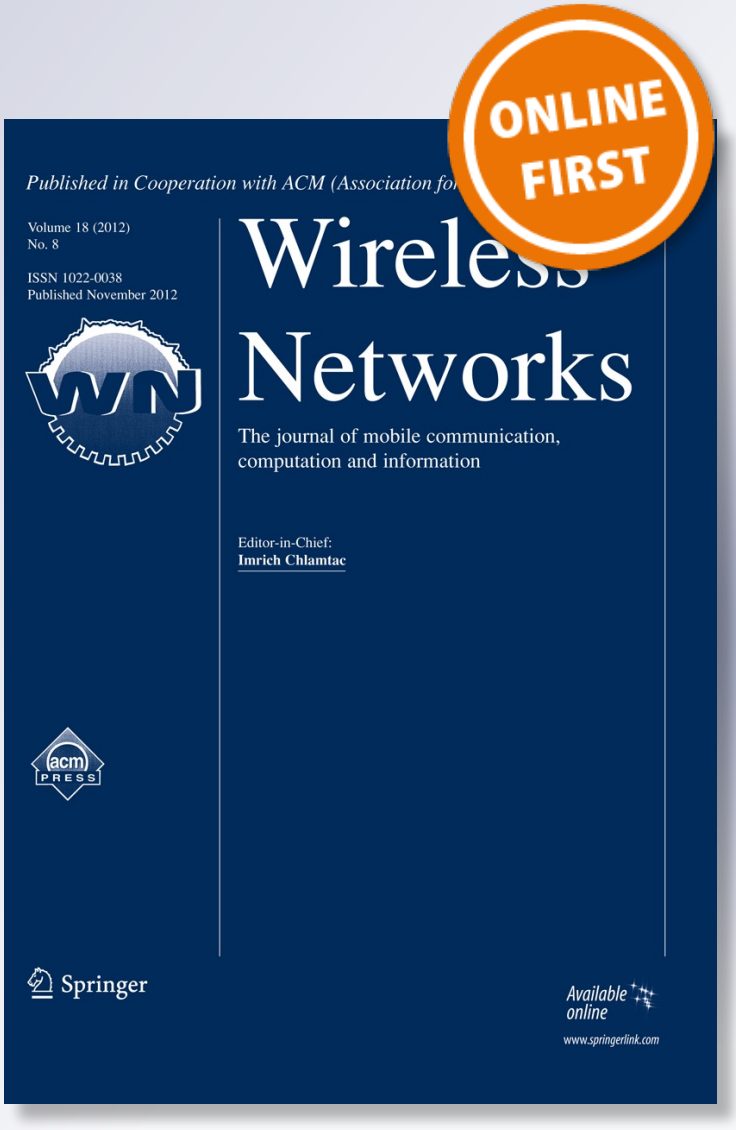
Deployment of a mobile wireless sensor network with k-coverage constraint: a cellular learning automata approach

M. Esnaashari & M. R. Meybodi

Wireless Networks
The Journal of Mobile Communication, Computation and Information

ISSN 1022-0038

Wireless Netw
DOI 10.1007/s11276-012-0511-7



Springer

Wireless Netw
DOI 10.1007/s11276-012-0511-7

Deployment of a mobile wireless sensor network with *k*-coverage constraint: a cellular learning automata approach

M. Esnaashari · M. R. Meybodi

Abstract Deployment of a wireless sensor network is a challenging problem, especially when the environment of the network does not allow either of the random deployment or the exact placement of sensor nodes. If sensor nodes are mobile, then one approach to overcome this problem is to first deploy sensor nodes randomly in some initial region within the area of the network, and then let the sensor nodes to move around and cooperatively and gradually increase the covered section of the area. Recently, a cellular learning automata-based deployment strategy, called CLA-DS, is introduced in literature which follows this approach and is robust against inaccuracies which may occur in the measurements of sensor positions or in the movements of sensor nodes. Despite its advantages, this deployment strategy covers every point within the area of the network with only one sensor node, which is not enough for applications with *k*-coverage requirement. In this paper, we extend CLA-DS so that it can address the *k*-coverage requirement. This extension, referred to as CLA-EDS, is also able to address *k*-coverage requirement with different values of *k* in different regions of the network area. Experimental results have shown that the proposed deployment strategy, in addition to the advantages it inherits from CLA-DS, outperforms existing algorithms such as DSSA, IDCA, and DSLE in covering the network area, especially when required degree of coverage differs in different regions of the network.

Keywords Mobile sensor network · Cellular learning automata · Self-regulated deployment · *k*-Coverage

1 Introduction

The very first step in constructing a wireless sensor network is to deploy sensor nodes throughout the targeted environment [1]. The main objective of the sensor deployment is to achieve desirable coverage of the network area. Sensor deployment strategies can be classified into the following four categories [2, 3]:

- Predetermined: This strategy is useful only if the network environment is completely known [2, 4–8].
- Randomly undetermined: In this strategy, sensor nodes are spread uniformly throughout the network area [3, 9–13].
- 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 region within the area of the network. After this initial placement, sensor nodes move around and cooperatively and gradually find their best positions within the area of the network [15–28].

In some contexts, neither predetermined nor random deployment of sensor nodes is feasible. This can be due to inaccessibility or hazardousness of the environment, cost-inefficiency of the random scattering of sensor nodes,

M. Esnaashari (✉) · M. R. Meybodi
Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran
e-mail: esnaashari@aut.ac.ir

M. R. Meybodi
e-mail: mmeybodi@aut.ac.ir

M. R. Meybodi
Institute for Studies in Theoretical Physics and Mathematics(IPM), School of Computer Science, Tehran, Iran

Published online: 05 November 2012

uncertainty about the resultant coverage, etc. Self-regulated deployment strategies on the other hand are well suited to such situations.

Recently, a cellular learning automata-based self-regulated deployment strategy, called CLA-DS, has been introduced [29], which is robust against inaccuracies which may occur in the measurements of sensor positions or in the movements of the sensor nodes. The novelty of this strategy is that it works without any sensor to know its position or its relative distance to other sensors. Despite its advantages, CLA-DS covers every point within the area of the network with only one sensor node, which is not enough for applications with k-coverage requirement such as intrusion detection [30, 31], data gathering [32, 33], and object tracking [34]. In such applications, it is required for every point within the area of the network to be covered by at least k different sensor nodes. k is referred to as the degree of coverage. To make CLA-DS capable of addressing the k-coverage requirement, in this paper we propose an extension to this deployment strategy, called CLA-EDS, which is able to provide the k-coverage of the entire area of the network. The key novelties and differences of this extended algorithm versus its basic algorithm, CLA-DS, are as follows:

- CLA-EDS algorithm is capable of providing k-coverage of the entire area of the network.
- This extended algorithm is also able to address k-coverage requirement with different values of k in different regions of the network area.
- Unlike CLA-DS which is only capable of deploying sensor nodes uniformly throughout the area of the network, CLA-EDS algorithm is able to deploy sensor nodes either uniformly or non-uniformly.
- If the value of k in the environment of the network changes from time to time, CLA-EDS algorithm is capable of adapting itself to such changes and providing the requested degree of coverage at any time.

Like CLA-DS, in CLA-EDS neighboring nodes apply forces to each other. Then, each node moves according to the resultant force vector applied to it from its neighbors. 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 fulfill the required degree of the coverage. The learning model used in CLA-DS is enhanced in CLA-EDS so that CLA-EDS is able to provide different degrees of coverage in different regions of the network.

To study the performance of the proposed deployment strategy, several experiments have been conducted and the results obtained from CLA-EDS are compared with the results obtained from existing self-regulated deployment strategies, capable of providing k-coverage, such as DSSA, IDCA, and DSLE. Experimental results show that, in terms of the network coverage, the proposed CLA-EDS strategy, like CLA-DS deployment strategy, can compete with the existing deployment strategies in noise-free environments, and outperforms existing deployment strategies 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. Results of the experiments also show that when the required degree of coverage differs in different regions of the network, CLA-EDS deployment strategy significantly outperforms DSSA, IDCA, DSLE, and CLA-DS algorithms with respect to the network coverage.

The rest of this paper is organized as follows. Section 2, gives a brief literature overview on the self-regulated deployment strategies and k-coverage providing algorithms in wireless sensor networks. Cellular learning automata will be discussed in Sect. 3. The problem statement is given in Sect. 4. In Sect. 5 the proposed deployment strategy is presented. Simulation results are given in Sect. 6. Section 7 is the conclusion.

2 Related work

In the following two subsections, we give a brief literature overview on the self-regulated deployment strategies and k-coverage providing algorithms in wireless sensor networks, respectively.

2.1 Self-regulated deployment strategies

Wang et al. in [35] groups the existing self-regulated deployment strategies for mobile sensor networks into the following three categories: coverage pattern based movement, virtual force based movement, and grid quorum based movement. Strategies in the coverage pattern based movement group [21, 36, 37] 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 [21], 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 [36] 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 [37] to provide *k*-coverage of the environment.

In the group of virtual force based node movement strategies [15–20], sensor nodes apply some sort 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 deployment strategy given in [15], the deployment strategy 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 two different distributed deployment strategies called DSSA, IDCA. 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. 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 strategies [22–28], 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.

2.2 *k*-Coverage providing algorithms

Notwithstanding the centralized approaches [38–43] in providing *k*-coverage in wireless sensor networks, most of the researches in this area are based on some sort of sleep/wake scheduling of sensor nodes [44–55]. Slijepcevic et al. in [44] introduced the "set *k*-cover" problem, which is to organize mutually exclusive sensor nodes into a number of covers or sets each of which can fully cover the network area. The goal in this problem is to maximize the number of covers. They first proved that this problem is NP complete and then proposed a heuristic algorithm with time complexity $O(N^2)$ for solving it. Makhoul et al. in [45] gave a fully distributed algorithm for providing *k* (not necessarily disjoint) cover sets for monitoring the entire area of a video wireless sensor network. In [46], a greedy centralized algorithm is given for constructing as many disjoint *k*-cover sets as possible from the set of deployed sensor nodes. This greedy algorithm makes use of the results reported in [47] to check if a set of sensor nodes *k*-cover the entire area. [48] provides a sufficiently and necessary rule which determines whether a node is eligible to sleep (is a redundant node) for the purpose of *k*-coverage or not. Using this rule, they propose an algorithm called CCP to schedule the work state of eligible nodes. [49] Proposes CGS algorithm, which makes use of the notion of bipartite graphs for selecting a subset of sensor nodes as active nodes so that every region of the network area is covered by at least *k* active sensor node. Performance of the CGS algorithm is analyzed in [50]. [51] Proposes two greedy algorithms (one centralized and one distributed) for selecting a subset of sensors from the set of deployed sensors so that the selected subset *k*-cover the entire area of the network. In [52], authors gave a polynomial time, distributed scheduling algorithm for maximizing the lifetime of the network. Lifetime is defined as the time from the network startup until the time at which the set of all sensors with non-zero remaining energy does not provide *k*–coverage of the entire area. In [53] authors derived, under the ideal case in which node density is sufficiently high, a set of optimality conditions under which a subset of active sensor nodes can be chosen for complete coverage. Based on the optimality conditions, they then devised a distributed algorithm, called OGDC, which can maintain coverage as well as connectivity. They also discuss how their algorithm can be extended to ensure *k*-coverage. Huang et al. in [54] proposed a decentralized algorithm that schedules sensors' active and sleeping periods to prolong the network lifetime while maintain the sensing field sufficiently covered. Fusco and Gupta in [55] defined a *d*-sensor as a sensor associated with multiple sensing regions, out of which only one is active, depending on the orientation assigned to the sensor. The problem considered is as follows: Given a set of *d*-sensors with fixed positions and a set of target points, select the minimum number of *d*-sensors and their assigned orientations, such that each target point is covered by at least *k* of the selected *d*-sensors. Authors designed a distributed greedy algorithm that *k*-covers at least half of the target points.

Another approach in providing *k*-coverage in wireless sensor networks is to use the concept of ε-net sets [56–58]. [56, 57] Use the concept of ε-net sets to give an efficient approximation algorithm for *k*-coverage problem which achieves a solution of the size within a logarithmic factor of the optimal. Authors proved that their algorithm is correct and analyzed its complexity. A fully distributed version of their algorithm is also presented. In [58], authors claimed that the results reported in [56, 57] is fundamentally flawed. Therefore, they tried to give a correct

extension of the ε-net technique for *k*-coverage problem. Their algorithm gives a $O(\log(M))$-approximation, where M is the number of sensors in an optimal solution. A fully distributed version of their algorithm is also presented.

Some other algorithms are also given in the literature for providing *k*-coverage. Ammari et al. [59, 60] analyzed the *k*-coverage problem in 3D wireless sensor networks and showed that the extension of the analysis in 2D space to 3D space is not straightforward due to the inherent characteristics of the Reuleaux tetrahedron. Therefore they proposed a new model that guarantees *k*-coverage for a 3D field. A distributed *k*-coverage protocol for 3D wireless sensor networks is also presented. In [61], authors claimed that *k*-coverage of the area can be obtained by first covering the entire area using a hexagonal grid pattern and then increasing the density of the network by decreasing the diameters of the hexagons gradually until the desired coverage level is obtained. In [62], authors proposed a mobility resilient coverage control mechanism to assure *k*-coverage in the presence of mobility. In [63], authors introduced a different version of *k*-coverage problem in which, by assuming that a number of sensor nodes are initially deployed within the network area, the goal is to add additional M sensor nodes so that all points within the network area are *k*-covered. They propose a heuristic approach based on a dynamic model resembling gas bubbles to solve this problem. Sh. Tang et al. in [64] introduced the problem of *k*-support coverage path in which, given a pair of points (S, D), it is required to find a path between S and D which is covered by at least k sensor nodes. They proposed an optimal polynomial time algorithm for this problem and proved that the time complexity of the proposed algorithm is $O(k^2n^2)$. In [65], a new concept of coverage called *k*-barrier coverage that is appropriate for intrusion detection applications is proposed. A sensor network provides *k*-barrier coverage if it guarantees that every penetrating object is detected by at least k sensors before crossing the barrier of the sensors. Ssu et al. in [66] by proposing a distributed algorithm for *k*-barrier coverage, argued that using directional antenna instead of omni-directional antenna in this problem results in fewer number of active nodes and more robust ability in constructing *k*-barrier coverage.

