

An Estimation of Distribution Improved Particle Swarm Optimization Algorithm

R. V. Kulkarni¹ and #G. K. Venayagamoorthy²

Real-Time Power and Intelligent Systems Laboratory, Department of Electrical and Computer Engineering
University of Missouri-Rolla, MO 65409, USA

¹arvie@ieee.org & ²gkumar@ieee.org

Abstract

PSO is a powerful evolutionary algorithm used for finding global solution to a multidimensional problem. Particles in PSO tend to re-explore already visited bad solution regions of search space because they do not learn as a whole. This is avoided by restricting particles into promising regions through probabilistic modeling of the archive of best solutions. This paper presents hybrids of estimation of distribution algorithm and two PSO variants. These algorithms are tested on benchmark functions having high dimensionalities. Results indicate that the methods strengthen the global optimization abilities of PSO and therefore, serve as attractive choices to determine solutions to optimization problems in areas including sensor networks.

1. INTRODUCTION

Particle swarm optimization algorithm (PSO), models the dynamics of societies of biological specimen like birds, insects and fish. It is a population based optimization technique in which a collection of test solutions interact with each other and search for the best solution to the given problem [1].

PSO consists of a population (or swarm) of particles, each of which represents an n dimensional potential solution. Particles are assigned random initial positions and they change their positions iteratively to reach the global optimal solution. The direction of position change is influenced by both particle's own experience and the knowledge the particle acquires from the flock. Each particle is evaluated using a fitness function, which indicates how close the particle is to the optimal solution. It is desired to maximize the fitness as the PSO iterations progress. A particle i has a memory to store the knowledge of position p_best_{id} , which is defined as the position at which the particle had best fitness. Besides, the best of p_best_{id} of all particles, called g_best_d , is stored too. At each iteration k , PSO modifies each dimension of the position x_{id} in a particle by adding a velocity v_{id} and moves the particle towards its p_best_{id} and g_best_d using (1) and (2).

$$v_{id}(k+1) = w \cdot v_{id}(k) + c_1 \text{rand}_1(p_{id} - x_{id}) + c_2 \text{rand}_2(p_{gd} - x_{id}) \quad (1)$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (2)$$

Velocity is never allowed to exceed V_{max} . This is done so to prevent particles from moving with large steps and going

outside the boundaries of the problem space. If V_{max} is very low, the particles move in very small steps and therefore, take long time to converge to the solution. Pseudocode for the PSO algorithm is given in Fig.1.

```

FOR each particle  $i$ 
  FOR each dimension  $d$ 
    Initialize position  $x_{id}$  randomly within permissible range
    Initialize velocity  $v_{id}$  randomly within permissible range
  END FOR
END FOR
Iteration  $k=1$ 
DO
  FOR each particle  $i$ 
    Calculate fitness value
    IF the fitness value is better than  $p\_best_{id}$  in history
      Set current fitness value as the  $p\_best_{id}$ 
    END IF
  END FOR
  Choose the particle having the best fitness value as the  $g\_best_d$ 
  FOR each particle  $i$ 
    FOR each dimension  $d$ 
      Calculate velocity according to the equation
       $v_{id}(k+1) = w \cdot v_{id}(k) + c_1 \text{rand}_1(p_{id} - x_{id}) + c_2 \text{rand}_2(p_{gd} - x_{id})$ 
      Update particle position according to the equation
       $x_{id}(k+1) = x_{id}(k) + v_{id}(k+1)$ 
    END FOR
  END FOR
   $k=k+1$ 
WHILE maximum iterations or minimum error criteria are not attained
  
```

Fig. 1. Pseudocode for PSO

PSO has found extensive applications in many optimization problems. In sensor networks, it has been used for cluster formation [2], optimal multicast routing [3], and distributed sensor placement problems [4]. Maximum likelihood estimation of target position [5] and sink node path optimization [6] are the other problems that have been addressed with PSO. A PSO variant has been applied in wavelength detection in FGB sensor network [7]. PSO has been used for odor source localization in mobile sensor networks [8].

