata called CLA-DS which guides the movements of sensor nodes within the area of the network without any sensor to know its position or its relative distance to other sensors. In the deployment strategy proposed in this paper, neighboring nodes apply forces to each other which make every node move according to the resultant force vector applied to it. Each node is equipped with a learning automaton. The Learning automaton of a node at any given time decides for the node whether to apply force to its neighbors or not. This way, each node in cooperation with its neighboring nodes gradually learns its best position within the area of the network so as to attain high coverage. Experimental results show that in terms of network coverage, the proposed deployment algorithm can compete with the existing deployment algorithms such as PF, DSSA, IDCA, and VEC in noise free environments, and outperforms the existing deployment algorithms in noisy environments, where utilized location estimation techniques such as GPS-based devices and localization algorithms experience inaccuracies in their measurements, or the movements of sensor nodes are noisy.

The rest of this paper is organized as follows. Section 2, gives a brief literature overview on deployment problem in mobile wireless sensor networks. Cellular learning automata will be discussed in section 3. The problem statement is given in section 4. In section 5 the proposed deployment algorithm is presented. Simulation results are given in section 6. Section 7 is the conclusion.

2. Related Work

Mobile sensor nodes can be used to improve the coverage of wireless sensor networks in different ways [38]. In a hybrid network consisting of both stationary and mobile sensor nodes, they can be used mainly to heal the coverage holes caused by the stationary nodes. In a mobile network consisting of only mobile nodes, they can be used to maximize the coverage of the area. And in event monitoring scenarios where some short-lived events may appear in different locations, mobile nodes can be dispatched to monitor the event sources for better event coverage.

Voronoi diagram is a major tool used to detect coverage holes in hybrid networks [39][40]. A Voronoi diagram is first constructed for all stationary sensor nodes, assuming that each node knows its own and its neighbors' positions. In the constructed diagram, if some points of a Voronoi cell are not covered by the cell's generating sensor node, these points will contribute to coverage holes. When a coverage hole detected, a stationary node can bid mobile nodes to heal its coverage hole.

In mobile sensor networks, nodes can adjust their positions after initial deployment in order to reduce their overlaps and maximize area coverage. Wang et. al. in [42] groups the existing movement schemes for mobile sensor networks into the following three categories: coverage pattern based movement, virtual force based movement, and grid quorum based movement. Methods in the coverage pattern based movement group [41][42][43] try to relocate mobile nodes based on a predefined coverage pattern. The most commonly used coverage pattern is the regular hexagon with the sensing range R_s as its side length. In [41], initially one node is selected as a seed. Seed node computes six locations surrounding itself so as to form a regular hexagon and greedily selects its nearest neighbors to each of the selected locations. Selected neighbors then move to the selected locations and become new seeds. Another hexagonal coverage pattern is given in [42] which can completely cover the sensor field. According to this coverage pattern, final locations of mobile nodes are specified. Sensor movement problem then converted into a maximum-weight maximum-matching problem and centrally be solved. This approach is extended in [43] to provide *k*-coverage of the environment.

In the group of virtual force based node movement methods [15][16][17][18][19][20], sensor nodes apply some kinds of virtual forces to their neighboring nodes. Resultant force vector applied to each sensor node from its neighboring nodes specifies the direction and distance that the node should move to. In [15], a distributed potential-field-based approach is presented in which each node is repelled by other neighboring nodes. In addition to the repulsive forces, nodes are also subject to a viscous friction force. This force is used to ensure that the network will eventually reach static equilibrium; i.e., to ensure that all nodes will ultimately come to a complete stop. In [16][18], a node may apply attractive or repulsive forces to its neighboring nodes according to its distance with them. Nearby neighbors are repelled and far away ones are attracted. In order to reduce the movement of sensor nodes, the algorithm is virtually executed on cluster heads and then, sensor nodes move directly to the locations specified for them by the result of the algorithm. Like the algorithm given in [15], the deployment algorithm proposed in [17] is also based on virtual potential fields, but it considers the constraint that in the deployed network, each of the nodes has at least K neighbors. Heo and Varshney in [19] proposed three different distributed deployment algorithms called DSSA, IDCA, and VDDA. DSSA uses virtual repulsive forces between neighboring nodes. IDCA modifies DSSA by filtering out nodes, for which local density is very near to the desired density, from moving. VDDA is a distributed deployment approach based on the Voronoi diagram. Wang et. al. in [20] give two sets of distributed protocols for controlling the movement of sensors, one favoring communication and one favoring movement. In each set of protocols, Voronoi diagrams are used to detect coverage holes.

In the group of grid quorum based movement methods [44][45][46][47][48][49][50], the area of the network is divided into a number of grid cells and each mobile sensor node must find a suitable cell as its final location and move to that cell.

In an event detection and monitoring sensor network, sensing tasks are performed in an on-demand manner. At first, sensors are used to detect events. After detecting an event, it is often desirable to relocate more mobile sensors close to the event location for performing the sensing task. Wang et. al. in [51] consider a hybrid network in which stationary nodes detect events. When a new event is detected, one mobile node is assigned to that event for more advanced sensing and analysis tasks. In [52] Butler and Rus proposed two moving strategies for the event coverage problem in a mobile sensor network; history-free strategy and history-based strategy. Either of these strategies moves sensors closer to the event locations. Using a Voronoi diagram, a mobile node stops moving if it detects that its movement causes coverage holes.

3. Dynamic Irregular Cellular Learning Automata (*DICLA*)

In this section we briefly review learning automata, cellular learning automata, irregular cellular learning automata, and then introduce dynamic irregular cellular learning automata.

3-1. Learning Automata

Learning Automata (*LA*) are adaptive decision-making devices that operate on unknown random environments. A learning Automaton 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 automaton, the environment gives a reinforcement signal with fixed unknown probability distribution. The automaton then updates its action probability vector depending upon the reinforcement signal at that stage, and evolves to some final desired behavior. A class of learning automata is called variable structure learning automata and are represented by quadruple $\{\alpha, \beta, p, T\}$ in which $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ represents the action set of the automata, $\beta = \{\beta_1, \beta_2, \cdots, \beta_r\}$ represents the input set, $p = \{p_1, p_2, \cdots, p_r\}$ 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 *n*, then the recurrence equation for updating *p* is defined as

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

for favorable responses, and

$$p_i(n+1) = (1-b).p_i(n)$$
$$p_j(n+1) = \frac{b}{r-1} + (1-b)p_j(n) \qquad \forall j \;\; j \neq i \qquad (2)$$

for unfavorable ones. In these equations, *a* and *b* are reward and penalty parameters respectively. For more information about learning automata the reader may refer to [22][23][24].

3-2. Cellular Learning Automata

Cellular learning automata (*CLA*), which is a combination of cellular automata (*CA*) [25] and learning automata (*LA*), is a powerful mathematical model for many decentralized problems and phenomena. The basic idea of *CLA* is to utilize learning automata to adjust the state transition of *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 *LA*s of any particular *LA* determine the reinforcement signal to the *LA* residing in a cell. The neighboring *LA*s of any particular *LA* constitute the local environment of that cell. The local environment of a cell is non-stationary because the action probability vectors of the neighboring *LA*s vary during evolution of the *CLA*. A *CLA* is called synchronous if all *LA*s are activated at the same time in parallel. *CLA* has found many applications such as image processing [27], rumor diffusion [29], channel assignment in cellular networks [30] and VLSI placement [31], to mention a few. For more information about *CLA* the reader may refer to [26][28][32][33].

3-3. Irregular Cellular Learning Automata

An Irregular cellular learning automata (*ICLA*) is a cellular learning automata (*CLA*) in which the restriction of rectangular grid structure in traditional *CLA* is removed. This generalization is expected because there are applications such as wireless sensor networks, immune network systems, graph related applications, etc. that cannot be adequately modeled with rectangular grids. An *ICLA* is defined as an undirected graph in which, each vertex represents a cell which is equipped with a learning automaton. Despite its irregular structure, *ICLA* operation is equivalent to that of *CLA*. *CLA* has found a number of applications in wireless sensor networks [34][35][36].

3-4. Dynamic Irregular Cellular Learning Automata

We define Dynamic *ICLA* (*DICLA*) as an undirected graph in which, each vertex represents a cell and a learning automaton is assigned to every cell (vertex). A finite set of interests is defined for *DICLA*. For each cell of *DICLA* a tendency vector is defined whose *j*th element shows the degree of tendency of the cell to the *j*th interest. In *DICLA*, the state of each cell consists of two parts; the action selected by the learning automaton and the tendency vector. Two cells are neighbors in *DICLA* if the distance between their tendency vectors is smaller than or equal to the neighborhood radius.

Like *CLA*, there is a local rule that *DICLA* operates under. The rule of *DICLA* and the actions selected by the neighboring learning automata of any particular learning automaton LA_i determine the followings: 1. the reinforcement signal to the learning automaton LA_i, and 2. the restructuring signal to the cell in which LA_i resides. Restructuring signal is used to update the tendency vector of the cell. Dynamicity of *DICLA* is the result of modifications made to the tendency vectors of its constituting cells. Figure 1 gives a schematic of *DICLA*. A *DICLA* is formally defined below.

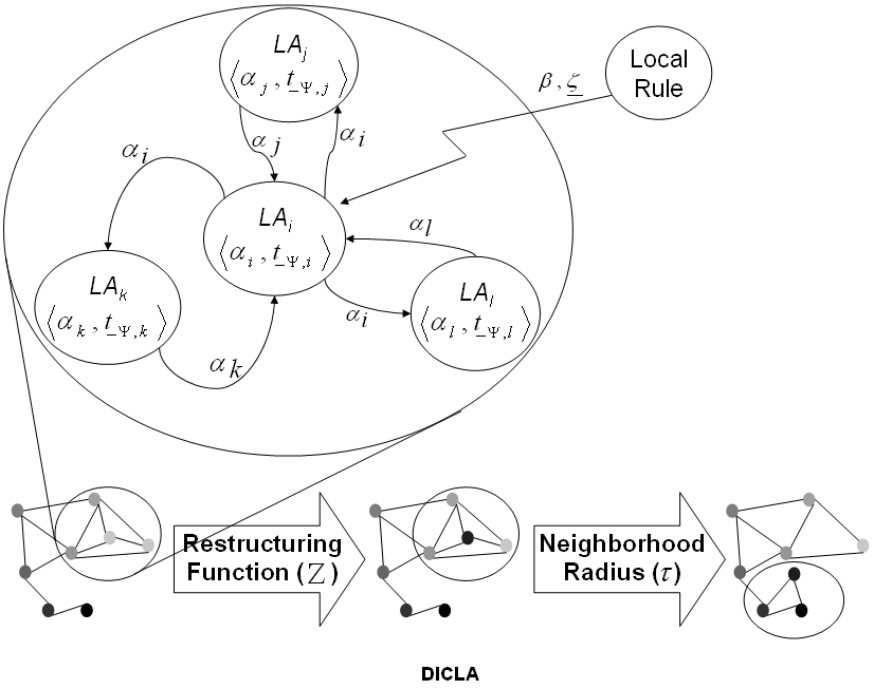

Figure 1. Dynamic Irregular cellular learning automata (*DICLA*)

Definition 1. A dynamic irregular cellular learning automata (*DICLA*) is a structure $A = (G < E, V >, \Psi, A, \Phi < \alpha, \underline{t}_{\Psi} >, \tau, F, Z)$ where

1. G is an undirected graph, with V as the set of vertices and E as the set of edges. Each vertex represents a cell in *DICLA*.
2. Ψ is a finite set of interests. Cardinality of Ψ is denoted by $|\Psi|$.
3. A is the set of learning automata each of which is assigned to one cell of *DICLA*.
4. $\Phi < \alpha, \underline{t}_{\Psi} >$ is the cell state. State of a cell c_i (Φ_i) consists of two parts; 1. α_i which is the action selected by the learning automaton of that cell, and 2. A vector $\underline{t}_{\Psi,i} = \left(t_{1,i}, t_{2,i}, \ldots, t_{|\Psi|,i}\right)^T$ which is called tendency vector of the cell. Each element $t_{k,i} \in [0,1]$ in tendency vector of the cell c_i shows the degree of tendency of c_i to the interest $\psi_k \in \Psi$.
5. τ is the neighborhood radius. Two cells c_i and c_j of *DICLA* are neighbors if $\left\| \underline{t}_{\Psi,i} - \underline{t}_{\Psi,j} \right\| \le \tau$. In other words, two cells of *DICLA* are neighbors if the distance between their tendency vectors is smaller than or equal to τ.
6. $F : \underline{\Phi}_i \rightarrow \left\langle \underline{\beta}, [0,1]^{|\Psi|} \right\rangle$ is the local rule of *DICLA* in each cell c_i, where $\underline{\Phi}_i = \left\{ \Phi_j \left| \left\| \underline{t}_{\Psi,i} - \underline{t}_{\Psi,j} \right\| \le \tau \right\} + \{\Phi_i\}$ is the set of states of all neighbors of c_i, $\underline{\beta}$ is the set of values that the reinforcement signal can take, and $[0,1]^{|\Psi|}$ is a $|\Psi|$-dimensional unit hypercube. From the current states of the neighboring cells of each cell c_i, local rule performs the followings: 1. gives the reinforcement signal to the learning automaton LA_i resides in c_i, and 2. produces a restructuring signal ($\underline{\zeta} = \left(\zeta_1, \zeta_1, \ldots, \zeta_{|\Psi|}\right)^T$) which is used to change the tendency vector of c_i. Each element ζ_i of the restructuring signal is a scalar value within the close interval $[0,1]$.
7. $Z : [0,1]^{|\Psi|} \times [0,1]^{|\Psi|} \rightarrow [0,1]^{|\Psi|}$ is the restructuring function which modifies the tendency vector of a cell using the restructuring signal produced by the local rule of the cell.