In [67] a generalized technique to extend any 1-coverage technique to a *k*-coverage technique is proposed. The general concept of this technique involves separating all sensors into k mutually exclusive groups. Each group uses 1-coverage algorithm to optimize its sensing range or chooses its sleep/wakeup schedule. Then, by layering the k groups, *k*-coverage can be achieved.

A number of theoretical studies are also given in the literature of the *k*-coverage problem [32, 47, 68–74].

Few attempts have been made to provide *k*-coverage during the deployment of the sensor network. Bai et al. in [75] proposed a centralized PSO-based algorithm which attempts to deploy a number of mobile sensor nodes throughout the network area so that the *k*-coverage requirement is satisfied. Li and Kao [76] proposed a distributed deployment strategy which aims at providing *k*-coverage of the network area. They first derived the minimum number of sensor nodes required to achieve *k*-coverage by modeling the sensing field using Voronoi diagrams. Using the results of this study, each sensor node is capable of locally determining the location it should move towards to ensure *k*-coverage. The proposed algorithm, called DSLE, makes use of this capability of sensor nodes to ensure *k*-coverage over the entire network area.

3 Heterogeneous dynamic irregular CLA (*HDICLA*)

In this section we briefly review learning automaton (*LA*), cellular learning automaton (*CLA*), irregular *CLA* (*ICLA*), dynamic irregular *CLA* (*DICLA*), and then introduce heterogeneous dynamic irregular *CLA* (*HDICLA*).

3.1 Learning automata

Learning Automaton (*LA*) is an adaptive decision-making device that operates on an unknown random environment. 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 $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the action set of the automata, $\beta = \{\beta_1, \beta_2, \ldots, \beta_r\}$ represents the input set, $p = \{p_1, p_2, \ldots, p_r\}$ 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

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

for favorable responses, and

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

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 [77–79].

3.2 Cellular learning automata

Cellular learning automaton (*CLA*), which is a combination of cellular automaton (*CA*) [80] and learning automaton (*LA*), is a powerful mathematical model for many decentralized problems and phenomena. The basic idea of the *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 the *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 [81], rumor diffusion [82], channel assignment in cellular networks [83] and VLSI placement [84], to mention a few. For more information about *CLA* the reader may refer to [85–88].

3.3 Irregular CLA

An Irregular cellular learning automaton (*ICLA*) is a cellular learning automaton (*CLA*) in which the restriction of the 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*. *ICLA* has found a number of applications in wireless ad hoc and sensor networks [89–91].

3.4 Dynamic irregular CLA

Dynamic irregular cellular learning automaton (*DICLA*) has been recently introduced in [29]. *DICLA* is an *ICLA* with dynamic structure. In other words, in *DICLA* the adjacency matrix of the underlying graph of the *ICLA* can be changed over time. This dynamicity is required in many applications such as mobile ad hoc and sensor networks, web mining, grid computing, etc.

3.5 Heterogeneous dynamic irregular CLA

We define Heterogeneous *DICLA* (*HDICLA*) 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 and a finite set of attributes are defined for the *HDICLA*. In *HDICLA*, the state of each cell consists of the following three parts:

- *Selected action*: The action selected by the learning automaton residing in the cell.
- *Tendency vector*: Tendency vector of the cell is a vector whose jth element shows the degree of tendency of the cell to the jth interest.
- *Attribute vector*: Attribute vector of the cell is a vector whose jth element shows the value of the jth attribute within the cell's locality.

Two cells are neighbors in *HDICLA* if the distance between their tendency vectors is smaller than or equal to the neighborhood radius.

Like *CLA*, there is a local rule that *HDICLA* operates under. The rule of *HDICLA*, the actions selected by the neighboring learning automata of any particular learning automaton LA_i, and the attribute vector of the cell in which LA_i resides (c_i) determine the followings: 1. The reinforcement signal to the learning automaton LA_i and 2. The restructuring signal to the cell c_i, which is used to update the tendency vector of the cell. Figure 1 gives a schematic of *HDICLA*. An *HDICLA* is formally defined below.

Definition 1 A heterogeneous dynamic irregular cellular learning automaton (*HDICLA*) is a structure $A = (G<E, V>, \Psi, \Lambda, A, \Phi<\alpha, \underline{\psi}, \underline{\lambda}>, \tau, F, Z)$ 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 *HDICLA*.
2. Ψ is a finite set of interests. Cardinality of Ψ is denoted by $|\Psi|$.
3. Λ is a finite set of attributes. Cardinality of Λ is denoted by $|\Lambda|$.
4. A is the set of learning automata each of which is assigned to one cell of the *HDICLA*.
5. $\Phi\left\langle \alpha, \underline{\psi}, \underline{\lambda} \right\rangle$ is the cell state. State of a cell c_i (Φ_i) consists of the following three parts:

   5-1 α_i which is the action selected by the learning automaton of that cell.

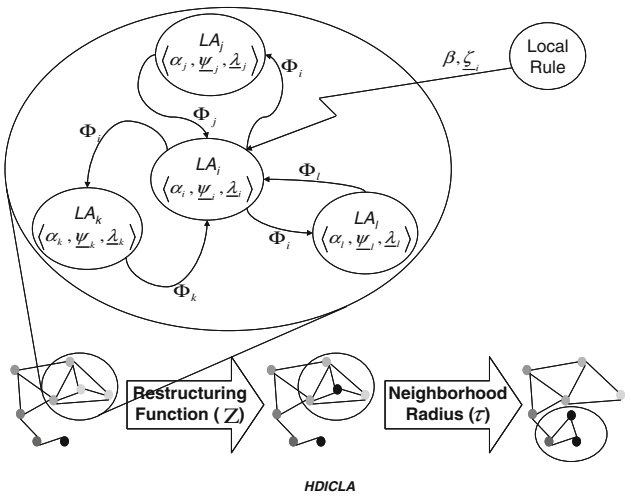


Fig. 1 Heterogeneous dynamic irregular cellular learning automaton (*HDICLA*)

5-2 A vector $\underline{\psi}_i = (\psi_{i1}, \ldots, \psi_{i|\Psi|})^T$ which is called tendency vector of the cell. Each element $\psi_{ik} \in [0, 1]$ in the tendency vector of the cell c_i shows the degree of tendency of c_i to the interest $\psi_{ik} \in \Psi$.

5-3 A vector $\underline{\lambda}_i = (\lambda_{i1}, \ldots, \lambda_{i|\Lambda|})^T$ which is called attribute vector of the cell. Each element $\lambda_{ik} \in \mathbb{R}$ in the attribute vector of the cell c_i shows the value of the attribute $\lambda_k \in \Lambda$ in the locality of the cell c_i.

6. τ is the neighborhood radius. Two cells c_i and c_j of the *HDICLA* are neighbors if $\left\|\underline{\psi}_i - \underline{\psi}_j\right\| \leq \tau$. In other words, two cells of the *HDICLA* are neighbors if the distance between their tendency vectors is smaller than or equal to τ.
7. $F : \underline{\Phi}_i \rightarrow \left\langle \underline{\beta}, [0, 1]^{|\Psi|} \right\rangle$ is the local rule of *HDICLA* in each cell c_i, where $\underline{\Phi}_i = \left\{ \Phi_j \middle| \left\|\underline{\psi}_i - \underline{\psi}_j\right\| \leq \tau \right\} + \{\Phi_i\}$ 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 ($\underline{\zeta}_i = (\zeta_{i1}, \ldots, \zeta_{i|\Psi|})^T$) which is used to change the tendency vector of c_i. Each element ζ_{ij} of the restructuring signal is a scalar value within the close interval [0, 1].
8. $Z : [0, 1]^{|\Psi|} \times [0, 1]^{|\Psi|} \rightarrow [0, 1]^{|\Psi|}$ 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 *HDICLA* with N cells. The learning automaton LA_i which has a finite action set $\underline{\alpha}_i$ is associated to cell c_i (for $i = 1, 2, \ldots, N$) of *HDICLA*. Let the cardinality of $\underline{\alpha}_i$ be m_i. The state of the *HDICLA* is represented by the triple $\left\langle \underline{p}, \underline{\psi}, \underline{\lambda} \right\rangle$, where

- $\underline{p} = (\underline{p}_1, \ldots, \underline{p}_N)^T$, where $\underline{p}_i = (p_{i1}, p_{i2}, \ldots, p_{im_i})^T$ is the action probability vector of LA_i.
- $\underline{\psi} = (\underline{\psi}_1, \ldots, \underline{\psi}_N)^T$, where ψ_i is the tendency vector of the cell c_i.
- $\underline{\lambda} = (\underline{\lambda}_1, \ldots, \underline{\lambda}_N)^T$, where λ_i is the attribute vector of the cell c_i.

The operation of the *HDICLA* 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 $F(\underline{\Phi}_i) \rightarrow \left\langle \underline{\beta}, [0, 1]^{|\Psi|} \right\rangle$. 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 (Eq. (3)).

$$\underline{\psi}_i(n+1) = Z\left(\underline{\psi}_i(n), \underline{\zeta}_i(n)\right). \tag{3}$$

After the application of the restructuring function, the attribute vector of the cell c_i may change due to the modifications made to its local environment.

An *HDICLA* 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 *HDICLA*, each cell is assumed to have an internal clock which wakes up the *LA* associated to that cell while in step-driven asynchronous *HDICLA* a cell is selected in fixed or random sequence.

3.5.1 HDICLA norms of behavior

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

restructurings of the *HDICLA* we use restructuring tendency.

3.5.1.1 Entropy Entropy, as introduced in the context of information theory by Shannon [92], is a measure of uncertainty associated with a random variable and is defined according to Eq. (4),

$$H(X) = -\sum_{X\in\chi} P(X)\cdot \ln(P(X)), \tag{4}$$

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 Eq. (5),

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

where m_i is the cardinality of the action set of the learning automaton LA_i. In the learning process, $H_i(n)$ represents the uncertainty associated with the decision of LA_i at time instant n. Larger values of $H_i(n)$ 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 the *HDICLA* in the environment is a macroscopic view of the operations of all of its constituting learning automata, we extend the concept of entropy through Eq. (6) in order to provide a metric for evaluating the uncertainty associated with the operation of a *HDICLA*.

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

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

3.5.1.2 Degree of expediency To introduce the "degree of expediency" measure, we have to first define the concept of expediency for the *HDICLA*.

Definition 2 The total average penalty received by an *HDICLA* at state $\left\langle \underline{p}, \underline{\psi}, \underline{\lambda} \right\rangle$ is the sum of the average penalties received by all of its learning automata, that is,

$$D(\underline{p}) = \sum_{i} D_i(\underline{p}), \tag{7}$$

where $D_i(\underline{p})$ is the average penalty received by the learning automaton LA_i.

Definition 3 A pure-chance automaton is an automaton that chooses each of its actions with equal probability i.e., by pure chance.

Definition 4 A pure-chance *HDICLA* is an *HDICLA* in which, every cell contains a pure-chance automaton instead of a learning automaton. The state of a pure-chance *HDICLA* is denoted by $\left\langle \underline{p}^0, \underline{\psi}, \underline{\lambda} \right\rangle$.

Definition 5 An *HDICLA* is said to be expedient if

$$\lim_{n\to\infty} \mathrm{E}\left[D(\underline{p}(n))\right] < D(\underline{p}^0). \tag{8}$$

In other words, an *HDICLA* is expedient if it performs better than a pure-chance *HDICLA*.

Degree of expediency (M_d), defined according to Eq. (9), is a measure for comparing the average penalty received by an *HDICLA* with the average penalty received by a pure-chance *HDICLA*. $M_d < 0$ indicates that the *HDICLA* performs worse than a pure-chance *HDICLA*, $M_d = 0$ indicates that the *HDICLA* is pure-chance, and $M_d > 0$ indicates that the *HDICLA* is expedient (performs better than a pure-chance *HDICLA*). Higher value of M_d means that the *HDICLA* is more expedient. M_d attains its maximum value ($M_d = 1$) if the *HDICLA* receives no penalty ($D(\underline{p}(n)) = 0$).

$$M_d = 1 - \left(\frac{\lim_{k\to\infty} \mathrm{E}\left[D\left(\underline{p}(k)\right)\right]}{D\left(\underline{p}^0\right)} \right). \tag{9}$$

3.5.1.3 Restructuring tendency Restructuring tendency (υ), as defined by Eq. (10), is used to measure the dynamicity of the structure of the *HDICLA*.

$$\upsilon(n) = \sum_{i=1}^{N} |\zeta_i(n)|. \tag{10}$$

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

4 Problem statement

Consider N mobile sensor nodes s_1, s_2, …, s_N with equal sensing ($R_s = r$) and transmission ranges ($R_t = 2.r$) which are initially deployed in some initial region within an

unknown 2 dimensional environment Ω. Assume that a rough estimate of the surface of Ω ($\widehat{S}_{\Omega}$) is available (using Google maps for example). Sensor nodes are able to move along 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.

