2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Particle Swarm Optimization with Population Adaptation

Nanda Dulal Jana[1], Jaya Sil[2], and Swagatam Das[3]

[1]Department of IT, National Institute of Technology, Durgapur-713209, India, nanda.jana@gmail.com
[2]Department of CST, Indian Institute of Engineering Science & Technology, Shibpur-711103, India, js@cs.becs.ac.in
[3]ECS Unit, Indian Statistical Institute, Kolkata- 700108, India, swagatam.das@isical.ac.in

***Abstract*—The Particle Swarm Optimization (PSO) algorithm is a novel population based swarm algorithm has shown good performance on well-known numerical test problems. However, PSO tends to suffer from premature convergence on multimodal test problems. This is due to lack of diversity of population in search space and leads to stuck at local optima and ultimately fitness stagnation of the population. To enhance the performance of PSO algorithms, in this paper, we propose a method of population adaptation (PA). The proposed method can identify the moment when the population diversity is poor or the population stagnates by measuring the Euclidean distance between particle position and particles average position of a population. When stagnation in the population is identified, the population will be regenerated by normal distribution to increase diversity in the population. The population adaptation is incorporated into the PSO algorithm and is tested on a set of 13 scalable CEC05 benchmark functions. The results show that the proposed population adaptation algorithm can significantly improve the performance of the PSO algorithm with standard PSO, ATREPSO and ARPSO.**

I. Introduction

Particle Swarm Optimization (PSO), introduced by Kennedy and Eberhart in 1995 [1], [2], is an efficient and effective evolutionary computation technique for global optimization problem. It is a population and iterative based optimization techniques that shares common properties with other evolutionary computing algorithms. Its model is based on social behaviour of birds flocking and fish schooling. In PSO, each particle represents a potential solution that flies through the search space with a velocity by adjusting flying trajectory according to its personal experience and its social experience.

PSO has become one of the most frequently used optimization techniques and has been successfully applied into many real-world applications [3] due to the simple concept and fast converging speed. Although PSO suffers from the premature convergence due to lack of diversity in the population i.e. the particles are easily trapped into local optima. Therefore, the particles are gathering into a small region of the search space. An algorithms search ability of exploration is decreased when premature convergence occurs and particles have a low possibility to explore new search areas.

Many approaches have been introduced to avoid premature convergence in PSO through population diversity metrics in [4], [5], [6], [7], [8], [9], [10], [11]. Shi and Eberhart proposed population diversity of particle swarms [4], using various measurement of population diversity - position based and velocity based population diversity. In [5], exploration and exploitation in PSO was measured by swarm diversity. Swarm diameter and swarm radius, average distance around the swarm center and swarm coherence was proposed for diversity measurement. In [6], population diversity of PSO was guided by predefined threshold values - d_{low} and d_{high}. During evolution, particle attraction and repulsion are associated to update particles velocity according to d_{high} and d_{low}. The modified version of [6] is presented in [7]. When diversity in between phase of the two threshold values, a new velocity update equation was considered. Shi et. al. [8], proposed diversity control in PSO, where position, velocity and cognition diversity are used for diversity measurement. Particles position are updated by adding random noise to the previous particles position. Zhi-hui Zhan et. al. [9], discussed experimental study on PSO diversity. Population diversity was further extend by diversity promotion by employing random and elitist re-initialization of population and the new algorithm was called promoting diversity in PSO [10]. Another new population diversity in PSO was proposed by Shi Cheng et. al. [11], where the inertia weight was adapted by evolving sigmoid function based on population diversity.

In a few words, there are several ways to measure the swarm diversity and even more variation of each of these methods are exits. But, in which moment of the evolution process, the diversity control can be considered and the way of maintaining the diversity in population are less studied. Our study focus on this issues.

