A New Diversity Guided Particle Swarm Optimization with Mutation

Radha Thangaraj[1], Millie Pant[1] and Ajith Abraham[2]
[1]Indian Institute of Technology Roorkee, India
[2]Norwegian University of Science and Technology, Norway
t.radha@ieee.org, millifpt@iitr.ernet.in, ajith.abraham@ieee.org

Abstract— **This paper presents a new diversity guided Particle Swarm Optimization algorithm (PSO) named Beta Mutation PSO or BMPSO for solving global optimization problems. The BMPSO algorithm makes use of an evolutionary programming based mutation operator to maintain the level of diversity in the swarm population, thereby maintaining a good balance between the exploration and exploitation phenomena and preventing premature convergence. Beta distribution is used to perform the mutation in the proposed BMPSO algorithm. The performance of the BMPSO algorithm is investigated on a set of ten standard benchmark problems and the results are compared with the original PSO algorithm. The numerical results show that the proposed algorithm outperforms the basic PSO algorithm in all the test cases taken in this study.**

Keywords-Particle Swarm Optimization; Diversity; Mutation; global optimization.

I. Introduction

The concept of Particle Swarm Optimization (PSO) was first suggested by Kennedy and Eberhart [1]. Since its development is 1995, PSO has become one of the most promising optimizing techniques for solving global optimization problems. It has been shown empirically in many studies to work well, outperforming other optimization techniques such as evolutionary algorithms. Although the rate of convergence of PSO is good due to fast information flow among the solution vectors, its diversity decreases very quickly in the successive iterations resulting in a suboptimal solution. One of the simplest methods to overcome the problem of diversity loss is to capitalize the strengths of EA and PSO together in an algorithm. A variety of methods combining the aspects of EA and PSO are available in literature [2] – [7] etc. Out of the EA operators, mutation is the most widely used EA tool applied in PSO [8], [9]. The concept of mutation is quite common to Evolutionary Programming (EP), Genetic Algorithms and Differential Evolution. Mutation has been introduced into the PSO as a mechanism to increase the diversity of PSO, and by doing so improving the exploration abilities of the algorithm. Mutation can be applied to different elements of a particle swarm. The effect of mutation depends on which elements of the swarm are mutated. If only the neighbourhood best position vectors are mutated, then effect is minimal, compared to mutation of particle position vectors. Velocity vector mutation is equivalent to particle's position vector mutation, under the condition that the same mutation operator is considered [10].

There are several instances in PSO where mutation is introduced in the swarm. Some mutation operators that have been applied to mutate the position vector in PSO include Gaussian Mutation [11] - [15], Cauchy [15], [16], Chaos mutation [17] – [19] etc. However, in our knowledge none of the above mentioned techniques uses diversity as a measure to guide the swarm. This paper presents a new PSO algorithm based on EP Mutation with the help of Beta distribution; also it uses a diversity enhancing mechanism to improve the performance of the swarm. The proposed Beta Mutation PSO algorithm is named as BMPSO. The BMPSO algorithm uses diversity threshold values d_{low} and d_{high} to guide the movement of the swarm. The threshold values are predefined by the user.

The rest of the paper is organized as follows: Section II describes the original Particle Swarm Optimization algorithm. In section III, the proposed BMPSO algorithm is discussed; section IV deals with the results and discussion. Finally, this paper concludes with section V.

II. Particle Swarm Optimization

PSO is a multi-agent parallel search technique developed by Eberhart and Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. The particles or members of the swarm fly through a multidimensional search space looking for a potential solution. Each particle adjusts its position in the search space from time to time according to the flying experience of its own and of its neighbors (or colleagues).

For a D-dimensional search space, the position of the i^th particle is represented as $X_i = (x_{i1}, x_{i2}, ..., x_{id}, ..., x_{iD})$. Each particle maintains a memory of its previous best position $P_i = (p_{i1}, p_{i2}, ..., p_{id}, ..., p_{iD})$. The best one among all the particles in the population is represented as $P_g = (p_{g1}, p_{g2}, ..., p_{gd}, ..., p_{gD})$. The velocity of each particle is represented as $V_i = (v_{i1}, v_{i2}, ..., v_{id}, ..., v_{iD})$.