Definition 6 Sensing region of a sensor node s_i denoted by $C(s_i)$ is a circle with radius R_s centered on s_i.

Definition 7 Coverage function $C_{x_l,y_l}(s_i)$ is defined according to the following equation:

$$C_{x_l,y_l}(s_i) = \begin{cases} 1; & (x_l, y_l) \in C(s_i) \\ 0; & otherwise \end{cases}. \quad (11)$$

In other words, $C_{x_l,y_l}(s_i)$=1, if (x_l, y_l) is within the sensing region of the sensor node s_i.

We assume that different regions within the network area require different degrees of coverage. Let Ω_1, Ω_2, …, Ω_M be M regions within the Ω with the following properties:

$$\Omega_i \cap \Omega_j = \emptyset; \quad for \quad i \neq j$$

$$\bigcup_{i=1}^{M} \Omega_i = \Omega$$

Let $\widehat{S}_{\Omega_i}$ and $rdc(\Omega_i)$ be the estimated surface and the required degree of coverage of the ith region respectively. It is straight forward that the required degree of coverage of every point (x_l, y_l)$\in\Omega_i$ is equal to $rdc(\Omega_i)$, that is, $(x_l, y_l) \in \Omega_i \Rightarrow rdc(x_l, y_l) = rdc(\Omega_i)$. We assume that there exists a notification-based mechanism in the network (using local base stations for example) which notifies the values of $\widehat{S}_{\Omega_i}$and $rdc(\Omega_i)$ in each region Ω_i to the sensor nodes within that region.

Consider the following definitions.

Definition 8 Covered sub-area denoted by $C(\Omega)$ refers to the set of the points (x_l, y_l) within the network area, each is covered with at least $rdc(x_l, y_l)$ sensor nodes. $C(\Omega)$ is stated through Eq. (12) given below.

$$C(\Omega) = \bigcup_{i=1}^{M} \left(\left\{ (x_l, y_l) \middle| (x_l, y_l) \in \Omega_i, \sum_{j=1}^{N} C_{x_l,y_l}(s_j) \geq rdc(\Omega_i) \right\} \right). \quad (12)$$

Definition 9 Covered section denoted by $S_{C(\Omega)}$is the surface of the covered sub-area.

Definition 10 Deployment strategy is an algorithm which gives for any given sensor node s_i a certain position within the area of the network.

Definition 11 Deployed network refers to a sensor network which results from a deployment strategy.

Definition 12 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 13 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 $\widehat{S}_{\Omega}$ so that the covered section of Ω ($S_{C(\Omega)}$) is maximized and the deployed network is connected.

5 CLA-EDS: an extension to CLA-DS deployment strategy

In this section, we first give a short description of CLA-DS deployment strategy and then propose CLA-EDS deployment strategy.

5.1 CLA-DS

CLA-DS deployment strategy consists of the following 4 major phases:

- *Initial deployment*: During the initialization phase, sensor nodes are initially deployed in some initial region within the area of the network.
- *Mapping phase*: The network topology is mapped into a *DICLA* model.
- *Deployment phase*: Deploying sensor nodes throughout the area of the network is performed during the deployment phase. Deployment phase for each sensor node s_i is divided into a number of rounds; each is started by the asynchronous activation of the cell c_i of the *DICLA*. Upon the startup of the nth round in the cell c_i, LA_i randomly selects one of its actions ("apply force to neighboring nodes", or "do not apply force to neighboring nodes"). This selection is then broadcasted within the neighborhood of the node s_i. Sensor node s_i then waits for certain duration to receive the selected actions of its neighboring nodes. When this duration is over, sensor node s_i collects following statistics from the received information: number of received packets and number of neighbors selecting "apply force to neighboring nodes" action. 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. Next, sensor node s_i uses vector $\underline{\zeta}_i(n)$ as its movement path. In other words, if s_i is located at $(x_i(n), y_i(n))$, it moves to $(x_i(n+1), y_i(n+1)) = (x_i(n) + \zeta_{1,i}, y_i(n) + \zeta_{2,i})$. Afterward, sensor node s_i waits for the next activation time of its corresponding cell c_i in the *DICLA* to start its next round. 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.
- *Oscillation*: Sensor node s_i oscillates between almost the same positions.

- *Maintenance phase*: Deployed network is maintained in order to compensate the effects of possibly node failures on the covered sub-area ($C(\Omega)$).

5.2 CLA-EDS

Like CLA-DS, CLA-EDS deployment strategy also consists of *initial deployment*, *mapping*, *deployment*, and *maintenance* phases. We explain these 4 phases in more details in the subsequent sections.

5.2.1 Initial deployment

Like CLA-DS, initial deployment of the sensor nodes in CLA-EDS deployment strategy can be done using any of the following strategies:

- *Random deployment*: In random deployment strategy, a random based deployment strategy [3, 9–14] is used to deploy sensor nodes uniformly at random throughout the network area (Ω).
- *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.

5.2.2 Mapping

The second phase of CLA-EDS deployment strategy involves the creation of a time-driven asynchronous *HDICLA*, which is isomorphic to the sensor network topology. In this *HDICLA*, any cell c_i corresponds to the sensor node s_i located at (x_i, y_i) in the sensor network.

The set of interests of the *HDICLA* has two members; X-axis and Y-axis of the network area. Tendency levels of each cell c_i to these two interests are initially set to $\psi_{i1}(0) = \frac{x_i(0)}{\mathrm{Max}(MaxX, MaxY)}$ and $\psi_{i2}(0) = \frac{y_i(0)}{\mathrm{Max}(MaxX, MaxY)}$ respectively. (*MaxX*, *MaxY*) refers to the farthest location within the network area at which a sensor node can be located.

Attribute set of the *HDICLA* has two member; *required degree of coverage* (*rdc*), and estimated surface ($\widehat{S}$) of the region requires this *rdc*. As it was mentioned before, the values of these attributes differ in different regions of the network and there exists a notification-based mechanism which notifies $rdc(\Omega_j)$ and $\widehat{S}_{\Omega_j}$ of each region Ω_j to the sensor nodes within that region. According to these assumptions, the local values of the *rdc* (rdc_i) and $\widehat{S}(\widehat{S}_i)$ attributes are known to every cell c_i of the *HDICLA*. Initially, $rdc_i(0)$ and $\widehat{S}_i(0)$ are set to 1 and $\widehat{S}_\Omega$ (the rough estimate of the surface of the network area) respectively.

The neighborhood radius (τ) of the *HDICLA* is set to $\frac{R_t}{\mathrm{Max}(MaxX, MaxY)}$. This means that two cells c_i and c_j in the *HDICLA* are adjacent to each other if their corresponding nodes s_i and s_j in the sensor network are within the transmission ranges of each other.

It should be noted that the values of $\psi_{i1}(0)$ and $\psi_{i2}(0)$ cannot be computed here due to the fact that the sensor nodes are not aware of their physical positions. As a consequence, one may think that the adjacent cells of a cell cannot be specified for the *HDICLA*. But this is not true due to the fact that the neighboring cells of each cell are implicitly specified here 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).

Each cell c_i of the *HDICLA* is equipped with a learning automaton LA_i with two actions; α_0 and α_1. Action α_0 is "apply force to the neighboring nodes" and action α_1 is "do not apply force to the neighboring nodes". The probability of selecting each of these actions is initially set to .5.

5.2.3 Deployment

In CLA-EDS, like CLA-DS, each sensor node can be in one of the following two states: *mobile* and *fixed*. The initial state of each sensor node is selected randomly with probability of selecting *fixed* state to be P_{fix}. P_{fix} is a constant which is known to all sensor nodes. The deployment phase is executed only on *mobile* sensor nodes.

Deployment phase of a "*mobile*" sensor node s_i consists of a number of rounds $R_i(0)$, $R_i(1)$,…. A new round $R_i(n)$ is

started by the asynchronous activation of the cell c_i of the *HDICLA*, which occurs at time $\delta_i + n \times ROUND\_DURATION$. Here, δ_i is a random delay, generated for cell c_i and is used for reducing the probability of collisions between neighboring nodes. *ROUND_DURATION* is an upper bound for the duration of a single round.

Following steps are performed during the execution of each round $R_i(n)$:

- LA_i selects one of its actions randomly according to its action probability vector. Selected action can be either of "apply force to the neighboring nodes", or "do not apply force to the neighboring nodes"
- Sensor node s_i creates an *APPLIED_FORCE* packet which includes the state of the node and the selected action of LA_i. This packet is then broadcasted in the neighborhood of s_i.
- Sensor node s_i starts collecting *APPLIED_FORCE* packets, sent from its neighboring nodes. This step lasts for certain duration (*COLLECTING_DURATION*). Collected packets are stored into a local database within the node.
- Sensor node s_i collects the 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 the neighboring nodes" action ($N_i^f(n)$).
- Considering the collected statistics, the local rule of the cell c_i is applied and the reinforcement signal $\beta_i(n)$ and the restructuring signal $\underline{\zeta}_i(n)$ are generated. Details on this step will be given in Sect. 5.2.3.1.
- Using the generated reinforcement signal $\beta_i(n)$, the selected action of LA_i is rewarded or penalized using Eqs. (1) or (2).
- 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 $(x_i(n), y_i(n))$, then it moves to $(x_i(n+1), y_i(n+1)) = (x_i(n) + \zeta_{i1}, y_i(n) + \zeta_{i2})$.
- Tendency levels of the cell c_i are updated by the application of the restructuring function Z. This is performed using Eq. (13) given below.
-

$$\begin{cases} \psi_{i1}(n+1) = \mathrm{Z}(\psi_{i1}(n), \zeta_{i1}(n)) = \psi_{i1}(n) + \dfrac{\zeta_{i1}(n)}{\mathrm{Max}(MaxX, MaxY)} \\ \psi_{i2}(n+1) = \mathrm{Z}(\psi_{i2}(n), \zeta_{i2}(n)) = \psi_{i2}(n) + \dfrac{\zeta_{i2}(n)}{\mathrm{Max}(MaxX, MaxY)} \end{cases}. \tag{13}$$

- Sensor node s_i waits for certain duration (*WAIT_FOR_LOCAL_INFO_DURATION*) to receive a packet which notifies the local values of rdc_i and $\widehat{S}_i$ in its new position. If such a notification packet is received within this duration, the local values of rdc_i and $\widehat{S}_i$ are updated accordingly.
- Sensor node s_i waits for the next activation time of its corresponding cell c_i in the *HDICLA*.

The above steps are performed iteratively until one of the following conditions occurs:

Stability: During consequent N_r rounds, sensor node s_i does not move more than *LEAST_DISTANCE*.

Oscillation: Sensor node s_i oscillates between two positions for more than N_o rounds.

On the occurrence of one of the above conditions, sensor node s_i sets its state to "*fixed*" and starts with the maintenance phase.

5.2.3.1 Applying local rule The local rule of a cell c_i generates the reinforcement and the restructuring signals (β_i and $\underline{\zeta}_i$) as given below in details.

Reinforcement Signal: To generate the reinforcement signal, a sensor node s_i first computes the minimum number of sensor nodes ($N_i^{Min}(n)$) that if exist within its transmission range, then the required degree of coverage of its local region is satisfied. Using the theorems and results given in [70] and by ignoring the border effect [93], $N_i^{Min}(n)$ can be computed using Eq. (14) given below.

$$N_i^{Min}(n) = \underset{l \geq 1}{Min}\left\{ \mathrm{E}\left[C_l^{rdc_i(n)}\right] \geq 1 - \varepsilon \right\} \cdot \rho_i(n) - 1, \tag{14}$$

where $\varepsilon \ll 1$ is a positive constant, $\rho_i(n) = \frac{\pi \cdot R_i^2}{\widehat{S}_i(n)}$, and $\mathrm{E}\left[C_l^{rdc_i(n)}\right]$ (given by the iterative Eq. (15)) is the expected surface that is rdc-$_i$-covered (covered with rdc_i sensor nodes) by randomly deploying l sensor nodes.

$$\mathrm{E}\left[C_l^{rdc_i(n)}\right] = (1 - \rho_i(n)) \cdot \mathrm{E}\left[C_{l-1}^{rdc_i(n)}\right] + \rho_i(n) \cdot \mathrm{E}\left[C_{l-1}^{rdc_i(n)-1}\right]. \tag{15}$$

The reinforcement signal for the nth round in a node s_i is generated based on a comparison between the number of neighbors of s_i ($N_i^r(n)$) and $N_i^{Min}(n)$. According to this comparison, following cases may occur:

- $N_i^{Min}(n) \leq N_i^r(n) \leq 2 \cdot N_i^{Min}(n)$: In this case, if the selected action of LA_i is "do not apply force to the neighboring nodes", then the reinforcement signal is to reward the action. Otherwise, the reinforcement signal is to penalize the action.
- $N_i^r(n) < N_i^{Min}(n)$ or $N_i^r(n) > 2 \cdot N_i^{Min}(n)$: In this case, if the selected action of LA_i is "apply force to the neighboring nodes", then the reinforcement signal is to reward the action. Otherwise, the reinforcement signal is to penalize the action.

In other words, if the number of neighboring nodes of the sensor node s_i is at least equal to the minimum required number of sensor nodes ($N_i^{Min}(n)$), but not greater than its twice ($2 \cdot N_i^{Min}(n)$), then it is better for this node not to apply any forces to its neighbors. Otherwise, and if the number of neighbors of s_i is less than $N_i^{Min}(n)$ or more than $2 \cdot N_i^{Min}(n)$, then it is better for it to apply forces to its neighbors.

