Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 6, No. 3, August 2015), Pages: 107-124
www.jacr.iausari.ac.ir

Coverage Improvement Using GLA (Genetic Learning Automata) Algorithm in Wireless Sensor Networks

Shirin Khezri[1], Amjad Osmani✉[2], Behdis Eslamnour[3]

1) Department of Computer Engineering, Payame Noor University, PO BOX 19395-3697, Tehran, i.r of Iran

2) Department of Computer Engineering, Saghez Branch, Islamic Azad University, Saghez ,Iran

3) Department of Electrical and Computer Engineering, Urmia University, Urmia, Iran

Sh.khezri@pnu.ac.ir; amjad.osmani@yahoo.com; b.eslamnour@urmia.ac.ir

Received: 2014/06/03; Accepted: 2015/01/03

Abstract

Coverage improvement is one of the main problems in wireless sensor networks. Given a finite number of sensors, improvement of the sensor deployment will provide sufficient sensor coverage and save cost of sensors for locating in grid points. For achieving good coverage, the sensors should be placed in adequate places. This paper uses the genetic and learning automata as intelligent methods for solving the blanket sensor placement. In this paper an NP-complete problem for arbitrary sensor fields is described which is one of the most important issues in the research fields, so the proposed algorithm is going to solve this problem by considering two factors: first, the complete coverage and second, the minimum used sensors. The proposed method is examined in different areas using MATLAB. The results confirm the successes of using this new method in sensor placement; also they show that the new method is more efficient than other methods like FAPBIL and MDPSO in large areas

Keywords: *Genetic Algorithms, Learning Automata, Wireless Sensor Networks, Sensor deployment.*

1. Introduction

Wireless sensor networks consist of certain amount of small and energy constrained nodes [1], [2], [3]. A typical wireless sensor network consists of thousands of sensor nodes, deployed either randomly or according to some predefined statistical distribution, over a geographic region of interest. A sensor node by itself has severe resource constraints, such as low battery power, limited signal processing, limited computation and communication capabilities, and a small amount of memory. However, when a group of sensor nodes collaborate with each other, they can accomplish a much bigger task efficiently. One of the primary advantages of deploying a wireless sensor network is its low deployment cost and freedom from requiring a messy wired communication backbone [1], [4].

For instance, a sensor network can be deployed in a remote island for monitoring wildlife habitat and animal behavior [5- 6], or near the crater of a volcano to measure temperature, pressure, and seismic activities. In many of these applications the

environment can be hostile where human intervention is not possible and hence, the sensor nodes will be deployed randomly or sprinkled from air and will remain unattended for months or years without any battery replacement. Therefore, energy consumption or, in general, resource management is of critical importance to these networks.

Sensor deployment strategies play a very important role in providing better QoS, which relates to the issue of how well each point in the sensing field is covered.
The coverage problem can be addressed in two main categories defined by Gage [7] in Figure 1:

1. Blanket coverage — to achieve a static arrangement of sensor nodes which maximizes the detection rate of targets appearing in the sensing field.
2. Sweep coverage — to move a number of sensor nodes across a sensing field, such that it addresses a specified balance between maximizing the detection rate and minimizing the number of missed detections per unit area.

This paper will focus mainly on the Blanket coverage, where the objective is to deploy sensor nodes in strategic ways such that optimal area coverage is achieved according to the needs of the underlying applications.

Figure 1. Coverage problem in two main categories

2. Related Works

Several deployment strategies have been studied for achieving an optimal sensor network architecture which would minimize cost, provides high sensing coverage, be resilient to random node failures, and so on. Some of the deployment algorithms try to find new optimal sensor locations after an initial random placement and move the sensors to those locations, achieving maximum coverage. These algorithms are applicable to only mobile sensor networks. Research has also been conducted in mixed-sensor networks, where some of the nodes are mobile and some are static and approaches are also proposed to detect coverage holes after an initial deployment and to try to heal or eliminate those holes by moving sensors [8],[9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [37].

In [37] the authors present how the mobility control can increase the coverage. They describe the model related to the sensing, coverage, and connectivity. [37] describes the evolutionary algorithms for optimization coverage and the classification of mobility exploited coverage is described. In particular, a concept of the coverage holes is explained in detail. The authors in [37] describe the dynamic optimization coverage using evolutionary algorithm with mobility to improve the network coverage.

In [26] a heuristic algorithm is proposed based on Simulation Annealing Algorithm to solve this problem considering the coverage and cost limitations.

In [27] they have used the Distribution Estimation Algorithms named LAEDA on sensor placement. In Learning Automata Estimation Distributed Algorithm (LAEDA), the independency of genome variables is assumed. In these algorithms a Learning Automata is used for each variable in genome. The number of actions of Learning Automata equals to the number of permitted values of corresponding variable of Learning Automata. For production of each genome sample, the Learning Automata of each variable is asked to select its own suitable action; afterwards, they give a corresponding value of selected action to the corresponding variable. Though, they can calculate the probability of a genome's production $X = (x_1, ..., x_n)$ based on equation (1).

$$p(X = x) = \prod_{i=1}^{n} p(X_i = x_i) = \prod_{i=1}^{n} Grad_i^j \qquad (1)$$

where, $1 \le j \le r_i$. So $Grad_i^j$ is equal to the probability of action of corresponding j to value of x_i by ith Learning Automata. By applying Automata in each stage, a number of N individual genomes are created, which is compatible with the number of population.Then the new population of genomes is evaluated using Evaluation Function, and Se genomes which are considered as the best genomes are chosen from this population. After applying some mechanisms which are dependent on Learning Automata Environment Model, a reinforcement signal vector is created and the learning process is applied to each Learning Automata. Having accomplished the learning process, a new generation is produced and the above stages will be continued until a termination condition is satisfied.