During each generation each particle is accelerated toward the particles previous best position and the global best position. At each iteration a new velocity value for each particle is calculated based on its current velocity, the distance from the global best position. The new velocity value is then used to calculate the next position of the particle in the search space. This process is then iterated a number of times or until a minimum error is achieved. The two basic equations which govern the working of PSO are that of velocity vector and position vector given by:

$$v_{id} = \omega * v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad \textbf{(1)}$$

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

Here ω is inertia weight, predefined by the user. c_1 and c_2 are acceleration constants. They represent the weighting of the stochastic acceleration terms that pull each particle toward personal best and global best positions. Therefore, adjustment of these constants changes the amount of tension in the system. Low value of these constants allow particles to roam far from the target regions before tugged back, while high values result in abrupt movement toward, or past, target regions [20]. The constants r_1, r_2 are the uniformly generated random numbers in the range of [0, 1]. The first part of equation (1) represents the inertia of the previous velocity, the second part tells us about the personal thinking of the particle and the third part represents the cooperation among particles and is therefore named as the social component [21].

III. Proposed BMPSO Algorithm

The Beta Mutation Particle Swarm Optimization is simple and modified version of Particle Swarm Optimization algorithm; it uses Beta distribution to mutate the particle. The BMPSO algorithm has two phases namely attraction phase and mutation phase. The attraction phase is same as that of the classical PSO, while in the mutation phase the swarm particles position vectors are mutated using Beta Distributed Mutation (BDM) operator. The BDM operator is EP based mutation operator and is defined as:

$$x_i^{t+1} = x_i^{t} + \sigma_i' * Betarand_j() \quad (3)$$

where, $\sigma_i' = \sigma_i * \exp(\tau\ N(0,1) + \tau'\ N_j(0,1))$

$N(0,1)$ denotes a normally distributed random number with mean zero and standard deviation one. $N_j(0,1)$ indicates that a different random number is generated for each value of j. τ and τ′ are set as $1/\sqrt{2n}$ and $1/\sqrt{2\sqrt{n}}$ respectively. *Betarand*$_j$ () is a random number generated by beta distribution with parameters less than 1.

The BMPSO starts like the classical PSO i.e. it uses attraction phase (Eqn. (1)) for updating velocity vector and uses (2) for updating position vector. In this phase, the swarm contracts rapidly due to the fast information flow among the particles, as a result, the diversity also decreases consequently the chances of the swarm particles to get trapped in some local region or some suboptimal solution also increases. In BMPSO algorithm, we keep a check on the decreasing value of diversity with the help of a user-defined parameter called d_{low}. When the diversity of population drops below, d_{low}, it switches over to the mutation phase, with the hope of increasing the diversity of the swarm population and thereby helping the swarm to escape the possible local regions. This process is repeated until a maximum number iteration is reached or the stopping criterion is reached.

The Beta distribution used in this study has the probability density function,

$$f(x) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \alpha > 0, \beta > 0 \quad (4)$$

with parameters less than 1.

The computational steps of BMPSO are given below:

```
Step 1   Initialize PSO parameters
Step 2   Initialize the positions and
         velocities of all particles
         using uniformly distributed
         random numbers
Step 3   Evaluate the objective function
         values of all particles in the
         swarm
Step 4   Update particles personal best
         position and global best
         position (i.e. P_i and P_g)
Step 5   Calculate the diversity of
         swarm
Step 6   If (diversity < d_low)
            Update particles position
            using Eqn. (3)
         Else
            Update particles velocity
            and position vectors
            according to equations (1)
            and (2) respectively
         End if
Step 7   Evaluate particle’s fitness
         values
Step 8   Update P_i and P_g
Step 9   If (Stopping criteria is
         reached) then go to step 10
         Else go to step 5
Step 10  Print the global best particle
         and the corresponding fitness
         function value
```

IV. Exoerimental Settings and Numerical Results

For comparison of PSO and BMPSO algorithms, a collection of 10 standard benchmark problems with box constraints is considered. Mathematical models of the problems along with the true optimum value are given in Table I. The entire set of test problems taken for the present study is scalable i.e. the problems can be tested for any number of variables. However, for the present study we have tested the problems for dimensions 10, 30 and 50.

In order to make a fair comparison of basic PSO and the proposed BMPSO algorithm, we fixed the same seed for random number generation so that the initial population is same for both the algorithms. The population size is taken as 20 for all the test problems. A linearly decreasing inertia weight is used which starts at 0.9 and ends at 0.4, with the user defined parameters c_1=2.0 and c_2=2.0. For each algorithm, the maximum number of generations is set as 1000, 2000 and 3000 generations for dimensions 10, 30 and 50 respectively. . A total of 30 runs for each experimental setting were conducted.

