H. Nezamabadi-pour, M. Rostami-shahrbabaki, M.M. Farsangi, “Binary Particle Swarm Optimization: challenges and New Solutions”, The Journal of Computer Society of Iran (CSI) On Computer Science and Engineering (JCSE), vol. 6, no. (1-A), pp. 21-32, 2008.

Binary Particle Swarm Optimization: Challenges and New Solutions

Hossein Nezamabadi-pour , Majid Rostami-sharbabaki , Malihe Maghfoori-Farsangi

Department of Electrical Engineering, Shahid Bahonar University of Kerman , P.O. Box 76169-133, Kerman, Iran.
E-mail: nezam@mail.uk.ac.ir
Tel & Fax : +98-341-3235900

Abstract— Particle Swarm Optimization (PSO) algorithm, originated as a simulation of a simplified social system, is an evolutionary computation technique developed successfully in recent years and have been applied to many optimization problems. PSO can be applied to continuous and discrete optimization problems through local and global models. In this paper, PSO is addressed in details. There are some difficulties with the standard PSO where causing slow convergence rate on some optimization problems. These difficulties are transferred to the origin binary PSO (BPSO) that makes the algorithm not to converge well. Due to these difficulties with the BPSO, in this paper a new BPSO (NBPSO) is introduced. Several benchmark problems including unimodal and multimodal functions are considered for testing the robustness and effectiveness of the proposed method over the original BPSO. The results show that NBPSO performs much better than BPSO. Since the obtained results show that NBPSO may trap in the local optima, further modification is carried out. Two different methods are suggested to improve NBPSO which are denoted as Guaranteed Convergence BPSO (GCBPSO) and Improved NBPSO (INBPSO). The results show the superiority of the INBPSO for solving optimization problems.

Index Terms— Particle swarm optimization, Binary PSO, Convergence characteristic.

1. Introduction

Over the last decades there has been a growing interest in algorithms inspired by the observation of natural phenomenon. It has been shown by many researches that these algorithms are good replacement as tools to solve complex computational problems. Various heuristic approaches have been adopted by researches including genetic algorithm, tabu search, simulated annealing, ant colony and particle swarm optimization.

PSO can be classified in swarm intelligence areas, where developed by Kennedy and Eberhart in 1995 [1]. Since 1995, it is being researched and utilized in different subjects by researches around the world. It is reported in the literature that the PSO technique can generate high-quality solution within shorter calculation time on some optimization problems.

The PSO technique conducts searches using a population of particles, corresponding to individuals. Each particle tries to search the best position (state) with time in a multidimensional space and adjusts its position in light of its own experience and the experiences of its neighbors, including the current velocity and position and the best previous position experienced by itself and its neighbors. The origin version of the particle swarm has been operated in continuous space. But many optimization problems are set in discrete space. In view of this, two years later the work carried out by Kennedy and Eberhart in 1997 [2] a reworking of the algorithm to operate on discrete binary variables. In spite of continuous PSO that trajectories are defined as changes in position on some number of dimensions, in the binary version of PSO, trajectories are changes in the probability that a coordinate will take on a zero or one value. As mentioned before, since 1995, PSO and BPSO are being researched and utilized in different subjects such as power systems [3][4][5], neural network learning[6][7], data clustering[8], FPGA routing[9], TSP modeling[10], feature selection[11], and other applications, by researches around the world. Some researches are shown that standard PSO and BPSO cannot converge properly. In view of this, a solution is given to overcome the difficulty associated with the standard PSO by Van den Bergh and Engelbrecht [13].

In this paper, a new discrete binary PSO (NBPSO) is presented that deals with the difficulties associated by BPSO. To show the effectiveness of the proposed algorithm, the algorithm is tested on several function optimization problems and compared with the original version of BPSO. The results obtained show that NBPSO converge very well.

Also, NBPSO may fell into a local optimum early in a run on some optimization problems. In the other word, the algorithm approaches the neighborhood of the global optimum but for some reasons it fails to converge to the global optimum. To overcome the premature convergence of NBPSO, two different methods are suggested.

This paper organized as follows: to make a proper background, next Section briefly presents an overview of PSO and BPSO. In Section 3 the difficulties associated with BPSO are addressed. Description of the proposed new binary PSO is given in Section 4. Section 5 gives the obtained results on the 11 benchmark functions using BPSO and NBPSO. Section 6 describes the reason of stagnation in NBPSO and follows by giving the solutions to overcome the stagnation in Section 7. Section 8 gives the obtained results by two improved algorithms. Finally, the paper is concluded in Section 9.

2. Overview of PSO Algorithm

PSO is motivated from the simulation of social behavior. This optimization approach update the population of individuals by applying an operator according to the fitness information obtained from the environment so that the individuals of the population can be expected to move towards better solution areas. The continuous and discrete versions of PSO are explained below.