This paper presents a simple and effective PSO algorithm with population adaptation, called PAPSO. The proposed method can identify the moment when the population diversity is unchanged or the population stagnates by measuring the Euclidean distance between particle position and particles average position of a population. When moment is identified, the population will be regenerated by normal distribution. The proposed algorithm is able to increase the population diversity automatically and enhance the performance of the PSO algorithms.

The rest of the paper is organized as follows. In section II, the frame work of PSO is presented. The proposed methodology for population adaptation in PSO are utilized and described in section III. In section IV, experimental results are described, including the benchmark functions and experimental settings of the algorithms. At last, conclusions are summarized and future work is highlighted in section V.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

II. PARTICLE SWARM OPTIMIZATION

The Particle Swarm Optimization (PSO) algorithm is a swarm (population) based optimization techniques introduced by Kennedy and Eberhart [1]. In PSO algorithm, each member of the population is called a 'particle' and each particle flies around in the D-dimension search space with a velocity, which is updated by the particles own experience and the experience of the whole swarm. The experience of a particle is recoded from the previous iterations and presented as personal best (*pbest*); the experience among all the particles is represented as global best (*gbest*). Each particle in the PSO has a position and a velocity, its evaluation is achieved using the objective function or fitness function (f) of a optimization problem, whose variables are the particle position dimensions. The particle updating method tries to move particles to better position by accelerating them towards *pbest* and *gbest*.

While optimizing a problem in a D-dimensional search space, each i^{th} particle in the swarm has a position vector $X_i = [x_{i1}, x_{i2}, ..., x_{iD}]$ and velocity vector $V_i = [v_{i1}, v_{i2}, ..., v_{iD}]$ to indicates its current status. Moreover, the i^{th} particle will keep its personal best position, named *pbest* and denoted as a vector $P_i = [p_{i1}, p_{i2}, ..., p_{iD}]$. All the current positions can be treated as the position population and all the *pbest* can be treated as the personal best population. In PSO, X_i and V_i are initialize randomly and are updated by learning from the personal best population to approach the global optimum.

In general, each i^{th} particle has a neighbourhoods and its search behaviour is influenced by its own personal best information P_i and the best personal best information P_n in the neighbourhoods. The velocity and position are updated as follows

$$V_i(t+1) = \omega.V_i(t) + c_1 r_1(X_{pbesti}(t) - X_i(t)) + c_2 r_2(X_{gbesti}(t) - X_i(t)) \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2)$$

Where ω is the inertia weight that used to control the global and local search abilities. The c_1 and c_2 are positive constants, called the acceleration coefficients, r_1 and r_2 are two uniformly distributed random number in the interval [0, 1]. In Eq. (1), ω is the inertia weight which provides the necessary diversity to the swarm by changing the momentum of particles. Generally, a maximum velocity (V_{max}) for each modulus of the velocity vector of the particles is defined in order to control excessive roaming of particles outside the user defined search space.

III. METHODOLOGY

In this section, we will introduce the PSO algorithm with population adaptation, called PAPSO. The PAPSO algorithm is able to enhance the population diversity when the population diversity is poor or it has been stagnates.

A. The Population Adaptation

Suppose that $X_{i,G} = [x_{i,1,G}, x_{i,2,G}, ..., x_{i,D,G}]$, a D-dimensional position vector of i^{th} particle at the G^{th} generation, $i = 1, 2, ..., N$, where N is the swarm size or population size. The diversity (D_G) of the swarm is calculated according to the diversity measure [12] as follows

$$D_G = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\sum_{j=1}^{D}(x_{ij} - \bar{x_j})^2} \quad (3)$$

where N is the swarm size, D is the dimensionality of the problem, is the j^{th} value of the i^{th} particle and $\bar{x_j}$ is the average of the j^{th} dimension overall particles, i.e.