The diversity measure of the swarm can be calculated as [10]:

$$Diversity(S(t)) = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\sum_{j=1}^{n_x} (x_{ij}(t) - \overline{x_j(t)})^2} \qquad (5)$$

where S is the swarm, $n_s = |S|$ is the swarm size, n_x is the dimensionality of the problem, x_{ij} is the j'th value of the i'th particle and $\overline{x_j(t)}$ is the average of the j-th dimension over all particles, i.e. $\overline{x_j(t)} = \frac{\sum_{i=1}^{n_s} x_{ij}(t)}{n_s}$

The results of the given benchmark problems for dimension 30 are shown in Table II in terms of mean best fitness, standard deviation, diversity, the improvement (%), and t-values of proposed BMPSO algorithm in comparison with original PSO. In addition, we have tested the benchmark problems f_1 – f_6 with the two more different dimensions 10 and 50; the corresponding results are given in Table III. Figures 1 - 6 shows the performance curves of PSO and BMPSO algorithms. From the numerical results of Table II, we can see that the proposed algorithm perform better than the original PSO algorithm by a significant difference. The first test function f_1 is Rastringin function, which is a multimodal function. For this function, BMPSO gave a remarkable percentage of improvement of approximately 72% in comparison with PSO. For the function f_2, which is also a multimodal function, the proposed BMPSO algorithm performs much better than the original PSO algorithm. For the functions f_4 and f_{10}, the proposed BMPSO algorithm performs little better than PSO, in these test cases BMPSO algorithm gave approximately 4% improvement in comparison to original PSO. Likewise all the other test cases also we can see that there is a noticeable percentage of improvement in average mean value by using the proposed diversity based mutation algorithm. From the numerical results of Table III also, we can see that the proposed algorithm gave better results than PSO.

