# Fuzzy Adaptive PBIL based Sensor Placement in Wireless Sensor Networks

Shirin Khezri
Islamic Azad University,
Mahabad Branch
Dept. Computer & IT
Mahabad, Iran
sh.khezri @ qiau.ac.ir

Mohammad Reza Meybodi
Amirkabir University of
Technology
Dept. Computer & IT
Tehran, Iran
mmeybodi @ aut.ac.ir

Amjad Osmani
Islamic Azad University,
Saghez Branch
Dept. Computer & IT
Saghez, Iran
a.osmani @ iausaghez.ac.ir

*Abstract*— **In this paper, 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. Extensive computational tests are carried out with Sensor Placement problem. The results compared with Random Search algorithm, standard PBIL algorithms and LAEDA show that the proposed algorithm performs them in calculation efficiency and search capability. The proposed algorithm can acquire stable high quality solution.**

***Keywords: Fuzzy logic System; PBIL; Sensor Plasement; Wireless Sensor Networks.***

## I. INTRODUCTION

Evolutionary Algorithms are a subset of stochastic optimization Algorithms. For optimization purposes, these algorithms are based on natural biological evolution rules. We use and apply these probability optimization algorithms for solving parametric optimization problems in which other formal approaches are unable to solve them. A major set of these algorithms are called Genetic Algorithms. Due to their simplicity and because there is no need to apply complicated differential equations, Classical Genetic Algorithms have been applied and used extensively in solving complicated and high-precision problems with rough landscape. Despite enjoying plenty of positive and beneficial advantages, traditional Genetic Algorithms can operate efficiently, only if the variables are independent from each other or situated in close distance inside the genome [13]. In other words, the performance of genetic algorithm is heavily depended on other parameters such as the way the mutation operation determined and production of generation, the probability of generation mutation and production, the size of populations and number of generations produced. As we mentioned earlier, while combining genomes together, not only there exists any improvement in the quality of solutions, but also the algorithm traps in local optimal points. To handle this problem, many models of genetic algorithm were created, which can be divided in to three distinct groups:

The first group includes those algorithms in which genetic operators along with evolutionary process are improving simultaneously and automatically. In second group, the way in which the problem is represented is changed along with the evolutionary process similarly. The third group is known as Estimation Distributed Algorithms. In these kinds of algorithms new genomes are generated using probability distributions of all former permitted solutions instead of using common operator in genetic algorithms. These algorithms are trying to augment the convergence degree toward a suitable solution via creating probability models from variables [1-14].

## II. RELATED WORK

In [15] a set of Stochastic Heuristic Algorithms based on population is presented as Distribution Estimation Algorithm in which they don't need smooth search space and complicated differential equations. In addition, they solved many problems of genetic algorithms. By generating a probability model from constitutive components of a genome in Distribution Estimation Algorithm, the velocity toward finding problem`s optimum solution will be accelerate. In these algorithms, new populations wont generated by using mutation operators and generation production, but new genomes are estimated based on probability distribution and they are created and modeled based on genomes have chosen from former generations. In fact estimating of this distribution function is the most challenging parts of Distribution Estimation Algorithms.

In [16] 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 number of permitted values for the corresponding variable of Learning Automata. For generation 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 $X = (x_1, \dots, x_n)$ 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 , $1 \leq j \leq r_i$, so $Grad_i^j$ equals to probability of action of corresponding $j$ to value of $x_i$ by $i^{th}$ 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 we apply the Learning process in each Learning Automata. Having accomplished the learning process, a new generation is produced and the above stages will be continued until a terminal condition is satisfied.

In [17, 18], they applied Fuzzy Logic System (FLS) to re-deploy the sensors. Each individual mobile sensor uses a FLS to self-adjust its location. Therefore the deployment scheme based on FLSs is a fully distributed approach.

In [19] they apply the modified binary particle swarm optimization algorithm for solving the sensor placement in distributed sensor networks. PSO is an inherent continuous algorithm, and the discrete PSO proposed to be adapted to discrete binary space.