In spite of its advantages like low computational complexity, the PSO suffers from the problem of premature convergence. This is overcome with a mutation operator with adaptive probability, and by replacing particles flying out of the solution space by newly generated random particles during the search process. The variant of PSO that uses adaptive mutation and regeneration is called Improved PSO (IPSO) [9].

In the classical PSO, particles depend on their individual memory and peer influence to explore the search space. However, the swarm as a whole does not use its collective experience (represented by the array of previous best positions) to guide its search. This causes re-exploration of already known bad regions in the search space. This paper proposes an approach in which swarm's collective memory is used to guide the particle's movement towards the estimated good regions in the search space.

This paper presents two hybrid versions of PSO that allow a particle swarm to estimate the distribution of promising solution regions and thus learn through the information assimilated during the process of optimization. This distribution is used to keep the particles within the promising solution regions. This algorithm is fused with two versions of PSO, namely classical PSO and IPSO. The estimation of the distribution is done by means of a mixture of normal distributions of previous best solutions. These hybrids borrow ideas from recent developments in Ant Colony Optimization (ACO) in which an archive of solutions is used to select the next point to explore in the search space.

PSO and the two hybrid versions of PSO proposed here are tested on five benchmark test functions. The rest of the paper is organized as follows: Section 2 describes the background of the estimation of distribution PSO algorithm (EDPSO). Section 3 covers the details of the estimation of distribution improved PSO algorithm (EDIPSO). Numeric simulation and results are presented in section 4 and conclusions are given in section 5.

2. ESTIMATION OF DISTRIBUTION PSO (EDPSO) ALGORITHM

Estimation of distribution algorithms (EDA) use information obtained during optimization to build probabilistic models of distribution of good solution regions and use this information to produce new solutions. EDAs yield fast convergence to global optimal solution because they approximate the joint probability distribution that characterizes the problem. A comprehensive comparison of some best-known EDA algorithms is given in [10]. This paper uses two hybrids, which progress like PSO algorithms but model the joint probability distribution in order to constrain particles in better areas of search space.

Ant Colony Optimization (ACO) is another popular swarm intelligence algorithm. This algorithm is used for combinatorial optimization problems. A recent development of ACO that is aimed at continuous optimization is called ACO_R [11]. This algorithm approximates the joint probability distribution, one dimension at a time, by using a mixture of weighted Gaussian functions. The weights represent quality of different search regions in solution space. Therefore, ACO_R can deal with multimodal functions.

The concept of ACO_R is given in Fig. 2. The algorithm uses an archive of existing solutions of size m (swarm size) as the source of information to parameterize univariate distributions. The i^{th} component of l^{th} solution is represented as s_{il} . For an n -dimensional problem, $1 \leq i \leq n$ and $1 \leq l \leq m$. For each

dimension i , the vector $\mu_i = \langle s_{i1}, s_{i2}, \dots, s_{im} \rangle$ represents vector of means used to model univariate probability distribution for the i^{th} dimension. The vector of weights $w = \langle w_1, w_2, \dots, w_m \rangle$ is the same across all dimensions because it is based on relative quality of complete solutions. In each iteration, solutions are ranked and weights are determined using (3),

$$w_l = \frac{1}{qm\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2(qm)^2}} \quad (3)$$

where q is the parameter that determines the degree of preference of good solutions. With a small value of q , best solutions are strongly preferred over weaker solutions to guide the search [12].

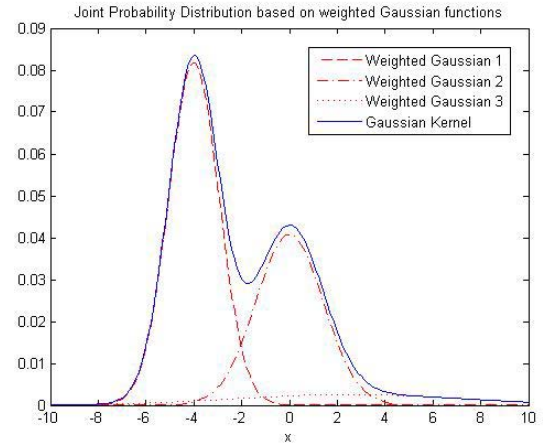


Fig. 2. Joint probability distribution based on weighted Gaussians

Because the algorithm samples a mixture of Gaussians, one will need to select one Gaussian function from the kernel probabilistically. The probability of choosing l^{th} Gaussian function is computed using (4).