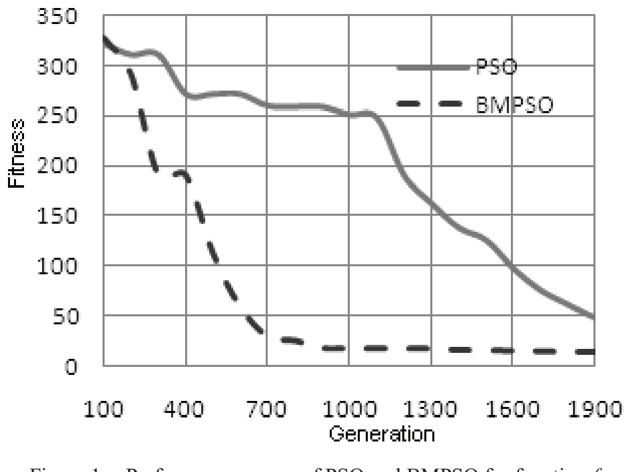

Figure 1. Performance curves of PSO and BMPSO for function f_1

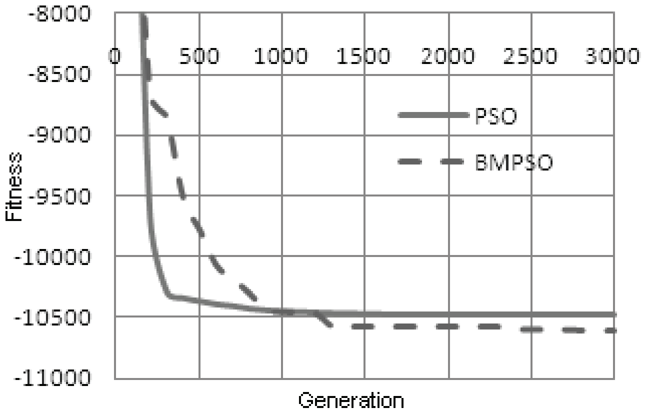

Figure 2. Performance curves of PSO and BMPSO for function f_4

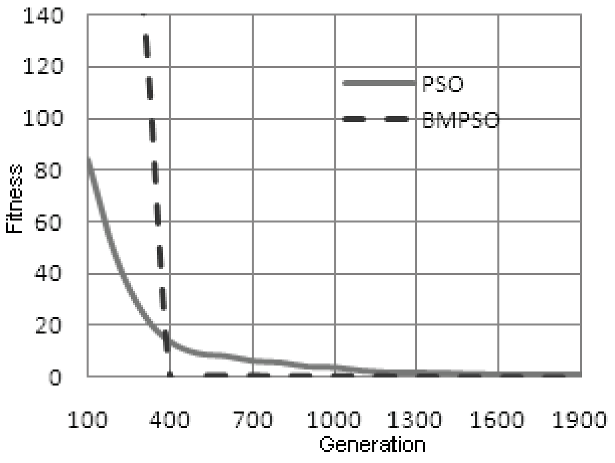

Figure 3. Performance curves of PSO and BMPSO for function f_5

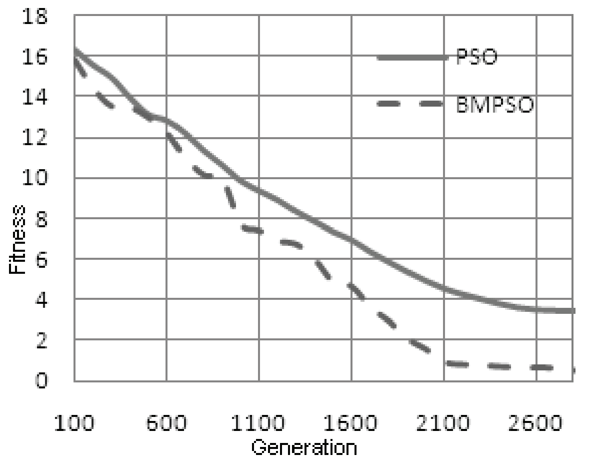

Figure 4. Performance curves of PSO and BMPSO for function f_6

TABLE I. NUMERICAL BENCHMARK PROBLEMS

Function	Function Definition	Range	Optimum
Rastringin Function	$f_1(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	[-5.12,5.12]	0
Griewank Function	$f_2(x)=\frac{1}{4000}\sum_{i=0}^{n-1}x_i^2+\sum_{i=0}^{n-1}\cos(\frac{x_i}{\sqrt{i+1}})+1$	[-600,600]	0
Rosenbrock Function	$f_3(x)=\sum_{i=0}^{n-1}100(x_{i+1}-x_i^2)^2+(x_i-1)^2$	[-30,30]	0
Schwefel Function	$f_4(x)=-\sum_{i=1}^{n}x_i\sin(\sqrt{\lvert x_i\rvert})$	[-500,500]	-12569.5
Noisy Function	$f_5(x)=(\sum_{i=0}^{n-1}(i+1)x_i^4)+rand[0,1]$	[-1.28,1.28]	0
Ackley Function	$f_6(x)=20+e-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})\ -\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))$	[-32,32]	0
Schwefel function 1.2	$f_7(x)=\sum_{i=0}^{n-1}(\sum_{j=0}^{i}x_i)^2$	[-100, 100]	0
Schwefel function 2.21	$f_8(x)=\max\lvert x_i\rvert\ ,\ 0\le i<n$	[-100, 100]	0
Schwefel function 2.22	$f_9(x)=\sum_{i=0}^{n-1}\lvert x_i\rvert+\prod_{i=0}^{n-1}\lvert x_i\rvert$	[-10, 10]	0
Shaffer's function 7	$f_{10}(x)=(\sum_{i=1}^{n}x_i^2)^{1/4}[\sin^2(50(\sum_{i=1}^{n}x_i^2)^{1/10})+1.0]$	[-32.767, 32.767]	0

TABLE II. COMPARISON RESULTS OF PSO AND BMPSO IN TERMS OF AVERAGE FITNESS FUNCTION VALUE, STANDARD DEVIATION AND DIVERSITY

Function	Fitness Value		Standard Deviation		Diversity		Improvement (%) and t-value of BMPSO with PSO	
	PSO	BMPSO	PSO	BMPSO	PSO	BMPSO	Improvement	t-value
f_1	47.29223	**13.34445**	11.06489	**4.690**	3.70371	3.01e-05	71.78	15.47
f_2	0.0182	**0.002525**	0.244025	**0.001589**	33.1326	0.009598	86.12	0.35
f_3	316.4468	**74.76106**	80.001	**24.37858**	4.15842	3.14e-05	76.37	15.82
f_4	-6466.19	**-6718.49**	**643.4821**	666.7723	0.42845	0.01666	3.9	1.49
f_5	0.617222	**0.072433**	0.492993	**0.21498**	0.547031	0.071049	88.26	5.54
f_6	1.70	**0.077461**	0.4530	**0.06742**	1.56783	0.60276	95.43	19.36
f_7	271.793	**28.938**	208.325	**98.7083**	4.7239	0.10343	89.35	5.77
f_8	15.2228	**9.96414**	3.652739	**2.00684**	63.3639	6.60096	34.54	6.91
f_9	0.209776	**0.169551**	**0.072407**	0.625909	0.239009	0.02924	19.17	0.34
f_{10}	4.22028	**4.10447**	**0.416**	0.460416	14.4341	0.806342	2.74	1.02