Another model of probability distribution estimation algorithms is Population Based Incremental Learning [28], [29] that is a technique which combines aspects of Genetic Algorithms and simple competitive learning. Like the GA, PBIL represents the solution set as a population set of solution vectors. In general, each solution vector in the population set, called an individual, is a possible solution of the problem. The population is produced randomly according to the probabilities specified in the probability vector. The population is evaluated and the knowledge about composing of the best individual in the population is acquired and then the probability vector is updated by pushing it towards generating good individuals in the population. After the probability vector being updated, a new generation population is produced according to the updated probability vector, and the cycle is continued until the termination condition is satisfied.

In [30] a Fuzzy Adaptive Population-Based Incremental Learning algorithm (FAPBIL) is presented based on analyzing the characteristics of traditional PBIL algorithm. Overcoming disadvantages of traditional PBIL algorithm, the proposed FAPBIL algorithm can adjust learning rate and mutation probability automatically according to the evolution degree of the algorithm's searching performed using Fuzzy Controller.

In [31], [32] the Modified Binary Particle Swarm Optimization algorithm is applied for solving the problem of sensor placement in distributed sensor networks. PSO is an inherent continuous algorithm, and the discrete PSO is proposed to be adapted to discrete binary space.

In [33], the SA (simulated annealing) algorithm is characterized by a rule for randomly generating a new solution in the neighborhood of the current solution. SAGLA [33] (Simulated Annealing + Genetic + Learning Automata) uses Simulated Annealing, Genetic and Learning Automata.

The rest of this work is organized as following. Learning automata and genetic algorithms are described in summary in section 3. In sections 4, 5 the sensor placement problem is described. Section 6 is about the proposed algorithm. The performance evaluations are in Section 7 and Section 8 concludes this paper.

3. About Used Intelligence Algorithms

Genetic and Learning Automata (LA) Algorithms are general-purpose stochastic optimization methodologies for solving search problems. These two techniques can both be constructed for asymptotical convergence to the global optimal solution through proper choice of their control parameters. The techniques share the following two principle properties.

1) Probabilistic operators are used by both of these approaches to efficiently explore regions of the search space where the probability of finding improvements in performance is high.
2) The algorithms only require the evaluation of an objective function to guide their search with no additional derivative or auxiliary knowledge required.

The GA is a form of evolutionary algorithm that uses a population of trial solutions to search the encoded space of interest. Through application of reproduction, mutation, competition, selection, and recombination operators, new solutions are generated which are then evaluated and the simulated evolution process iteratively repeated. Though originally proposed as a general model of adaptive processes, they have emerged in recent years as one of the leading methodologies for search and optimization problems involving high-dimensional search spaces. Discrete stochastic learning automata were originally developed to model the behavior of biological systems and have since been developed as models of learning systems where they have been extensively studied [34],[35]. They are typically used as the basis of a learning system that, through interaction with a stochastic and unknown environment, dynamically learns the optimal action for that environment. The learning automaton tries to determine, iteratively, the optimal action to apply to the environment from a finite number of actions that are available to it. The environment is generally noisy; therefore, repeated evaluation of the actions must be made before their true affects can be accurately determined.

The environment returns a reinforcement signal that provides some relevant and predefined measure of performance of the action in the environment. It is this measure of the (sometimes relative) success or failure of the action that is used to change the probability of selecting actions at future iterations.

3.1 Learning Automata

Learning automata operate through interaction with a random or unknown environment by selecting actions in a stochastic manner, using reinforcement to improve some relevant and predefined measure (or measures) of system performance.

Figure 2 shows a typical learning system layout. For a single learning automaton system, the automaton selects an action, probabilistically from a discrete set, which is then evaluated in the environment. The performance evaluation function then provides a signal representing the effectiveness of the selected action on the environment. This is employed by the automaton to update an internal probability distribution that is utilized for future action selection. Those actions that have produced an improvement in system performance (in relation to other actions) have their corresponding selection probabilities increased. Those actions that have degraded the system performance are generally penalized and the probability of their future selection is reduced, although some probability update rules leave the selection probabilities unchanged in this case.

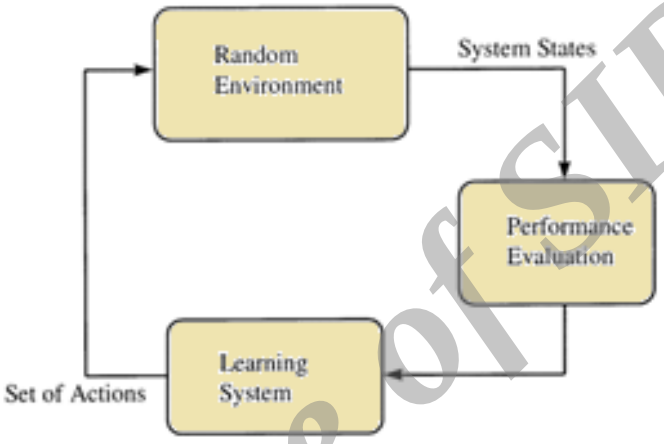

Figure 2. Learning Automata System

The learning automata algorithm is shown in Figure 3. Initially, since no information is known about the actions, they are all given an equal probability of being selected. An action is then selected using this distribution and applied to the environment. This is evaluated and a cost β, a value between zero and one, is returned. This performance evaluation signal β is then employed by the learning automata rules to update the probability distribution that is then used to select the next action.

Begin

1- Initialize P to $[1/r, 1/r, \ldots, 1/r]$

2- Select an action α_i *based on the probability distribution P*

3- Evaluate action and return a cost β

4- Update probability distribution using one of the learning automata rules e.g. L_{RP}

5- If max probability < Threshold (typically 0.999) then

Go to step 2

6- Else

The best action is the one with the highest probability

End

Figure.3- Stochastic Learning Automata (SLA)

1.2 Genetic Algorithm (GA)

The GA, as first introduced by Holland [36], is a general adaptive search and optimization technique that is loosely based on the principle of Darwinian natural selection and on the mechanisms of evolution in nature. The population undergoes a process of simulated evolution involving reproduction, mutation, recombination, competition, and selection operators. Through the use of these operators, the population converges, over many generations (iterations), to a population of fitter individuals. The chromosome is typically a fixed-length bit string, where each bit position is called an allele. In multidimensional optimization problems, the bit string of the chromosome is used to encode the values for the different parameters being optimized. While the chromosome encodes the genetic information or genotype, the phenotype represents its effect in the environment and is measured by a performance index or fitness function. It is this function that is optimized by the simulated evolution process.