Restructuring Signal: Since the interest set of the *HDI-CLA* has two members, the restructuring signal must be a 2 dimensional vector. To generate this vector, first an angle θ is selected randomly and uniformly from the range [0, 2π] and then the elements of the vector are computed according to the following equation:

$$\begin{cases} \zeta_{i1}(n) = N_i^f(n) \cdot \cos(\theta) \\ \zeta_{i2}(n) = N_i^f(n) \cdot \sin(\theta) \end{cases}. \tag{16}$$

In the above equation, $N_i^f(n)$ is the number of neighbors selecting "apply force to the neighboring nodes" action.

5.2.4 Maintenance

Maintenance phase of CLA-EDS deployment strategy is the same as the maintenance phase of CLA-DS. This phase is identical to the deployment phase with the exception that the sensor nodes do not move. Additionally, each sensor node s_i keeps track of its "*fixed*" neighboring nodes during this phase. If one of these neighbors is unreachable (a neighbor is considered unreachable if no *APPLIED_FORCE* packet can be received from it for more than t rounds), then it can be concluded 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.

6 Experimental results

To evaluate the performance of CLA-EDS deployment strategy, several experiments have been conducted and the results are compared with the results obtained from DSSA and IDCA algorithms given in [19], DSLE algorithm given in [76], and CLA-DS algorithm. It should be mentioned here that DSSA and IDCA algorithms are not primarily designed for providing k-coverage. Instead, the goal of these two algorithms is to control the movements of sensor nodes in such a way that in the deployed network, the average distance between sensor nodes is almost equal to the desired distance of sensor nodes in a uniform distribution. This approach enables us to use these algorithms for providing k-coverage by simply changing the input parameter "desired distance of sensor nodes" from d_{avg} to $\frac{d_{avg}}{k}$. In the experiments given in this section, we use DSSA and IDCA algorithms with this modification.

For comparison of the mentioned algorithms, we use the following criteria:

- Coverage: Fraction of the area which is covered by the deployed network. Coverage is specified according to Eq. (17).

$$Coverage = \frac{S_{C(\Omega)}}{S_\Omega} \tag{17}$$

- Node separation: Average distance from the nearest-neighbor in the deployed network. Node separation can be computed using Eq. (18). 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.

$$Node\ separation = \frac{1}{N}\sum_{i=1}^{N} \underset{s_j \in Nei(s_i)}{\text{Min}} (dist(s_i, s_j)) \tag{18}$$

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

Network area is assumed to be a 100 (m) × 100 (m) rectangle. Networks of different sizes from $N = 500$ to $N = 1{,}500$ 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 [94]. Table 1 gives the values for different parameters of the algorithm.

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 CLA-EDS algorithm with that of DSSA, IDCA, DSLE, and CLA-DS algorithms in terms of coverage as defined by Eq. (17). For

Table 1 Parameters of CLA-EDS algorithm and their values

Parameter	Value
Pfix	0
ROUND_DURATION	11 (s)
COLLECTING_DURATION	10 (s)
WAIT_FOR_LOCAL_INFO_DURATION	2 (s)
LEAST_DISTANCE	1 (m)
Nr	3 Rounds
No	6 Rounds
ε	.01
a (Reward parameter)	.25
b (Penalty parameter)	.25
t	3 rounds

this experiment, sensor nodes are initially deployed using a hybrid deployment method within a square with side length 10 (m) centered on the center of the network area. We assume that $M = 1$, that is, the required degree of coverage (k) throughout the network area is uniform. The experiment is performed for $k = 1, 2, 3, 4$, and 5. Figures 2, 3, and 4 give the results of this experiment for networks of different sizes ($N = 500$, 1,000, and 1,500). From the results we can conclude the following:

- The performance of CLA-EDS algorithm in covering the network area, for different values of k and networks of different sizes, can compete with that of DSSA and IDCA algorithms. Such performance, when considering the fact that CLA-EDS algorithm does not use any information regarding sensor positions or their relative distances to each other, indicates the efficiency of the learning automata in guiding the sensor nodes throughout the network area for finding their best positions.
- Performances of CLA-EDS and DSLE algorithms in covering the network area are almost the same in networks of small sizes ($N < 1{,}000$), but in networks of large sizes ($N > 1{,}000$), CLA-EDS outperforms DSLE in terms of the coverage criterion. The reason behind this phenomenon is that DSLE algorithm makes use of the Voronoi diagram for guiding the movements of sensor nodes throughout the network area, but the information coded in the Voronoi diagram of a highly dense network is not as useful as that coded in the Voronoi diagram of a sparse network.
- CLA-EDS outperforms CLA-DS algorithm in terms of the coverage criterion when $k > 1$. This superiority is expected since CLA-DS is designed for providing 1-coverage of the network area, whereas its extension, CLA-EDS, is designed for providing k-coverage.

6.2 Experiment 2

In this experiment, we compare CLA-EDS algorithm with DSSA, IDCA, DSLE, and CLA-DS algorithms in terms of the node separation criterion given by Eq. (18). The simulation settings of experiment 1 are also used for this experiment. Figures 5, 6, and 7 give the results of this experiment for networks of different sizes. 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:

- Performances of CLA-EDS, CLA-DS, and DSLE algorithms in terms of the node separation criterion for different values of k and networks of different sizes are almost the same. In other words, the overlapping areas resulted from CLA-EDS, CLA-DS, and DSLE algorithms are almost the same.
- Node separation of the CLA-EDS algorithm is less than that of DSSA and IDCA algorithms. Since the coverage of CLA-EDS algorithm is almost equal to that of DSSA and IDCA algorithms in almost all cases, having smaller node separation or more overlapped area makes CLA-EDS superior to DSSA and IDCA algorithms due to the following two reasons:
  - The fraction of the network area, which is under the supervision of more than one sensor node, is higher in CLA-EDS algorithm than in DSSA and IDCA 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 heal the holes when node separation is smaller.

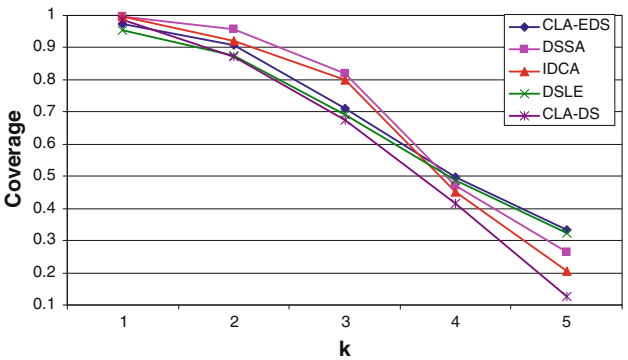


Fig. 2 Comparison of CLA-EDS with existing deployment algorithms in terms of the coverage criterion for $N = 500$ sensor nodes

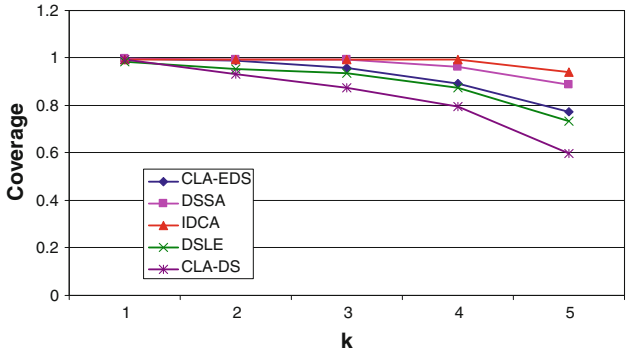


Fig. 3 Comparison of CLA-EDS with existing deployment algorithms in terms of the coverage criterion for $N = 1{,}000$ sensor nodes

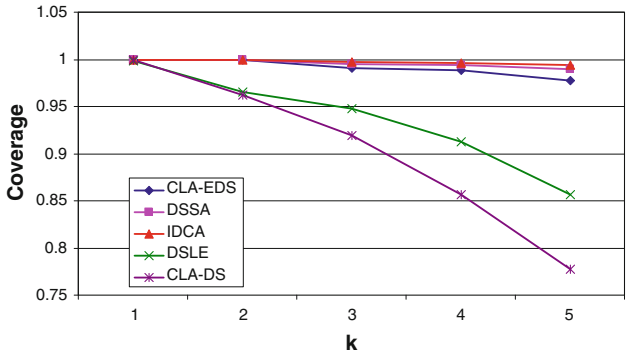


Fig. 4 Comparison of CLA-EDS with existing deployment algorithms in terms of the coverage criterion for $N = 1{,}500$ sensor nodes

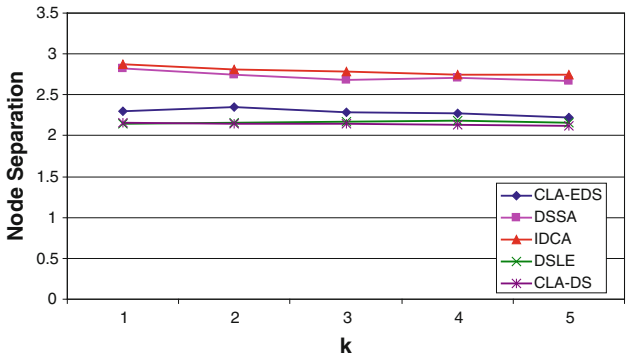


Fig. 5 Comparison of CLA-EDS with existing deployment algorithms in terms of the node separation criterion for $N = 500$ sensor nodes

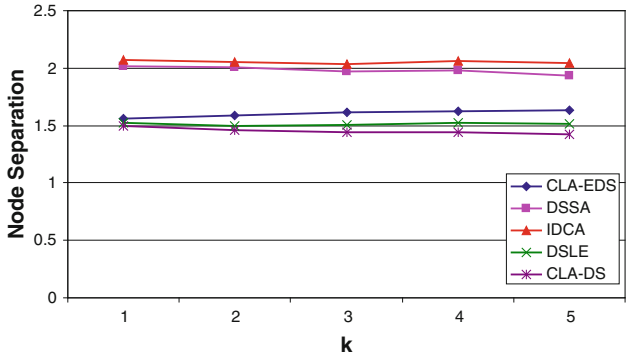


Fig. 6 Comparison of CLA-EDS with existing deployment algorithms in terms of the node separation criterion for $N = 1{,}000$ sensor nodes

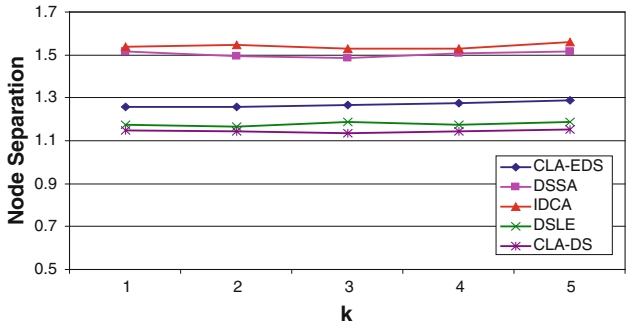


Fig. 7 Comparison of CLA-EDS with existing deployment algorithms in terms of the node separation criterion for $N = 1{,}500$ sensor nodes

6.3 Experiment 3

In this experiment, CLA-EDS algorithm is compared with DSSA, IDCA, DSLE, and CLA-DS algorithms in terms of the distance criterion. The simulation settings of experiment 1 are also used for this experiment. Figures 8, 9, and 10 give the results of this experiment for networks of different sizes. These figures show that in terms of the distance criterion, CLA-EDS is the worst algorithm among the compared algorithms. This is due to the fact that CLA-EDS algorithm does not use any information regarding the position of the sensor nodes or their relative distances to each other. To compensate this weakness, CLA-EDS algorithm has a parameter p_{fix} which can be used to make a tradeoff between the distance and coverage criteria. By increasing the value of p_{fix}, one can decrease the average distance moved by sensor nodes at the expense of less network coverage (refer to experiment 7).