TABLE III. COMPARISON RESULTS OF PSO AND BMPSO FOR FUNCTIONS $F_1 - F_6$ WITH DIMENSION 10 AND 50

Function	Dimension	PSO	BMPSO	Improvement (%) of BMPSO: comparison with PSO
f_1	10	5.5572	0.012055	99.78307
	50	104.03725	27.889415	73.19286
f_2	10	0.0919	3.726e-06	99.99595
	50	0.014701	0.00415	71.77063
f_3	10	96.1715	7.864022	91.82292
	50	533.64808	178.32484	66.58381
f_4	10	-2389.365	-2764.422	15.6969
	50	-10473.09	-10605.52	1.26448
f_5	10	0.502671	0.405912	19.24897
	50	0.788322	0.409282	48.08188
f_6	10	6.965e-12	0.004697	-
	50	3.43866	0.313801	90.87432

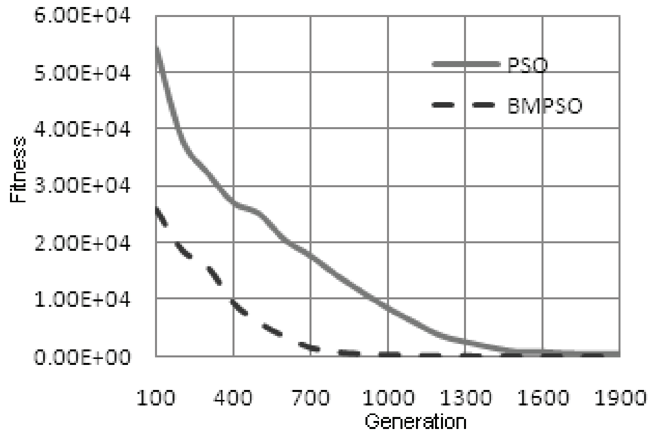

Figure 5. Performance curves of PSO and BMPSO for function f_7

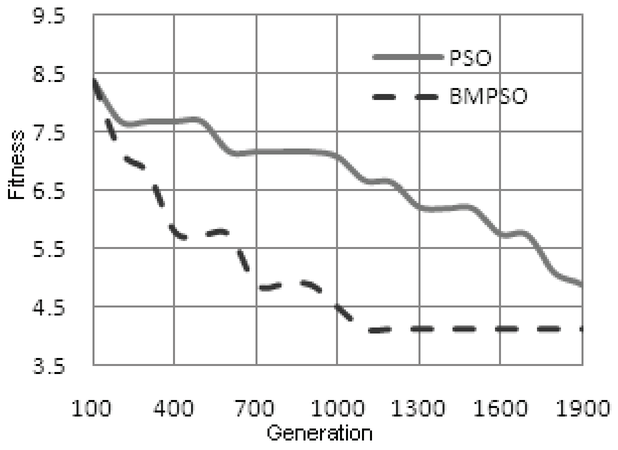

Figure 6. Performance curves of PSO and BMPSO for function f_{10}

V. CONCLUSION

In this paper, a simple and modified version of Particle Swarm Optimization named BMPSO has been proposed. The novelty of the present work is the use of diversity for activating the mutation. Another new feature is the use of a beta distributed mutation operator. The performance of BMPSO algorithm is tested with ten standard benchmark functions with various dimensions 10, 30 and 50. The empirical results show that the proposed BMPSO algorithm is efficient than the original PSO algorithm and is quite competent for solving problems of dimensions up to 50. Thus, from the numerical results it is concluded that the BMPSO algorithm is simple and yet effective algorithm for solving different kind of optimization problems.

REFERENCES

[1] Kennedy, J. and Eberhart, R. C., "Particle Swarm Optimization", IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, Vol. IV, 1995, pp. 1942-1948.

[2] Robinson, J., Sinton, S. and Rahmat-Samii, Y., "Particle Swarm, Genetic Algorithm, and Their Hybrids: Optimization of a Profiled Corrugated Horn Antenna", In Proc. of the IEEE Antennas and Propagation Society International Symposium and URSI national radio Science meeting, Vol. 1, 2002, pp. 314 – 317.