The GA, as depicted in Figure 4, involves the following cycle:

Begin

1) Evaluate the fitness of all of the individuals in the population.

2) Create a new population using fitness-proportionate reproduction.

3) Apply the genetic operations such as crossover and mutation to individuals in the new population.

4) Discard the old population and iterate using the new population.

The cycle is then repeated where an iteration of this loop is referred to as a generation.

End

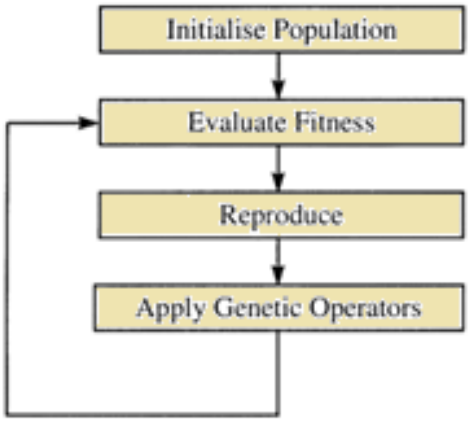

Figure 4. Flowchart for Genetic Algorithm.

The initial population consists of randomly generated individuals. From then on, the genetic operations, in concert with the fitness measure, operate to modify the population. The GA can, with suitable selection and variation operators, be constructed to asymptotically converge to the globally optimal solutions. Fitness-proportionate reproduction and the primary genetic operators of crossover and mutation are also required.

4. Definition of Problem

A grid-based sensor field can be represented as a collection of two- or three-dimensional grid points [20]. A set of sensors can be deployed on the grid points to monitor the sensor field. In this paper the detection model of a sensor is considered to be a 0/1 coverage model. The coverage is assumed to be full (1) if the distance between the grid point and the sensor is less than the detection radius of the sensor (rd). Otherwise, the coverage is assumed to be ineffective (0). If any grid point in a sensor field can be detected by at least one sensor, the field is called completely covered, as shown in Figure 5. A power vector is defined for each grid point to indicate whether sensors can cover a grid point in a field. As Figure 5 shows, the power vector of grid point 1 is (1, 0, 0, 0) corresponding to sensor 2, 8, 9 and 15. In a completely covered sensor field, when each grid point is identified by a unique power vector, the sensor field is said to be completely discriminated, as shown in Figure 5. Sometimes, due to some resource limitations, a completely discriminated sensor field cannot be constructed. Consequently, these may lead to wrong determinations, when a target occurs at any one of the indistinguishable grid points. Positioning accuracy, therefore, becomes a major consideration in solving the problem. Distance error is one of the most natural criteria to measure positioning accuracy. The distance error of two indistinguishable grid points is defined as the Euclidean distance between them.

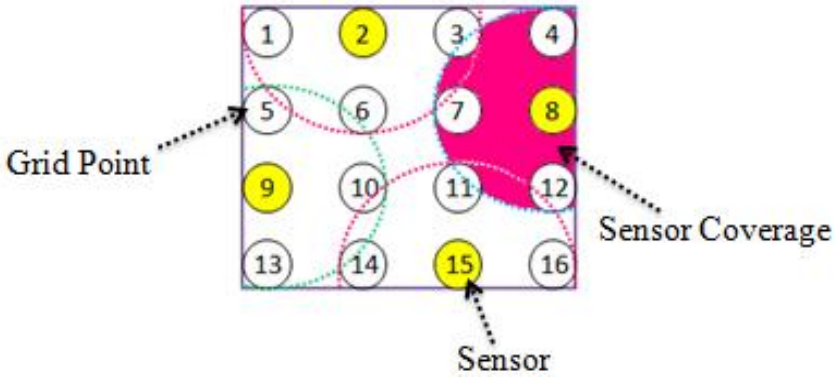

Figure 5. A complete covered and discriminated sensor field with radius =1

5. Mathematical Model

The sensor placement problem is formulated herein as a combinatorial optimization problem. Complete discrimination requires that the minimum Hamming distance of the power vectors associated with any pair of grid points be at least one. High discrimination requires that the maximum distance error be minimized. The problem is, therefore, defined as a min-max model.

5.1 Given Parameters:

A= {*1, 2, …, m*}: Index set of the sensors' candidate locations.
B= {*1, 2, …, n*}: Index set of the locations in the sensor field, $m \le n$.
r_k : Detection radius of the sensor located at $k, k \in A$.
d_{ij} : Euclidean distance between location *i* and $j, i, j \in B$.
c_k : The cost of the sensor allocated at location *k*, $k \in A$.
G: Total cost limitation.

5.2 Decision Variables:

y_k : 1, if a sensor is allocated at location *k* and 0 otherwise, $k \in A$.

$v_i = (v_{i1}, v_{i2}, \ldots, v_{ik})$: The power vector of location *i*, where v_{ik} is 1 if the target at location *i* can be detected by the sensor at location *k* and 0 otherwise, where $i \in B$, $k \in A$.

5.3 Objective Function:

Objective Function is cost limitation and the complete coverage. The cost limitation formula is based on equation 4.
Subject to:

$$y_k = \begin{cases} 1, & \text{if a sensor is allocated at location } k \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

$$v_{ik} = y_k \quad , \forall k \in A, i \in B, i \neq K \tag{3}$$

$$\sum_{k=1}^{m} c_k y_k \leq G \tag{4}$$

$$\sum_{k=1}^{m} v_{ik \geq 1} \quad , \forall i \in B \tag{5}$$

If a target appears at grid point *i* and the grid is inside the coverage of sensor *k*, the sensor can detect the target if sensor *k* is available. Constraint (2) is an integer constraint. *K* is an arbitrarily large number.
Constraint (4) requires that the total deployment cost of sensors be limited by cost *G*. Constraint (5) is the complete coverage limitation.