6.4 Experiment 4

In this experiment, CLA-EDS algorithm is compared with DSSA, IDCA, DSLE, and CLA-DS 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 [95] and localization techniques adopted to wireless sensor networks [96, 97]. For simulating a noise level of $0 < \lambda < 1$,

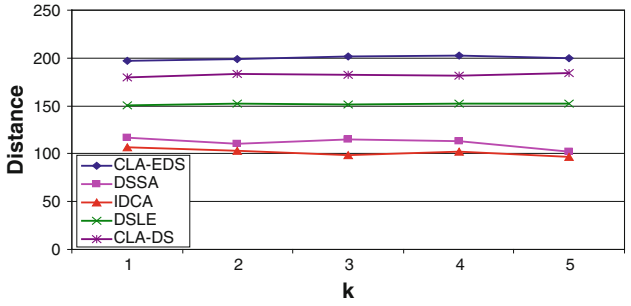


Fig. 8 Comparison of CLA-EDS with existing deployment algorithms in terms of the distance criterion for $N = 500$ sensor nodes

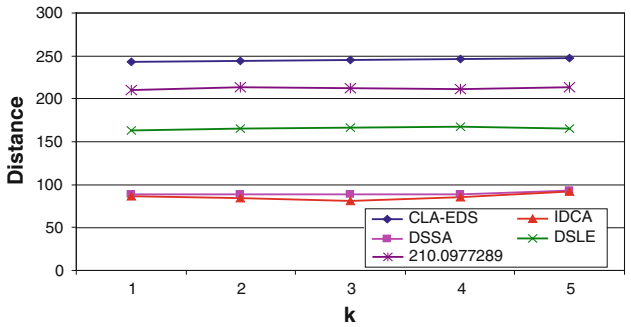


Fig. 9 Comparison of CLA-EDS with existing deployment algorithms in terms of the distance criterion for $N = 1{,}000$ sensor nodes

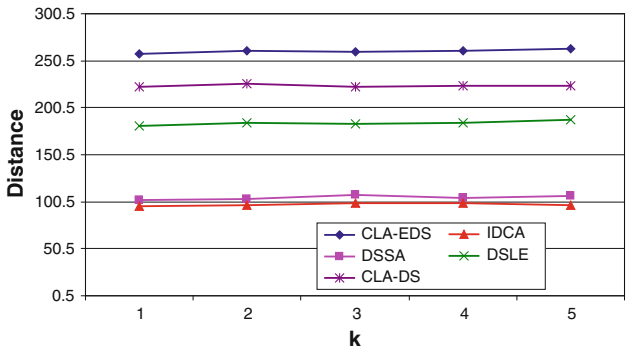


Fig. 10 Comparison of CLA-EDS with existing deployment algorithms in terms of the distance criterion for $N = 1{,}500$ sensor nodes

for each sensor node s_i two numbers $Rnd_i(x)$ and $Rnd_i(y)$ are selected uniformly at random from the ranges $[-MaxX, MaxX]$ and $[-MaxY, MaxY]$ respectively and are used for modifying the position (x_i, y_i) of the node according to Eq. (19). For this study, λ is assumed to be one of the following: .1, .2, .3, .4, and .5.

$$\begin{cases} x_i^{Noisy} = x_i + \lambda.Rnd_i(x) \\ y_i^{Noisy} = y_i + \lambda.Rnd_i(y) \end{cases} \quad (19)$$

We assume that $M = 1$, $N = 1{,}500$, and $k = 5$. Sensor nodes are initially deployed using a hybrid deployment method within a square with side length 10 (m) centered on the center of the network area. Figures 11, 1213 give the results of this experiment for different criteria. These figures show that:

1. Noise level has no effect on the performance of CLA-EDS and CLA-DS algorithms with respect to coverage, node separation and distance criteria. This is due to the fact that CLA-EDS and CLA-DS algorithms do not use any information about the position of the sensor nodes.
2. DSLE algorithm is highly affected by increasing the noise level. This is because the deployment strategy used in this algorithm is highly dependent on the exact positions of sensor nodes.
3. The impact of noise level on the performances of DSSA and IDCA algorithms with respect to coverage and node separation criteria is not too significant. This is due to the fact that these algorithms, like CLA-EDS 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 distance criterion. This is because DSSA and IDCA algorithms, unlike CLA-EDS, 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.

6.5 Experiment 5

In this experiment, we compare the behavior of CLA-EDS, DSSA, IDCA, DSLE, and CLA-DS 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

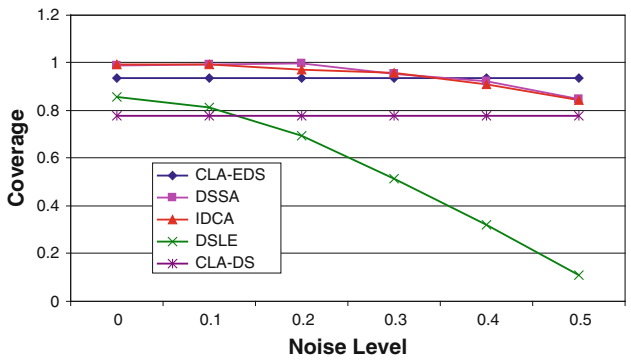


Fig. 11 Comparison of CLA-EDS with existing deployment algorithms in terms of the coverage criterion in the presence of noise in estimating the position of sensor nodes

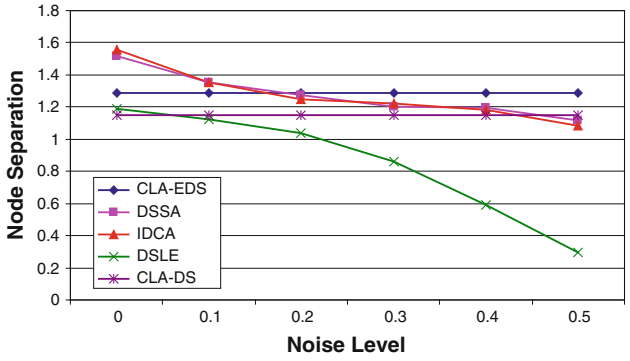


Fig. 12 Comparison of CLA-EDS with existing deployment algorithms in terms of the node separation criterion in presence of the noise in estimating the position of sensor nodes

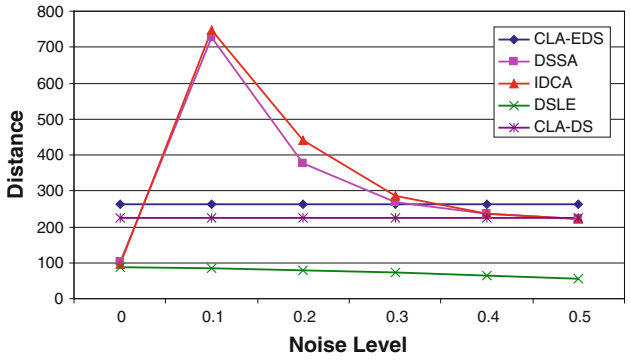


Fig. 13 Comparison of CLA-EDS with existing deployment algorithms in terms of the distance criterion in presence of the noise in estimating the position of sensor nodes

world scenarios. We use the probabilistic motion model of sensor nodes given in [98]. 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:

$$\begin{cases} x_i(n+1) = x_i(n) + D_i(n)\cdot\cos\left(\theta_i(n) + \frac{T_i(n)}{2}\right) \\ \qquad\qquad + C_i(n)\cdot\cos\left(\theta_i(n) + \frac{T_i(n)+\pi}{2}\right) \\ y_i(n+1) = y_i(n) + D_i(n)\cdot\sin\left(\theta_i(n) + \frac{T_i(n)}{2}\right) \\ \qquad\qquad + C_i(n)\cdot\sin\left(\theta_i(n) + \frac{T_i(n)+\pi}{2}\right) \\ \theta_i(n+1) = (\theta_i(n) + T_i(n)) \mod (2\pi) \end{cases} \quad (20)$$

In Eq. (20), $\theta_i(n) + \frac{T_i(n)}{2}$ is referred to as the major axis of movement, $\theta_i(n) + \frac{T_i(n)+\pi}{2}$ 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 $r_i(n)$:

$$\begin{cases} D_i(n) \sim N\left(d_i(n), d_i^2(n)\cdot\sigma_{D_d}^2 + r_i^2(n)\cdot\sigma_{D_r}^2 + \sigma_{D_1}^2\right) \\ T_i(n) \sim N\left(r_i(n), d_i^2(n)\cdot\sigma_{T_d}^2 + r_i^2(n)\cdot\sigma_{T_r}^2 + \sigma_{T_1}^2\right) \\ C_i(n) \sim N\left(0, d_i^2(n)\cdot\sigma_{C_d}^2 + r_i^2(n)\cdot\sigma_{C_r}^2 + \sigma_{C_1}^2\right) \end{cases} \quad (21)$$

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.

We assume that $M = 1$, $k = 5$, and $N = 1{,}500$. Sensor nodes are initially deployed using a hybrid deployment method within a square with side length 10 (m) centered on the center of the network area. Figures 14, 1516 show the results of this experiment. The results indicate the following facts:

- Using the probabilistic motion model instead of the perfect motion model significantly degrades the performances of DSSA, IDCA, and DSLE algorithms in terms all three criteria, but does not affect the performances of CLA-EDS and CLA-DS algorithms substantially.
- When the probabilistic motion model is used, CLA-EDS algorithm outperforms the existing algorithms in terms of coverage and node separation criteria.
- For probabilistic motion model, the average distance moved by sensor nodes for all algorithms is higher than when perfect motion model is used. When the probabilistic motion model is used the performances of CLA-EDS, DSSA, IDCA, DSLE, and CLA-DS algorithms in terms of distance criterion are degraded by 6, 200, 201, 153, and 4 % respectively. This indicates that CLA-EDS and CLA-DS algorithms are more robust to the deviations in the movements of sensor nodes than DSSA, IDCA, and DSLE algorithms.

6.6 Experiment 6

This experiment is conducted to study the behavior of CLA-EDS, DSSA, IDCA, and DSLE 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-EDS algorithm gradually learns the expected local density. For this study, we assume that $M = 1$ and $k = 5$. Sensor nodes are initially deployed using a hybrid deployment method within a square with side length 10 (m) centered on the center of the network area. We repeat the experiment for $N = 500$, 1,000, and 1,500. Figures 17, 1819 give the results of this experiment.

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

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

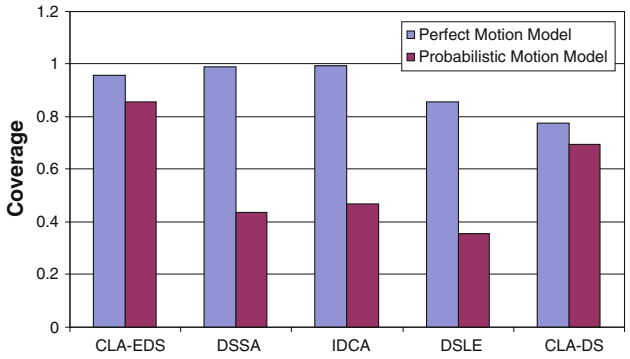


Fig. 14 Comparison of CLA-EDS with existing deployment algorithms in terms of the coverage criterion when movements of sensor nodes follow a probabilistic motion model

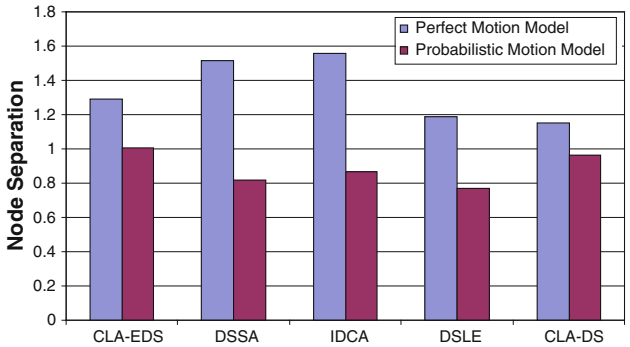


Fig. 15 Comparison of CLA-EDS with existing deployment algorithms in terms of the node separation criterion when movements of sensor nodes follow a probabilistic motion model

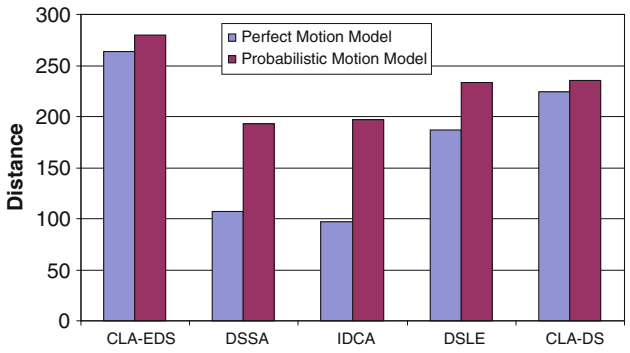


Fig. 16 Comparison of CLA-EDS with existing deployment algorithms in terms of the distance criterion when movements of sensor nodes follow a probabilistic motion model

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

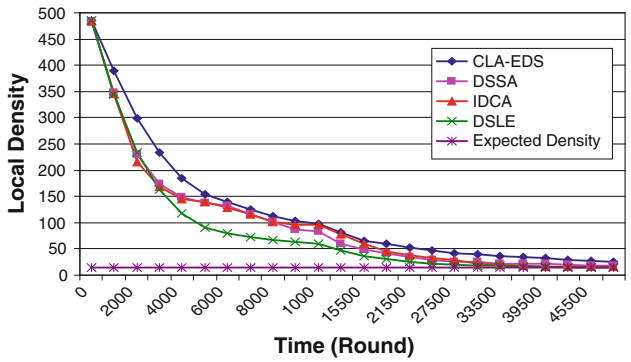


Fig. 17 Local density of sensor nodes during the deployment process when $N = 500$

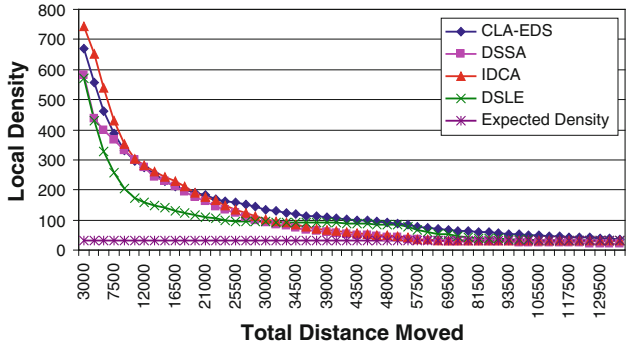


Fig. 18 Local density of sensor nodes during the deployment process when $N = 1{,}000$

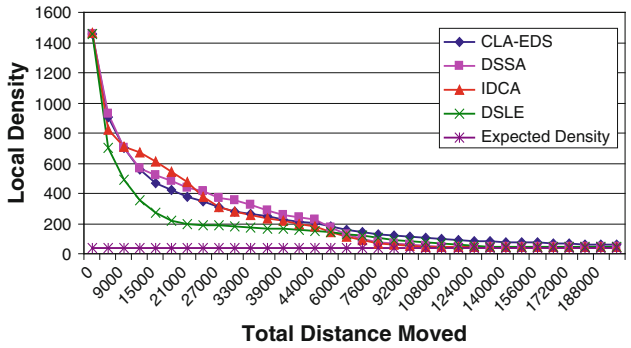


Fig. 19 Local density of sensor nodes during the deployment process when $N = 1{,}500$

the deployment process. These figures show that CLA-EDS 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.