$$p_l = \frac{w_l}{\sum_{j=1}^m w_j} \quad (4)$$

The standard deviation of the Gaussian functions is computed using (5).

$$\sigma_{ii} = \zeta \sum_{j=1}^m \frac{|s_{ij} - s_{il}|}{m-1} \quad (5)$$

where ζ is a parameter that allows algorithm to balance its exploration-exploitation behavior. This has the same value for all dimensions. ACO_R samples a Gaussian function and generates a new solution component in every iteration. This paper borrows the idea from ACO_R [11].

In estimation of distribution PSO (EDPSO) hybrid, ACO_R is fused with PSO in order to exploit the useful properties of both the algorithms. The p_best matrix is used here as the archive of solution over which ACO_R builds its probabilistic model. The EDPSO algorithm progresses as the normal PSO does. For

every particle in the swarm, two particles are generated, one using PSO and another using estimation of distribution. In each iteration, the location to which a particle will be moved is determined using PSO position update equation. Such a particle is names as PSO version of the particle. In addition, a Gaussian distribution function is probabilistically chosen from the kernel and a new particle is produced by sampling it in all n dimensions. This gives the EDA version of the particle. The fitness functions are evaluated for both versions of a particle. The particle that exhibits the better objective function is selected to enter the next iteration. The pseudocode for the EDPSO algorithm is presented in Fig. 3. Other variants of EDPSO have been reported. This paper uses central theme of the work reported in [12] but uses the selection criterion to choose either PSO or EDA version of a particle.

```

FOR each particle  $i$ 
  FOR each dimension  $d$ 
    Initialize position  $x_{id}$  randomly within permissible range
    Initialize velocity  $v_{id}$  randomly within permissible range
  End FOR
END FOR
Iteration  $k=1$ 
DO
  FOR each particle  $i$ 
    Calculate fitness value
    IF the fitness value is better than  $p\_best_{id}$  in history
      Set current fitness value as the  $p\_best_{id}$ 
    END IF
  END FOR
  Choose the particle having the best fitness value as the  $g\_best_d$ 
  FOR each particle  $i$ 
    FOR each dimension  $d$ 
      Calculate velocity according to the equation
       $v_{id}(k+1) = w v_{id}(k) + c_1 rand_1(p_{id}-x_{id}) + c_2 rand_2(p_{gd}-x_{id})$ 
      Update particle position according to the equation,
       $x_{id}(k+1) = x_{id}(k) + v_{id}(k+1)$  get  $d^{th}$  dimension of PSO-particle
      Compute weights if Gaussian functions in kernel  $w_i$ 
      Select a Gaussian function  $g_i$  from kernel according to  $p_i$ 
      Sample  $g_i$  to get  $d^{th}$  dimension of EDA-particle
    END FOR
  END FOR
  FOR each particle  $i$ 
    Evaluate fitness of  $i^{th}$  PSO-Particle
    Evaluate fitness of  $i^{th}$  EDA-Particle
    IF fitness(PSO-Particle) < fitness (EDA-Particle)
       $x(k+1) = \text{PSO-Particle}$ 
    ELSE
       $x(k+1) = \text{EDA-Particle}$ 
    ENDIF
  END FOR
   $k=k+1$ 
WHILE maximum iterations or minimum error criteria are not attained

```

Fig. 3. Pseudocode for EDPSO algorithm

3. ESTIMATION OF DISTRIBUTION IMPROVED PSO (EDIPSO) ALGORITHM

As seen in equation (1), the velocity update of the particle consists of three parts: the first term is inertia of particles; the second term is cognitive acceleration which represents the particle's own experiences; and the third term is social acceleration which represents the social interaction between the

particles. From this, it can be reasoned that when a particle's current position coincides with the global best position, the particle will leave this place only if the inertia weight w and its current velocity v_{id} are not equal to 0. If the particles' current velocities are very close to 0, then the particles will not move if they get caught up with the best particle. This means that all the particles will converge to the best position g_best_d . If this position is not the global best, then this phenomenon leads to premature convergence.