In what follows, we consider *DICLA* with N cells. The learning automaton LA_i which has a finite action set $\underline{\alpha}_i$ is associated to cell c_i (for i=1, 2, …, N) of *DICLA*. Let the cardinality of $\underline{\alpha}_i$ be m_i. The state of *DICLA* represented by $p=(p_1, p_2, ..., p_N)$, where $p_i = \left(p_{i1}, p_{i2}, \ldots, p_{im_i} \right)$ is the action probability vector of LA_i.

The operation of *DICLA* takes place as the following iterations. At iteration n, each learning automaton chooses an action. Let $\alpha_i \in \underline{\alpha}_i$ be the action chosen by LA_i. Then, each learning automaton receives a reinforcement signal. Let $\beta_i \in \underline{\beta}$ be the reinforcement signal received by LA_i. This reinforcement signal is produced by the application of the local rule $F\left(\underline{\Phi}_j\right) \to \left\langle \underline{\beta}, [0,1]^{|\Psi|} \right\rangle$. Each *LA* updates its action probability vector on the basis of the supplied reinforcement signal and the action chosen by the cell. Next, each cell c_i updates its tendency vector using the restructuring function Z (equation (3)).

$$\underline{t}_{\Psi,i}\left(n+1\right) = Z\left(\underline{t}_{\Psi,i}\left(n\right), \underline{\zeta}_i\left(n\right)\right) \qquad (3)$$

A *DICLA* is called asynchronous if at a given time only some cells are activated independently from each other, rather than all together in parallel. The cells may be activated in either time-driven or step-driven manner. In time-driven asynchronous *DICLA*, each cell is assumed to have an internal clock which wakes up the *LA* associated to that cell while in step-driven asynchronous *DICLA* a cell is selected in fixed or random sequence.

3-4-1. *DICLA* Norms of Behavior

Behavior of *DICLA* within its environment can be studied from two different aspects; the operation of *DICLA* in the environment, which is a macroscopic view of the actions performed by its constituting learning automata, and the restructurings of *DICLA*, which is the result of application of the restructuring function. To study the operation of *DICLA* in the environment we use entropy and to study the restructurings of *DICLA* we use restructuring tendency.

3-4-1-1.Entropy

Entropy, as introduced in the context of information theory by Shannon [57], is a measure of uncertainty associated with a random variable and is defined according to equation (4),

$$H\left(X\right) = -\sum_{X \in \chi} P\left(X\right) \cdot \ln\left(P\left(X\right)\right) \qquad (4)$$

where X represents a random variable with set of values χ and probability mass function $P(X)$. Considering the action chosen by a learning automaton LA_i as a random variable, the concept of entropy can be used to measure the uncertainty associated with this random variable at any given time instant n according to equation (5),

$$H_i\left(n\right) = -\sum_{j=1}^{m_i} p_{ij}\left(n\right) \cdot \ln\left(p_{ij}\left(n\right)\right) \qquad (5)$$

where m_i is the cardinality of the action set of learning automaton LA_i. In the learning process, $H_i\left(n\right)$ represents the uncertainty associated with the decision of LA_i at time instant n. Larger values of $H_i\left(n\right)$ mean more uncertainty in the decision of the learning automaton LA_i. H_i can only represent the uncertainty associated with the operation of a single learning automaton, but as the operation of *DICLA* in the environment is a macroscopic view of the operations of all of its constituting learning automata, we extend the concept of entropy through equation (6) in order to provide a metric for evaluating the uncertainty associated with the operation of a *DICLA*.

$$H(n)=\sum_{i=1}^{N}H_i(n) \qquad (6)$$

In the above equation, N is the number of learning automata in *DICLA*. The value of zero for $H(n)$ means $p_{ij}(n)\in\{0,1\}$, $\forall i,j$. This means that no learning automaton in *DICLA* changes its selected action over time, or in other words, the behavior of *DICLA* remains unchanged over time. Higher values of $H(n)$ mean higher rates of changes in the behavior of *DICLA*.

3-4-1-2. Restructuring Tendency

Restructuring tendency (υ), as defined by equation (7), is used to measure the dynamicity of the structure of *DICLA*.

$$\upsilon(n)=\sum_{i=1}^{N}\left|\zeta_i(n)\right| \qquad (7)$$

The value of zero for $\upsilon(n)$ means that tendency vector of no cell of *DICLA* during nth iteration has changed which means that no changes has occurred in the structure of *DICLA* during the nth iteration. Higher values of $\upsilon(n)$ mean higher changes in the structure of *DICLA* during the nth iteration.

4. Problem Statement

Consider N mobile sensor nodes $s_1, s_2, \ldots, s_N$ with equal sensing ranges of $R_s=r$ and transmission ranges of $R_t=2.r$ which are initially deployed in some initial positions within an unknown 2 dimensional environment Ω. Assume that a rough estimate of the surface of Ω ($\hat{S}_\Omega$) is available (using Google maps for example). According to this estimate, expected number of neighbors for a sensor node (number of sensors residing within its transmission range) referred to as $E[N_{nei}]$ is equal to $N\cdot\frac{\pi\cdot R_t^2}{\hat{S}_\Omega}-1$. $E[N_{nei}]$ is provided to all sensor nodes prior to their initial deployments. Sensor nodes are able to move in any desired direction within the area of the network at a constant speed, but they cannot cross the barrier of Ω. We assume that sensor nodes have no mechanism for estimating their physical positions or their relative distances to each other.

Consider the following definitions:

Definition 2. Sensing region of a node s_i denoted by $C(s_i)$ is a circle with radius R_s centered on s_i.

Definition 3. Covered sub-area denoted by $C(\Omega)$ refers to any point within the network area which is in the sensing region of at least one of the sensor nodes. $C(\Omega)$ is stated using equation (8)

$$C(\Omega)=\bigcap\left(\Omega,\ \bigcup_{i=1}^{N}\left(C(s_i)\right)\right) \qquad (8)$$

Definition 4. Covered section denoted by $S_{C(\Omega)}$ is the surface of the covered sub-area.

Definition 5. Deployment strategy is an algorithm which gives for any given sensor node s_i a certain position within the area of the network.

Definition 6. Deployed network refers to a sensor network which results from a deployment strategy.

Definition 7. Self-regulated deployment strategy is a deployment strategy in which each sensor node finds its proper position within the network area by exploring the area and cooperating with its neighboring sensor nodes.

Definition 8. A connected network is a network in which there is a route between any two sensor nodes.

Using the above definitions and assumptions, the problem considered in this paper can be stated as follows: Propose a self-regulated deployment strategy which deploys *N* mobile sensor nodes throughout an unknown network area Ω with estimated surface $\hat{S}_{\Omega}$ so that the covered section of Ω ($S_{C(\Omega)}$) is maximized and the deployed network is connected.

5. CLA-DS: A Cellular Learning Automata-based Deployment Strategy

The proposed deployment strategy consists of 4 major phases; Initial deployment, mapping, deployment, and maintenance. During the initialization phase, sensor nodes are initially deployed in some initial positions within the area of the network. Mapping the network topology to a *DICLA* is done in the mapping phase. Deploying sensor nodes throughout the area of the network is performed during the deployment phase. Finally, in the maintenance phase, deployed network is maintained in order to compensate the effect of possibly node failures on the covered sub-area ($C(\Omega)$). We explain these 4 phases in more details in the subsequent sections.

5-1. Initial Deployment

Three main strategies exist for initially deploying sensor nodes in some initial positions within the area of the network (Ω):

- **Random deployment**: In random deployment strategy, a random based deployment strategy [3][9][10][11][12][13][14] is used to deploy sensor nodes uniformly at random throughout Ω. Using this strategy for initial deployment reduces the cost of the deployment phase due to the fact that after such initial deployment, sensor nodes need only to heel coverage holes within their vicinities rather than exploring Ω for finding their proper positions.
- **Sub-area deployment**: In this deployment strategy, sensor nodes are placed manually in some initial positions within a small accessible sub-area of the network.
- **Hybrid deployment**: In this approach, sensor nodes are deployed randomly within a small sub-area of the network.

Any of the random deployment, sub-area deployment, or hybrid deployment strategy can be used for the initial deployment phase of CLA-DS.

5-2. Mapping

In the mapping phase, a time-driven asynchronous *DICLA*, which is isomorphic to the sensor network topology, is created. Each sensor node s_i located at (x_i, y_i) in the sensor network corresponds to the cell c_i in *DICLA*. Interest set of *DICLA* consists of two members; X-axis and Y-axis of the network area. Initially (at time instant 0), tendency levels of each cell c_i to these interests are $t_{1,i}(0) = \frac{x_i(0)}{\max(MaxX, MaxY)}$ and $t_{2,i}(0) = \frac{y_i(0)}{\max(MaxX, MaxY)}$ respectively. (*MaxX*, *MaxY*) is the farthest location within the network area at which a sensor node can be located. Neighborhood radius (τ) of *DICLA* is equal to $\frac{R_t}{\max(MaxX, MaxY)}$, and hence two cells c_i and c_j in *DICLA* are adjacent to each other if s_i and s_j in the sensor network are close enough to hear each other's signals.

According to the definition of *DICLA*, values of tendency levels are used along with the neighborhood radius of *DICLA* to specify the neighboring cells of a cell. But here, values of $t_{1,i}(0)$ and $t_{2,i}(0)$ cannot be computed due to the fact that sensor nodes are not aware of their physical positions. This does not cause any problems due to the fact that in a *DICLA* which is mapped into a sensor network, neighboring cells of each cell are implicitly specified according to the topology of the network (two cells are adjacent to each other if their corresponding sensor nodes are within the transmission ranges of each other).

The learning automaton in each cell c_i of *DICLA*, referred to as LA_i, has two actions α_0, and α_1. Action α_0 is "apply force to neighboring nodes", and action α_1 is "do not apply force to neighboring nodes". The probability of selecting each of these actions is initially set to .5.

5-3. Deployment

Each sensor node has a state which can be "*mobile*" or "*fixed*". Initially, each sensor node selects its state randomly with probability of selecting "*fixed*" state to be P_{fix}. P_{fix} is a constant which is known to all sensor nodes. A "*fixed*" sensor node leaves the deployment phase immediately and starts with the maintenance phase.

Deployment phase for each "*mobile*" sensor node s_i is divided into a number of rounds $R_i(0)$, $R_i(1)$, $R_i(2)$, ... Each round $R_i(n)$ is started by the asynchronous activation of cell c_i of *DICLA*. The *n*th activation of cell c_i occurs at time $\delta_i + n \times ROUND\_DURATION$ where δ_i is a random number generated for cell c_i and *ROUND_DURATION* is an upper bound for the duration of a single round. We call *n* the local iteration number for the cell. Delays δ_i are chosen randomly in order to reduce the probability of collisions between neighboring nodes.