Another model of probability distribution estimation algorithms is Population Based Incremental Learning which firstly proposed by Baluja in 1994 [1, 3]. PBIL is a technique that 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. A simple and universal procedure of the PBIL algorithm is given as following:

***//Initialize probability vector;***

***While (Not termination condition)***

***{***

*//Produce the population of solutions according to Probability vector;*

*//Evaluate the population;*

*//Acquire knowledge from best individual of the Population;*

*//Update the probability vector according to the Knowledge;*

***}***

***//Output the solution***

Figure 1. PBIL algorithm

The three main operators of PBIL used in this paper are: probability vector (PV), Learning rate (LR) and mutation, that LR and Pm are fixed here. The individuals are evaluated according to the objective function. The "best" individual is used to update the probability vector so as to produce solutions similar to the current best individuals.

In general, a fixed learning rate is usually adapted to accumulate probability of each gene of the individual. However, a fixed learning rate and mutation probability cannot ensure that the algorithm search well in the whole performing process. On the one hand, a small learning rate will result in very slow convergent process and a larger leaning rate will bring about that the algorithm cannot find high quality solution. On the other hand, a small mutation probability cannot ensure the population dispersal enough to make algorithm escape from local optima and a larger mutation probability will result in entire stochastic search process. Therefore, it is necessary to choose reasonable parameters to ensure that algorithm performs effectively and finds high quality solution. In fact, it needs different learning rate and mutation probability in different periods of algorithm running.

The rest of this work is organized as following. In section III, we state sensor placement problem. In section IV, we describe mathematic model of sensor placement problem. Section V is about the proposed algorithm. The performance evaluations are in Section VI and Section VII concludes the paper.

## III. DEFINITION OF PROBLEM

The sensor network based on grid-based could be considered as a two or three dimensional network [20]. A set of sensors are settled on the grid points to monitor the sensor area. In this paper, we consider the detection model of a sensor to be a 0/1 coverage model. Now if the Euclidean distance between the grid point and the sensor is less than the detection radius of the sensor ($d<r$), so the coverage is assumed to be full (1), 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, we call the field is completely covered, as shown in Fig. 2. A power vector is defined for each grid point to indicate whether sensors can cover a grid point in field. In Fig. 2, a complete covered and discriminated sensor field of 7*4 with radius =1 is illustrated, that a target can be detected at any place in the field. In Fig.2, the power vector for point 19 is (0, 0, 0, 1, 0, 1, 0, 0) corresponding to sensor 3, 7, 8, 12, 14, 18, 23 and 27. 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 Fig.2. In this case, as soon as a target occurs in a grid of sensor field, it can be located by the back-end according to power vector of the grid.

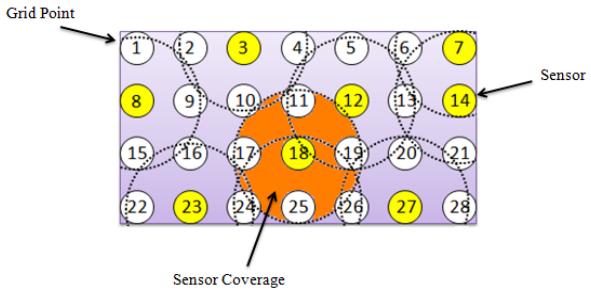


Figure 2. A complete covered and discriminated sensor field with radius =1, achieved by proposed algorithm

## IV. Mathematical Model

The sensor placement problem is a NP-complete problem, and is formulated here as a combinatorial optimization problem. The formulation can plan a sensor network that provides either complete or high, discrimination, depending on the cost limitation.

**Given Parameters:**

$A$= {1, 2… $m$}: Index set of the sensor's candidate locations.
$B$= {1, 2… $n$}: index set of the location in the sensor field, $m \leq n$.
$r_k$: 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 located at $k$, $k \in A$.
$G$: Total Cost limitation

**Decision Variables:**

$y_k$: 1, if a sensor is allocated at location $k$ and 0 otherwise, $k \in A$.
$pv_i = (pv_{i1}, pv_{i2}, \dots, pv_{ik})$: The power vector of location $i$, where $pv_{ik}$ is 1 if the target at location $i$ can be detected by the sensor at location $k$ and 0 otherwise, where $i \in B, k \in A$.

**Objective Function:**

Objective Function is cost limitation and the complete coverage that cost limitation formula is in equation (2).

$$\sum_{k=1}^{m} c_k y_k \leq G \qquad (2)$$

## V. Proposed Algorithm

In evolutionary computing, a maximum evolution generation is usually used as a terminal condition according to experience. However, the evolution and convergence degree are different for different problems. It is almost impossible to simply use the maximum generation to determine the evolution and convergence degree of the algorithm accurately. From analysis of PBIL algorithm we know that probability of each gene will change gradually from initial values to 0 or 1 when algorithm is running and evolving. Determining evolution degree is the same as determining decentralization of probability of each gene in population. Therefore, the convergence degree can be determined through census of the probability value of each gene.