Improved PSO (IPSO) uses adaptive mutation to avoid premature convergence [9]. If $x=x_1+x_2...x_n$ is the particle chosen with mutation probability P_m , then the mutation result of this particle is

$$x_d = g_best_d + 0.50 \times randn(g_best_d) \quad (6)$$

$$d = 1, 2, \dots n$$

In the mutation operation, the mutation probability P_m is dynamically adjusted according to the diversity in the swarm. The ratio between mean and the maximum of the fitness function of all particles in an iteration is used to measure the diversity div , such that $0 < div < 1$. If $div \approx 1$, it means that all the particles have gathered at the same position. Under these circumstances, the mutation probability should be increased to allow more particles to search in different unexplored zones. On the contrary, if $div \ll 1$ it indicates that there is a great diversity of particles in the swarm, in which case the P_m must be reduced to avoid a basically random search. The pseudocode for the dynamic adaptation of the mutation is shown in Fig. 4.

```

IF  $div > V_{max}$ 
   $P_m = k_m \times P_m$ 
ELSE IF  $div < V_{min}$ 
   $P_m = P_m / k_m$ 
END IF
IF  $P_m > P_{mmax}$   $P_m = P_{mmax}$  END IF
IF  $P_m < P_{mmin}$   $P_m = P_{mmin}$  END IF

```

Fig. 4. Pseudocode for dynamic adaptive mutation

The standard PSO algorithm generally uses boundary condition to constrain particles in the search region. On the other hand, IPSO algorithm produces the same number of random particles to replace the particles that fly out of the search space [2] [9]. The pseudocode for the regeneration is given in Fig 5. Pseudocode for EDIPSO algorithm is shown in Fig. 6.

```

IF  $x_{id} > x_{max}$  OR  $x_{id} < x_{min}$ 
   $x_{id} = x_{min} + rand \times (x_{max} - x_{min})$ 
END IF

```

Fig. 5. Pseudocode for regeneration

The EDIPSO proposed here is a hybrid of IPSO and the ACO_R . The difference between EDPSO and EDIPSO lies in the fact that the latter uses both dynamic adaptive mutation and particle regeneration.

```

FOR each particle  $i$ 
  FOR each dimension  $d$ 
    Initialize position  $x_{id}$  randomly within permissible range
    Initialize velocity  $v_{id}$  randomly within permissible range
  End FOR
END FOR
Iteration  $k=1$ 
DO
  FOR each particle  $i$ 
    Calculate fitness value
    IF the fitness value is better than  $p\_best_{id}$  in history
      Set current fitness value as the  $p\_best_{id}$ 
    END IF
  END FOR
  Choose the particle having the best fitness value as the  $g\_best_d$ 
  Update  $p_m$  depending upon the diversity of solutions
  FOR each particle  $i$ 
    FOR each dimension  $d$ 
      Calculate velocity according to the equation
       $v_{id}(k+1) = w v_{id}(k) + c_1 rand_1(p_{id} - x_{id}) + c_2 rand_2(p_{gd} - x_{id})$ 
      Update particle position according to
       $x_{id}(k+1) = x_{id}(k) + v_{id}(k+1)$  to get  $d^{th}$  dimension of PSO-particle
      Apply adaptive mutation with probability  $p_m$ 
      Apply regeneration if a particle flies out of search space
      Compute weights if Gaussian functions in kernel  $w_i$ 
      Select a Gaussian function  $g_i$  from kernel according to  $p_i$ 
      Sample  $g_i$  to get  $d^{th}$  dimension of EDA-particle
    END FOR
  END FOR
  FOR each particle  $i$ 
    Evaluate fitness of  $i^{th}$  PSO-Particle
    Evaluate fitness of  $i^{th}$  EDA-Particle
    IF fitness(PSO-Particle) < fitness(EDA-Particle)
       $x(k+1) = \text{PSO-Particle}$ 
    ELSE
       $x(k+1) = \text{EDA-Particle}$ 
    ENDIF
  END FOR
 $k=k+1$ 
WHILE maximum iterations or minimum error criteria are not attained

```

Fig 6. Pseudocode for EDIPSO algorithm

4. NUMERICAL SIMULATION AND RESULTS

All simulations are carried out on the same computer using Matlab. Relative performance of PSO, EDPSO and EDIPSO is tested to minimize five standard benchmark functions given in Table I. All the functions have global minima at zero.