2.1 Standard PSO (Continues Version of PSO)

Similar to evolutionary algorithms, the PSO technique conducts searches using a population of particles, corresponding to individuals. Each particle represents a candidate solution to the problem at hand. In a PSO algorithm, particles change their positions by flying around in a multidimensional search space until a relatively unchanged position has been encountered, or until computational limitations are exceeded.

Bird flocking optimizes a certain objective function. Each particle (individual) knows its best value so far and its position (called as personal best or p_best_i for i^{th} particle). The information corresponds to personal experiences of each agent. Moreover, each particle knows the best value so far in the group among (known as global best or g_best).

Namely, each particle tries to modify its position (x_i) using the following information:

• the distance between the current position and p_best.

• the distance between the current position and g_best.

This modification can be represented by the concept of velocity (v_i). The velocity and the position of each particle in a d-dimensional space, can be modified by the following equations [1]:

$$v_{id}(t+1) = w.v_{id}(t) + c_1.rand()(p\_best_{id} - x_{id}) + c_2.Rand()(g\_best_d - x_{id}) \qquad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \qquad (2)$$

where $rand()$ and $Rand()$ are two random functions in the range [0,1], c_1 and c_2 are positive constants and w is the inertia weight. $x_i = (x_{i1}, x_{i2},..., x_{iD})$ represents the position of i^{th} particle. The rate of the position change (velocity) for particle i is represented as $v_i = (v_{i1}, v_{i2},..., v_{id})$. The best previous position (the position giving the best fitness value) of the i^{th} particle is recorded and represented as $p\_best_i = (p\_best_{i1}, p\_best_{i2},..., p\_best_{id})$.

As a particle moves through the search space, it compares its fitness value at the current position to the best fitness value it has ever attained at any time up to the current time.

There are two PSO models known as global model (or g-best) and local model (or l-best). The equations (1)-(2) represent global model. In the global model, all the particles in the swarm interact with the g_best while in the local model, each particle interact with the local best particle (p_best_i for i^{th} particle).

2.2 BPSO algorithm

The search space in BPSO is considered as a hypercube in which a particle may be seen to move to nearer and farther corners of the hypercube by flipping various numbers of bits. The moving velocity is defined in terms of changes of probabilities that a bit will be in one state or the other. Thus a particle moves in a state space restricted to 0 and 1 on each dimension, where each v_{id} represents the probability of bit x_{id} taking the value 1. With this definition p_{id} and x_{id} are integers in {0, 1} and v_{id} , since it is a probability, must be constrained to the interval [0.0, 1.0]. By defining a logistic function transformation $S(v_{id})$ in equation (3), the position will be updated according to equation (4).

$$S(v_{id}) = Sigmoid(v_{id}) = \frac{1}{1+e^{-v_{id}}} \tag{3}$$

$$\begin{aligned} &if \quad rand() < S(v_{id}(t+1)) \quad then \quad x_{id}(t+1) = 1 \\ &\qquad\qquad\qquad\qquad\qquad\quad else \quad x_{id}(t+1) = 0 \end{aligned} \tag{4}$$

where $S(v_{id})$ is a sigmoid limiting transformation and $rand()$ is a quasi-random number selected from a uniform distribution in [0.1, 1.0].

In the continuous version of PSO, v_{id} is limited by a value $v_{\max}$. Also in the discrete version of PSO, v_{id} is limited in the range of [-$v_{\max}$, $v_{\max}$]. Usually $v_{\max}$ is set to be 6. Although this setting limits the probability to be in [0.0025 0.9975] but will be resulted in a better convergence characteristics.

It should be noted that as standard PSO, the BPSO can be implemented through global and local models. In this paper, both models are used.

3. Disadvantages of the BPSO

As mentioned in Section 2 particle in the continuous PSO are defined by x_{id} and v_{id} in which x_{id} the position of the particle representing a candidate solution to the problem and v_{id} describes the velocity. A big value of the velocity shows that the current position of the particle is not proper and there is a big distance reaching to the optimum position. It means that greater movement is required to reach to the optimum position (equation 2). On the other hand, having small value for v_{id} implies neighboring to the optimum solution for which particle velocity becomes zero.

While updating of the particle position is realized in the continuous PSO by the position and the velocity information, in BPSO the particle position is not realized by the position and the velocity information. BPSO updates the velocity based on the equation 4 and consider the new position to be 1 or 0 with a probability. On the other hand, the value of v_{id} represents the probability of x_{id}, having value of 1 or 0. The probability is obtained by applying a sigmoid transformation to the velocity (equation 3) in which sigmoid transformation is a limiting transformation on v_{id}. The sigmoid transformation function is shown in Fig. 1.