In this paper, to escape local optima, entropy is used to express the evolution degree of the algorithm, and a Fuzzy Adaptive PBIL algorithm is presented. Entropy is usually used to reflect the consistency of the system. It will decrease as the evolutionary process continues. At the end of the evolution process, the entropy should be 0, or at least close to 0, and at the same time the probability of each solution bit (gene) should be towards 0 or 1. So, entropy of the population can be used to determine convergence degree of the algorithm. Suppose that there are $N$ genes in an individual, and each gene has $L$ possible values, $P_{ij}$ denotes the probability that the $i^{th}$ gene takes the $j^{th}$ value, the entropy formula is equation (3).

$$E = -\sum_{i=1}^{N}\sum_{j=1}^{L} P_{ij} \,.\, log2(P_{ij}) \qquad (3)$$

The proposed algorithm adopts adaptive functions to adjust learning rate and mutation probability automatically according to the evolution degree of the algorithm. For learning rate, the adjusting strategy is as following: to take a small learning rate in initial period of algorithm's running in order to ensure population diversity; to take a large learning rate in the middle period of algorithm's running to accelerate evolution, and take a relative little learning rate in the end period of algorithm's performing to ensure algorithm to search subtly and obtain high quality solution. For mutation probability, the adjusting strategy is to take a small mutation probability in initial period of algorithm's running and make mutation probability increase gradually with algorithm evolution to avoid searching process getting into local optima.

To tune the parameters of proposed algorithm we applied Fuzzy Controller. Fig.3 shows structure of fuzzy system.

The inputs of Fuzzy system are as following:

***Entropy***: is equals to the amount of information in probability vector.

***Diff-Fit***: is a norm of distance between the best solution and worst solution.

The outputs of Fuzzy System include:

***LR***: Learning Rate

***PM***: Probability Mutation

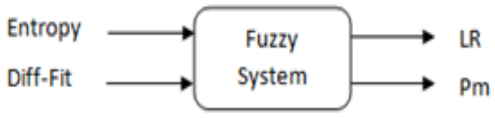


Figure 3. Structure of Fuzzy System

Fig.4 shows Membership functions and Table I presents fuzzy rules.

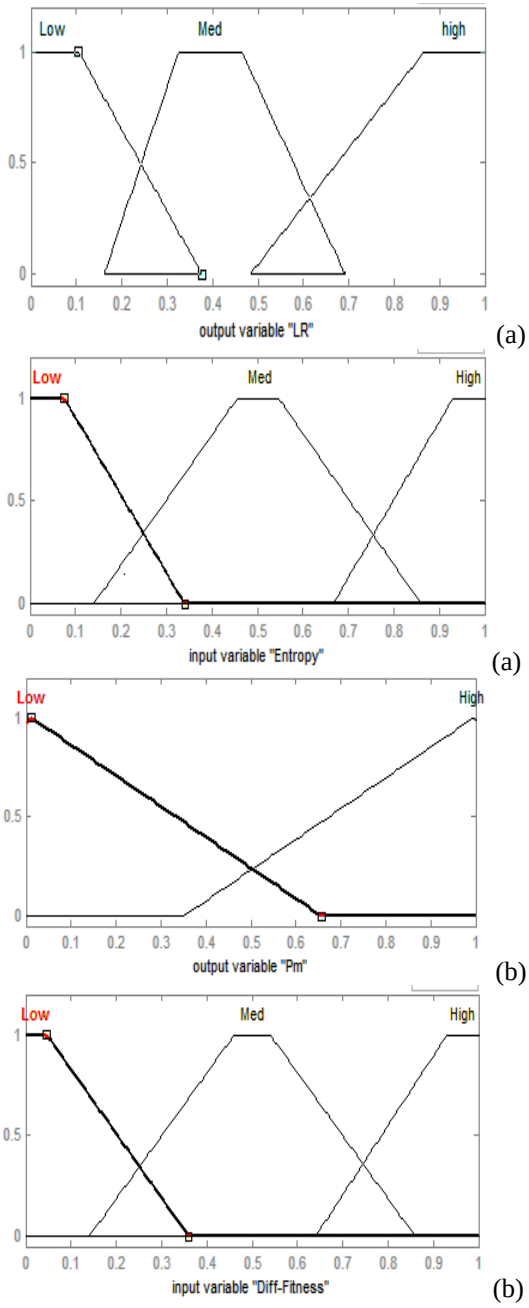


Figure 4. (a) Membership Functions of inputs (b) outputs

The proposed algorithm based Sensor Placement is described in Fig. 5.