$$\bar{x_j} = \frac{\sum_{i=1}^{N} x_{ij}}{N} \quad (4)$$

The diversity measure in Eq.(3) is independent of swarm size and the dimensionality of the problem as well as the search range in each dimension. When the population diversity (D_G) is poor or unchanged in two consecutive iterations, the population has converged at an optimum and in that case D_G will not change any more. In the case of local optima or stagnates situation, the algorithm may occasionally stop proceeding toward the global optimum. When population stuck at local optima, D_G is unchanged or small change will be occur in the consecutive generations. Z_G is a flag to denote whether the population diversity is poor at the G^{th} generation.

$$Z_j = \begin{cases} 1 & \text{if } abs(D_G - D_{G-1}) \leq (1E-3) \\ 0 & \text{Otherwise} \end{cases} \quad (5)$$

If D_G maintains unchanged or absolute value of the difference of the diversity in two consecutive generations is less than equal to a specific error $(1E-3)$, it indicates that the algorithm can't generate better population. In this case, $Z_G = 1$, otherwise $Z_G = 0$ (in Eq.(5)). If $Z_G = 1$, the algorithm needs to regenerate the population of the particles position. When $Z_G = 1$, the new population of particles position $X_{i,G+1}$, $i = 1, 2, ..., N$, is generated as follows

$$x_{i,j,G+1} = a_{j,G} + (b_{j,G} - a_{j,G}).randN_{j,G}, j = 1, 2, ..., D. \quad (6)$$

where

$$a_{j,G} = min(m_{j,G}, x_{min,j}) \quad (7)$$

$$b_{j,G} = max(m_{j,G}, x_{max,j}) \quad (8)$$

$x_{min,j}$ and $x_{max,j}$ are the predefined lower and upper bounds for the j^{th} dimension, respectively. $m_{j,G}$ is the j^{th}-dimension value of the *gbest* particle in the population. $randN_{j,G}$ is a random number with normal distribution of mean $\mu_{j,G}$ and variance $\sigma^2_{J,G}$ and then truncated to [0, 1]. The values of $\mu_{j,G}$ and $\sigma^2_{J,G}$ are as follows

$$\mu_{j,G} = \frac{m_{j,G} - a_{j,G}}{b_{j,G} - a_{j,G}} \quad (9)$$

$$\sigma_{J,G} = (1 - \frac{k}{T}).\overline{\sigma_{J,G}} \quad (10)$$

where

$$\overline{\sigma_{J,G}} = max(\mu_{j,G}, 1 - \mu_{j,G}) \quad (11)$$

In Eq.(10), T is a predefined maximum number of generation and k is the current generation. $\sigma_{J,G}$ decreases with the evolutionary progress by Eq.(10), so the diversity of the j^{th} dimension also decrease with evolutionary progress. From

Eq.(9) and (10), the new value of $x_{i,j,G+1}$ is generated nearby the best particle $m_{j,G}$ with a large probability but far way from $m_{j,G}$ with a small probability. This scheme is able to enable the algorithm to further exploit the local area of a revisited location if the area is not sufficiently searched. The algorithm is also able to explore other promising areas.

Algorithm 1 illustrates the framework of the population adaptation (PA). In order to prevent the current best solution from being destroyed, the PA approach does not re-diversify the best solution found so far by the whole population.

Algorithm 1 Population Adaptation (PA)

```
1: Compute D_G
2: Compute Z_G using Eq. (5)
3: if Z_G = 1 then
4:    for each dimension j = 1, 2, ..., D do
5:       for each particle X_{i,G}, i = 1, 2, ..., N do
6:          if x_{i,G} is not the best particle then
7:             Regenerate x − i, j, G + 1 of the next generation
               using Eq. (6)
8:          end if
9:       end for
10:   end for
11:   Evaluate the fitness for each particle X_{i,G+1}, i =
      1, 2, ..., N
12: end if
```