Upon the startup of a new round $R_i(n)$, LA_i selects one of its actions randomly according to its action probability vector. Selected action, which is one of "apply force to neighboring nodes", or "do not apply force to neighboring nodes", specifies whether sensor node s_i will apply any forces to its neighboring nodes during the current round or not. Sensor node s_i then creates a packet called *APPLIED_FORCE* containing its state and the selected action and broadcasts it in its neighborhood. After broadcasting the *APPLIED_FORCE* packet, sensor node s_i waits for certain duration (*RECIEVE_DURATION*) to receive *APPLIED_FORCE* packets from its neighboring nodes. Received packets are stored into a local database within the node.

When *RECIEVE_DURATION* is over, sensor node s_i collects following statistics from the stored information in its local database: number of received packets ($N_i^r(n)$), and number of neighbors selecting "apply force to neighboring nodes" action ($N_i^f(n)$). According to the collected statistics, local rule of the cell c_i computes the reinforcement signal $\beta_i(n)$ and the restructuring signal $\underline{\zeta}_i(n)$ which are used to update the action probability vector of LA_i and the tendency vector of c_i. Details on this will be given in section 5-3-1.

Next, sensor node s_i uses vector $\underline{\zeta}_i(n)$ as its movement path for the current round. In other words, if s_i is located at $(x_i(n), y_i(n))$, it moves to $(x_i(n+1), y_i(n+1)) = (x_i(n) + \zeta_{1,i}, y_i(n) + \zeta_{2,i})$. Last step during round $R_i(n)$ is the application of the restructuring function Z which updates the tendency levels of cell c_i using the restructuring signal $\underline{\zeta}_i(n)$ according to equation (9).

$$\begin{cases} t_{1,i}(n+1) = t_{1,i}(n) + \dfrac{\zeta_{1,i}(n)}{\max(MaxX, MaxY)} \\ t_{2,i}(n+1) = t_{2,i}(n) + \dfrac{\zeta_{2,i}(n)}{\max(MaxX, MaxY)} \end{cases} \quad (9)$$

When this step is done, sensor node s_i waits for the next activation time of its corresponding cell c_i in *DICLA* to start its next round ($R_i(n+1)$).

Deployment phase for a sensor node s_i is completed upon the occurrence of one of the followings:

- Stability: Sensor node s_i moves less than a specified threshold (*LEAST_DISTANCE*) during its last N_r rounds.
- Oscillation: Sensor node s_i oscillates between almost the same positions for more than N_o rounds.

When a sensor node s_i completes the deployment phase of CLA-DS, its state is changed to "*fixed*" and it starts with the maintenance phase.

5-3-1. Applying Local Rule

Local rule of a cell c_i based on the states of the cell and its neighboring cells computes the reinforcement signal β_i and the restructuring signal $\underline{\zeta}_i$.

Reinforcement Signal: The reinforcement signal for the *n*th round in a node s_i is computed based on a comparison between the number of neighbors of s_i ($N_i^r(n)$) and the expected number of neighbors ($E[N_{nei}]$). According to this comparison, following two cases may occur:

- $N_i^r(n)$ is almost equal to $E[N_{nei}]$ or formally $\left|N_i^r(n) - E[N_{nei}]\right| < \varepsilon$ where ε is a specified constant: In this case, if the selected action of LA_i is "do not apply force to neighboring nodes", then the reinforcement signal is to reward the action. Otherwise, the reinforcement signal is to penalize the action.
  - $N_i^r(n)$ is smaller or greater than $E[N_{nei}]$ or formally $\left|N_i^r(n) - E[N_{nei}]\right| > \varepsilon$: In this case, if the selected action of LA_i is "apply force to neighboring nodes", then the reinforcement signal is to reward the action. Otherwise, the reinforcement signal is to penalize the action.

Equations (1) and (2) are used for rewarding or penalizing the selected action of LA_i. The idea behind the above method of computing the reinforcement signal is that to have a uniform deployment of sensor nodes, one way is to minimize the difference between the number of neighbors of each sensor node and the expected number of neighbors ($E[N_{nei}]$). When $N_i^r(n)$ is almost equal to $E[N_{nei}]$, it means that the number of neighbors of sensor node s_i is as it must be, and hence it is better for this node not to apply any forces to its neighbors. In the other case when $\left|N_i^r(n) - E[N_{nei}]\right| > \varepsilon$, the number of neighbors of s_i differs from that expected, and hence it is better for it to apply forces to its neighbors.

Restructuring Signal: The restructuring signal $\underline{\zeta}_i(n) = \left(\zeta_{1,i}(n), \zeta_{2,i}(n)\right)^T$ is a 2 dimensional vector with a random orientation whose magnitude is $N_i^f(n)$ (number of neighbors selecting "apply force to neighboring nodes" action). The elements of this vector are computed according to equation (10) using an angle θ which is selected uniformly at random from the range $[0, 2\pi]$.

$$\begin{cases} \zeta_{1,i}(n) = N_i^f(n) \cdot \cos(\theta) \\ \zeta_{2,i}(n) = N_i^f(n) \cdot \sin(\theta) \end{cases} \qquad (10)$$

This vector specifies the orientation, direction, and distance of movement for the node s_i during the current round.

5-4. Maintenance

Maintenance phase in a sensor node s_i is similar to the deployment phase except that s_i remains fixed during this phase and the force vectors applied by its neighbors have no effect on it. Additionally, during this phase, sensor node s_i collects a list of its "*fixed*" neighboring nodes (neighboring nodes which are in maintenance phase). If something happens to a member s_j of this list (its battery exhausted, it experiences some failures, it leaves the sensing region of s_i, …) then sensor node s_i does not receive *APPLIED_FORCE* packets from s_j anymore. This indicates that a hole may occur in the vicinity of s_i. As a result, s_i leaves the maintenance phase, set its state to "*mobile*" and starts over with the deployment phase in order to fill any probable holes. Since collisions may also result in not receiving *APPLIED_FORCE* packets from neighbors, a node s_i starts over with the

deployment phase only if it does not receive *APPLIED_FORCE* packets from one of its "*fixed*" neighbors for more than *k* rounds.

6. Experimental Results

To evaluate the performance of CLA-DS several experiments have been conducted and the results are compared with the results obtained for potential field-based algorithm given in [15] referred to as PF hereafter, DSSA and IDCA algorithms given in [19], and the basic VEC algorithm given in [20]. The Algorithms are compared with respect to three criteria: coverage, node separation, and distance.

- Coverage: Fraction of the area which is covered by the deployed network. Coverage is specified according to equation (11).

$$Coverage = \frac{S_{C(\Omega)}}{S_{\Omega}} \qquad (11)$$

- Node separation: Average distance from the nearest-neighbor in the deployed network. Node separation can be computed using equation (12). In this equation, $dist(s_i, s_j)$ is the Euclidean distance between sensor nodes s_i and s_j. Node separation is a measure of the overlapping area between the sensing regions of sensor nodes; smaller node separation means more overlapping.

$$Node\ separation = \frac{1}{N}\sum_{i=1}^{N} \min_{s_j \in Nei(s_i)} \left(dist\left(s_i, s_j\right)\right) \qquad (12)$$

- Distance: The average distance traveled by each node. This criterion is directly related to the energy consumed by the sensor nodes.

Experiments are performed for three different simulation areas which are shown in Figure 2. Networks of different sizes from N=50 to N=500 sensor nodes are considered for simulations. Sensing ranges (R_s=r) and transmission ranges (R_t=2.r) of sensor nodes are assumed to be 5(m) and 10(m) respectively. Energy consumption of nodes follows the energy model of the J-Sim simulator [37]. Table 1 gives the values for different parameters of the algorithm.

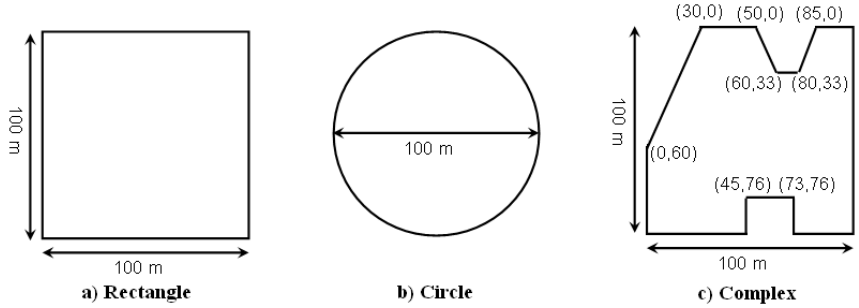

Figure 2. Different simulation areas

Table 1. Parameters of CLA-DS algorithm and their values

Parameter	Value
P_{fix}	.1
ROUND_DURATION	11 (s)
RECEIVE_DURATION	10 (s)
LEAST_DISTANCE	1 (m)
N_o	6 rounds
ε	1
a (reward parameter)	.25

b (penalty parameter)	.25
K	3 rounds

All simulations have been implemented using J-Sim simulator. J-Sim is a java based simulator which is implemented on top of a component-based software architecture. Using this component-based architecture, new protocols and algorithms can be designed, implemented and tested in the simulator without any changes to the rest of the simulator's codes.

All reported results are averaged over 50 runs. We have used CSMA as the MAC layer protocol, free space model as the propagation model, binary sensing model and Omni-directional antenna.

6-1. Experiment 1

In this experiment we compare the behavior of the CLA-DS algorithm with that of PS, DSSA, IDCA, and VEC algorithms in terms of coverage as defined by equation (11). Experiment is performed for N=50, 100, 200, 300, 400, and 500 sensor nodes which are initially deployed using a hybrid deployment method within a square with side length 10 (m) centered on the center of the network area. Figure 3 to Figure 5 give the results of this experiment for different network areas given in Figure 2. From the results we can conclude the following:

1. Although CLA-DS algorithm does not use any information about sensor positions or their relative distances to each other, its performance in covering the network area in all three environments is almost equal to that of PF algorithm in which sensor nodes have information about their relative distances to each other. This indicates the efficiency of the learning automata in guiding sensor nodes through the network area for finding their best positions.
2. In sparse networks (N<400), DSSA and IDCA algorithms better cover the network area than other algorithms. In dense networks, CLA-DS and PF algorithms outperform DSSA and IDCA algorithms in terms of coverage. This is due to the fact that the repulsive forces between neighboring nodes in DSSA and IDCA algorithms are stronger in sparse networks than in dense networks, and hence in sparse networks, sensor nodes can better spread through the network area.
3. VEC algorithm performs the same as CLA-DS and PF algorithms in sparse networks, but in dense networks, its performance is degraded. The reason for low performance of VEC algorithm in a dense network is that in a dense network, Voronoi cells of sensor nodes are very small and are covered very quickly and therefore many of sensor nodes stop moving during initial rounds without enough exploration through the network area for finding better positions. Note that in VEC algorithm, a node moves only if its movement increases the coverage of its Voronoi cell.
4. Coverage for all algorithms is better in circular network area than in rectangular and complex network areas. This means that sensor nodes are uniformly spread in all directions through the network area.