A big value for v_{id} in BPSO does not mean a big change (movement) is needed for x_{id} . Having a big value for v_{id} (in the direction of positive values) increase the probability of x_{id} to take the value 1 without considering of previous position. In the same way, having a big value for v_{id} (in the direction of negative values) increase the probability of x_{id} to take the value 0 without considering of previous position. Also, if v_{id} becomes zero, the position (x_{id}) still will be changed, and it takes the value of 1 or 0 with the probability of 0.5.

According to the above descriptions, the following disadvantages can be appointed to the original BPSO:

- The first drawback relates to sigmoid function (equation 3, Fig.1). In the standard PSO there is no difference between a big value of v_{id} in the positive and negative direction and it just shows that the greater movement is required based on the previous position. However in the binary PSO, a difference is associated so that increasing the value in the positive direction causes bigger probability (probability of 1) for the particle position and increasing in the negative direction causes probability of zero. Also, in the standard PSO while the particle velocity for a particular dimension goes to zero, it means that particle has a suitable position in that dimension. While, in the BPSO using sigmoid function, the position may be changed and with the probability of 0.5, x_{id}, takes the value of 1 or 0.

- The second disadvantage relates to the position updating equation (Equation 4). Updating the position is performed without considering the previous position.

The obtained results by BPSO in solving different problems show that the average cost function is improving at the first few iterations. It means that the algorithm is getting close to the optimal solution but as the algorithm continues the particles diverge from the optimal solution and may trapped in local optimum. The reason of the divergence can be found in the first disadvantage explained above. When the algorithm is reached to the optimum solution, the probability of changing the position of the particle must be near to zero, while at this point using sigmoid function, the position will change by taking the value of 1 or 0 with the probability of 0.5. This causes the algorithm not to converge well.

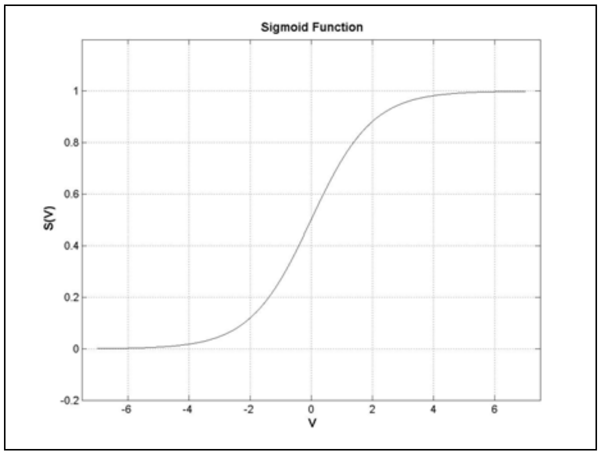


Fig.1 sigmoid function

4. NBPSO Algorithm

To overcome the first disadvantage associated with the BPSO, a proper probability function can be defined. In the algorithm proposed in this paper, instead of sigmoid function ($S(V_{id})$ equation 3), Equation 5 is used which is shown in Fig.2. By considering Equation 5, there is no difference between the big value of positive and negative velocities. In the other word, when v_{id} has a big value then $S'(V_{id})$ will get a big value. Also, for v_{id} close to zero the probability is close to zero.

$$S'(V_{id}) = 2 \times \left|(Sigmoid(v_{id}) - 0.5)\right| \qquad (5)$$

It should be noted that we may use $S'(V_{id}) = |\tanh(\alpha v_{id})|$ instead of equation 5.

To overcome the second disadvantages of BPSO, Equation 6 is substituted of Equation 4, in which a big value for v_{id} shows the position is not good and it changes from 0 to 1 or vice versa. Also a small value for v_{id}, decrease the probability of changing the position and when v_{id} becomes zero, the position will remain unchanged.

$$\begin{array}{l} if \quad rand() < S'(v_{id}(t+1)) \quad then \quad x_{id}(t+1) = exchange(x_{id}(t)) \\ \qquad\qquad\qquad\qquad\qquad\quad else \quad x_{id}(t+1) = x_{id}(t) \end{array} \tag{6}$$

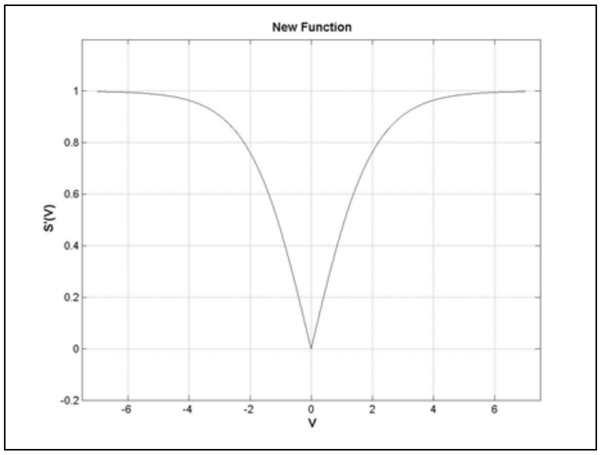


Fig 2. The proposed function in (5)