TABLE I. FUZZY RULES

| Rules | Entropy | Diff-Fitness | Learning Rate | Probability of Mutation |
|---|---|---|---|---|
| 1 | High | Low | Med | High |
| 2 | High | High | Med | Low |
| 3 | High | Med | High | Low |
| 4 | Low | Low | Low | Low |
| 5 | Low | High | High | Low |
| 6 | Low | Med | Med | High |
| 7 | Med | Low | Med | High |
| 8 | Med | High | High | Low |
| 9 | Med | Med | High | High |

```
/* Initialize the probability matrix P */
For (i=1; i<=N; i++)
        P_ij = 1/L;
While (Not termination condition)
{
 /* Sampling and Evaluation*/
  For (k=1; k<=Pop-Size; k++)
        {
           Repeat
                 Individual [k] =Sample (P);
           Until Coverage is satisfy base on equation 3;
           Fitness [k] = Evaluate (Individual [k]);
         }
Best Individual = Find Best Individual (Fitness);

  /*Active the Fuzzy System for calculate LR & Pm */
 /* Revise P according to Best Individual adaptively */
For (i=1; i<=N; i++)


/* Revise P according to Mutation Probability */
For (i=1; i<=N; i++)
 {
       For (j=1; j<=L; j++)
         If Random (0, 1) < p_m
            P_ij = P_ij + Pm ;
 }
/* normalize the probability matrix P */
For (i=1; i<=N; i++)
  {
     For (j=1; j<=L; j++)
           P_ij = P_ij / Σ_{j=1}^{L} P_ij ;
   }
}
```

Figure 5. FAPBIL Pseudo code for Sensor Placement

## VI. SIMULATION RESULTS

This section presents the computational results. First, the performance of the proposed algorithm is evaluated when small sensor fields are deployed. The purpose of the experiment is to examine whether the algorithm can find the optimal solution under a minimum cost constraint. Then, the performance

results in the case of larger sensor fields are presented under various cost constraints.

The parameters of FAPBIL, PBIL and LAEDA are set as Table II. In the table, *Pop-Size* means population size in each generation, *Pm* means mutation probability, *LR* means learning rate and *Se* is selection genomes for next generation across current generation.

In LAEDA and PBIL algorithms, a high value of Se genomes has chosen for updating the genome`s probability model. In all experiments, we assume the value of *Se* as a value equals to half of population of each generation. Each algorithm runs 10 times for each problem and average results for different areas are calculated and compared in Table III. The algorithms are implemented in Matlab (v 7.6) on a personal computer (3G).

### A. *Experiment I*

Experiment I evaluates the performance of the proposed algorithm for smaller rectangular sensor fields that have no more than 30 grid points. The results are compared with Random Search [21], LAEDA [16] and PBIL [1, 3].

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

Table III shows the number of sensors used by four algorithms when they cover the sensor field with various areas completely. In all cases, the proposed algorithm achieves the best deployment with a minimum sensor density. The required sensor density is between 25% and 33%. Fig.6 confirms the superiority of the proposed algorithm against the PBIL, LAEDA and Random Search algorithms considering Sensor density (in #Sensors) vs. target area parameter.

TABLE II. THE PARAMETERS OF FAPBIL, PBIL AND LAEDA

| Parameters | Pop-Size | Pm | LR | Se |
|---|---|---|---|---|
| LAEDA | 50 | - | 0.01 | Pop/2 |
| PBIL | 50 | 0.2 | 0.01 | Pop/2 |
| FAPBIL | 50 | FA* | FA | - |