B. PAPSO Algorithm

The PSO algorithm with PA, called PAPSO, applies PA approach after PSO operator at each iteration. The pseudo-code of PAPSO is presented in Algorithm 2. Compared with the original PSO algorithm, only the step 10 is added to perform Algorithm 1. When velocity and position of a particle exceeds the search range after the velocity and position update rule, we map $x_{i,j}$ and $v_{i,j}$ legal as follows

$$v_{i,j} = \begin{cases} V_{max} & \text{if } v_{i,j} > V_{max} \\ v_{i,j} & \text{if } -V_{max} \leq v_{i,j} \leq V_{max} \\ -V_{max} & \text{if } v_{i,j} < -V_{max} \end{cases} \quad (12)$$

$$x_{i,j} = \begin{cases} X_{max} & \text{if } x_{i,j} > X_{max} \\ x_{i,j} & \text{if } X_{min} \leq x_{i,j} \leq X_{max} \\ X_{min} & \text{if } x_{i,j} < X_{min} \end{cases} \quad (13)$$

In this paper, this method is used for all algorithms to handle the situation when particles are beyond the search range.

Algorithm 2 PAPSO Algorithm

```
1: Initialize velocity and position randomly for each particle
   P_0
2: G = 1
3: while The stoping criterion is not satisfied do
4:    Adapt inertia weight (ω)
5:    Evaluate the fitness of each particle
6:    Update pbest and gbest
7:    for each particle do
8:       update particle's velocity and position according to
         the Eqs. (1) & (2)
9:    end for
10:   Implement the population adaptation using Algorithm
      1
11:   G = G + 1
12: end while
```

IV. EXPERIMENTAL RESULTS

A. Benchmark Functions

In this section, PAPSO is applied to minimize a set of 13 scalable CEC05 benchmark functions in dimensions $D = 10$ and $D = 30$. In Table I, these CEC05 functions f_1 - f_{13} include shifted functions, rotated functions and rotated shifted functions. A more detailed description and parameter settings of these CEC05 functions can be found in [13]. The functions f_1 - f_5 are unimodal functions, f_6 - f_{11} are basic multimodal functions, f_{12} - f_{13} are expanded multimodal functions.

B. Experimental setup

For the purpose of performance evaluation, we compare the purposed PAPSO algorithm with the standard PSO (SPSO) [14], ATREPSO [7] and ARPSO [6]. In order to make a fair comparison of these algorithms, we fixed the same seed for random number generation so that the initial population is same for all the four algorithms. The number of particles in the population is 50. For each algorithm, the maximum number of iterations is allowed 2000 for 10-dimensions and 6000 for 30-dimensions. The parameter settings of all the algorithms are as follows

a) PAPSO: A linearly decreasing inertia weight is used which start at 0.9 and end at 0.4 with C_1 = 1.49 and C_2 = 1.49.

b) SPSO: C_1 = 1.49612, C_2 = 1.496172 and ω = 0.72984 as used or recommended in [11].

c) ATREPSO: Linearly decreasing inertia weight from 0.9 to 0.4 with C_1 = C_2 = 2. The diversity controlling parameters d_{low} = $5.0 * 10 - 6$ and d_{high} = 0.25 respectively.

d) ARPSO: The ω is the linearly decreasing function, $\omega(t)$ = $(1 - \frac{t}{t_{max}})$, where t is the current iteration number and t_{max} is the maximum iteration number. The diversity parameters d_{low} and d_{high} same as ATREPSO.

A total of 30 runs for each experimental setting were conducted and the average error best result throughout the run was recorded. All these algorithms are implemented using

TABLE I: CEC05 Benchmark functions. D denotes the dimensionality of the test problem, S denotes the range of the variables and f_{min} is the function value of global optimum