6. Proposed Intelligent Algorithm

Herein a new intelligent algorithm for better placement of sensors in sensor field area is described. GLA (Genetic + Learning Automata) uses Genetic and Learning Automata. The optimized placement of sensors in this paper is how some grid points are chosen to hold the sensors. Our algorithm represents the relationship between the potential answers and the chromosomes. So the chromosomes of the potential answers $\Phi_t (t = 1, 2, ..., T)$ can be represented by binary bits, that is $\Phi_t \{s_1, s_2, s_3 ..., s_m\}, s_k \in \{0, 1\}$ Where *m* is the number of all candidate grid points, s_k is the decision variable which indicates whether a sensor is allocated to the candidate grid point *k* (s_k = 1 means allocated, s_k = 0 means not allocated).

GLA generates populations of chromosomes and computes their fitness. Then the algorithm orders answers based on fitness values. Each chromosome is generated based on a probability vector. First population is generated randomly. That is to say, the bits (as genes) in each binary string (as chromosome) are selected randomly and set to 1 or 0. It means probability vector is set to 0.5 for each gene of each chromosome. At the end of each iteration the probability vector is updated and used for regenerating new population at the first of next round. So it means a probability vector in background is used for each chromosome. Figure 6, illustrates how probability vector maps to sensor grid state where 1 means a sensor is allocated to a grid point and vice versa.

The GLA generates each population based on probability vector. It is important to say that each chromosome can map to a state for sensor grid and it must satisfy complete coverage. Our proposed Algorithm computes fitness of each chromosome using fitness function in formula (4).

In GLA, the state with minimum fitness value is selected as the best state and its value as f_{best}. GLA uses usual genetic operators like crossover and mutation. In mutation operator probability vector can be reset to 0.5 or *1-P*. This operation is performed with the probability of *Pm(*probability of mutation*)*. Crossover in a genetic algorithm, which selects the genes from the parent chromosomes and creates a new offspring is the key operator. Single parent or two parents can be used for creating new generation. In this paper, two parents method is used. In this type the algorithm uses probability vectors of two parents in current population as parents of crossover operation for generation of two

children. The algorithm performs crossover operation like Figure 7. This operation is performed with the probability of *Pc(*probability of crossover*)*.

The algorithm uses learning automata (formula (6)) for updating probability vector for next round.

$$Pi(n+1)=\begin{cases} Pi(n)+\theta\beta(n)(1-Pi(n)) & (6.1) \\ Pi(n)-\theta\beta(n)(Pi(n)) & (6.2) \end{cases}$$

where θ is learning rate and β(n) is computed using formula (7).

$$\beta(n)=\frac{fi(n)-\min(f)}{\max(f)-\min(f)} \qquad (7)$$

where *min(f)* and *max(f)* are minimum and maximum fitness values in current generated population. *fi(n)* is fitness value of chromosome *i* in nth generation. *Pi(n)* means probability vector of chromosome *i* in *n*th generation. It is computed for each gene in each chromosome *i*. Therefore it is updated based on formula (6. 1) if gene j^{th} of chromosome *i* is 1, otherwise formula (6. 2) is used.

Figure 8 Shows the GLA steps by pseudo code.