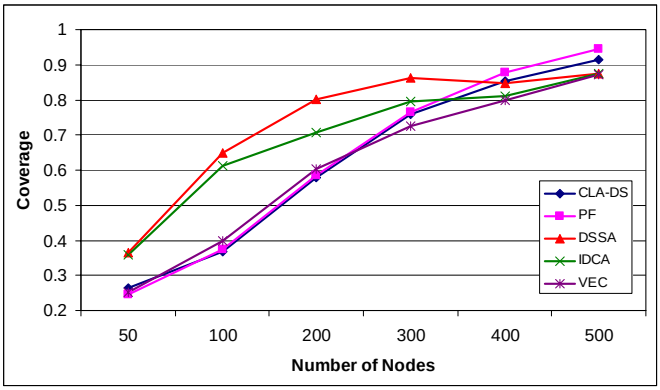

Figure 3. Comparison of CLA-DS with existing deployment algorithms in terms of coverage criterion within the rectangular network area

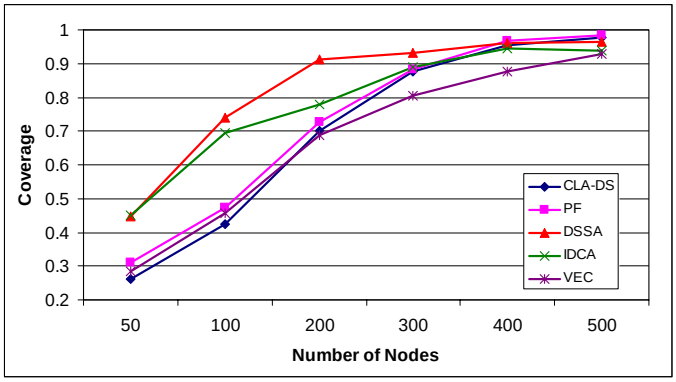

Figure 4. Comparison of CLA-DS with existing deployment algorithms in terms of coverage criterion within the circular network area

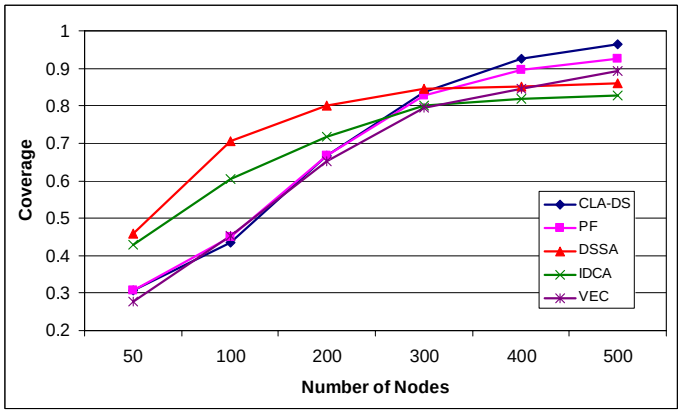

Figure 5. Comparison of CLA-DS with existing deployment algorithms in terms of coverage criterion within the complex network area

6-2. Experiment 2

In this experiment, we compare CLA-DS algorithm with PS, DSSA, IDCA, and VEC algorithms in terms of the node separation criterion given by equation (12). The simulation settings of experiment 1 are also used for this experiment. Figure 6 to Figure 8 give the results of this experiment for different network areas given in Figure 2. Node separation is a measure of the overlapping area between the sensing regions of sensor nodes; smaller node separation means more overlapping. Results of this experiment indicate that the overlapping area between sensor nodes in CLA-DS is more than in the PS, DSSA, IDCA, and VEC algorithms. Since the coverage of CLA-DS algorithm is better than or equal to the existing algorithms, having smaller node separation or more overlapped area makes CLA-DS superior to the existing algorithms due to the following reasons:

- The fraction of the network area, which is under the supervision of more than one sensor node, is higher in CLA-DS algorithm than the existing algorithms. This increases the tolerance of the network against node failures.
- In occurrences of coverage holes (due to node failures or deaths for example), neighboring nodes need fewer movements to heel the holes when node separation is smaller.

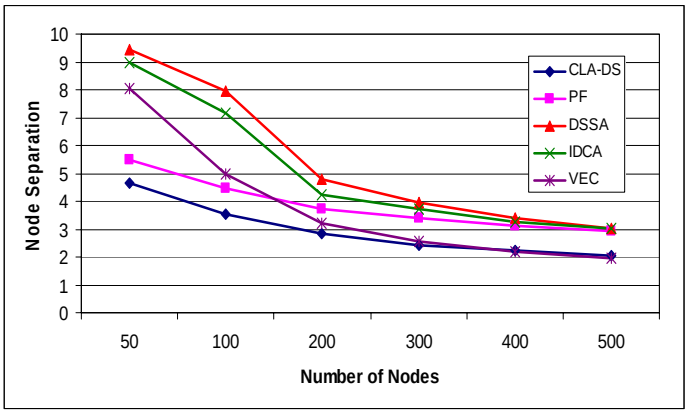

Figure 6. Comparison of CLA-DS with existing deployment algorithms in terms of node separation criterion within the rectangular network area

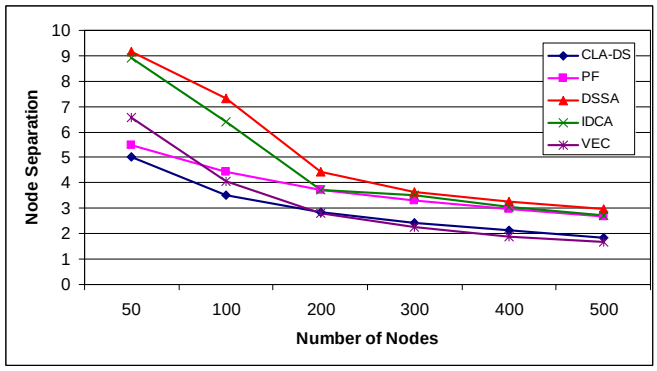

Figure 7. Comparison of CLA-DS with existing deployment algorithms in terms of node separation criterion within the circular network area

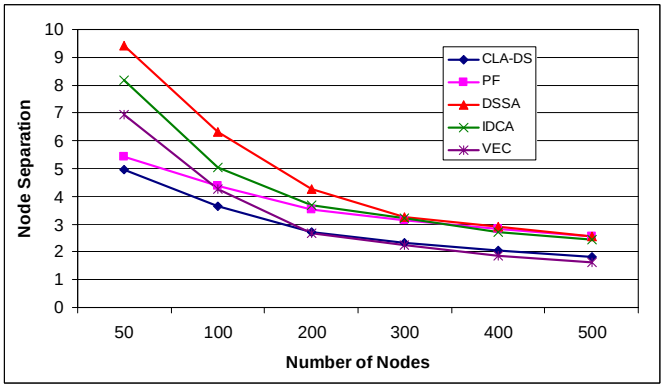

Figure 8. Comparison of CLA-DS with existing deployment algorithms in terms of node separation criterion within the complex network area

6-3. Experiment 3

In this experiment, CLA-DS algorithm is compared with PS, DSSA, IDCA, and VEC algorithms in terms of the distance criterion. The simulation settings of experiment 1 are also used for this experiment. Figure 9 to Figure 11 give the results of this experiment for different network areas given in Figure 2. From the results obtained for this experiment one may conclude the following:

1. In terms of distance criterion, the most efficient algorithm among CLA-DS, PF, DSSA, IDCA, and VEC algorithms is PF algorithm. This is due to the fact that in PF algorithm, unlike other algorithms, movements of sensor nodes are directed by the application of Newton's second law of motion [53].
2. In highly sparse networks (N<200), CLA-DS outperforms DSSA and IDCA algorithms in terms of distance criterion, but in dense networks, DSSA and IDCA algorithms perform better than

CLA-DS. This is again due to the fact that the repulsive forces between neighboring nodes in DSSA and IDCA algorithms are stronger in sparse networks than in dense networks, and hence in dense networks, movements of sensor nodes are more limited than in sparse networks. As it is shown in experiment 1, this limited movement degrades the performance of DSSA and IDCA algorithms in covering the network area.

3. For VEC algorithm, the average distance moved by sensor nodes for networks with different sizes is almost the same. This is due to the fact that in this algorithm, the number of sensor nodes which do not explore the network area sufficiently increases as the density of the network increases.

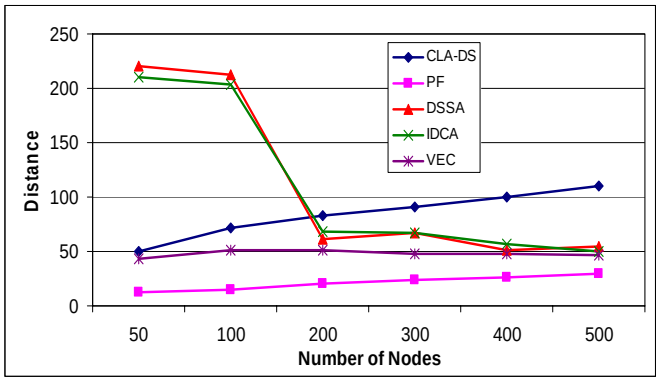

Figure 9. Comparison of CLA-DS with existing deployment algorithms in terms of distance criterion within the rectangular network area

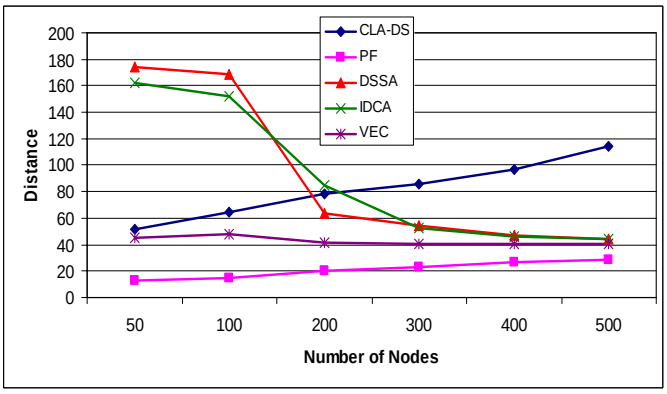

Figure 10. Comparison of CLA-DS with existing deployment algorithms in terms of distance criterion within the circular network area

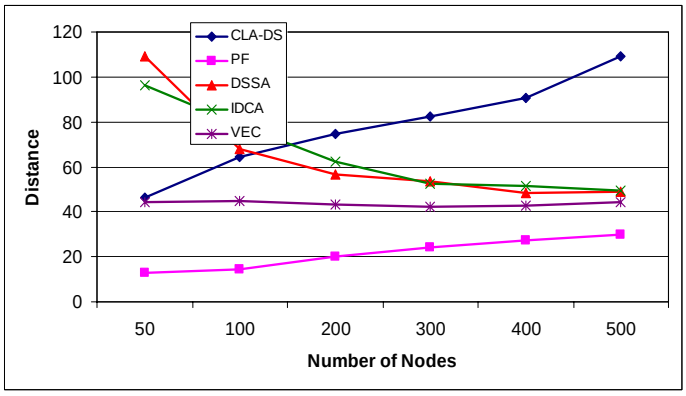

Figure 11. Comparison of CLA-DS with existing deployment algorithms in terms of distance criterion within the complex network area

6-4. Experiment 4

This experiment is conducted to study the behavior of CLA-DS algorithm in comparison to PS, DSSA, IDCA, and VEC algorithms in terms of coverage, node separation, and distance criteria when random deployment used as the initial deployment method. Experiment is performed for N=50, 100, 200, 300, 400, and 500 sensor nodes which are randomly deployed throughout the complex network area given in Figure 2. Figure 12 to Figure 14 give the results of this experiment. From these figures we may conclude the following:

1. In terms of coverage criterion, performances of all algorithms are almost the same.
2. In terms of node separation criterion, CLA-DS algorithm outperforms other algorithms.
3. In terms of distance criterion, CLA-DS algorithm has the worst performance among the existing algorithms. This is because CLA-DS does not use any information regarding the sensor positions or their relative distances to each other.