F	Function Name	S	f_{min}
f_1	Shifted Sphere Function	$[-100, 100]^D$	-450
f_2	Shifted Schwefel's Problem 1.2	$[-100, 100]^D$	-450
f_3	Shifted Rotated High Conditioned Elliptic Function	$[-100, 100]^D$	-450
f_4	Shifted Schwefel's Problem 1.2 with Noise in Fitness	$[-100, 100]^D$	-450
f_5	Schwefel,s Problem 2.6 with Global Optimum on Bounds	$[-100, 100]^D$	-310
f_6	Shifted Rosenbrock's Function	$[-100, 100]^D$	-180
f_7	Shifted Rotated Ackley,s Function with Global Optimum on Bounds	$[-32, 32]^D$	-140
f_8	Shifted Rastrigin's Function	$[-5, 5]^D$	-330
f_9	Shifted Rotated Rastrigin's Function	$[-5, 5]^D$	-330
f_10	Shifted Rotated Weierstrass Function	$[-0.5, 0.5]^D$	90
f_11	Schwefel,s Problem 2.13	$[-\pi, \pi]^D$	-460
f_12	Expanded Extended Griewank's plus Rosenbrock's Function	$[-0.5, 0.5]^D$	-130
f_13	Shifted Rotated Expanded Scaffer's Rastrigin's Function	$[-100, 100]^D$	-300

MATLAB 7.6.0 (R2008a) applied on Intel (R) Core (TM) i7-2670QM CPU @ 2.20 GHz with 8 GB RAM on windows 7 Home Premium platform.

C. Results and Discussions

Table II and Table III summarize the mean and standard deviation of the average error results over 30 independent runs for each algorithm on each function with $D = 10$ and $D = 30$ respectively. For each functions error best value of the results got by all the algorithms is shown in bold front. For 10-dimensions problems, from Table II, it can be seen that PAPSO performs better than SPSO, ATREPSO and ARPSO on 13 benchmark unimodal, basic and expanded multimodal functions. In the functions f_7, f_9, f_{12} and f_{13}, PAPSO performs better with compared to other algorithms. However, there is no significant difference is observed with respect to mean error values for the functions f_7, f_9, f_{12} and f_{13}. The standard deviation of the error values on the functions f_7, f_{12} and f_{13} are better in ARPSO than PAPSO. But PAPSO performs significantly better than SPSO, ATREPSO and ARPSO in terms of mean and standard deviation on the functions f_1 to f_6, f_8, f_{10} and f_{11} because of the PA approach does improve the performance of PSO. In Table III, mean error values obtained by PAPSO perform better than SPSO, ATREPSO and ARPSO in unimodal, basic multimodal and expanded multimodal functions except f_3. Moreover, mean error result is not highly significant than the PAPSO in SPSO for f_3. But standard deviation of the error value achieved by the proposed algorithm dominates all other algorithms on function f_3. PAPSO performs significantly better than SPSO, ATREPSO and ARPSO in terms of mean error values on the functions f_1, f_2, f_5, f_6, f_8, f_{11} and f_{12}. So the overall performance observed by proposed algorithm on the 13 benchmark functions with 10 and 30-dimensions is efficient and robust from the other algorithms.

We have shown the convergence graph in Fig. 1 and Fig. 2 for the basic multimodal function f_8 and the expanded multimodal function f_{13} with $D = 10$. The graph record the mean error of the best results over 30 independent runs for the functions. From Fig. 1, it can be seen that beginning of the iterations, SPSO converge faster than PAPSO. But end of the iterations PAPSO converge faster than other PSO algorithms. In Fig. 2, for the expanded multimodal function f_{13}, converge faster than SPSO, ATREPSO and ARPSO. In the Fig. 1 and Fig. 2, when the population is trapped in a local optimum or stagnates situation occur, PAPSO can jump out from the local optimum and continue to evolve because of the population adaptation and shown the faster convergence speed.

To study the working mechanism of population adaptation, we apply PAPSO to minimize the multimodal function f_8 in 10-dimensions. The Fig. 3 and Fig. 4 shows the change of D_G and the error result of the best particle. When the population diversity is unchanged or small change of D_G in two consecutive iterations and the algorithm stops evolving. Then PA is executed and the population diversity is enhanced. The algorithm continues to evolve and enhance the performance. From the figure, it has been shown that that when the population diversity unchanged, PA can enhance it. The new population generated by PA is beneficial to make the best particle to get better results.