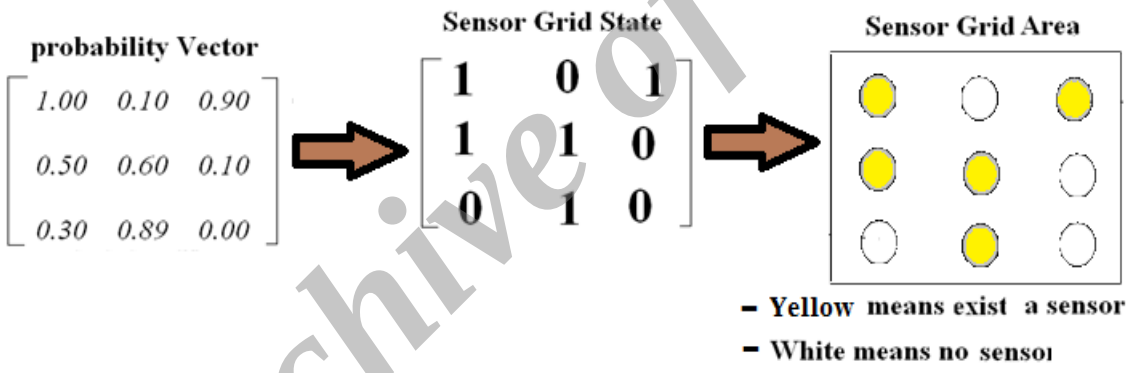

Figure.6- An example for mapping probability vector to sensor grid area

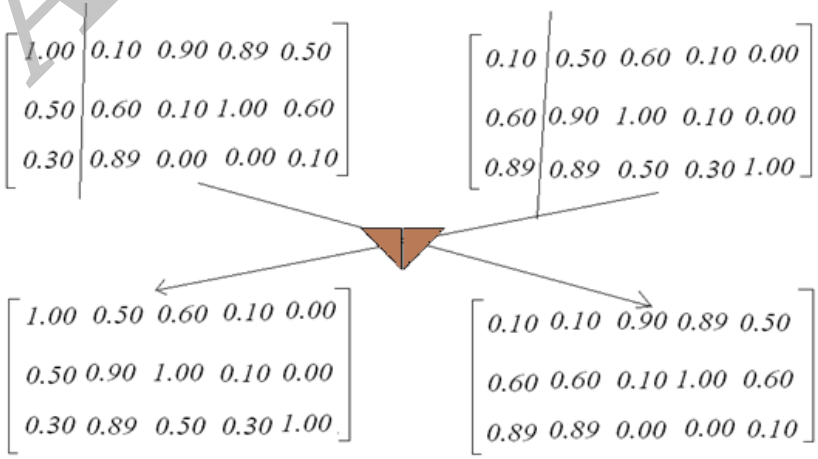

Figure.7- An example for performing crossover operation

Begin

1 Generating the first population(with t1 chromosome)

1.1 Generating the first probability vector of each chromosome

1.1.1 Where Pi(n=1)= 0.5 (i ∈ t1)

1.2 Generating the first state of each chromosome using the first probability vector

2 Computing fitness of each chromosome using formula (4)

3 Selecting best chromosome

3.1 Where f_{best}= best state value

4 Updating probability vector of each chromosome by learning automata and genetic operators

Repeat

5 Generating the next population

5.1 Generating next state of each chromosome using updated probability vector

6 Computing fitness of each chromosome using formula (4)

7 Comparing new best chromosome and last best chromosome

7.1 Where f_{best}= smallest state value

8 Updating probability vector of each chromosome by learning automata and genetic operators

Until { number of generated populations reaches to a desired value }

End

Figure. 8- GLA pseudo code for Sensor Placement

7. Experimental Section

This section presents the computational results. Firstly the performance of the proposed algorithm is evaluated when small sensor fields are deployed. The purpose of this experiment is to examine whether the algorithm can find the optimal solution in small sensor fields. Then, the performance results in the case of larger sensor fields are presented.

The radius of each sensor is one. In MDPSO it is assumed that the population size is 30, $c_1 = c_2 = 2$ and $v_{\max} = 6$. The value of w is considered between 0.2 and 0.9. The parameters of FAPBIL, PBIL and LAEDA are set as Table 1. In the table, Pop-Size means population size in each generation, *pm* means mutation probability, *LR* means learning rate and *Se* is selection size for next generation across previous generation. In LAEDA and PBIL algorithms, a high value of *Se* genomes was chosen for updating the genome`s probability model. In all experiments, the value of *Se* is assumed as a value equals to half of the population of each generation and Learning Rate is 0.01. In SA algorithm, the parameters of the cooling schedule are $\alpha=0.75$ and $\beta=1.3$. The initial values of r and t are respectively $5n$ and 0.1 and n is the number of grids in the sensor field. The frozen temperature, t_f, is $t_0/30$. In SAGLA algorithm [33], the parameter of the cooling schedule is $\alpha=0.55$. θ is equal to 0.41, *pc*=0.4 and *pm*=0.3 (*pc* means crossover probability and *pm* is mutation probability) and k is 1000. In GLA θ is equal to 0.3, population size is 30, *pc*=0.7 and *pm*=0.4. Detection radius of each sensor (*rd*) is 1. Parameter of *e* in SAGLA is equal to 3.

In simulated algorithms here, as all sensors have the same deployment cost, the cost constraint (4), is considered as a limit on the number of sensors.

Each algorithm is run for 20 times and average results for different areas are calculated and compared. The above methods are examined in a benchmark environment that has been provided by MATLAB.

TABLE1. THE PARAMETERS OF FAPBIL, PBIL AND LAEDA

Methods / Parameters	Pop-Size	Pm	LR	Se
LAEDA	50	-	0.01	Pop/2
PBIL	50	0.2	0.01	Pop/2
FAPBIL	50	Fuzzy Adaptive	Fuzzy Adaptive	-

7.1 Experiment 1

Experiment I, evaluates the performance of the proposed algorithm for small rectangular sensor fields which have no more than 30 grid points. The results are compared with those obtained in SA, MDPSO, FAPBIL and LAEDA and SAGLA. Each algorithm is run 20 times and the average results for different areas are calculated and compared in Figure 9 and Table.2 that confirm the superiority of the proposed algorithm against the SA, SAGLA algorithms considering Sensor Density (in #Sensors) vs. target area parameter.

First, a minimum sensor density is found for a complete covered and discriminated sensor field. Then, an attempt is made to obtain a better result using the proposed algorithm under a sensor density constraint.

Figure 9 shows the number of sensors used by six algorithms when they completely cover the sensor field with various areas. In all cases, the proposed algorithm achieves good deployment with a good sensor density. The required sensor density is between 25% and 37%. Figure 9 confirms the superiority of the proposed algorithm over the SA considering Sensor density (in #Sensors) vs. target area parameter.

7.2 Experiment 2

In this experiment, a large sensor field, with 15 × 15 grid points is considered. The radius of the each sensor is one. Obtained results using the proposed algorithm are compared with the best solution obtained by SAGLA, MDPSO, FAPBIL and LAEDA approaches which is shown in Figure 10.

The best solution that has a minimum objective value is found in 1000 arbitrarily generated solutions. Figure 10 shows the density for the desired solution obtained by the proposed algorithm 71 in 1000 arbitrarily generation. In contrast, the other approaches are associated with a relatively high density (90 and higher). Also GLA reaches to desired solution at the same time or faster than other algorithms which is an important advantage. The proposed algorithm can achieve completely covered placement at a very low sensor density. Proposed algorithm gives better results especially in larger networks compared to other algorithms. It can trade off global search against local search more efficient than others. So Figure 10 confirms the superiority of the proposed algorithm over the PBIL, LAEDA, FAPBIL, MDPSO and SAGLA algorithms. SAGLA results are near to those of GLA but GLA performs significantly better than other algorithms.