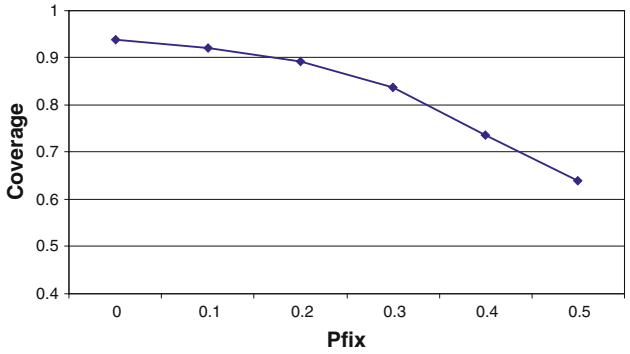


Fig. 20 Effect of parameter P_{fix} on the performance of CLA-EDS in terms of coverage criterion

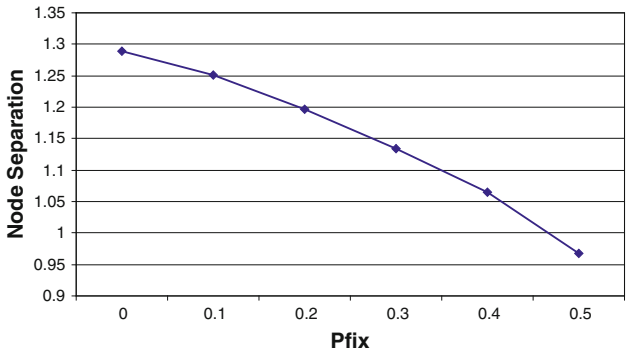


Fig. 21 Effect of parameter P_{fix} on the performance of CLA-EDS in terms of node separation criterion

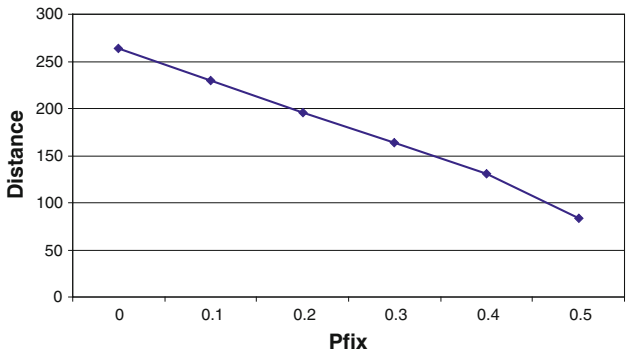


Fig. 22 Effect of parameter P_{fix} on the performance of CLA-EDS in terms of distance criterion

Of course, it takes longer time for CLA-EDS to achieve this. This is due to the fact that CLA-EDS does not use any information regarding the physical positions of sensor nodes or their relative distances to each other.

6.7 Experiment 7

This experiment is conducted to study the effect of the parameter P_{fix} on the performance of CLA-EDS algorithm. For this study, we let $M = 1$ and $k = 5$. We consider a network of $N = 1{,}500$ 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. Figures 20, 2122 give the results of this experiment. These figures indicate that by increasing the value of parameter P_{fix} in CLA-EDS 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.

6.8 Experiment 8

In this experiment, we study the behavior of CLA-EDS, DSSA, IDCA, DSLE, and CLA-DS algorithms with respect to coverage, node separation, and distance criteria when the required degree of coverage differs in different regions of the network. For this experiment, we consider $M = 10$ different regions. Each region is a circular sub-area within the network area with a radius, which is randomly selected from the range [5, 37] meters, and a required degree of coverage, which is selected randomly from the range [1, 5]. The experiment is performed for $N = 500$, 1,000, and 1,500 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. Figures 23, 2425 give the results of this experiment. From these figures, one may conclude the following facts:

- CLA-EDS algorithm significantly outperforms DSSA, IDCA, DSLE, and CLA-DS with respect to the coverage criterion in networks of different sizes, especially when $N < 1000$. This indicates that the learning mechanism utilized in CLA-EDS algorithm is

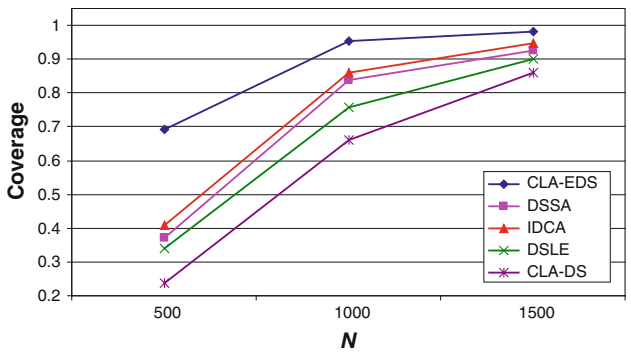


Fig. 23 Comparison of CLA-EDS with existing deployment algorithms in terms of the coverage criterion when the required degree of coverage differs in different regions of the network

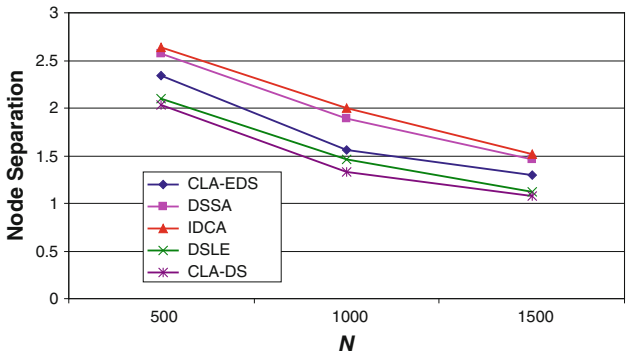


Fig. 24 Comparison of CLA-EDS with existing deployment algorithms in terms of the node separation criterion when the required degree of coverage differs in different regions of the network

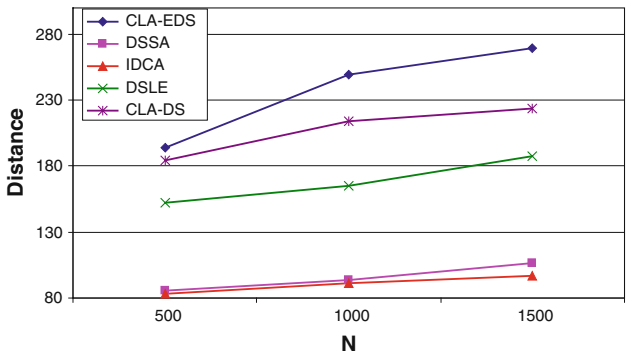


Fig. 25 Comparison of CLA-EDS with existing deployment algorithms in terms of the distance criterion when the required degree of coverage differs in different regions of the network

able to adapt itself to the different degrees of coverage needed in different regions of the network.

- According to Fig. 24, CLA-EDS is superior to CLA-DS and DSLE algorithms in terms of the node separation criterion, that is CLA-EDS better spreads sensor nodes throughout the network area than CLA-DS and DSLE algorithms. This figure also shows that the node separation of DSSA and IDCA algorithms are higher than that of CLA-EDS algorithm. But with a similar discussion given in experiment 2, this also indicates that CLA-EDS algorithm is superior to IDCA and DSSA algorithms with respect to the node separation criterion.
- In terms of the distance criterion, CLA-EDS has the worst performance among the compared algorithms. This inferiority is expected as it was mentioned in experiment 3.

6.9 Experiment 9

The aim of conducting this experiment is to study the behavior of CLA-EDS deployment strategy in controlling the local density of sensor nodes within different regions, each having a different requirement of degree of coverage, during the deployment process. The simulation settings of the experiment 8 are also used for this experiment. The experiment is performed for $N = 1{,}500$ 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. Figures 26, 2728 give the results of this experiment for 3 randomly selected regions with the following properties:

- Region 1: Radius = 24 m, $k = 5$
- Region 2: Radius = 16 m, $k = 3$
- Region 3: Radius = 36 m, $k = 4$

These figures show that CLA-EDS 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 in each region approaches its expected value in that region. The figures also show that DSSA, IDCA, DSLE, and CLA-DS algorithms are not able to perfectly control the local density of sensor nodes when the expected local density differs in different regions of the network.

6.10 Experiment 10

This experiment is conducted to study the behavior of the *HDICLA* as a learning model in CLA-EDS algorithm. For this study, we set $M = 1$, $k = 5$, and $N = 1{,}500$. Sensor nodes 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 29 depicts the action probability vector of a randomly selected learning automaton from the *HDICLA*. 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 valued.

Figure 30 shows the *HDICLA* entropy during the deployment process. This figure indicates that the entropy of the *HDICLA* is high at initial rounds of CLA-EDS algorithm, but gradually decreases as time passes. It goes below 70 at about round number 100. This means that after this round, the entropy of each learning automaton LA_i in the *HDICLA* is on average below .047. If the entropy of a two-action learning automaton is below .047, then it can be concluded that the action probability of one of its actions is higher than .992. This means that action switching in each learning automaton in the *HDICLA* rarely occurs after round number 100.

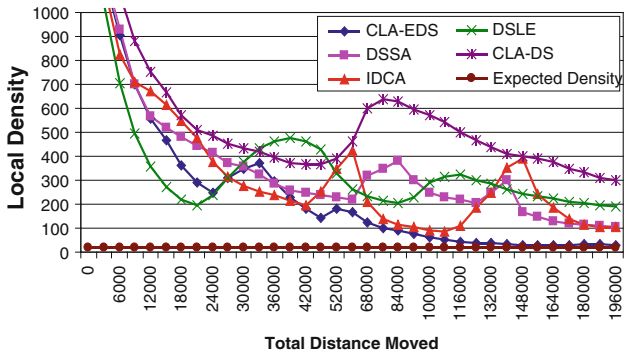


Fig. 26 Local density of sensor nodes during the deployment process in region 1

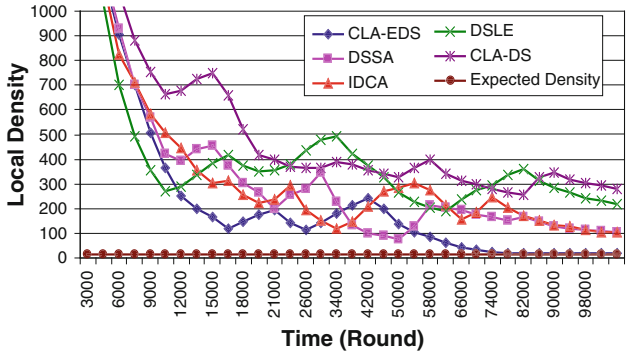


Fig. 27 Local density of sensor nodes during the deployment process in region 2

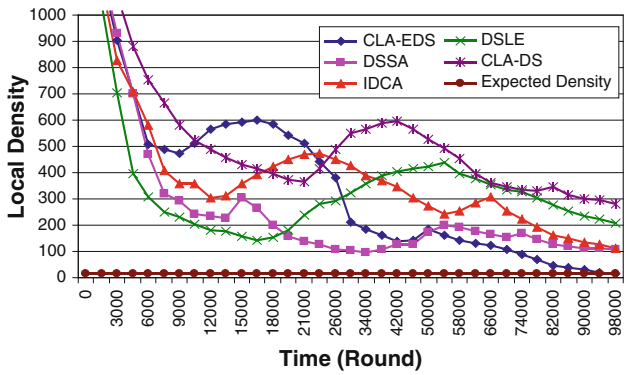


Fig. 28 Local density of sensor nodes during the deployment process in region 3

Figure 31 gives the degree of expediency (M_d) of the *HDICLA* during the deployment process of CLA-EDS algorithm. As it can be seen from this figure, M_d is initially low, but as time passes, it gradually increases. In other words, the *HDICLA*, with the local rule of CLA-EDS algorithm, gradually improves its performance and learns to receive fewer penalties from the environment in comparison to a pure-chance *HDICLA*. To have a better understanding of the expediency concept, we change the local rule of the *HDICLA* in CLA-EDS algorithm to form

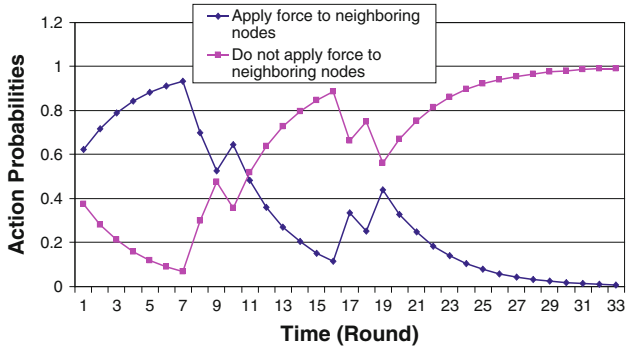


Fig. 29 Action probabilities of a randomly selected learning automaton from *HDICLA*

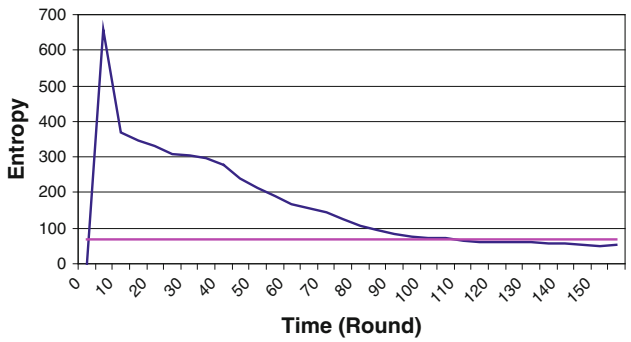


Fig. 30 *HDICLA* entropy

three new algorithms called Modified-CLA-EDS-1, Modified-CLA-EDS-2, and Modified-CLA-EDS-3. The local rules used in these modified algorithms are as follows:

- Modified-CLA-EDS-1 algorithm:
  - $N_i^{Min}(n) \le N_i^r(n) \le 3 \cdot N_i^{Min}(n)$: 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.
  - $N_i^r(n) < N_i^{Min}(n)$ or $N_i^r(n) > 3 \cdot N_i^{Min}(n)$: 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.
- Modified-CLA-EDS-2 algorithm:
  - If $\left|N_i^{Min}(n) - N_i^r(n)\right| \le 1$ then: 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.
  - If $\left|N_i^{Min}(n) - N_i^r(n)\right| > 1$ then: if the selected action of LA_i is "apply force to neighboring nodes", then the reinforcement signal is to reward the action.