V. Conclusion

The possible moves given by a population, some moves are beneficial in the search for the optimum while some others are ineffective and result in a waste of computational effort. Therefore, maintaining the diversity in the population is needed in such a way that the maximum number of moves will be beneficial in the search for optimum. This paper has studied the population adaptation on PSO and proposed a method of population adaptation, called PA.

Through measuring the Euclidean distances between particles position and average particles position of a population, PA can identify the moment when the population diversity is unchanged or small changed in two consecutive iterations. When the moment is identified, PA can regenerate the population by normal distribution. The new values are generated nearby the best particle by the proposed method. The experimental results show that PA can enhance the population diversity and improve the performance of PSO algorithm. In the experiment on 13 CEC05 benchmark functions, our proposed PAPSO algorithm outperforms from other algorithms.

Future research will involve investigating the effects of different distance measures e.g. Manhattan distance, Quadratic distance, etc. on the robustness of the different measures. Lastly, the behavior of the different diversity control methods will be investigated.

TABLE II: Average error values archived for 13 10-Dimensions CEC05 benchmark functions over 30 independent runs

F	PAPSO		SPSO		ATREPSO		ARPSO	
	Mean	Std. Dev	Mean	Std. Dev	Mean	Std. Dev	Mean	Std. Dev
f_1	**8.323E-03**	**1.249E-03**	4.832E+02	5.550E+02	6.818E+02	6.105E+02	6.709E+02	3.971E+02
f_2	**8.323E-03**	**1.249E-03**	4.832E+02	5.550E+02	6.818E+02	6.105E+02	6.709E+02	3.971E+02
f_3	**5.413E+05**	**6.371E+05**	6.369E+05	1.009E+06	1.820E+06	2.914E+06	4.093E+06	6.906E+06
f_4	**4.011E+01**	**5.932E+01**	2.789E+02	3.995E+02	9.056E+02	1.436E+03	4.685E+02	4.403E+02
f_5	**7.925E-03**	**1.705E-03**	1.822E+02	9.977E+02	1.048E+03	2.394E+03	7.833E+02	2.040E+03
f_6	**2.681E+04**	**1.447E+05**	7.436E+06	1.620E+07	5.106E+07	8.266E+07	1.625E+07	2.525E+07
f_7	**2.027E+01**	7.926E-02	2.028E+01	9.189E-02	2.029E+01	9.787E-02	2.031E+01	**5.661E-02**
f_8	**5.297E+00**	**4.611E+00**	8.958E+00	6.392E+00	7.151E+00	6.023E+00	8.819E+00	7.856E+00
f_9	**2.229E+01**	**6.025E+00**	2.408E+01	1.443E+01	2.320E+01	1.032E+01	2.748E+01	1.020E+01
f_{10}	**4.262E+00**	**1.159E+00**	5.212E+00	1.435E+00	5.494E+00	1.289E+00	5.396E+00	1.676E+00
f_{11}	**1.715E+03**	**2.640E+03**	3.950E+03	7.003E+03	2.399E+03	5.323E+03	3.586E+03	5.266E+03
f_{12}	**6.076E-01**	2.165E-01	6.551E-01	2.484E-01	6.512E-01	2.758E-01	6.243E-01	**1.988E-01**
f_{13}	**2.926E+00**	4.737E-01	3.321E+00	4.857E-01	3.017E+00	4.768E-01	3.078E+00	**4.453E-01**

TABLE III: Average error values archived for 13 30-Dimensions CEC05 benchmark functions over 30 independent runs