TABLE2. COMPARISON BETWEEN GLA AND OTHER ALGORITHMS FOR SOME TARGET AREA VALUES

Area	Number of Sensors						
	SA	LAEDA	MDPSO	FAPBIL	SAGLA	GLA	GLA's Sensor Density (%)
4*3	6	4	4	4	4	4	33
5*3	6	5	4	4	5	5	33
4*4	7	4	4	4	5	4	25
6*3	8	6	5	6	5	6	33
7*3	9	7	7	7	7	7	33
6*4	10	7	7	7	7	8	33
8*3	10	9	7	8	8	8	33
5*5	10	9	7	8	8	8	32
9*3	11	9	8	9	9	10	37
7*4	12	9	8	8	9	10	38
6*5	12	10	9	9	9	9	30
10*3	12	11	9	10	10	9	30

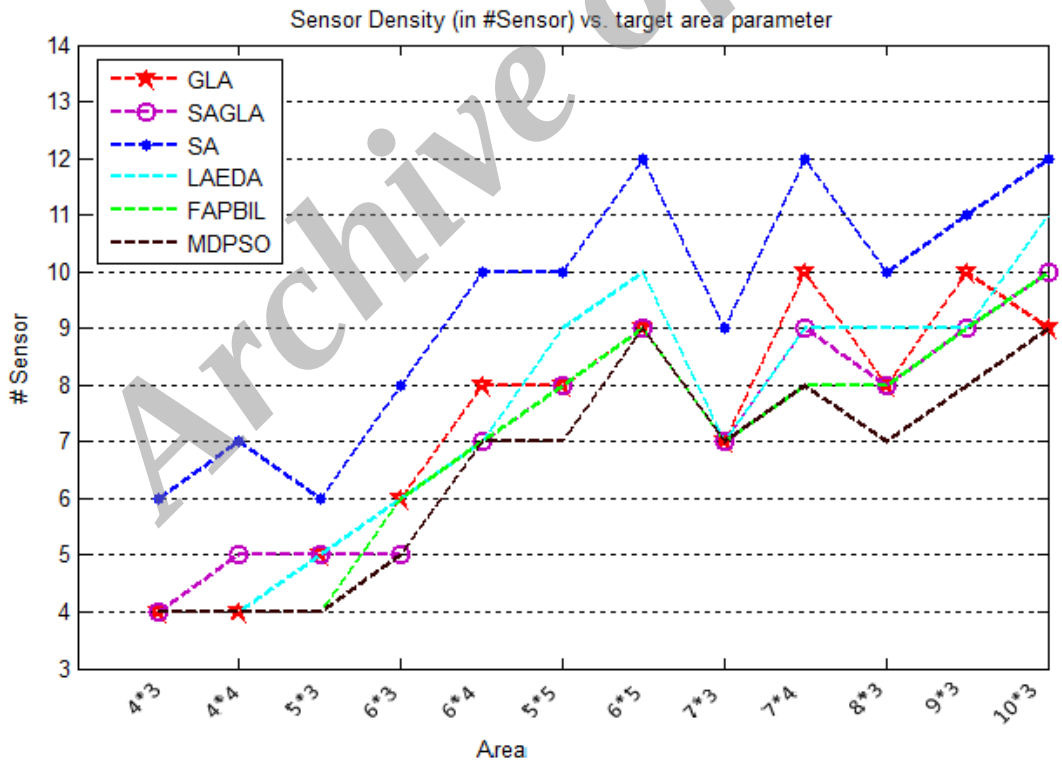

Figure 9. Sensor density (in #Sensor) vs. target area parameter

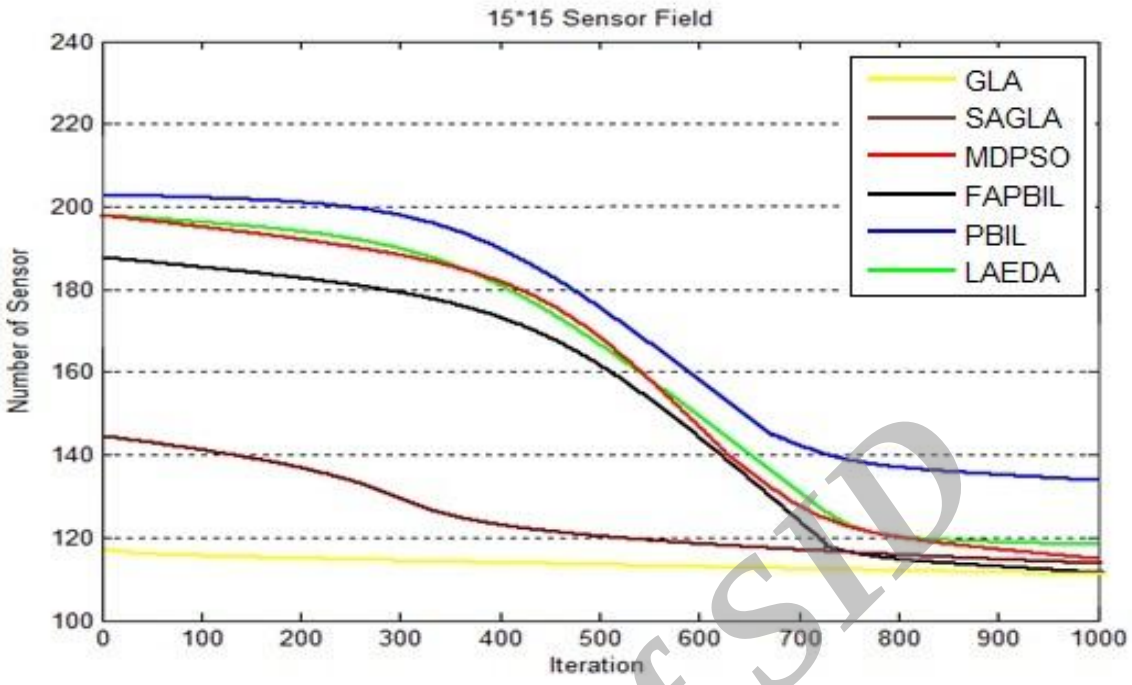

Figure 10. Sensor density (in #Sensor) for 15*15 sensor field

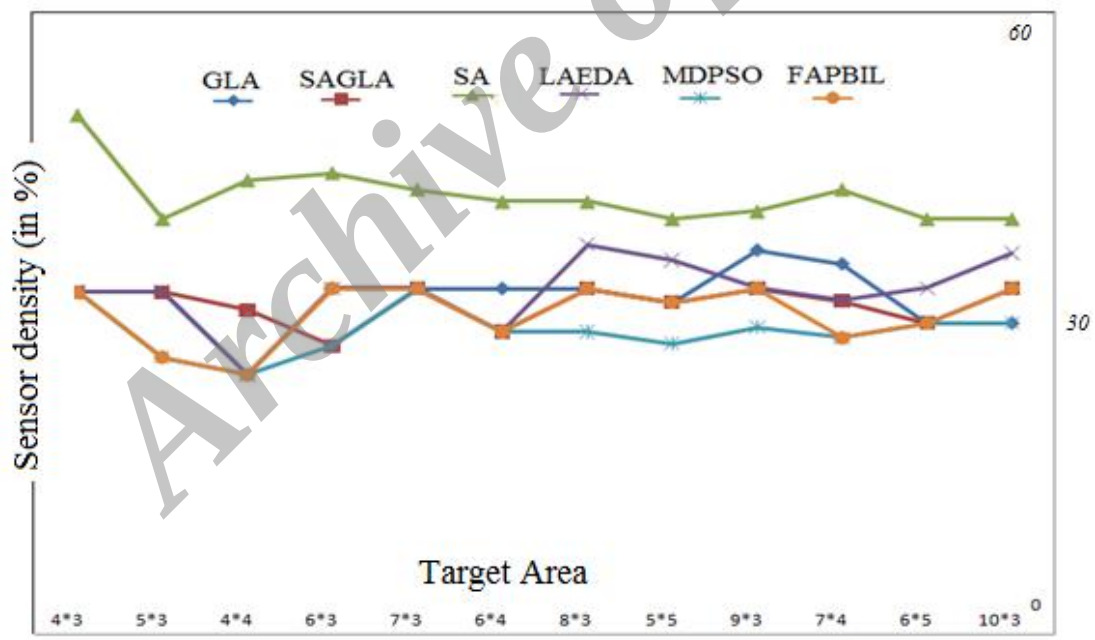

Figure 11. Sensor density (in %) vs. target area parameter

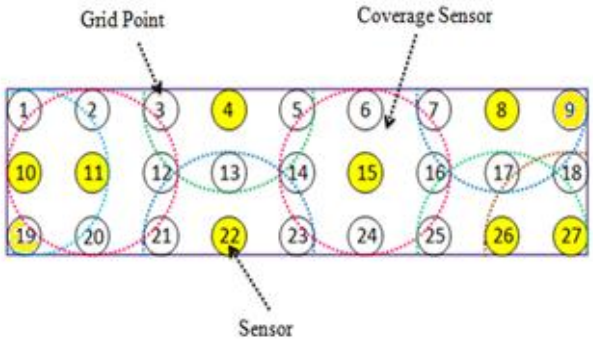

Figure 12. sensor area coverage with 9 × 3 grid points which is obtained using GLA

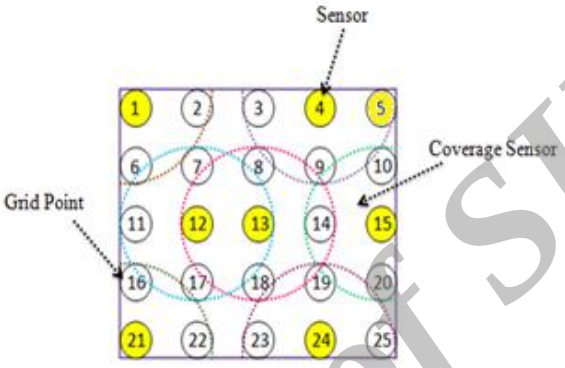

Figure 13. sensor area coverage with 5 × 5 grid points which is obtained using GLA

Figure 11 shows sensor density (in percent) vs. target area parameter. Sensor density is defined in formula (8).

$$Sensor\ density(\%) = \left(\sum_{k=1}^{m} \frac{y_k}{n} \right) \times 100\% \quad (8)$$

where: $y_k = \begin{cases} 1, if\ a\ sensor\ is\ allocated\ at\ location\ k \\ 0, otherwise \end{cases}$

and n is the number of grids in sensor field.

As illustrated in Figure 11, the proposed algorithm has results near to LAEDA, FPBIL and MDPSO. SA algorithm produced the highest sensor density among simulated algorithms in this research.

Figures 12, 13 show sensor areas with 9 × 3 and 5 × 5 grid points when GLA reaches to its solution after 20 runs.