5. Benchmark Functions and Implementation of NBPSO

To show the capability of NBPSO in solving different problems, 11 benchmark functions are used [16]. These functions are given in Table 1, where *n* is the dimension of the function, f_{opt} is the optimum value of the function and $S \subseteq R^n$. The first ten functions will be minimized and the last one will be maximized. The first seven functions (f_1 to f_7) are unimodal functions where for unimodal functions, the convergence rates of the algorithm are more interesting than the final results of optimization (since there are some approaches that are specifically designed to optimize unimodal functions). f_8 to f_{10} are multimodal functions having many local minima and the algorithm must be capable in finding the optimum solution (or a good near-global optimum) and it should not be trapped in local optima. The last function (f_{11}) has discrete nature and maximize 1’s in a binary string.

Table1. The 11 benchmark functions used in the experiment

	Function’s name	**Test Function**	***S***	f_{opt} **(optimum solution)**
1	Sphere Model	$f_1(x)=\sum_{i=1}^{n} x_i^2$	$[-100,100]^n$	0
2	Schwefel’s Problem 2.22	$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]^n$	0
3	Schwefel’s Problem 1.2	$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_i\right)^2$	$[-100,100]^n$	0
4	Schwefel’s Problem 2.21	$f_4(x)=\max_i\{\lvert x_i\rvert, 1\le i\le n\}$	$[-100,100]^n$	0
5	Generalized Rosenbrock’s Function	$f_5(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	$[-30,30]^n$	0
6	Step Function	$f_6(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]^n$	0
7	Quadratic Function i.e. Noise	$f_7(x)=\sum_{i=1}^{n} i x_i^4+random[0,1)$	$[-1.28,1.28]^n$	0
8	Generalized Schwefel’s Problem 2.26	$f_8(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500,500]^n$	depends on *n*
9	Generalized Rastrigin’s Function	$f_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$[-5.12,5.12]^n$	0
10	Ackley’s Function	$f_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-32,32]^n$	0
11	Max-Ones Problem	$f_{11}(x)=\sum_{i=1}^{100} x_i$	$\{0,1\}^{100}$	100

In applying BPSO and NBPSO on different benchmark functions, both global and local model are used and a comparison between two models is carried out. The implementation of both models in BPSO and NPSO is as below.

5.1 The use of global model in BPSO and NBPSO

To implement the BPSO and NBPSO on different benchmark functions, the following setting is used.

In the local model, the neiburhood is considered to be 3. For the first ten function, the dimension size is set to be $n=5$ by considering a string length of 15 bits for each dimension (the dimension of each particle is 75). The number of iteration is considered to be 70, which is the stopping criteria.

For f_8 to f_{10}, the number of population is 50 and the number of iteration is considered to be 70. The max-ones problem (f_{11}) maximize 1’s in a binary string. A string length of 100 bits is used in this paper.

The parameter in (1) must be tuned where, in this paper, $c_1 = c_2 = 2$, $v_{\max} = 6$ and the weight w is decreasing linearly from 0.6 to 0.2.

Finding the optimum solution is based on 50 independent runs under different random seeds. The average best-so-far and the average mean fitness of each run are recorded and averaged over 50 independent runs. To have a better clarity, the convergence characteristics in finding the solution for some of the benchmark functions listed in Table 1 are given in Figs. 3-7.