F	PAPSO		SPSO		ATREPSO		ARPSO	
	Mean	Std. Dev	Mean	Std. Dev	Mean	Std. Dev	Mean	Std. Dev
f_1	**9.372E-03**	**5.922E-04**	8.584E+03	7.669E+03	9.998E+03	4.617E+03	1.363E+04	5.188E+03
f_2	**1.819E+02**	**1.196E+02**	9.553E+03	9.686E+03	2.089E+04	8.624E+03	2.770E+04	1.588E+04
f_3	2.747E+07	**1.879E+07**	**2.463E+07**	2.026E+07	6.997E+07	4.794E+07	1.367E+08	9.299E+07
f_4	**1.260E+04**	**1.010E+04**	1.384E+04	1.475E+04	2.622E+04	1.609E+04	3.028E+04	1.880E+04
f_5	**9.583E+03**	**2.053E+03**	1.297E+04	3.440E+03	1.315E+04	3.688E+03	1.462E+04	3.861E+03
f_6	**7.044E+06**	**2.174E+07**	1.399E+09	1.253E+09	3.628E+09	2.736E+09	4.692E+09	2.705E+09
f_7	**2.087E+01**	6.348E-02	2.088E+01	**5.822E-02**	2.088E+01	6.134E-02	2.089E+01	6.664E-02
f_8	**7.855E+01**	3.580E+01	1.175E+02	3.019E+01	1.391E+02	3.122E+01	1.467E+02	**2.802E+01**
f_9	**1.594E+02**	**3.956E+01**	1.836E+02	4.558E+01	2.163E+02	5.070E+01	2.626E+02	4.582E+01
f_{10}	**2.418E+01**	**3.041E+00**	2.620E+01	3.919E+00	2.635E+01	3.686E+00	2.664E+01	3.293E+00
f_{11}	**9.739E+04**	**1.142E+05**	1.597E+05	1.188E+05	2.855E+05	1.411E+05	2.893E+05	1.178E+05
f_{12}	**4.264E+00**	**1.549E+00**	6.304E+02	2.381E+03	7.243E+03	1.537E+04	5.801E+03	9.415E+03
f_{13}	**1.229E+01**	6.128E-01	1.251E+01	4.819E-01	1.243E+01	4.057E-01	1.264E+01	**3.554E-01**

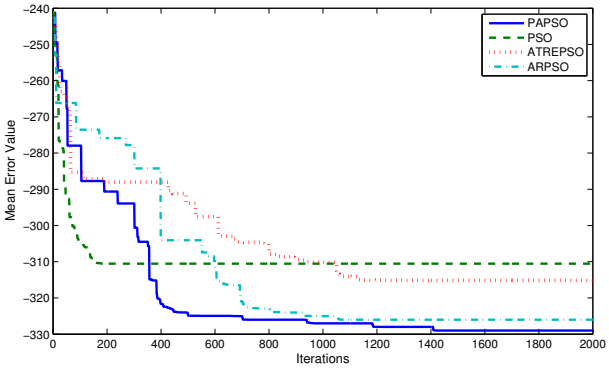

Fig. 1: Convergence graph for f_8 with 10-Dimensions

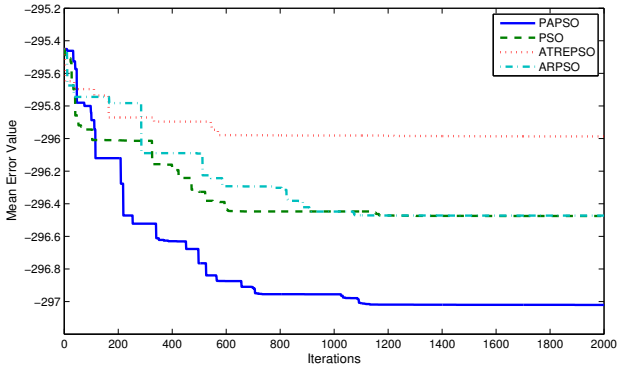

Fig. 2: Convergence graph for f_{13} with 10-Dimensions

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of IEEE International Conference on Neural Networks (ICNN95)*, Perth, IEEE Press, Australia, 1995, pp.1942-1948.