8. Conclusion

This paper describes the sensor deployment problem for locating targets under constraints (complete coverage of sensor network with minimum number of used sensors for coverage). Firstly, an NP-complete problem, then the method of GLA for solving the problem was defined. The results show that GLA is more efficient than other methods like FAPBIL, PBIL and LAEDA in solving the optimization problem in large sensor fields. The proposed algorithm can achieve completely covered placement

at a very low sensor density. Since sensor deployment in the Wireless Sensor Networks (WSN) is important, more efficient intelligent algorithms should be found.

References

[1] Chong, C. Y., Kumar, S. P, "Sensor networks: evolution, opportunities, and challenges", Proceedings of the IEEE, (2003).

[2] Pottie, G.J. X, "Wireless sensor networks" , Proceedings of Information Theory Workshop, (2003).

[3] Pottie, G.J., Caiser, W, "Wireless sensor networks", Proceedings of the Commun .ACM, (2000).

[4] Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E, "Wireless sensor networks A survey", Proceedings of the Comput . Networks, (2002).

[5] Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., Anderson, J., "Wireless sensor networks for habit at monitoring", Proceedings of the 1st ACMInt.Workshop on Wireless Sensor Networks and Applications (2002).

[6] Qi, H., Iyengar, S. S., Chakrabarty, K., "Distributed sensor fusion – a review of recent research", Proceedings of the Journal of the Franklin Institute, (2001).