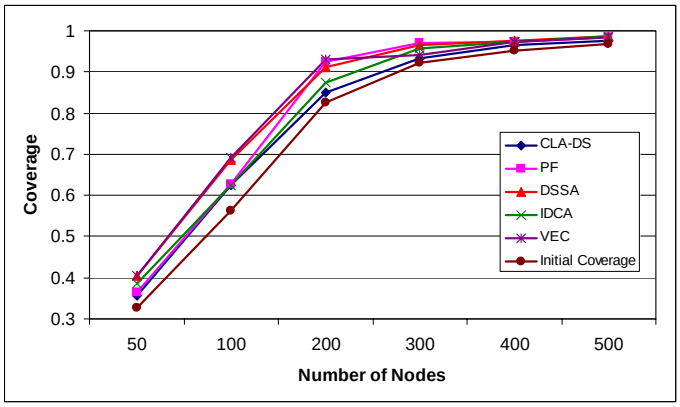

Figure 12. Comparison of CLA-DS with existing deployment algorithms in terms of coverage criterion when random deployment used as the initial deployment method

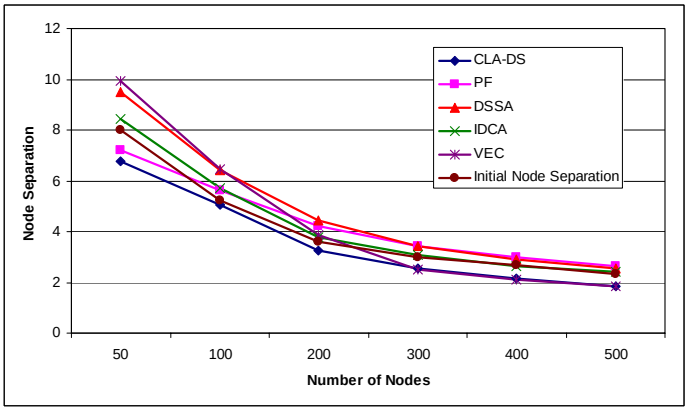

Figure 13. Comparison of CLA-DS with existing deployment algorithms in terms of node separation criterion when random deployment used as the initial deployment method

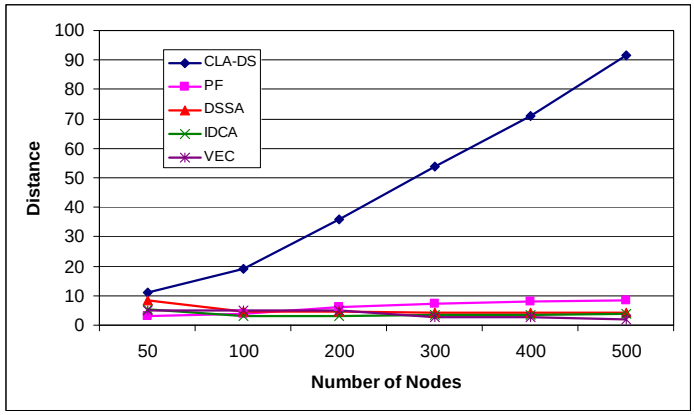

Figure 14. Comparison of CLA-DS with existing deployment algorithms in terms of distance criterion when random deployment used as the initial deployment method

6-5. Experiment 5

In this experiment, we compare CLA-DS algorithm with PS, DSSA, IDCA, and VEC algorithms in terms of coverage, node separation, and distance criteria when the region, within which sensor nodes are initially deployed, changes. For this study, we consider a network of N=500 sensor nodes which are initially deployed using a hybrid deployment method for complex network area given in Figure 2. We experiment with the following four different regions:

- A square with side length 10 (m) located at the up-left corner of the area (Up-Left).
- A square with side length 10 (m) centered on the center of the area (Center).
- Down half of the area (Down Half)
- Left half of the area (Left Half)

Figure 15 to Figure 17 give the results of this experiment. From the results one can say the following:

1. In terms of distance criterion, performances of all algorithms are degraded when the initial deployment region is in a corner of the network area.
2. Coverage and node separation of CLA-DS and PF algorithms are not highly affected by changing the initial deployment region of sensor nodes. This indicates the efficiency of these two algorithms in spreading sensor nodes through the network area.
3. Starting from a corner of the region highly affects the performance of DSSA, IDCA, and VEC algorithms in terms of coverage and node separation criteria. To explain the reason of this phenomenon, we may say that in these algorithms, sensor nodes have limited movements which do not allow them to sufficiently explore the network area. This drawback becomes more serious when the initial deployment region is in a corner of the area.

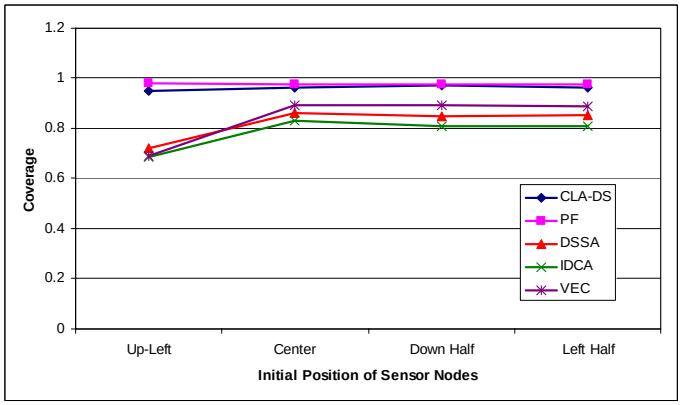

Figure 15. Comparison of CLA-DS with existing deployment algorithms in terms of coverage criterion in different initial deployment regions

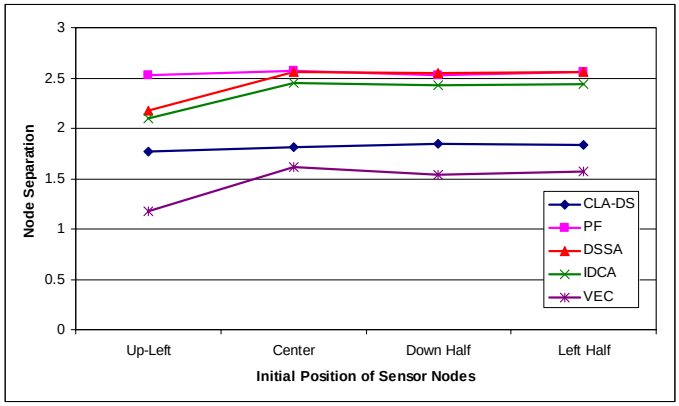

Figure 16. Comparison of CLA-DS with existing deployment algorithms in terms of node separation criterion in different initial deployment regions

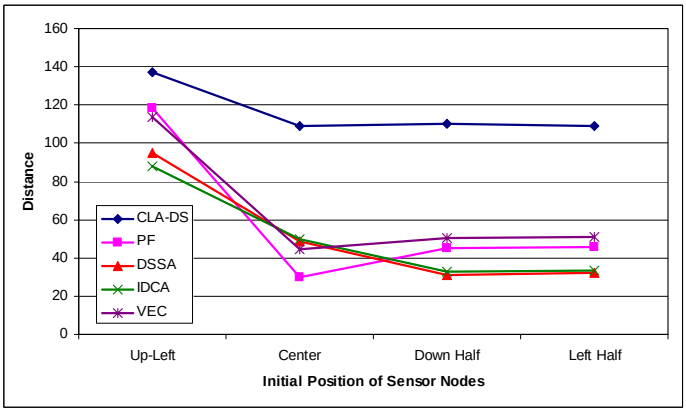

Figure 17. Comparison of CLA-DS with existing deployment algorithms in terms of distance criterion in different initial deployment regions

6-6. Experiment 6

In this experiment, CLA-DS algorithm is compared with PS, DSSA, IDCA, and VEC algorithms in terms of coverage, node separation, and distance criteria when devices or algorithms used for location estimation in sensor nodes experience different levels of noise. Such noises due to inaccuracies in measurements are common both in GPS-based location estimator devices [54] and localization techniques adopted to wireless sensor networks [55][56]. For simulating a noise level of $0 < \lambda < 1$, for each sensor node s_i two numbers $Rnd_i(x)$ and $Rnd_i(y)$ are selected uniformly at random from the ranges [0, *MaxX*] and [0, *MaxY*] respectively and are used for modifying the position (x_i, y_i) of the node according to equation (13). For this study, λ is assumed to be one of the following: .2, .25, .35, and .5.

$$\begin{cases} x_i^{Noisy} = x_i + \lambda . Rnd_i\left(x\right) \\ y_i^{Noisy} = y_i + \lambda . Rnd_i\left(y\right) \end{cases} \qquad (13)$$

The experiment is performed for N=500 sensor nodes which are initially deployed using a hybrid deployment method within a square with side length 10 (m) centered on the center of the complex network area given in Figure 2. Figure 18 to Figure 20 give the results of this experiment for different criteria. These figures show that:

1. Noise level has no effect on the performance of CLA-DS algorithm with respect to coverage, node separation and distance criteria. This is due to the fact that CLA-DS algorithm does not use any information about the position of sensor nodes.

2. PF algorithm is highly affected by increasing the noise level. This is because in PF, movements of sensor nodes are directed by the application of Newton's second law of motion [53] which is highly sensitive to noise.
3. Noise level does not highly affect the performances of DSSA and IDCA algorithms with respect to coverage criterion. This is due to the fact that these algorithms, like CLA-DS algorithm, try to minimize the difference between local density and expected local density of sensor nodes which is not sensitive to noise level.
4. Noise level highly affects the performances of DSSA and IDCA algorithms with respect to node separation and distance criteria. This is because DSSA and IDCA algorithms, unlike CLA-DS, use the relative distances of neighboring sensor nodes, which is sensitive to noise level, in order to minimize the difference between local density and expected local density of sensor nodes.

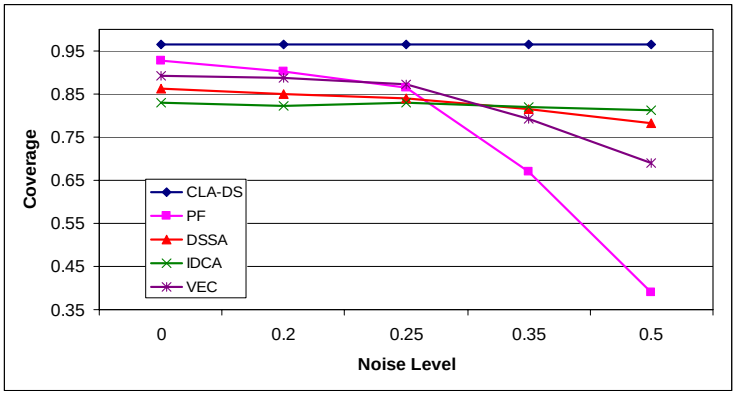

Figure 18. Comparison of CLA-DS with existing deployment algorithms in terms of coverage criterion in presence of noise in estimating the position of sensor nodes

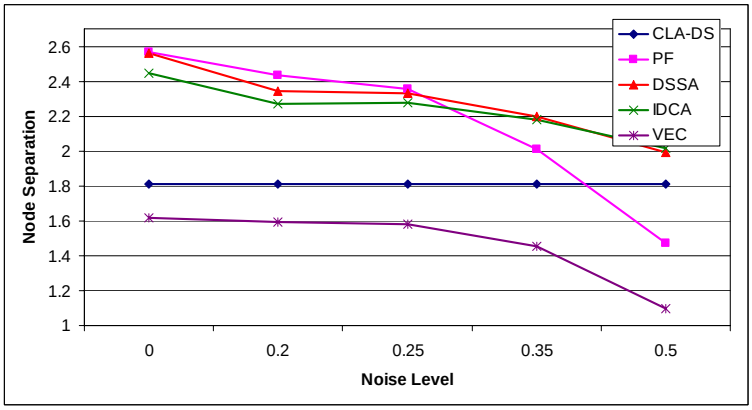

Figure 19. Comparison of CLA-DS with existing deployment algorithms in terms of node separation criterion in presence of noise in estimating the position of sensor nodes

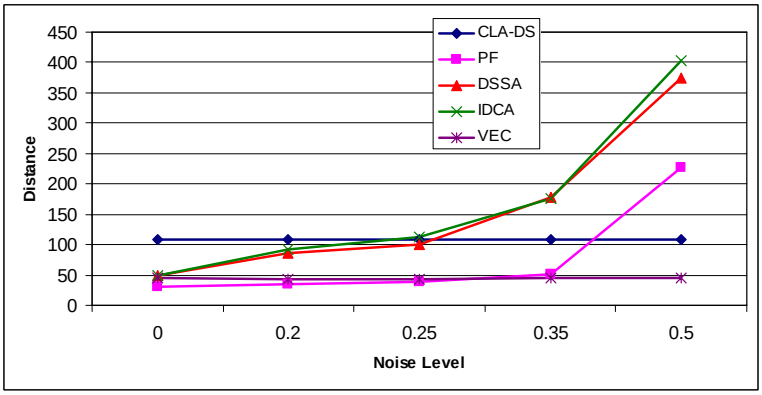

Figure 20. Comparison of CLA-DS with existing deployment algorithms in terms of distance criterion in presence of noise in estimating the position of sensor nodes