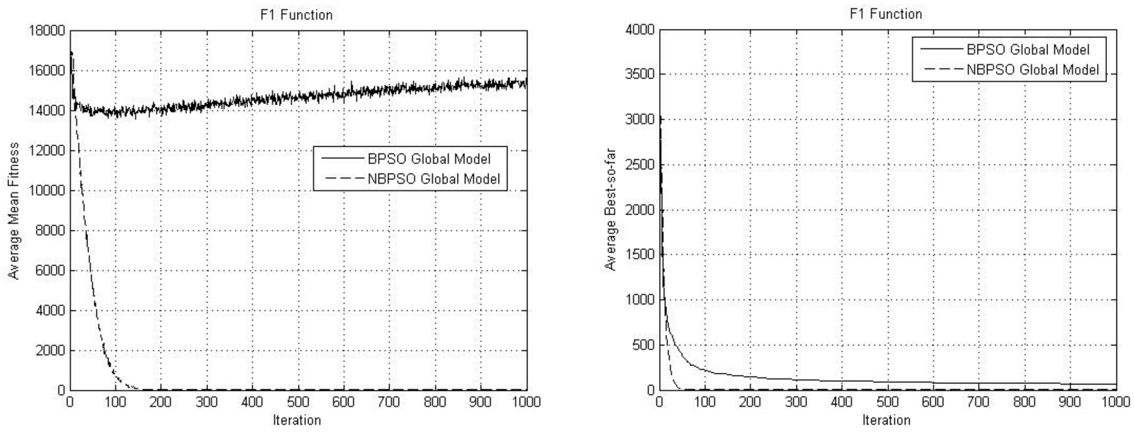
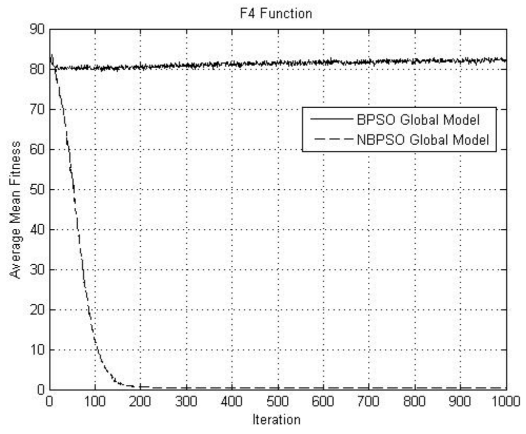


Fig. 3. Convergence characteristics of BPSO and NBPSO on f_1 in global model
a: average mean fitness; b: average best-so-far.



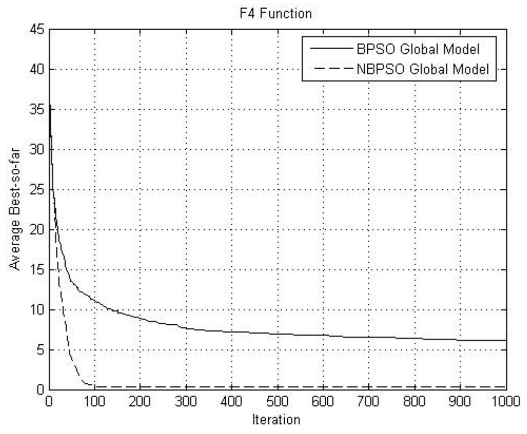
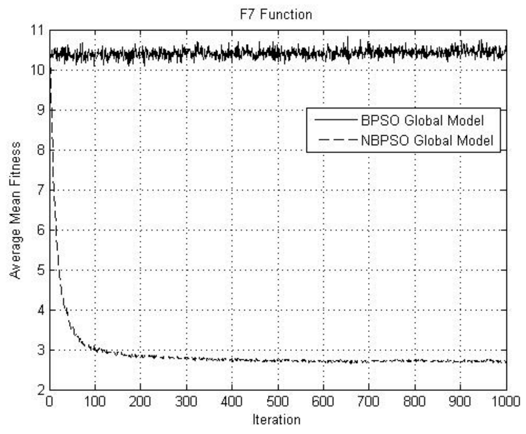


Fig. 4. Convergence characteristics of BPSO and NBPSO on f_4 in global model
a: average mean fitness; b: average best-so-far.



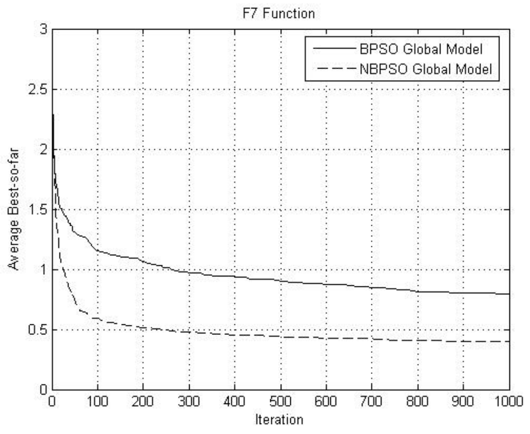
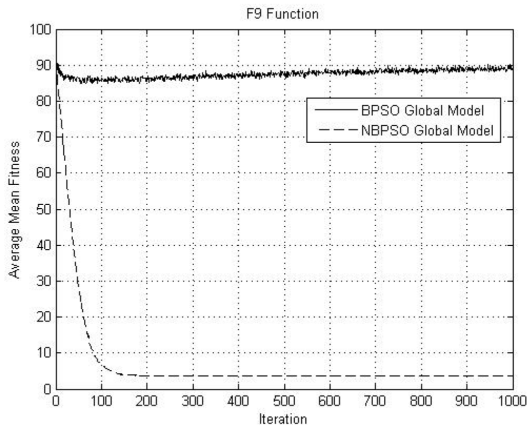


Fig. 5. Convergence characteristics of BPSO and NBPSO on f_7 in global model
a: average mean fitness; b: average best-so-far.