[3] Shi, Y. and Krohling, R. A., "Co-Evolutionary Particle Swarm Optimization to Solve Min-Max Problems", In Proc. of the IEEE Congress on Evolutionary Computation, Vol.2, 2002, pp. 1682 – 1687.

[4] Shi, X., Hao, J., Zhou, J. and Liang, Y., "Hybrid Evolutionary Algorithms Based on PSO and GA", In Proc. of the IEEE Congrss on Evolutionary Computation, Vol. 4, 2003, pp. 2393 – 2399.

[5] Zhang, W - J. and Xie, X - F., "DEPSO: Hybrid Particle Swarm with Differential Evolution Operator", In Proc. of IEEE Int. Conf. on Systems, Man & Cybernetics, 2003, pp. 3816 – 3821.

[6] Hao, Z-F., Guo, G-H. and Huang, H., "A Particle Swarm Optimization Algorithm with Differential Evolution", In Proc. of the 6th Int. Conf. on Machine Learning and Cybernetics, 2007, pp. 1031 – 1035.

[7] Yang, B., Chen, Y. and Zhao, Z., "A Hybrid Evolutionary Algorithm by Combination of PSO and GA for Unconstrained and Constrained optimization Problems", In Proc. of the IEEE Int. Conf. on Control and Automation, 2007, pp. 166 – 170.

[8] Hu, X., Eberhart, R. C. and Shi, Y., "Swarm Intelligence for Permutation Optimization: A Case Study on n-Queens problem", In Proc. of IEEE Swarm Intelligence Symposium, 2003, pp. 243 – 246.

[9] Juang, C - F., "A hybrid of genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design", IEEE Trans. On Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 34(2), 2003, pp. 997 – 1006.

[10] Engelbrecht, A. P., "Fundamentals of Computational Swarm Intelligence", England: John Wiley & Sons Ltd, 2005.

[11] Wei, C., He, Z., Zheng, Y. and Pi, W., "Swarm Directions Embedded in Past Evolutionary Programming", In Proc. of IEEE Congress on Evolutionary Computation, Vol. 2, 2002, pp. 1278 – 1283.

[12] Higashi, H. and Iba, H., "Particle Swarm Optimization with Gaussian Mutation", In Proc. of the IEEE Swarm Intelligence Symposium, 2003, pp. 72 – 79.

[13] Secrest, B. R. and Lamont, G. B., "Visualizing Particle Swarm Optimization – Gaussian Particle Swarm Optimization", In Proceedings of the IEEE Swarm Intelligence Symposium, 2003, pp. 198 – 204.

[14] Sriyanyong, P., "Solving Economic Dispatch Using Particle Swarm Optimization Combined with Gaussian Mutation", In Proc. of ECTI-CON, 2008, pp. 885 – 888.

[15] Krohling, R. A., "Gaussian Particle Swarm with Jumps", In. Proc. of the IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2005, pp. 1226-1231.

[16] Stacey, A., Jancic, M. and Grundy, I., "Particle Swarm Optimization with Mutation", In Proc. of the IEEE Congress on Evolutionary Computation, 2003, pp. 1425 – 1430.

[17] Dong, D., Jie, J., Zeng, J. and Wang, M., "Chaos-Mutation-Based Particle Swarm Optimizer for Dynamic Environment", In Proc. of the 3rd Int. Conf. on Intelligent System and Knowledge Engineering, 2008, pp. 1032 – 1037.

[18] Yang, M., Huang, H. and Xiao, G., "A Novel Dynamic Particle Swarm Optimization Algorithm Based on Chaotic Mutation", Int. Workshop on Knowledge Discovery and Data Mining, 2009, pp. 656 – 659.

[19] Yue-lin, G., Xiao-hui, A. and Jun-min, L., "A Particle Swarm optimization Algorithm with Logarithm Decreasing Inertia Weight and Chaos Mutation, In Proc. of Int. Conference on Computational Intelligence and Security, 2008, pp. 61 – 65.

[20] Eberhart, R. C. and Shi, Y., "Particle Swarm Optimization: Developments, Applications and Resources", In Proc. of IEEE Congress of Evolutionary Computation, Vol. 1, 2001, pp. 27 - 30.

[21] Kennedy, J., "The Particle Swarm: Social Adaptation of Knowledge", In Proc. of the IEEE Int. Conf. on Evolutionary Computation, 1997, pp. 303 – 308.