Initial assignment of weights and maximum values of velocities and positions are taken as shown in Table II. In all of the functions, dimensionalities $n=50$ and $n=100$ are tested. The parameters chosen are:

- Number of particles in the swarm: 30
- Inertia weight is computed by (12) where $w_{\max} = 0.9$ and $w_{\min} = 0.4$.

$$w_{\max} = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times iter \quad (12)$$

- Acceleration constants $c_1 = 2.0$ and $c_2 = 2.0$
- Initial mutation probability in EDIPSO, $P_m = 0.08$
- $P_{m \max} = 0.15$, $P_{m \min} = 0.01$
- For $n=50$, Maximum iterations $iter_{\max} = 5000$

TABLE I. LIST OF BENCHMARK FUNCTIONS ON WHICH PSO, EDPSO AND EDIPSO ARE TESTED

Sphere function	$f_0(x) = \sum_{i=1}^n x_i^2 \quad (7)$
Rosenbrock Function	$f_1(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \quad (8)$
Rastrigrin Function	$f_2(x) = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi \cdot x_i) + 10) \quad (9)$
Griewank Function	$f_3(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (10)$
Ackley Function	$f_4(x) = e + 20 - \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi \cdot x_i)\right) \quad (11)$

- For $n=100$, Maximum iterations $iter_{\max} = 10000$.
- In EDPSO and EDIPSO, $q=0.05$ and $\zeta=0.85$
- PSO, EDPSO and EDIPSO algorithms are tested with the same set of random initial particles.

Each algorithm is tested for 20 trials. Average fitness and standard deviation of fitness are computed. The results obtained are presented in Table III. Fig.7 through Fig.11 indicate the improvements in fitness values with respect to iterations in one particular trial run for the five benchmark functions.

TABLE II. PARTICLE INITIALIZATION RANGES IN OPTIMIZATION OF BENCHMARK FUNCTIONS

Function	Initialization range	V_{\max}	X_{\max}
Sphere	$(50,100)^n$	100	100
Rosenbrock	$(15,30)^n$	100	100
Rastrigrin	$(2.56, 5.12)^n$	10	10
Griewank	$(-50,50)^n$	50	50
Ackley	$(-32,32)^n$	10	32