[7] Gage, D.W., 'Command control for many-robot systems", Proceedings of the 19th Annual AUVS Technical Symp. Reprinted in Unmanned Syst.Mag. (1992).

[8] Zou., Y., Chakrabarty, K., "Sensor deployment and target localization in distributed Sensor networks", Proceedings of the Trans .IEEE Embedded Comput .Syst. (2004).

[9] Wang, G., Cao, G., La Porta, T., "Movement-assisted sensor deployment", Proceedings of the IEEE Transactions On Mobile Computing, (2006).

[10] Bose, P., Morin, P., Stojmenovic, I., Urrutia, J., "Routing with guaranteed delivery in Adhoc wirelesss networks", Proceedings of the Wireless Networks, (2001).

[11] Wang, G., Cao, G., La Porta, T., "Proxy-based sensor deployment for mobile sensor networks", Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (2004).

[12] Yang, Sh., Li, M., Wu, J., "Scan-Based Movement-Assisted Sensor Deployment Methods in Wireless Sensor Networks", Proceedings of the IEEE Transactions On Parallel And Distributed Systems, (2007).

[13] Stojmenovic, I., "A Scalable Quorum Based Location Update Scheme for Routing In Ad Hoc Wireless Network", Technical Report TR-99-09, SITE, University of Ottawa (1999).

[14] WU, X., CHO, J., AURIOL, B. J.D', LEE, S., "Mobility-Assisted Relocation for Self-Deployment in Wireless Sensor Networks', Proceedings of the IEICE TRANS.COMMUN., (2007).

[15] Rahman, Md. O., Razzaque, Md. A., Hong, Ch. S., "Probabilistic Sensor Deployment in Wireless Sensor Network: A New Approach", Proceedings of the ICACT, (2007).

[16] Meguerdichian, S., Koushanfar, F., Potkonjak, M., Srivastava, M. B., "Coverage Problems in Wireless Ad-hoc SensorNetworks', Proceedings of the International Conference on Mobile Computing and Networking (2001).

[17] Padmavathi, M., Yong, R., Yelena, Y., 'Key frame-based Video Summarization using Delanunay Clustering", Proceedings of the International Journal on Digital Libraries, (2006).

[18] http://www.darpa.mil/ato/progarms/shm/index.html (2007)..

[19] Chellappan, S., Bai, X., Mam B., Xuan, D., "Mobility Limited Flip-Based Sensor Networks Deployment", Proceedings of the IEEE Mobile Ad Hoc and Sensor Systems (2005).

[20] Dhillon, S.S., Chakrabarty, K., Iyengar, S.S., "Sensor Placement for Grid Coverage under Imprecise Detections", Proceedings of the Fifth International Conference on Information Fusion, (2002).

[21] Dhillon, S.S., Chakrabarty, K., "Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks", Proceedings of the lEEE WCNC, (2003).

[22] Chakrabarty, K., Iyengar, S.S., Qi, H., Cho, E., . "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks", Proceedings of the IEEE Trans. on Computers, (2002).

[23] Sasikumar, P., Vasudevan, S. K., Vivek, C. , Subashri, V., "Heuristic Approaches with Probabilistic Management for Node Placement in Wireless Sensor Networks", Proceedings of the International Journal of Recent Trends in Engineering, (2009).

[24] Rajagopalan, R., Niu, R., Mohan, Ch. K., Varshney, P. K., Drozd, A.L., "Sensor placement algorithms for target localization in sensor networks", Proceedings of the IEEE Radar Conference, (2008).

[25] Indu, S., Chaudhury, S., Mittal, N. R., Bhattacharyya, A., "Optimal Sensor Placement for Surveillance of Large Spaces", Proceedings of the Third International Conference on Digital Object Identifier, (2009).

[26] Lin, F. Y. S., Chiu, P. L., "A Near-Optimal Sensor Placement Algorithm to Achieve Complete Coverage/Discrimination in Sensor Networks", Proceedings of the IEEE Communication letters, (2005).

[27] Khezri, S., Osmani, A., Gholami, M., "Estimation of Distribution Algorithm Based on Learning Automata for Sensors Placement in Wireless Sensor Networks", Proceedings of the 3nd National Conference on Computer/Electrical and IT Engineering (2011).

[28] Baluja, S., "Population Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning", Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania (1994).

[29] Baluja, S., Caruana, R., "Removing The Genetics from The Standard Genetic Algorithm", Proceedings of ICML, (1995).

[30] Khezri, S., Meybodi, M., Osmani, A., "Fuzzy Adaptive PBIL based Sensor Placement in WSN", Proceedings of the International Symposium on Computer Networks and Distributed Systems, (2011).

[31] Khezri, S., Faez, K., Osmani, A., "Modified Discrete Binary PSO Based Sensor Placement in WSN", Proceedings of the International Conference on Computational Intelligence and Communication Networks, (2011).

[32] Khezri, S., Faez, K., Osmani, A., "An Intelligent Sensor Placement Method to Reach a High Coverage in Wireless Sensor Networks", International journal of grid and high performance computing (2011).

[33] Osmani, A., "Design and evaluation of new intelligent sensor placement algorithm to improve coverage problem in wireless sensor networks", J. Basic. Appl. Sci. Res., (2012).

[34] N. Baba, "New Topics in Learning Automata Theory and Applications", New York: Springer-Verlag, Lecture Notes in Control and Information Sciences (1984).

[35] K. S. Narendra and M. A. L. Thathachar, "Learning Automata: An Introduction". Englewood Cliffs, NJ: Prentice-Hall (1989).

[36] H. Holland, "Adaptation in Natural and Artificial Systems". Ann Arbor, MI: Univ. of Michigan Press (1975).

[37] M. Abbasi, M. S. B. Abd Latiff, and H. Chizari, "Bioinspired Evolutionary Algorithm Based for Improving Network Coverage in Wireless Sensor Networks," The Scientific World Journal, (2014).