[2] R. Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995, pp.39-43.

[3] X. D. Li and A. P. Engelbrecht, "Particle Swarm Optimization: an introduction and its recent developments," in *Proceedings of Conference on Genetic and Evolutionary Computation*, 2007, pp.3391-3414.

[4] Y. H. Shi and R. Eberhart, "Population Diversity of Particle swarms," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2008, pp. 1063-1067.

[5] O. Olorunda and A. P. Engelbrecht, "Measuring Exploration/Exploitation in Particle Swarms using swarm diversity," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2008, pp. 1128-1134.

[6] J. Riget and A. Vesterstrom, "A Diversity Guided particle swarm optimizer - the ARPSO," EVALife Project Group, Department of Computer Science, Aarhus Universitym, Technical Report, 2002-02, 2002.

[7] M. Pant, T. Radha and V. P. Sing, "A Simple Diversity Guided Particle Swarm Optimization," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2007, pp. 3294-3299.

[8] S. Cheng and Y. Shi, "Diversity Control in particle swarm optimization," in *Proceedings of IEEE Swarm Intelligence Symposium*, Paris,France, 2011, .

[9] Z. Zhan, J. Zhang and Y. Shi, "Experimental Study on PSO Diversity," in *Proceedings of 3rd International Workshop on Advanced Computing intelligence*, 2010, pp. 310-317.

[10] S. Cheng, Y. Shi and Q. Qin, "Promoting Diversity in Particle Swarm Optimization to Solve Multimodal Problems," in *ICONIP*, LNCS, 2011, 7063, pp. 228-237.

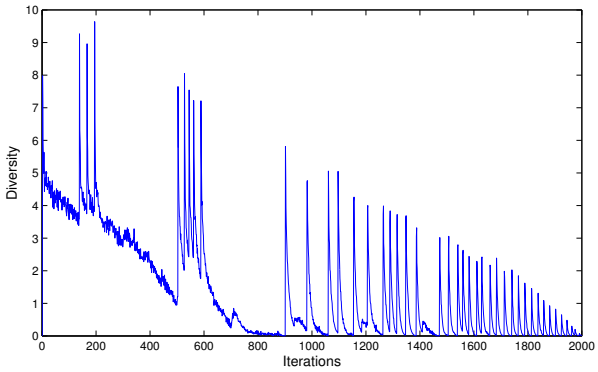

Fig. 3: The change of diversity (D_G) for the function f_8

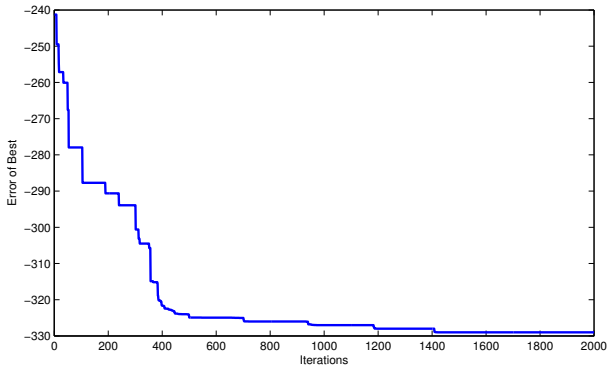

Fig. 4: The error result of the best particle for the function f_8

[11] S. Cheng, Y. Shi, Q. Qin and T. O. Ting, "Population Diversity Based Inertia weight Adaptation in Particle Swarm Optimization," in *Proceedings of 5th International Conference on Advanced Computational Intelligence*, 2012, pp. 395-403.

[12] A. P. Engelbrecht, "Fundamentals of Computational Swarm Intelligence," *John Wiley & Sons Ltd*, 2005.

[13] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization," Technical Report, NTU, Singapore, 2005.

[14] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *proceedings of the IEEE swarm intelligence Symposium (SIS 2007)*, April, 2007, pp. 120-127.