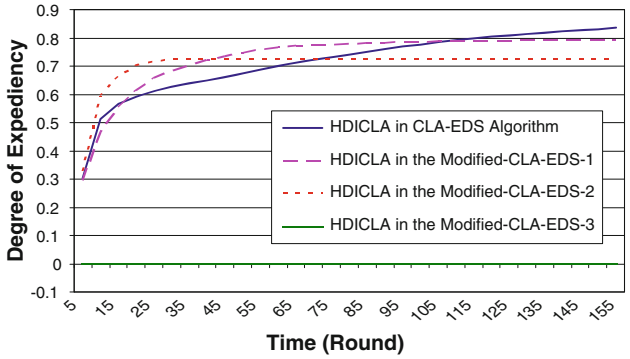


Fig. 31 Degree of expediency of the *HDICLA*

Otherwise, the reinforcement signal is to penalize the action.

- Modified-CLA-EDS-3 algorithm: In this algorithm, a pure-chance *HDICLA* is used and therefore, local rule is useless.

Figure 31, compares the degree of expediency of the *HDICLA*, used in CLA-EDS algorithm, with that of the *HDICLA*s, used in the modified versions of CLA-EDS algorithm. As it can be seen from this figure, the *HDICLA*, used in CLA-EDS algorithm, is more expedient than that used in the modified versions of CLA-EDS algorithm. To study the effect of the degree of expediency on the performance of the *HDICLA*, we compare CLA-EDS algorithm with its modified versions in terms of the coverage, node separation, and distance criteria. The results of this comparison, which are given in Table 3, show that:

- CLA-EDS algorithm outperforms all of its modified versions in terms of all mentioned criteria.
- Modified-CLA-EDS-1 algorithm, which is more expedient than all algorithms but CLA-EDS, outperforms Modified-CLA-EDS-2 and Modified-CLA-EDS-3 algorithms in terms of all mentioned criteria.
- Modified-CLA-EDS-3 algorithm, in which a pure-chance *HDICLA* is used, has the worst performance among CLA-EDS algorithm and its modified versions in terms of coverage, node separation, and distance criteria.

In other words, the performance of an *HDICLA*, which uses the local rule of CLA-EDS algorithm, is higher than that of an *HDICLA*, which uses the local rule of the modified versions of CLA-EDS algorithm. This indicates that the degree of expediency is a measure of the performance of the *HDICLA*; higher values of M_d results in better performance of the *HDICLA*. Since the degree of expediency of an *HDICLA* with a specific structure depends directly on the local rule of that *HDICLA*, we can conclude that for increasing the performance of an *HDICLA*, it is enough to devise local rules which increase the degree of expediency of that *HDICLA*.

Figure 32 depicts the changes in the *HDICLA* restructuring tendency during the deployment process. This figure shows that the restructuring tendency of the *HDICLA* 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 the sensor nodes approaching its expected value which results in the magnitude of the force vector applied to each sensor node to approach zero.

7 Summary of results

In this study, we compared the performance of CLA-EDS deployment algorithm with respect to coverage, node separation, and distance criteria with DSSA, IDCA, DSLE, and CLA-DS deployment algorithms. Comparisons were made for different network sizes, different degrees of coverage, and noise free and noisy environments. From the results of this study we can conclude that:

- In noise free environments, the proposed algorithm (CLA-EDS) can compete with existing algorithms in terms of the coverage criterion, outperforms existing algorithms in terms of the node separation criterion, and performs worse than the existing algorithms in terms of the distance criterion.
- CLA-EDS and CLA-DS algorithms, unlike DSSA, IDCA, and DSLE algorithms, do not use any information regarding the position of the 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, outperform existing algorithms in terms of the coverage and node separation criteria.
- The algorithms which are least affected by the selection of the "node movement model" of sensor nodes are CLA-EDS and CLA-DS.
- When required degree of coverage differs in different regions of the network, CLA-EDS performs significantly better than the existing algorithms with respect to the coverage and the node separation criteria.
- CLA-EDS 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.

Table 3 Comparison of CLA-EDS with its modified versions in terms of coverage, node separation, and distance criteria

$N = 500$ sensor nodes

$N = 1{,}000$ sensor nodes

	CLA-EDS	Modified-CLA-EDS-1	Modified-CLA-EDS-2	Modified-CLA-EDS-3
Coverage	.93744	.83215	.68156	.33198
Node separation	1.28853	1.06329	.96755	.86207
Distance	263.53069	304.18386	376.20975	637.95104

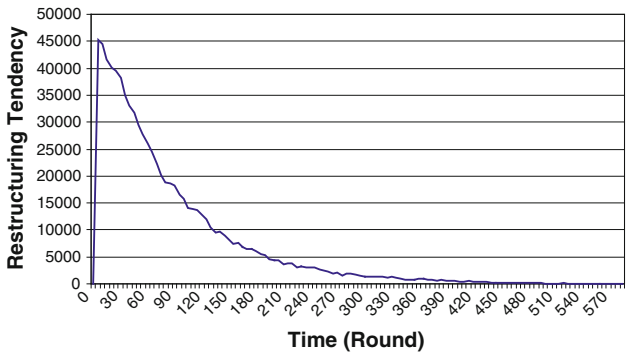


Fig. 32 *HDICLA* restructuring tendency

8 Conclusion

In this paper, a new deployment strategy based on cellular learning automata called CLA-EDS for mobile sensor networks was proposed. The proposed deployment strategy is an extension to a recently introduced deployment strategy, called CLA-DS, which can provide k-coverage of the network area even if k (required degree of coverage) differs in different regions of the network. CLA-EDS deployment strategy, unlike similar existing deployment strategies, does not use any information regarding the 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 the proposed deployment strategy, in addition to the advantages it inherits from CLA-DS, outperforms existing algorithms such as DSSA, IDCA, and DSLE in providing k-coverage, especially when required degree of coverage differs in different regions of the network.

References

1. Ilyas, M., & Mahgoub, I. (2005). *Handbook of sensor networks: Compact wireless and wired sensing systems*. London, Washington, DC: CRC Press.
2. Dhillon, S. S., Chakrabarty, K., & Iyengar, S. S. (2002). Sensor placement for grid coverage under imprecise detections. *International conference on information fusion (FUSION) 2002*, Annapolis, 2002, pp. 1581–1587.
3. Tilak, S., Abu-Ghazaleh, N. B., & Heinzelman, W. (2002, September). Infrastructure trade-offs for sensor networks. *ACM WSNA'02*, pp. 49–58.
4. Musman, S., Lehner, P. E., & Elsaesser, C. (1997). Sensor planning for elusive targets. *Journal of Computer Mathematical Modeling, 25*(3), 103–115.
5. Salhieh, A., Weinmann, J., Kochhal, M., & Schwiebert, L. (2001). *Power efficient topologies for wireless sensor network* (pp. 156–163). Spain: International Conference on Parallel Processing.
6. Schwiebert, L., Gupta, S. K. S., & Weinamann, J. (2001, July). Research challenges in wireless networks of biomedical sensors. *ACM SIGMOBILE 2001*, Rome, pp. 151–165.
7. Petriu, E. M., Georganas, N. D., Petriu, D., Makrakis, D., & Groza, V. Z. (2000, December). Sensor-based information appliances. *IEEE Instrumentation Measurement Magazine*, pp. 31–35.
8. Chakrabarty, K., Iyengar, S. S., Qi, H., & Cho, E. (2002). Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers, 51*, 1448–1453.
9. Heinzelman, W., Chandrakasan, A., & Balakrishnan, H. (2000, January). Energy-efficient communication protocol for wireless microsensor networks. *HICSS 2000*, Maui, pp. 8020–8029.
10. Heinzelman, W. (2000, June). *Application-specific protocol architecture for wireless networks*. Ph.D. dissertation, Massachusetts Institute of Technology.
11. Heinzelman, W. B., Chandrakasan, A. P., & Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications, 1*(4), 660–670.
12. Lindsey, S., & Raghavendra, C. S. (2002, March). PEGASIS: Power-efficient gathering in sensor information systems. *IEEE aerospace conference*, pp. 1125–1130.
13. Lindsey, S., Raghavendra, C., & Sivalingam, K. M. (2002). Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel Distributed Systems, 13*(9), 924–935.
14. Willig, A., Shah, R., Rabaey, J., & Wolisz, A. (2002, August) Altruists in the PicoRadio sensor network. *4th IEEE International Workshop on Factory Communications Systems*, Sweden, pp. 175–184.
15. Howard, A., Mataric, M. J., & Sukhatme, G. S. (2002, June), Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem. *International symposium on distributed autonomous robotics systems*, Fukuoka, Japan, pp. 299–308.
16. Zou, Y., & Chakrabarty, K. (2003). Sensor deployment and target localization based on virtual forces. In *Proceedings of IEEE infocom conference*, pp. 1293–1303.
17. Poduri, S., & Sukhatme, G. (2004). Constrained coverage for mobile sensor networks. In *Proceedings of IEEE international conference on robotics and automation (ICRA'04)*.
18. Zou, Y., & Chakrabarty, K. (2004, February). Sensor deployment and target localization in distributed sensor networks. *ACM*