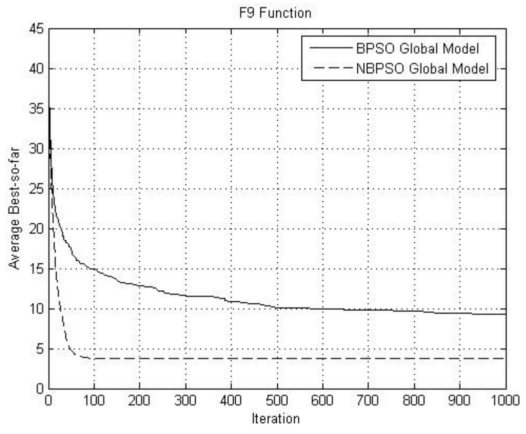


Fig. 6. Convergence characteristics of BPSO and NBPSO on f_9 in global model
a: average mean fitness; b: average best-so-far.

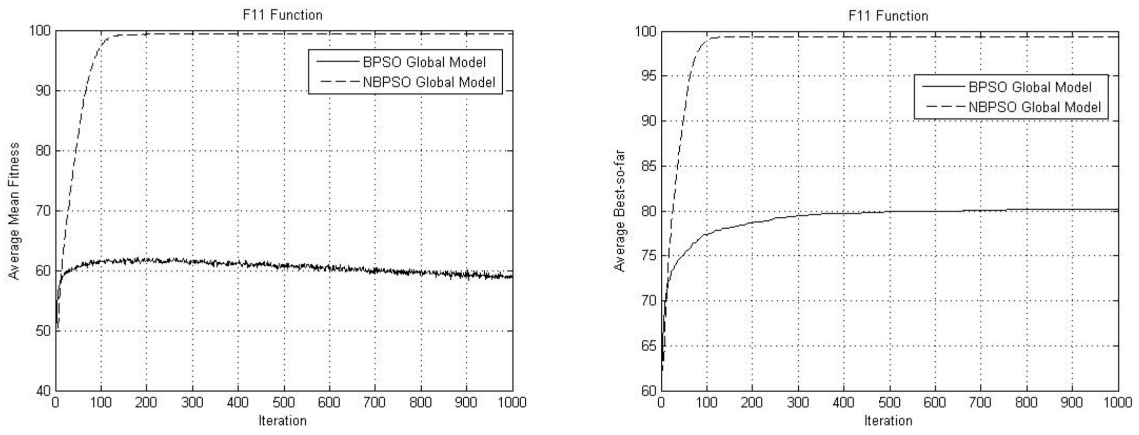


Fig. 7. Convergence characteristics of BPSO and NBPSO on f_{11} in global model
a: average mean fitness; b: average best-so-far.

The results shown in Figs. 3-7, reveal the difficulties associated by BPSO and rapid convergence of NBPSO supports our explanation in Sections 3-4 . As can be seen in Figs.3-7, the average mean fitness in BPSO did not converge to the optimum solution and is improved significantly by NBPSO. Also, NBPSO makes a significant improvement in average best-so-far comparing to BPSO.

5.2 The use of local model in BPSO and NBPS.

The same setting as global model in subsection 5.1 is used for local model in BPSO and NBPSO for different benchmark functions. Once again the obtained results by local model of both algorithms show that NBPSO performs better than BPSO in terms of convergence rate. The convergence characteristics of f_9 is shown in Fig. 8.

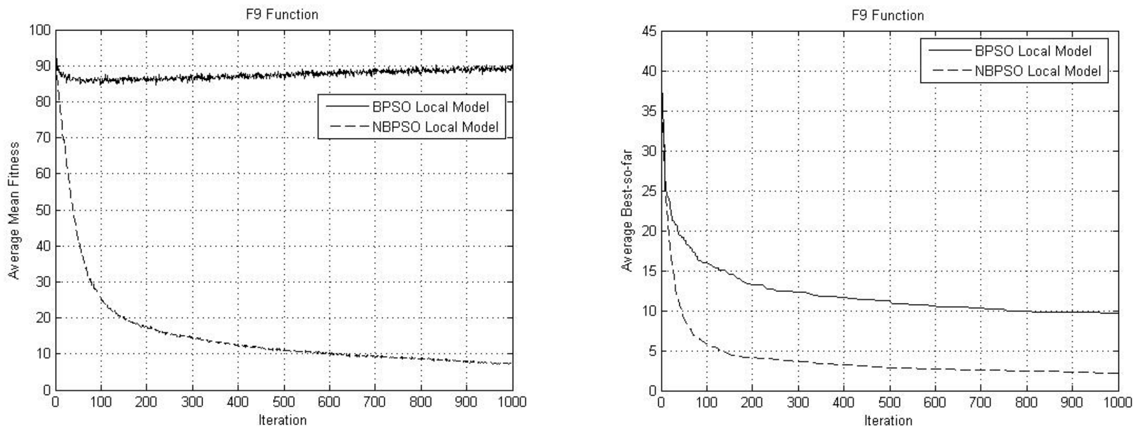


Fig. 8. Convergence characteristics of BPSO and NBPSO on f_9 in local model
a: average mean fitness; b: average best-so-far.