**Fuzzy Adaptive*

TABLE III. COMPARISON BETWEEN FOUR ALGORITHMS AND THE

| Area | #Sensors | | | | |
|---|---|---|---|---|---|
| | Random Search | LAEDA | PBIL | FAPBIL | FAPBIL's Sensor Density |
| 4*3 | 6 | 4 | 6 | 4 | 0.33 |
| 4*4 | 7 | 4 | 6 | 4 | 0.25 |
| 6*3 | 8 | 6 | 8 | 6 | 0.33 |
| 6*4 | 10 | 7 | 9 | 7 | 0.29 |
| 7*3 | 9 | 7 | 8 | 7 | 0.33 |
| 8*3 | 10 | 9 | 10 | 8 | 0.33 |
| 9*3 | 11 | 9 | 10 | 9 | 0.33 |
| 5*3 | 6 | 5 | 5 | 4 | 0.26 |
| 5*5 | 10 | 9 | 10 | 8 | 0.32 |
| 6*5 | 12 | 10 | 11 | 9 | 0.3 |
| 7*4 | 12 | 9 | 11 | 8 | 0.28 |
| 10*3 | 12 | 11 | 12 | 10 | 0.33 |

As all sensors have the same deployment cost, the cost constraint, constraint (2), can be express as a limit on the number of sensors. This section use a normalized term, sensor density, in the constraint. Sensor density is defined in equation (4).

$$Sensor\ density(\%) = \left(\sum_{k=1}^{m} \frac{y_k}{n}\right) \times 100\% \quad (4)$$

Where:

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

$and\ n\ \ is\ the\ number\ of\ \ grids\ in\ sensor\ field.$

The proposed algorithm can achieve completely covered placement at a very low sensor density.

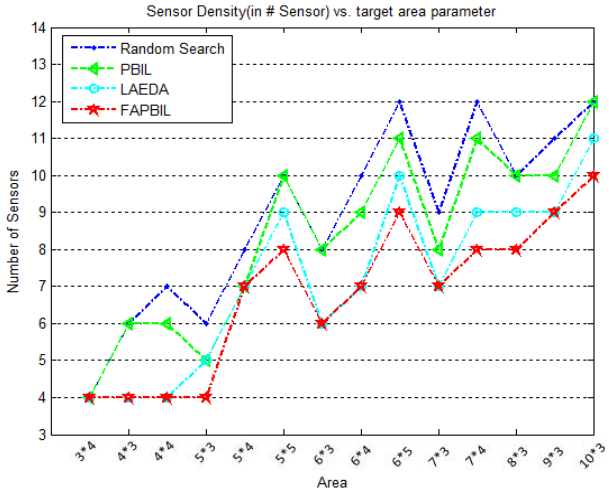


Figure 6. Sensor density (in #Sensors) vs. target area

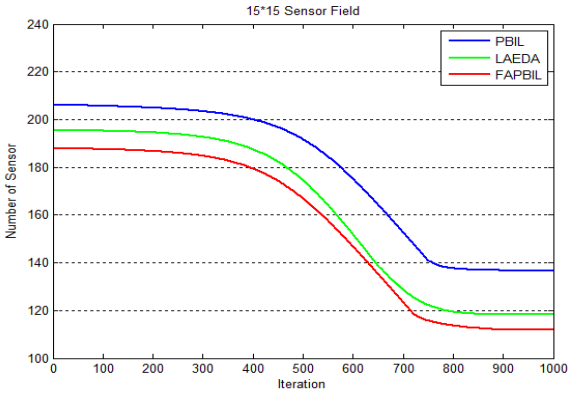


Figure 7. Sensor density (in #Sensors) for 15*15 sensor field

### B. *Experiment II*

In this experiment, a larger sensor area, with 15 × 15 grid points is considered. The radius of each sensor is one. The results obtained using the proposed algorithm is compared with the best solution obtained by the Random Search, LAEDA and PBIL approaches. The best solution that has a minimum objective value is founded in 1000 arbitrarily generated solutions. Fig. 7 shows that the required density for the desired solution obtained by the proposed algorithm 51% in 1000 arbitrarily generation. In contrast, the other approaches are associated with a relatively high density (58% and 62%). The proposed algorithm can achieve completely covered placement at a very low sensor density. Fuzzy Adaptive PBIL gives better results especially in larger networks compared to PBIL algorithm due to using algorithm`s evolution rate, so it can operate better than PBIL, LAEDA and Random search in scalability parameter. It determines the learning rate and probability mutation according to fuzzy rules and as per simulation results FPBIL is able to trade off global search against local search more efficient than PBIL.

## VII. CONCLUSION

This paper proposes an improved version of PBIL algorithm. Based on entropy of the population, the proposed algorithm adjusts learning rate and mutation probability automatically according to the evolution degree of the algorithm using Fuzzy Controller. The algorithm applied to solve Sensor Placement problem and Simulation results show that the proposed algorithm improves PBIL, LAEDA and Random Search. Since sensor placement in the Wireless Sensor Networks (WSN) is important, we should find better intelligent algorithms.