6-7. Experiment 7

In this experiment, we compare the behavior of CLA-DS, PS, DSSA, IDCA, and VEC algorithms with respect to coverage, node separation and distance criteria when the movements of sensor nodes are not perfect and follow a probabilistic motion model. A probabilistic motion model can better describe the movements of sensor nodes in real world scenarios. We use the probabilistic motion model of sensor nodes given in [58]. In this probabilistic motion model, movements of a sensor node s_i for a given drive ($d_i(n)$) and turn ($r_i(n)$) command is described using the following equations:

$$\begin{cases} x_i(n+1) = x_i(n) + D_i(n)\cdot\cos\left(\theta_i(n) + \dfrac{T_i(n)}{2}\right) + \\ \qquad\qquad C_i(n)\cdot\cos\left(\theta_i(n) + \dfrac{T_i(n)+\pi}{2}\right) \\ y_i(n+1) = y_i(n) + D_i(n)\cdot\sin\left(\theta_i(n) + \dfrac{T_i(n)}{2}\right) + \\ \qquad\qquad C_i(n)\cdot\sin\left(\theta_i(n) + \dfrac{T_i(n)+\pi}{2}\right) \\ \theta_i(n+1) = \left(\theta_i(n) + T_i(n)\right) \bmod (2\pi) \end{cases} \tag{14}$$

In equation (14), $\theta_i(n) + \dfrac{T_i(n)}{2}$ is referred to as the major axis of movement, $\theta_i(n) + \dfrac{T_i(n)+\pi}{2}$ is the minor axis of movement (orthogonal to the major axis), and $C_i(n)$ is an extra lateral translation term to account for the shift in the orthogonal direction to the major axis. $D_i(n)$, $T_i(n)$, and $C_i(n)$ are all independent and conditionally Gaussian given $d_i(n)$ and $r_i(n)$:

$$\begin{cases} D_i(n) \sim N\left(d_i(n), d_i^2(n)\cdot\sigma_{D_d}^2 + r_i^2(n)\cdot\sigma_{D_r}^2 + \sigma_{D_1}^2\right) \\ T_i(n) \sim N\left(r_i(n), d_i^2(n)\cdot\sigma_{T_d}^2 + r_i^2(n)\cdot\sigma_{T_r}^2 + \sigma_{T_1}^2\right) \\ C_i(n) \sim N\left(0, d_i^2(n)\cdot\sigma_{C_d}^2 + r_i^2(n)\cdot\sigma_{C_r}^2 + \sigma_{C_1}^2\right) \end{cases} \tag{15}$$

where $N(a,b)$ is a Gaussian distribution with mean a and variance b, and $\sigma_{D_d}^2$, $\sigma_{D_r}^2$, $\sigma_{D_1}^2$, $\sigma_{T_d}^2$, $\sigma_{T_r}^2$, $\sigma_{T_1}^2$, $\sigma_{C_d}^2$, $\sigma_{C_r}^2$, and $\sigma_{C_1}^2$ are all parameters of the specified motion model. Table 2 gives values of these parameters used for this experiment.

Table 2. Parameters of the specified probabilistic motion model and their corresponding values

Parameter	$\sigma_{D_d}^2$	$\sigma_{D_r}^2$	$\sigma_{D_1}^2$	$\sigma_{T_d}^2$	$\sigma_{T_r}^2$	$\sigma_{T_1}^2$	$\sigma_{C_d}^2$	$\sigma_{C_r}^2$	$\sigma_{C_1}^2$
Value	.021869	.010731	.000001	.000345	.338267	.666048	.008588	.013427	.000014

The experiment is performed for N=500 sensor nodes which are initially deployed using a hybrid deployment method within a square with side length 10 (m) centered on the center of the complex network area given in Figure 2. Figure 21 to Figure 23 show the results of this experiment. The results indicate the following facts:

- Using the probabilistic motion model instead of the perfect motion model degrades significantly the performances of PF, DSSA, and IDCA algorithms in terms all three criteria, but does not affect the performances of VEC and CLA-DS algorithms substantially.

- When the probabilistic motion model is used CLA-DS and VEC algorithms outperform the existing algorithms in terms of coverage criterion.
- Node separation of CLA-DS algorithm is smaller than that of VEC algorithm. This indicates the superiority of CLA-DS algorithm over VEC algorithm in terms of node separation criterion using a similar discussion to that given in experiment 2.
- Node separation of CLA-DS algorithm is larger than that of PF, DSSA, and IDCA algorithms. This means that CLA-DS has more overlapping area between sensor nodes.
- For probabilistic motion model, the average distance moved by sensor nodes for all algorithms except for VEC algorithm is higher than when perfect motion model is used. When the probabilistic motion model is used the performances of CLA-DS, PF, DSSA, and IDCA algorithms are degraded by 24%, 136%, 81%, and 100% respectively. This indicates that CLA-DS algorithm is more robust to deviations in the movements of sensor nodes than PF, DSSA, and IDCA algorithms.
- For VEC algorithm, the average distance moved by sensor nodes does not change significantly when the probabilistic motion model is used. This is due to following two reasons: 1. In VEC algorithm, many of the sensor nodes stop moving during initial rounds without enough exploration through the network area, and 2. Sensor nodes move within their Voronoi cells in VEC algorithm and hence their movements are very limited.

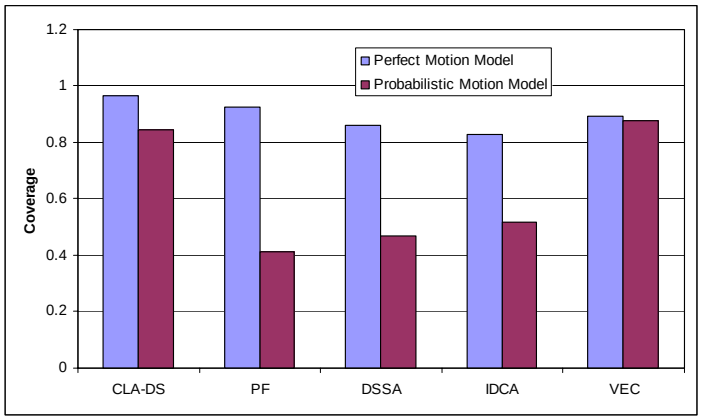

Figure 21. Comparison of CLA-DS with existing deployment algorithms in terms of coverage criterion when movements of sensor nodes follow a probabilistic motion model

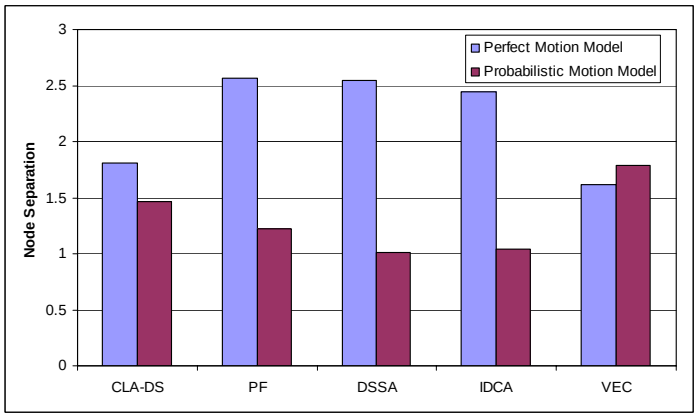

Figure 22. Comparison of CLA-DS with existing deployment algorithms in terms of node separation criterion when movements of sensor nodes follow a probabilistic motion model

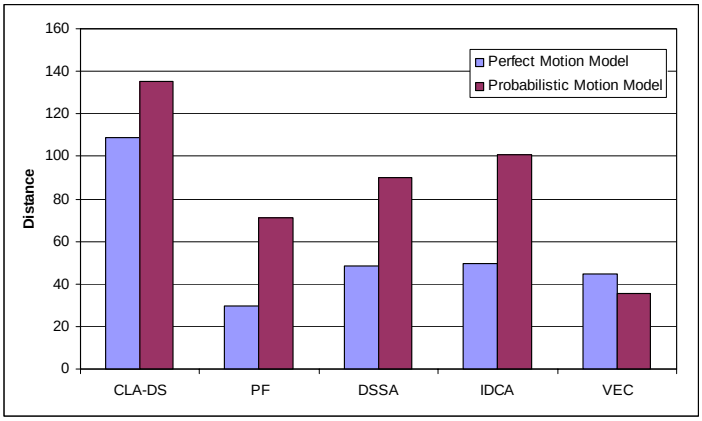

Figure 23. Comparison of CLA-DS with existing deployment algorithms in terms of distance criterion when movements of sensor nodes follow a probabilistic motion model

6-8. Experiment 8

This experiment is conducted to study the behavior of CLA-DS, PS, DSSA, IDCA, and VEC algorithms in controlling the local density of sensor nodes in the network during the deployment process. The results of this experiment also show that CLA-DS algorithm gradually learns the expected local density. For this study, we consider networks of N=500, 200, and 50 sensor nodes which are initially deployed using a hybrid deployment method within a square with side length 10 (m) centered on the center of the complex network area given in Figure 2. Figure 24 to Figure 26 give the results of this experiment. In these figures, X-axis gives the overall distance moved by all sensor nodes during the deployment process and Y-axis shows the local density of sensor nodes as measured during the deployment process. These figures show that CLA-DS algorithm, without any node knowing its physical position or its relative distances to its neighbors, controls the local density of sensor nodes in such a way that the local density approaches its expected value just as other algorithms do. Of course, it takes longer time for CLA-DS to achieve this. This is due to the fact that CLA-DS does not use any information regarding the physical positions of sensor nodes or their relative distances to each other.

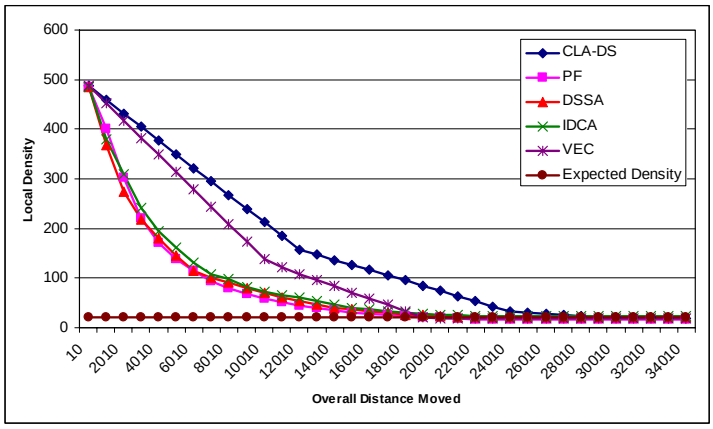

Figure 24. Local density of sensor nodes during the deployment process when N=500

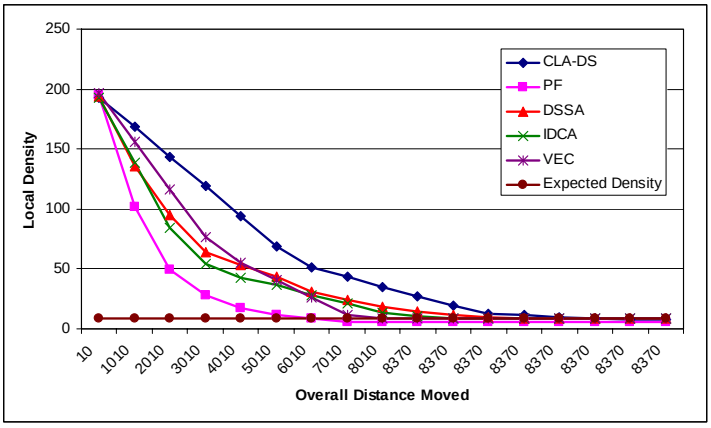

Figure 25. Local density of sensor nodes during the deployment process when N=200

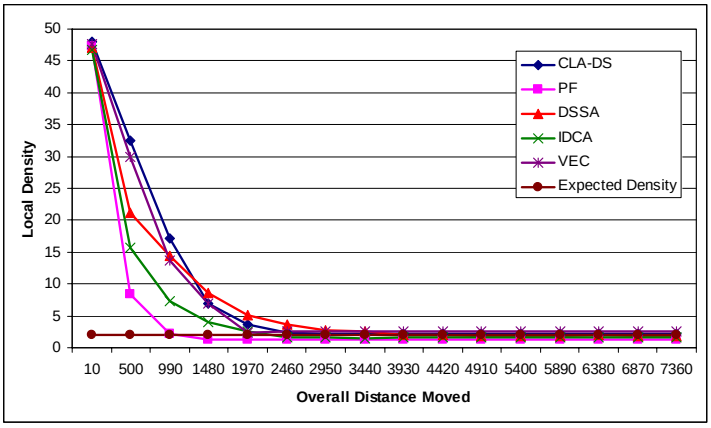

Figure 26. Local density of sensor nodes during the deployment process when N=50

6-9. Experiment 9

This experiment is conducted to study the effect of parameter P_{fix} on the performance of CLA-DS algorithm. For this study, we consider a network of N=500 sensor nodes which are initially deployed using a hybrid deployment method within a square with side length 10 (m) centered on the center of the complex network area given in Figure 2. Figure 27 to Figure 29 give the results of this experiment. These figures indicate that by increasing the value of parameter P_{fix} in CLA-DS algorithm, the covered section of the area, node separation and the average distance traveled by each sensor node decrease. In other words, higher values of P_{fix} results in more energy saving during the deployment process at the expense of poor coverage. Determination of P_{fix} for an application is very crucial and is a matter of cost versus precision. For better coverage, higher price must be paid.