5.3 Comparison of local and global models in NBPSO

To have a comparison between the local model and global model of PSO, the convergence characteristic of f_6 (unimodal function) and f_9 (multimodal function) are shown in Fig. 9-10. These figures show that the global model finds the optimal solution faster than local model. This is because of the nature of two algorithms. In the global model, all particles in the swarm follow the best particle (global best) while in the local model, each particle interacts with the local best particle. Therefore it is expecting that the convergence rate of global model is faster than local model.

Comparing of Fig. 9 and Fig. 10 reveal that using the global model for the unimodal function is much better due to having fast convergence rate while using the local model for the multimodal is better than global model. Fig. 9 shows the reason. Multimodal functions having many local minima and the algorithm may trap in a local optimum. Therefore using the local model helps the algorithm to escape from local optimum and give a better solution.

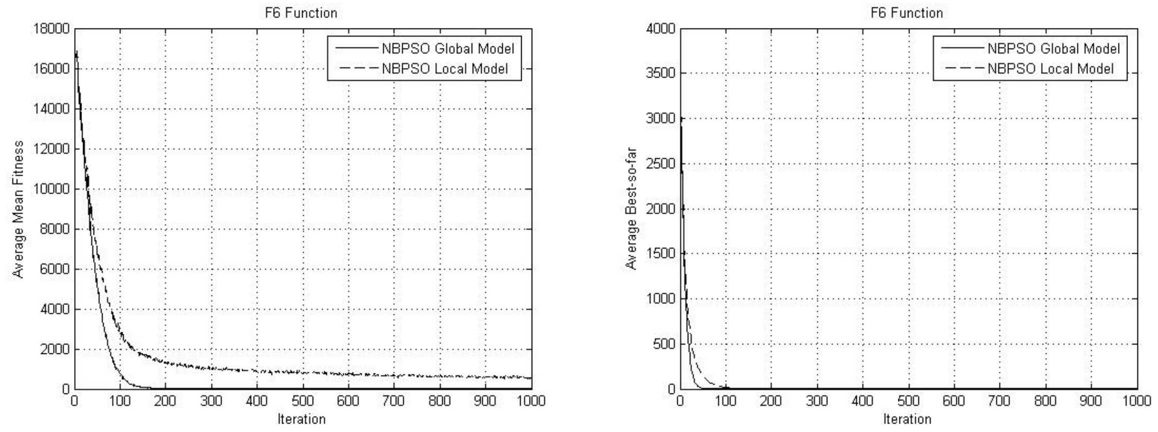


Fig. 9. Convergence characteristics of BPSO and NBPSO on f_6 in global model and local model
a: average mean fitness; b: average best-so-far.

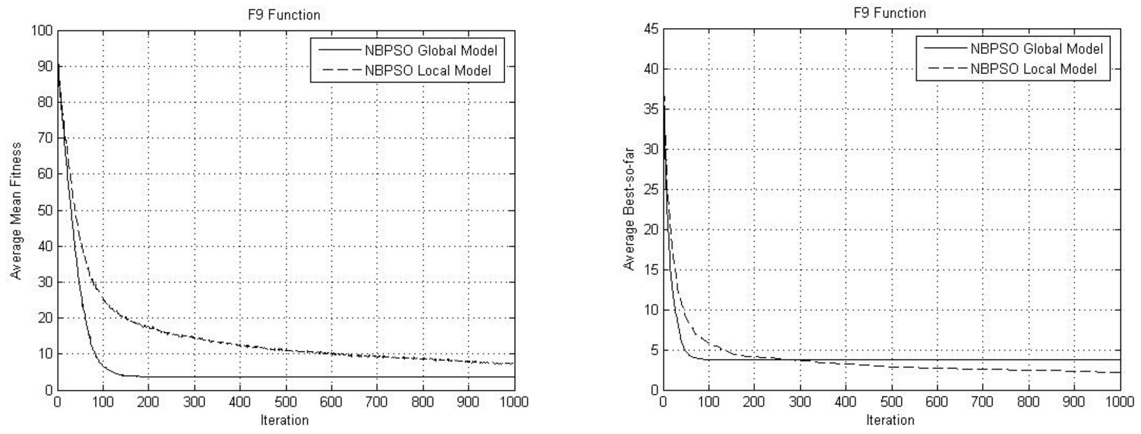


Fig. 10. Convergence characteristics of BPSO and NBPSO on f_9 in global model and local model
a: average mean fitness; b: average best-so-far.