## REFERENCES


[1] 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.

[2] Baluja, S., Davies, S., "*Using Optimal Dependency Trees for Combinatorial Optimization: Learning the Structure of Search Space*", Technical Report CMU-CS-97-107, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1997.

[3] Baluja, S., Caruana, R., "*Removing The Genetics from The Standard Genetic Algorithm*", Proceedings of ICML, pp.38-46, Morgan Kaufmann Publishers, 1995.

[4] De Bonet, J. S., Isbell, C. L., Viola, P., "*MIMIC: Finding Optima by Estimating Probability Densities*", Proceedings of NIPS, pp.424-431, MIT Press, Cambridge, 1997.

[5] Gordon, T. J., Marsh, C., Wu, Q. H., "*Stochastic optimal control of active vehicle suspensions using learning automata*," Journal of Systems and Control Engineering, vol. 207, pp. 143-152, 1993.

[6] Harik, G. R., Lobo, F. G., Goldberg, D. E., "*The Compact Genetic Algorithm*", IEEE Transaction on Evolutionary Computing, vol. 3, no. 4, pp. 287-297, 1999.

[7] Harik, G., "*Learning Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms*", Illinois Genetic Algorithm Report, No. 97005, Illinois University, USA, 1997.

[8] Howell, M. N., Gordon, T. J., Brandao, F. V., "*Genetic Learning Automata for Function Optimization*", IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol 32, No. 6, pp. 804-815, 2002.

[9] Mühlenbein, H., Mahnig, T., "*Evolutionary Algorithms: From Recombination to Search Distributions*", Theoretical Aspects of Evolutionary Computing, Springer Publication, 2001.

[10] Mühlenbein, H., Mahnig, T., "*The Factorized Distribution Algorithm for additively decomposed functions*", Proceedings of the 1999 Congress on Evolutionary Computation, IEEE press, pp. 752-759, 1999.

[11] Pelikan, M., Goldberg, D. E., Cant-Paz, E., "*Linkage Problem, Distribution Estimation and Bayesian Networks*", Evolutionary Computation, vol. 8, no. 3, pp. 311-340, 2000.

[12] Pelikan, M., Goldberg, D. E., Lobo, F., "*A Survey of Optimization by Building and Using Probabilistic Model*", Illinois Genetic Algorithm Report, no. 99018, Illinois University, USA, September 1999.

[13] Riopka, T. P., Bock, P., "*Intelligent Recombination Using Individual Learning in a Collective Learning Genetic Algorithm*", Proceedings of the Genetic and Evolutionary Computation Conf. (GECCO-2000), pp. 104-111, 2000.

[14] Smith, J., Fogarty, T. C., "*Self Adaptation of Mutation Rates in a Steady State Genetic Algorithm*", Proceedings of 3rd IEEE Conf. on Evolutionary Comp. IEEE Press, 1996.

[15] Muhlenbein. H, Paab. G, "*From recombination of genes to the estimation of distributions binary parameters*", Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature PPSN IV, pp. 178-187, 1996.

[16] 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 (CEIT), Iran, 2011.

[17] Osmani, A., Haghighat, A. T., Dehghan, M., Emdadi, P., "*FSPNS: Fuzzy Sensor Placement Based on Neighbors State"*, Proceedings of the International Conference on Computer Modeling and Simulation, (IEEE) UKSIM, England, 2010.

[18] Osmani, A., Dehghan, M., Pourakbar, H., Emdadi, P., "*Fuzzy-Based Movement-Assisted Sensor Deployment Method in Wireless Sensor Networks*", Proceedings of the International Conference on Computational Intelligence, Communication Systems and Networks, (IEEE) CICSYN, India, 2009.

[19] Khezri, S., Faez, K., Osmani, A., "*Modified Discrete Binary PSO Based Sensor Placement in WSN*", International Conference on Computational Intelligence and Communication Networks, (IEEE) CICN, India, 2010.

[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, Vol. 2, No.3, pp. 1581-1587, 2002.

[21] Lin, Frank Y. S. Chiu, P. L., "*A Near-Optimal Sensor Placement Algorithm to Achieve Complete Coverage/Discrimination in Sensor Networks*", IEEE Communication letters, Vol. 9, No. 1, 2005.