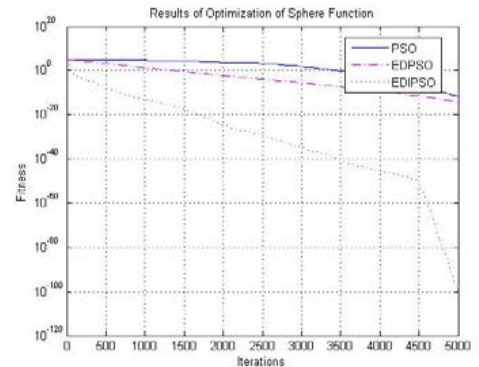


Fig. 7. Results of Optimization of Sphere Function

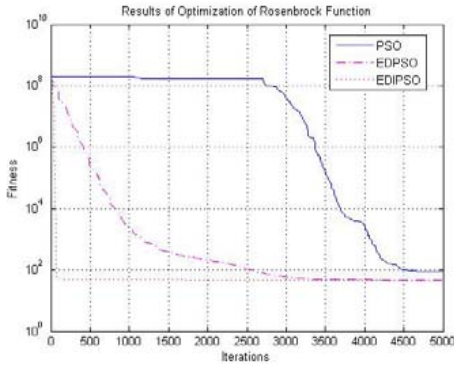


Fig. 8. Results of Optimization of Rosenbrock Function

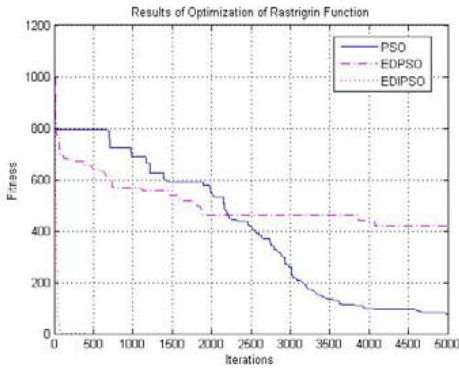


Fig. 9. Results of Optimization of Rastrigrin Function

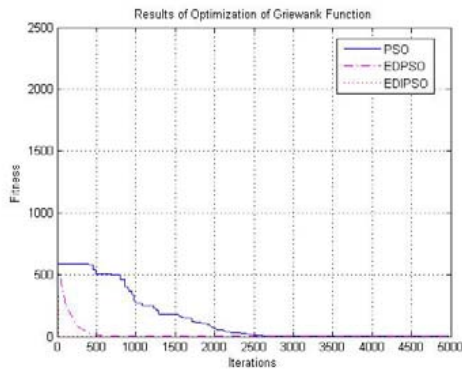


Fig. 10. Results of Optimization of Griewank Function

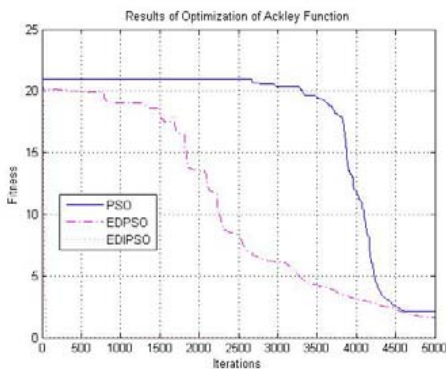


Fig. 11. Results of Optimization of Ackley Function

TABLE III. AVERAGE FITNESS ACHIEVED BY PSO, EDPSO AND EDIPSO IN OPTIMIZATION OF BENCHMARK FUNCTIONS OVER 20 TRIAL RUNS. STANDARD DEVIATION IS SHOWN IN BRACKETS.

Function	PSO	EDPSO	EDIPSO	
Sphere	3.10×10^{-9} (6.56×10^{-9})	4.38×10^{-15} (5.708×10^{-15})	3.86×10^{-96} (1.73×10^{-95})	Dimensions: 50 Iterations: 5000
Rosenbrock	462.7562 (649.0585)	66.488 (42.168)	41.81 (0.27)	
Rastrigrin	112.9276 (24.3408)	320.308 (36.508)	0 (0)	
Griewank	0.0130 (0.0244)	0.003 (0.008)	0 (0)	
Ackley	5.8618 (6.2995)	2.30 (0.55)	4.44×10^{-15} (0)	Dimensions: 100 Iterations: 10000
Sphere	0.01 (0.0243)	1.00×10^{-13} (6.96×10^{-14})	1.33×10^{-169} (0)	
Rosenbrock	3.11×10^{-6} (6.29×10^{-6})	357.30 (90.09)	90.39 (0.32)	
Rastrigrin	781.57 (448.24)	795.19 (153.93)	0 (0)	
Griewank	11.50 (8.68)	0.0012 (0.003)	0 (0)	Dimensions: 100 Iterations: 10000
Ackley	19.62 (0.89)	1.76 (0.75)	4.44×10^{-15} (0)	

Results of optimization of benchmark functions reveal that in all functions except Rastrigrin, EDPSO algorithm produces better quality of solutions than classical PSO, both in terms of fitness and standard deviation. This is because particles are guided towards the better solution zones due to probabilistic modeling. Further, it is observed that the concepts of adaptive mutation and regeneration yield more efficient search for global optimal solution in EDIPSO algorithm. Best solutions determined by EDIPSO are several orders lesser than the best solutions determined by EDPSO algorithm.