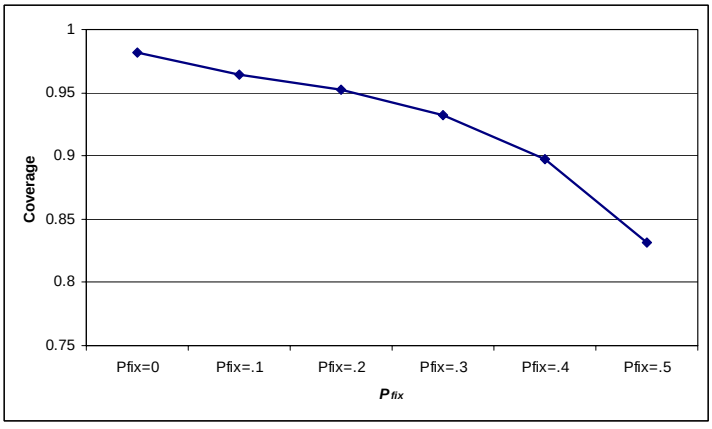

Figure 27. Effect of parameter P_{fix} on the performance of CLA-DS in terms of coverage criterion

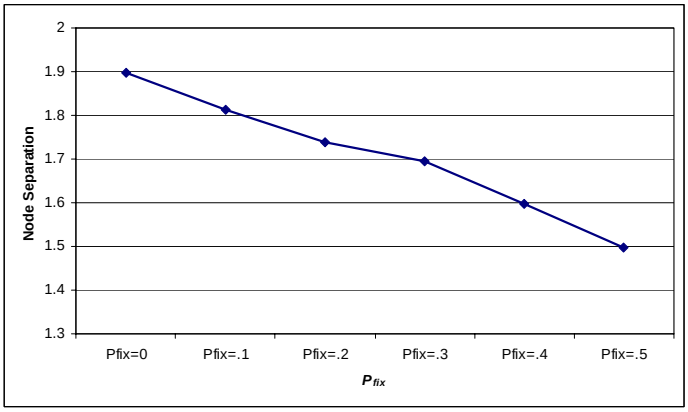

Figure 28. Effect of parameter P_{fix} on the performance of CLA-DS in terms of node separation criterion

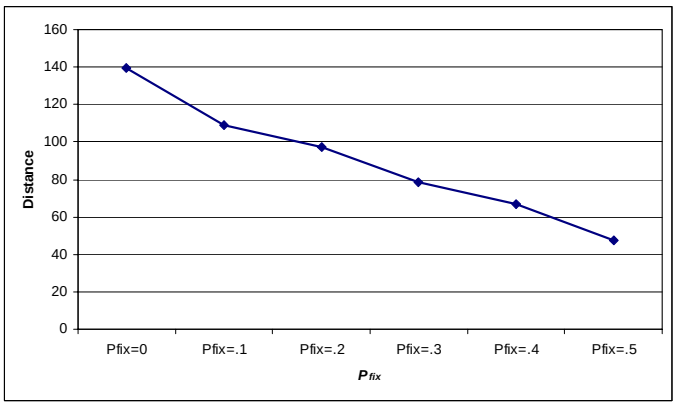

Figure 29. Effect of parameter P_{fix} on the performance of CLA-DS in terms of distance criterion

6-10. Experiment 10

This experiment is conducted to study the behavior of *DICLA* as a learning model in CLA-DS algorithm. For this study, we consider a network of N=500 sensor nodes which are initially deployed using a hybrid deployment method within a square with side length 10 (m) centered on the center of the complex network area given in Figure 2. Figure 30 depicts the action probability vector of a randomly selected learning automaton from *DICLA*. As it can be seen, at the beginning of the deployment process, the action probability of "apply force to neighboring nodes" action increases. This is due to the fact that the density of sensor nodes in the initial hybrid deployment method is very high and hence, sensor nodes must apply force to each other to spread through the area. As time passes, the action probability of "do not apply force to neighboring nodes" action gradually increases and approaches unity. As a result, local node density gradually approaches its desired value ($E[N_{nei}]$).

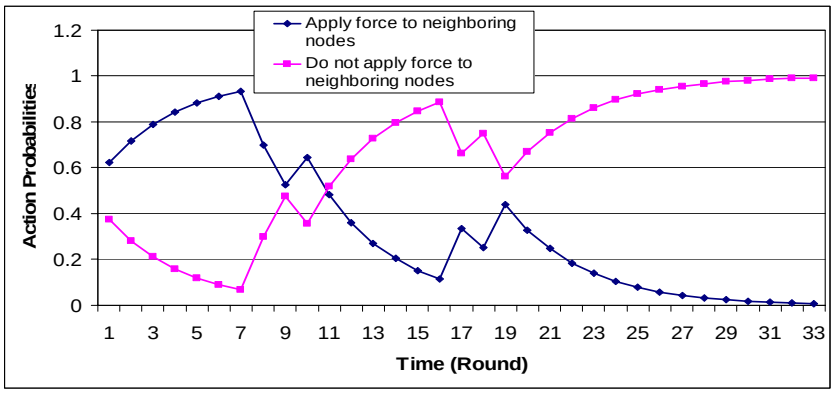

Figure 30. Action probabilities of a randomly selected learning automaton from *DICLA*

Figure 31 shows the *DICLA* entropy during the deployment process. This figure indicates that the entropy of *DICLA* is high at initial rounds of CLA-DS algorithm, but gradually decreases as time passes. It goes below 45 at about round number 750. This means that after this round, the entropy of each learning automaton LA_i in *DICLA* is on average below .09. If the entropy of a two-action learning automaton is below .09, then it can be concluded that the action probability of one of its actions is higher than .982. This means that action switching in each learning automaton in *DICLA* rarely occurs after round number 750.

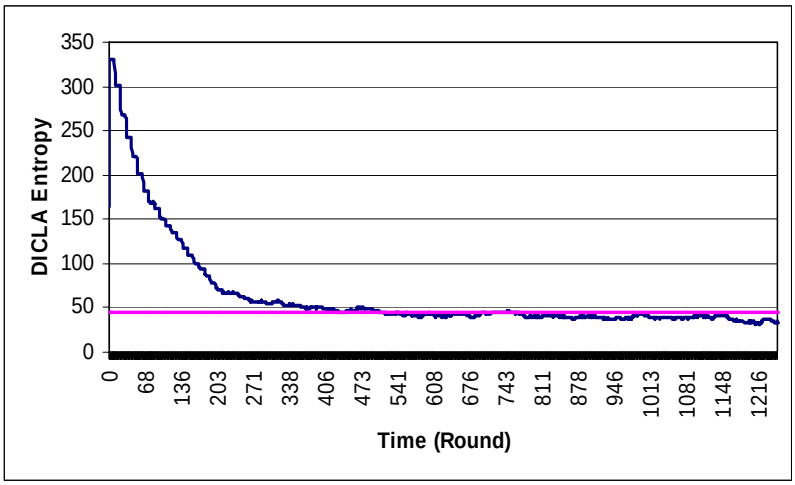

Figure 31. *DICLA* entropy

Figure 32 depicts the changes in *DICLA* restructuring tendency during deployment process. This figure shows that the restructuring tendency of *DICLA* is initially high and gradually approaches zero. It is initially high because during initial rounds, the magnitude of the force vector applied to each sensor node is large, and it gradually approaches zero because as time passes, the local density of sensor nodes approaching its expected value which results in the magnitude of the force vector applied to each sensor node to approach zero.

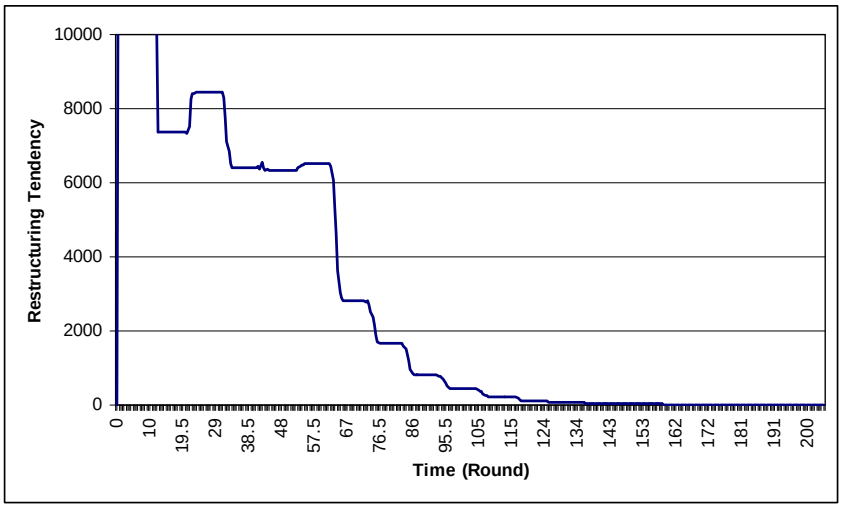

Figure 32. *DICLA* restructuring tendency

6-11. Summary of Results

In this study, we compared the performance of CLA-DS deployment algorithm in terms of coverage, node separation, and distance criteria with PF, DSSA, IDCA, and VEC deployment algorithms. Comparisons were made for different network sizes and areas, different initial deployment strategies and regions, and noise free and noisy environments. From the results of this study we can conclude that:

- In all noise free environments and starting from either of hybrid or random initial deployment strategies, the proposed algorithm (CLA-DS) can compete with existing algorithms in terms of coverage criterion, outperforms existing algorithms in terms of node separation criterion, and performs worse than existing algorithms in terms of distance criterion.
- The coverage and node separation for CLA-DS and PF do not change when the initial deployment region of sensor nodes within the area of the network changes.
- CLA-DS algorithm, unlike existing algorithms, does not use any information regarding the position of sensor nodes or their relative distances to each other and therefore, in noisy environments, where utilized location estimation techniques such as GPS-based devices and localization algorithms experience inaccuracies in their measurements, CLA-DS algorithm outperforms existing algorithms in terms of all the criteria.
- The algorithms which are least affected by the selection of "node movement model" of sensor nodes are CLA-DS and VEC.
- CLA-DS algorithm, unlike existing algorithms, has a parameter (P_{fix}) for controlling the tradeoff between the network coverage and the average distance traveled by sensor nodes.

7. Conclusion

In this paper, a deployment strategy based on cellular learning automata called CLA-DS for mobile sensor networks was proposed. CLA-DS, unlike similar existing deployment strategies, does not use any information regarding sensor positions or their relative distances to each other. Using the proposed movement strategy, each node in cooperation with its neighboring nodes gradually learns its best position within the area of the network in order to attain high coverage. Experimental results showed that in noise free environments, CLA-DS algorithm can compete with the existing algorithms such as PF, DSSA, IDCA, and VEC in terms of network coverage. It was also showed that in noisy environment, where utilized location estimation techniques such as GPS-based devices and localization algorithms experience inaccuracies in their measurements, or the movements of sensor nodes are not perfect and follow a probabilistic motion model, CLA-DS algorithm outperforms the existing algorithms in terms of network coverage.