Transactions on Embedded Computing Systems, Special Issue on Networked Embedded Computing: Tools, Architectures and Applications, 3(1), 61–91.
19. Heo, N., & Varshney, P. K. (2005). Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics (Part A)*, *35*(1), 78–92.
20. Wang, G., Cao, G. & La Porta, T. (2006, June). Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing, 5*(6), 640–652.
21. Wang, P. C., Hou, T. W., & Yan, R. H. (2006). Maintaining coverage by progressive crystallattice permutation in mobile wireless sensor networks. *IEEE international conference on systems and networks communication (ICSNC)*, pp. 1–6.
22. Chellappan, S., Bai, X., Ma, B., Xuan, D., & Xu, C. (2007). Mobility limited flip-based sensor networks deployment. *IEEE Transactions on Parallel and Distributed Systems*, *18*(2), 199–211.
23. Chellappan, S., Gu, W., Bai, X., Xuan, D., Ma, B., & Zhang, K. (2007). Deploying wireless sensor networks under limited mobility constraints. *IEEE Transactions on Mobile Computing, 6*(10), 1142–1157.
24. Jiang, Z., Wu, J., Kline, R., & Krantz, J. (2008). Mobility control for complete coverage in wireless sensor networks. In *Proceedings of the 28th international conference on distributed computing systems workshops (ICDCS)*, pp. 291–296.
25. Wang, G., Cao, G., Porta, T. L., & Zhang, W. (2005). Sensor relocation in mobile sensor networks. *IEEE INFOCOM*, pp. 2302–2312.
26. Yang, S., Li, M., & Wu, J. (2007). Scan-based movement-assisted sensor deployment methods in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems, 18*(8), 1108–1121.
27. Wu, J., & Yang, S. (2007). Optimal movement-assisted sensor deployment and its extensions in wireless sensor networks. *Simulation Modeling Practice and Theory, 15*(4), 383–399.
28. Yang, S., Wu, J., & Dai, F. (2006). Localized movement-assisted sensor deployment in wireless sensor networks. *IEEE workshops in the international conference on mobile adhoc and sensor systems (MASS)*, pp. 753–758.
29. Esnaashari, M., & Meybodi, M. R. (2011). A cellular learning automata-based deployment strategy for mobile wireless sensor networks. *Journal of Parallel and Distributed Computing, 71*(7), 988–1001.
30. Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., et al. (2004, December). A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks, 46*(5), 605–634.
31. Mehta, D., Lopez, M., & Lin, L. (2003, May). Optimal coverage paths in ad hoc sensor networks. *IEEE international conference on communications*, pp. 507–511.
32. Kumar, S., Lai, T. H., & Balogh, J. (2004). *On k-Coverage in a Mostly Sleeping Sensor Network* (pp. 144–158). Philadelphia, Pennsylvania, USA: 10th Annual International Conference on Mobile Computing and Networking.
33. Zhao, Z., & Govindan, R. (2003, November). Understanding packet delivery performance in dense wireless sensor networks. In *3th ACM conference on embedded networked sensor systems*, Los Angeles, CA, pp. 1–13.
34. Hall, D., & Llinas, J. (2001). *Handbook of multisensor data fusion*. New York: CRC Press.
35. Wang, B., Lim, H. B., & Ma, D. (2009). A survey of movements strategies for improving network coverage in wireless sensor networks. *Computer Communications, 32*, 1427–1436.
36. Wang, Y. C., Hu, C. C., & Tseng, Y. C. (2008). Efficient placement and dispatch of sensors in a wireless sensor network. *IEEE Transactions on Mobile Computing, 7*(2), 262–274.
37. Wang, Y. C., & Tseng, Y. C. (2008, September). Distributed deployment schemes for mobile wireless sensor networks to ensure multi-level coverage. *IEEE Transactions on Parallel and Distributed Systems, 19*(9), 1280–1294.
38. Katsuma, R., Murata, Y., Shibata, N., Yasumoto, K., & Ito, M. (2010, April). Extending *k*-coverage lifetime of wireless sensor networks with surplus nodes. In *Proceedings of the 5th international conference on mobile computing and ubiquitous networking (ICMU2010)*, pp. 9–16.
39. Mo, W., Qiao, D., & Wang, Z. (2006). Lifetime maximization of sensor networks under connectivity and *k*-coverage constraints. *Lecture Notes in Computer Science, 4026*, 422–442. Berlin/Heidelberg: Springer.
40. Mo, W., Qiao, D., & Wang, Z. (2005, September). Mostly-sleeping wireless sensor networks: connectivity, k-coverage, and α-lifetime. *Allerton Conference*, UIUC.
41. Kalayci, T. E., Yildirim, K. S., & Ugur, A. (2007, October). Maximizing coverage in a connected and *k*-covered wireless sensor network using genetic algorithms. *International Journal of Applied Mathematics and Informatics, 1*(3), 123–130.
42. Abeyweera, I. S. (2007, February). A coverage control mechanism satisfying application requirements in a wireless sensor network. *Master's Thesis*, Department of Information Networking, Osaka University.
43. Ye, M., Chan, E., Chen, G., & Wu, J. (2006, July). Energy efficient fractional coverage schemes for low cost wireless sensor networks. In *26th IEEE International Conference on Distributed Computing Systems Workshop*, pp. 74–79.
44. Slijepcevic, S., & Potkonjak, M. (2001, June). Power efficient organization of wireless sensor networks. In *IEEE international conference on communications*, pp. 472–476.
45. Makhoul, A., Saadi, R., & Pham, C. (2009, October). Coverage and adaptive scheduling algorithms for criticality management on video wireless sensor networks. In *International conference on ultra modern telecommunications & workshops*, pp. 1–8.
46. Gao, S., Vu, C. T., Li, Y. (2006). Sensor scheduling for *k*-coverage in wireless sensor networks. *Lecture Notes in Computer Science*, 4325, 268–280. Berlin/Heidelberg: Springer.
47. Huang, C. F., & Tseng, Y. C. (2005). The coverage problem in a wireless sensor network. *Mobile Networks and Applications, 10*(4), 519–528.
48. Liu, Y., Pu, J., Zhang, S., Liu, Y., & Xiong, Z. (2009). A localized coverage preserving protocol for wireless sensor networks. *Sensors, 9*, 281–302.
49. Simon, G., Molnar, M., Gonczy, L., & Cousin, B. (2007). *Dependable k-Coverage Algorithms for Sensor Networks* (pp. 1–6). Warsaw, Poland: Instrumentation and Measurement Technology Conference.
50. Pocquet, A., Cousin, B., Molnar, M., & Parraud, P. (2008, August). Performance analysis of CGS, a k-coverage algorithm based on one-hop neighboring knowledge. In *2nd international conference on sensor technologies and applications*, pp. 115–122.
51. Zhou, Z., Das, S., & Gupta, H. (2004, October). Connected k-coverage problem in sensor networks. In *13th international conference on computer communications and networks*, Chicago, IL, pp. 373–378.
52. Kasbekar, G. S., Bejerano, Y., & Sarkar, S. (2009, September). Lifetime and coverage guarantees through distributed coordinate-free sensor activation. In *15th annual international conference on mobile computing and networking*. Beijing, China, pp. 169–180.
53. Zhang, H., & Hou, J. C. (2005, Marach). Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc & Sensor Wireless Networks, 1*, 89–124.
54. Huang, C. F., Lo, L. C., & Tseng, Y. C. (2006, May). Decentralized energy-conserving and coverage-preserving protocols for

wireless sensor networks. *ACM Transactions on Sensor Networks, 2*(2), 182–187.
55. Fusco, G., & Gupta, H. (2009, June) Selection and orientation of directional sensors for coverage maximization. In *6th Annual IEEE communications society conference on sensor, Mesh, and ad hoc communications and networks*. Rome, pp. 1–9.
56. Bagheri, M. (2007). Efficient *k*-coverage algorithms for wireless sensor networks and their applications to early detection of forest fires. *PhD thesis*, Simon Fraser University.
57. Hefeeda, M., & Bagheri, M. (2007, May). Randomized *k*-coverage algorithms for dense sensor networks. *IEEE INFOCOM 2007*, Anchorage, AK, pp. 2376–2380.
58. Fusco, G., & Gupta, H. (2009). ε-Net approach to sensor *k*-coverage, In *4th International conference on wireless algorithms, systems, and applications*, pp. 104–114.
59. Ammari, H. M., & Das, S. K. (2008, June). Joint *k*-coverage and hybrid forwarding in duty-cycled three-dimensional wireless sensor networks. In *5th annual IEEE communications society conference on sensor, mesh, and ad hoc communications and networks*. San Francisco, CA, pp. 170–178.
60. Ammari, H. M. (2008, May). Energy efficient connected *k*-coverage, duty-cycling, and geographic forwarding in wireless sensor networks. In *PhD thesis*, University of Texas at Arlington.
61. Noy, A. B., Brown, T., Johnson, M. P., Porta, T. L., Sarioz, D., Verma, D., et al. (2007, October). Robust and efficient coverage in dense sensor deployment. *ITA technical paper*.
62. Yu, H., Iyer, J., Kim, H., Kim, E. J., Yum, K. H., & Mah, P. S. (2006). *Assuring k-coverage in the presence of mobility in wireless sensor networks*. San Francisco, CA: IEEE GLOBECOM.
63. Leibnitz, K., Abeyweera, I. S., Wakamiya, N., & Murata, M. (2008). A heuristic approach for *k*-coverage extension with energy-efficient sleep scheduling in sensor networks. In *3rd international conference on bio-inspired models of network, information, and computing systems*.
64. Tang, S., Mao, X., & Li, X. Y. (2009, March). Optimal k-support coverage paths in wireless sensor networks. In *IEEE international conference on pervasive computing and communications*. Galveston, TX, pp. 1–6.
65. Kumar, S. (2006). Foundations of coverage in wireless sensor networks. *PhD thesis*, Ohio State University.
66. Ssu, K. F., Wang, W. T., Wu, F. K., & Wu, T. T. (2009, March). *k*-barrier coverage with a directional sensing model. *International Journal on Smart Sensing and Intelligent Systems, 2*(1), 75–93.
67. Wang, J. (2008, November). Communication protocols and sensing coverage in mobile ad hoc and wireless sensor networks. *PhD thesis*. Washington State University.
68. Zaidi, S. A. R., Hafeez, M., McLernon, D. C., & Ghogho, M. (2008). *A probabilistic model of k-coverage in minimum cost wireless sensor networks*. Madrid, Spain: ACM CoNEXT Conference.
69. Wang, Y., Wang, X., Agrawal, D. P., & Minai, A. A. (2006). *Impact of heterogeneity on coverage and broadcast reachability in wireless sensor networks* (pp. 63–67). Arlington, VA: 15th International Conference on Computer Communications and Networks.
70. Yen, L. H., Yu, C. W., & Cheng, Y. M. (2006). Expected *k*-coverage in wireless sensor networks. *Ad Hoc Networks, 4*, 636–650.
71. Wan, P. J., & Yi, C. W. (2006, June). Coverage by randomly deployed wireless sensor networks. *IEEE Transactions on Information Theory, 52*(6), 2658–2669.
72. Lazos, L., & Poovendran, R. (2006, August). Stochastic coverage in heterogeneous sensor networks. *ACM Transactions on Sensor Networks, 2*(3), 325–358.
73. Lazos, L., & Poovendran, R. (2006, April). Coverage in heterogeneous sensor networks. In *4th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks*, pp. 1–10.
74. Bejerano, Y. (2008, April). Simple and efficient *k*-coverage verification without location information. In *27th IEEE conference on computer communications*. Phoenix, AZ, pp. 291–295.
75. Bai, X., Li, S., & Xu, J. (2010, March, April). Mobile sensor deployment optimization for *k*-coverage n wireless sensor networks with a limited mobility model. *IETE Technical Review, 27*(2), 124–137.
76. Li, J. S., & Kao, H. C. (2010). Distributed *k*-coverage self-location estimation scheme based on voronoi diagram. *IET Communications, 4*(2), 167–177.
77. Thathachar, M. A. L., & Sastry, P. S. (2002). Varieties of learning automata: An overview. *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, 32*(6), 711–722.
78. Narendra, K. S., & Thathachar, M. A. L. (1989). *Learning automata: An introduction*. Upper Saddle River, NJ: Prentice-Hall Inc.
79. Thathachar, M. A. L., & Sastry, P. S. (2004). *Networks of learning automata*. Boston: Kluwer Academic Publishers.
80. Wolfram, S. (2002). *A new kind of science*. Champaign, IL: Wolfram Media Inc.
81. Meybodi, M. R., Beygi, H., & Taherkhani, M. (2004). Cellular learning automata and its applications. *Journal of Science Technology*, Sharif (Sharif University of Technology, Tehran, Iran), pp. 54–77.
82. Meybodi, M. R., & Taherkhani, M. (2001, May). Application of cellular learning automata in modeling of rumor diffusion. In *Proceedings of 9th conference on electrical engineering*. Power and Water Institute of Technology, Tehran, Iran, pp. 102–110.
83. Beigy, H., & Meybodi, M. R. (2003). A self-organizing channel assignment algorithm: A cellular learning automata approach. *Springer-Verlag Lecture Notes in Computer Science, 2690*, 119–126.
84. Meybodi, M. R., & Mehdipour, F. (2004, Summer). Application of cellular learning automata with input to VLSI placement. *Journal of Modarres*, University of Tarbeit Modarres, Vol. 16, pp. 81–95.
85. Beigy, H., & Meybodi, M. R. (2004, September, December). A mathematical framework for cellular learning automata. *Advances in Complex Systems, 7*(3, 4), 295–319.
86. Beigy, H., & Meybodi, M. R. (2010). Cellular learning automata with multiple learning automata in each cell and its applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics, 40*(1), 54–66.
87. Beigy, H., & Meybodi, M. R. (2007). Open synchronous cellular learning automata. *Advances in Complex Systems, 10*(4), 1–30.
88. Beigy, H., & Meybodi, M. R. (2008, May) Asynchronous cellular learning automata. *Automatica, Journal of International Federation of Automatic Control, 44*(5), 1350-1357.
89. Esnaashari, M., & Meybodi, M. R. (2008). A cellular learning automata based clustering algorithm for wireless sensor networks. *Sensor Letters, 6*(5), 723–735.
90. Esnaashari, M., & Meybodi, M. R. (2010). Dynamic point coverage problem in wireless sensor networks: A cellular learning automata approach. *Journal of Ad hoc and Sensors Wireless Networks*, *10*(2–3), 193–234.
91. Ghaderi, R., Esnaashari, M., & Meybodi, M. R. (2010, February 20–22). Maintaining coverage and connectivity in sensor networks: A cellular learning automata approach. In *Proceedings of the 15th annual CSI computer conference (CSICC'10),* Tehran, Iran.

92. Shannon, C. E. (1948, July, October). A mathematical theory of communication. *Bell System Technical Journal, 27*, 379–423, 623–656.
93. Bettstetter, C., & Krause, O. (2001, August). On border effects in modeling and simulation of wireless ad hoc networks. In *IEEE international conference on mobile and wireless communication networks*. Recife, Brazil, pp. 20–27.
94. Sobeih, A., Chen, W. P., Hou, J. C., Kung, L. C., Li, N., Lim, H., et al. (2006). J-Sim: A simulation and emulation environment for wireless sensor networks. *IEEE Wireless Communications, 13*(4), 104–119.
95. Zhou, Y., Schembri, J., Lamont, L., & Bird, J. (2009). *Analysis of Stand-Alone GPS for relative location discovery in wireless sensor network* (pp. 437–441). Newfoundland, Canada: Canadian Conference on Electrical and Computer Engineering.
96. Lee, H., Dong, H., & Aghajan, H. (2006, September). Robot-assisted localization techniques for wireless image sensor networks. In *IEEE international conference on sensor, mesh, and ad hoc communications and networks (SECON)*, pp. 383–392.
97. Ramadurai, V., & Sichitiu, M. L. (2003). Localization in wireless sensor networks: A probabilistic approach. *International conference on wireless networks*, pp. 275–281.
98. Yap, T. N., & Shelton, Ch. R. (2008). Simultaneous learning of motion and sensor model parameters for mobile robots. In *Proceedings of IEEE international conference on robotics and automation*, pp. 2091–2097.

Author Biographies

Mehdi Esnaashari received the B.S., M.S., and Ph.D. degrees in Computer Engineering all from the Amirkabir University of Technology, Tehran, Iran, in 2002, 2005, and 2011 respectively. His research interests include Computer networks, learning systems and soft computing.

M. R. Meybodi received the B.S. and M.S. degrees in Economics from the Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degrees from the Oklahoma University, USA, in 1980 and 1983, respectively, in Computer Science. Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA. His research interests include, channel management in cellular networks, learning systems, parallel algorithms, soft computing and software development.