EDIPSO algorithm exhibits remarkable speed in optimizing all the functions tested. Besides, it produces consistent performance over trial runs as indicated by the small standard deviation. This makes EDIPSO algorithm one of the very promising algorithms available for optimization of multimodal functions.

5. CONCLUSIONS AND FUTURE WORK

In this paper, two versions of EDA-PSO hybrid are introduced. Results of optimization of benchmark functions indicate that EDPSO and EDIPSO have abilities to find better quality of solutions than that of PSO. This renders these algorithms attractive for optimization problems in sensor networks like cluster formation, multicast routing, distributed sensor placement and sink node path optimization.

Different EDIPSO optimization parameters are required for solving different problems in practical application, such as the number of agents (individuals), weight factors and, acceleration factors and the limits for change in velocity. Sensitivity analysis of optimization parameters for finding the best solutions is one of the future works. Further scope for research lies in hybrids of other forms of EDA and PSO and their applications to specific optimization problems in sensor networks.

REFERENCES

- [1] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. IEEE Int. Conf. Neural Networks*, Vol. IV, Perth, Australia, 1995, pp. 1942–1948.
- [2] S. M. Guru, S.K. Halgamuge, S.Fernando, "Particle Swarm Optimizers for Cluster formation in Wireless Sensor Networks," *Proc. Int. Conf. Intelligent Sensors, Sensor Networks and Information Processing*, 8 Dec. 2005, pp.19 – 324.
- [3] P. Yuan; C. Ji, Y. Zhang, Y. Wang, "Optimal multicast routing in wireless ad hoc sensor networks," *Proc. IEEE Int. Conf. Networking, Sensing and Control*, Vol. 1, 21-23 March 2004, pp. 367–371.
- [4] P.N Ngatchou, W.L.J. Fox, M.A. El-Sharkawi, "Distributed sensor placement with sequential particle swarm optimization," *Proc. IEEE Swarm Intelligence Symposium 2005*, 8-10 June 2005, pp. 385–388.
- [5] M.M Noel, P.P Joshi, T.C. Jannett, "Improved Maximum Likelihood Estimation of Target Position in Wireless Sensor Networks using Particle Swarm Optimization," *Proc. Third Int. Conf. Information Technology: New Generations, 2006. ITNG 2006*, 10-12 April 2006, pp. 274 – 279
- [6] C. Mendis, S.M Guru, S. Halgamuge, S. Fernando, "Optimized sink node path using particle swarm optimization," *Proc. Int. Conf. Advanced Information Networking and Applications, 2006. AINA 2006*, Vol. 2, 18-20 April 2006.
- [7] J. J. Liang, P.N. Suganthan, C.C. Chan, V.L. Huang, "Wavelength detection in FBG sensor network using tree search DMS-PSO," *IEEE Photonics Technology Letters*, Vol. 18, Issue 12, June 2006, pp. 1305–1307.
- [8] W. Jatmiko, K. Sekiyama, T. Fukuda, "A Mobile Robots PSO-Based for Odor Source Localization in Extreme Dynamic Advection-Diffusion Environment with Obstacle," *Proc. 5th IEEE Conf. Sensors*, Oct. 2006, pp. 526–529.
- [9] J. Chen, Z. Ren, and X. Fan, "Particle Swarm Optimization with Adaptive Mutation and Its Application Research in Tuning of PID Parameters", *Proc. 1st International Symposium on Systems and Control in Aerospace and Astronautics, 2006. ISSCAA 2006*, 19-21 Jan. 2006, pp. 990-994.
- [10] Martin Pelican, David E. Goldberg, Francis Lobo, "A survey of optimization by building and using Probabilistic Models," IlliGAL Report No. 99018, ACC 2000.
- [11] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," Universite Libre de Bruxelles, IRIDI Technical Report Series, Technical Report No. TR/IRIDIA/2006-037, Dec 2005.
- [12] M. Iqbal, A. Marco, M. de Oca, "An Estimation of Distribution Algorithm Particle Swarm Optimization Algorithm". *Proc. Fifth International Workshop on Ant Colony Optimization and Swarm Intelligence, ANTS 2006*, Brussels, Be