References

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

[2] S.S. Dhillon, K. Chakrabarty, and S.S. Iyengar, "Sensor Placement for Grid Coverage under Imprecise Detections", *Int. Conf. on Information Fusion (FUSION) 2002*, Annapolis, 2002, pp. 1581–1587.

[3] S. Tilak, N.B. Abu–Ghazaleh, and W. Heinzelman, "Infrastructure Trade-offs for Sensor Networks", *ACM WSNA'02*, September 2002, pp. 49–58.

[4] S. Musman, P.E. Lehner, and C. Elsaesser, "Sensor Planning for Elusive Targets", *Journal of Computer Math. Modeling*, Vol. 25, No. 3, pp. 103–115, 1997.

[5] A. Salhieh, J. Weinmann, M. Kochhal, and L. Schwiebert, "Power Efficient Topologies for Wireless Sensor Network", *Int. Conf. on Parallel Processing*, Spain, 2001, pp. 156–163.

[6] L. Schwiebert, S.K.S. Gupta, and J. Weinamann, "Research Challenges in Wireless Networks of Biomedical Sensors", *ACM SIGMOBILE 2001*, Rome, July 2001, pp. 151–165.

[7] E. M. Petriu, N. D. Georganas, D. Petriu, D. Makrakis, and V. Z. Groza, "Sensor-based Information Appliances", *IEEE Instrumentation Measurement Mag.*, pp. 31–35, December 2000.

[8] K. Chakrabarty, S.S. Iyengar, H. Qi, and E. Cho, "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks", *IEEE Trans. on Computers*, Vol. 51, pp. 1448–1453, 2002.

[9] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient Communication Protocol for Wireless Microsensor Networks", *HICSS 2000*, Maui, January 2000, pp. 8020–8029.

[10] W. Heinzelman, "Application-specific Protocol Architecture for Wireless Networks", Ph.D. dissertation, Massachusetts Institute of Technology, June 2000.

[11] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan, "An application-specific Protocol Architecture for Wireless Microsensor Networks", *IEEE Trans. on Wireless Communications*, Vol. 1, No. 4, pp. 660–670, October 2002.

[12] S. Lindsey, and C.S. Raghavendra, "PEGASIS: Power-efficient Gathering in Sensor Information Systems", *IEEE Aerospace Conf.*, March 2002, pp. 1125–1130.

[13] S. Lindsey, C. Raghavendra, and K.M. Sivalingam, "Data Gathering Algorithms in Sensor Networks using Energy Metrics", *IEEE Trans. on Parallel Distributed Systems*, Vol. 13, No. 9, pp. 924–935, September 2002.
[14] A. Willig, R. Shah, J. Rabaey, and A. Wolisz, "Altruists in the PicoRadio Sensor Network", *4th IEEE Int. Workshop on Factory Commun. Syst*., Sweden, August 2002, pp. 175–184.
[15] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile Sensor Network Deployment using Potential Fields: A Distributed Scalable Solution to the Area Coverage Problem", *Intl. Symposium on Distributed Autonomous Robotics Systems*, Fukuoka, Japan, June, 2002, pp. 299-308.
[16] Y. Zou, and K. Chakrabarty, "Sensor Deployment and Target Localization Based on Virtual Forces", *Proc. of IEEE Infocom Conf.*, 2003, pp. 1293-1303.
[17] S. Poduri, and G. Sukhatme, "Constrained Coverage for Mobile Sensor Networks", *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA'04)*, 2004.
[18] Y. Zou, and K. Chakrabarty, "Sensor Deployment and Target Localization in Distributed Sensor Networks", *ACM Trans. on Embedded Computing Systems, Special Issue on Networked Embedded Computing: Tools, Architectures and Applications*, Vol. 3, No. 1, pp. 61-91, February 2004.
[19] N. Heo, and P. K. Varshney, "Energy-Efficient Deployment of Intelligent Mobile Sensor Networks", *IEEE Trans. on Systems, Man, and Cybernetics (Part A)*, Vol. 35, No. 1, pp. 78-92, January, 2005.
[20] G. Wang, G. Cao, Th, and F. La Porta, "Movement-Assisted Sensor Deployment", *IEEE Trans. on Mobile Computing*, Vol. 5, No. 6, pp. 640-652, June, 2006.
[21] I. Z. Emiris, Ch. Fragoudakis, and E. Markou, "Maximizing the Guarded Interior of an Art Gallery", 22nd European Workshop on Computational Geometry, Delphi, March, 2006, pp. 165-168.
[22] M. A. L. Thathachar, and 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.
[23] K. S. Narendra, and M. A. L. Thathachar, "Learning Automata: An Introduction", Prentice-Hall Inc, 1989.
[24] M. A. L. Thathachar, and P. S. Sastry, "Networks of Learning Automata", Kluwer Academic Publishers, 2004.
[25] S. Wolfram, "A New Kind of Science", Wolfram Media Inc., 2002.
[26] H. Beigy, and M. R. Meybodi, "A Mathematical Framework for Cellular Learning Automata", *Advances in Complex Systems*, Vol. 7, Nos. 3 & 4, pp. 295-319, September & December 2004.
[27] M. R. Meybodi, H. Beygi, and M. Taherkhani, "Cellular Learning Automata and its Applications", *Journal of Science Tech.*, Sharif (Sharif University of Technology, Tehran, Iran), pp. 54-77, 2004.
[28] H. Beigy, and M. R. Meybodi, "Cellular Learning Automata with Multiple Learning Automata in Each Cell and Its Applications", *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 40, No. 1, pp. 54-66, Feb. 2010.
[29] M. R. Meybodi, and M. Taherkhani, "Application of Cellular Learning Automata in Modeling of Rumor Diffusion", *in Proc. of 9th Conf. on Electrical Engineering*, Power and Water Institute of Technology, Tehran, Iran, pp. 102-110, May 2001.
[30] H. Beigy, and 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.
[31] M. R. Meybodi, and F. Mehdipour, "Application of Cellular Learning Automata with Input to VLSI Placement" , *Journal of Modarres*, University of Tarbeit Modarres, Vol. 16, pp. 81-95, summer 2004.
[32] H. Beigy, and M. R. Meybodi, "Open Synchronous Cellular Learning Automata", *Advances in Complex Systems*, Vol. 10, No. 4, pp. 1-30, December 2007.
[33] H. Beigy, and M. R. Meybodi, "Asynchronous Cellular Learning Automata", *Automatica, Journal of International Federation of Automatic Control*, Vol. 44, No. 5, May 2008, to appear.
[34] M. Esnaashari, and M. R. Meybodi, "A Cellular Learning Automata based Clustering Algorithm for Wireless Sensor Networks", *Sensor Letters*, Vol. 6, No. 5, pp. 723-735, October 2008.
[35] M. Esnaashari, and M. R. Meybodi, "Dynamic Point Coverage Problem in Wireless Sensor Networks: A Cellular Learning Automata Approach", *Journal of Ad hoc and Sensors Wireless Networks*, 2010, to appear.
[36] R. Ghaderi, M. Esnaashari, and M. R. Meybodi, "Maintaining Coverage and Connectivity in Sensor Networks: A Cellular Learning Automata Approach", *Proc. of the 15th Annual CSI Computer Conf. (CSICC'10)*, Tehran, Iran, Feb. 20-22, 2010.
[37] A. Sobeih, W. P. Chen, J. C. Hou, L.C. Kung, N. Li, H. Lim, H.Y. Tyan, and H. Zhang, "J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks", *IEEE Wireless Communications*, Vol. 13, Issue 4, pp. 104-119, August 2006.
[38] B. Wang, H. B. Lim, and D. Ma, "A Survey of Movements Strategies for Improving Network Coverage in Wireless Sensor Networks", *Computer Communications*, Vol. 32, pp. 1427-1436, 2009.
[39] A. Ghosh, "Estimating Coverage Holes and Enhancing Coverage in Mixed Sensor Networks", *IEEE Intl. Conf. on Local Computer Networks*, 2004, pp. 68–76.

[40] G. Wang, G. Cao, P. Berman, and T. F. L. Porta, "Bidding Protocols for Deploying Mobile Sensors", *IEEE Trans. on Mobile Computing*, Vol. 6, No. 5, pp. 515–528, 2007.
[41] P. C. Wang, T. W. Hou, and R. H. Yan, "Maintaining Coverage by Progressive Crystallattice Permutation in Mobile Wireless Sensor Networks", *IEEE Intl. Conf. on Systems and Networks Communication (ICSNC)*, 2006, pp. 1–6.
[42] Y. C. Wang, C. C. Hu, and Y.C. Tseng, "Efficient Placement and Dispatch of Sensors in a Wireless Sensor Network", *IEEE Trans. on Mobile Computing*, Vol. 7, No. 2, pp. 262-274, 2008.
[43] Y. C. Wang, and Y. C. Tseng, "Distributed Deployment Schemes for Mobile Wireless Sensor Networks to Ensure Multi-level Coverage", *IEEE Trans. on Parallel and Distributed Systems*, 2008.
[44] S. Chellappan, X. Bai, B. Ma, D. Xuan, and C. Xu, "Mobility Limited Flip-based Sensor Networks Deployment", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 18, No. 2, pp. 199-211, 2007.
[45] S. Chellappan, W. Gu, X. Bai, D. Xuan, and B. Ma, K. Zhang, "Deploying Wireless Sensor Networks under Limited Mobility Constraints", *IEEE Trans. on Mobile Computing*, Vol. 6, No. 10, pp. 1142–1157, 2007.
[46] Z. Jiang, J. Wu, R. Kline, and J. Krantz, "Mobility Control for Complete Coverage in Wireless Sensor Networks", *28th Intl. Conf. on Distributed Computing Systems Workshops (ICDCS)*, 2008, pp. 291–296.
[47] G. Wang, G. Cao, T.L. Porta, and W. Zhang, "Sensor Relocation in Mobile Sensor Networks", *IEEE INFOCOM*, 2005, pp. 2302–2312.
[48] S. Yang, M. Li, and J. Wu, "Scan-based Movement-assisted Sensor Deployment Methods in Wireless Sensor Networks", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 18, No. 8, pp. 1108–1121, 2007.
[49] J. Wu, and S. Yang, "Optimal Movement-assisted Sensor Deployment and Its Extensions in Wireless Sensor Networks", *Simulation Modeling Practice and Theory*, Vol. 15, No. 4, pp. 383–399, 2007.
[50] S. Yang, J. Wu, and F. Dai, "Localized Movement-assisted Sensor Deployment in Wireless Sensor Networks", *IEEE Workshops in the Interl. Conf. on Mobile Adhoc and Sensor Systems (MASS)*, 2006, pp. 753–758.
[51] Y. C. Wang, W. C. Peng, M. H. Chang, and Y. C. Tseng, "Exploring Load-balance to Dispatch Mobile Sensors in Wireless Sensor Networks", *16th IEEE Intl. Conf. on Computer Communications and Networks (ICCCN)*, 2007, pp. 669–674.
[52] Z. Butler, and D. Rus, "Event-based Motion Control for Mobile Sensor Networks", *IEEE Pervasive Computing Magazine*, Vol. 2, No. 4, pp. 34–42, 2003.
[53] B. Crowell, "Newtonian Physics", Light and Matter, 2004.
[54] Y. Zhou, J. Schembri, L. Lamont, and J. Bird, "Analysis of Stand-Alone GPS for Relative Location Discovery in Wireless Sensor Network", *Canadian Conf. on Electrical and Computer Engineering*, Newfoundland, Canada, 2009, pp. 437-441.
[55] H. Lee, H. Dong, H. Aghajan, "Robot-Assisted Localization Techniques for Wireless Image Sensor Networks", *IEEE Intl. Conf. on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON)*, September, 2006, pp. 383-392.
[56] V. Ramadurai, and M. L. Sichitiu, "Localization in Wireless Sensor Networks: A Probabilistic Approach", *Intl. Conf. on Wireless Networks*, 2003, pp. 275-281.
[57] C. E. Shannon, "A Mathematical Theory of Communication", *Bell System Technical Journal*, Vol. 27, pp. 379-423, 623-656, July, October, 1948.
[58] T. N. Yap, and Ch. R. Shelton, "Simultaneous Learning of Motion and Sensor Model Parameters for Mobile Robots", *in Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2008, pp. 2091-2097.