Table 2 shows the comparison between BPSO and NBPSO on benchmark functions using global and local models. The results are average over 50 runs and the average best-so-far, the average mean fitness function and the median of the best solution in the last iteration are summarized in Table 3.

Table 2. Comparison between BPSO and NBPSO on benchmark functions where “Ave.best” indicates the average best-so-far found in the last iteration, and “Ave.fit” stands for the average mean fitness function.

Local model						Global model						benchmark function
NBPSO			BPSO			NBPSO			BPSO			
Median	Ave.best	Ave.fit	Median	Ave.best	Ave.fit	Median	Ave.best	Ave.fit	Median	Ave.best	Ave.fit	
0.0114	0.3614	599.139	87.149	86.493	1.504E4	**0.0035**	**0.0440**	**0.0440**	71.976	69.132	1.534E4	F_1
0.0076	0.0231	8.913	1.456	1.401	2219.2	0.0082	**0.0174**	**0.0174**	1.270	1.259	2210	F_2
5.205	**10.683**	2004.4	92.794	97.187	4.679E4	**4.999**	24.9435	**24.943**	62.484	66.436	4.69E4	F_3
0.4181	0.6893	5.4800	7.165	6.940	81.858	**0.1983**	**0.2885**	**0.2887**	5.935	6.082	82.443	F_4
7.988	**17.577**	2.272E6	925.61	1095.8	5.910E7	**4.508**	155.80	**155.80**	584.95	735.158	6.115E7	F_5
0.6541	0.7335	543.68	80.169	87.514	15367.3	**0.5970**	**0.6819**	**0.6819**	65.573	65.359	15274.2	F_6
0.4037	0.3990	3.223	0.7904	0.8101	10.359	**0.3940**	**0.3947**	**2.727**	0.8352	0.7948	10.450	F_7
-2094.8	**-2094.4**	**-2094.0**	-1963.3	-1973.4	-244.74	-2060.3	-2040.3	-2040.3	-2002.3	-2000.3	-224.86	F_8
2.261	**2.135**	7.234	9.796	9.683	88.949	3.288	3.692	**3.692**	9.250	9.256	88.747	F_9
0.1625	0.5734	2.488	5.902	6.001	20.753	**0.0716**	**0.4821**	**0.4821**	5.482	5.522	20.730	F_{10}
100	**100**	**100**	79	78.82	57.75	99	99.3	99.3	80	80.22	59.098	F_{11}

Based on the results shown in Table 2 and Figs. 9-10 the following can be concluded:

- Using global model for unimodal functions performs a better and faster solution than local model.
- Using local model for multimodal functions performs a better solution than global model.

- NBPSO may fell into a local optimum early in a run for both global and local models on some optimization functions. In the other word, the algorithm approaches the neighborhood of the global optimum but for some reasons it fails to converge to the global optimum. The reason is investigated in the next Section.

6. Stagnation and Premature Convergence of NBPSO in Local and Global Model

The reason of trapping the NBPSO in local optimum can be found in standard PSO. Standard PSO may converge at the early stage: the best particle moves based only on the inertia term since $P_g = pb_i = gb$ at the time step when it became the best. Later, its position may improve where $P_g = pb_i = gb$ holds again. Also, its position will worsen where it will be drawn back to $pb_i = gb$ by the social component. Therefore, it is possible for the inertia weight to drive all velocities to zero before the swarms manage to reach a local optimum. When all the particles collapse with zero velocity on a given position in the search space, then the swarms have converged, but this does not mean that the algorithm has converged on a local optimum. It merely means that all the particles have converged on the best position discovered so far by the swarm. This phenomenon is referred to as stagnation [13]. Thus; it is possible for the standard PSO to converge prematurely without finding even a local optimum.

The same thing will happen for NBPSO. When all particles are moving toward the best position (P_g), the velocity of the particles become zero. Then the probability of changing the position becoming zero based on equations 5-6. This phenomenon can be referred to as stagnation in NBPSO.

7. Improvement of NBPSO

Two different suggestions are given to improve the NBPSO algorithm. The first one is to improve the standard PSO, resulting in improvement of NBPSO. The second one, NBPSO will be improved directly in the discrete space without considering standard PSO. These two methods are described below.

7.1 Improvement of NBPSO by improvement of standard PSO

The Guaranteed Convergence PSO (GCPSO) was introduced by Van den Bergh and Engelbrecht [13] to address the issue of premature convergence to solutions that are not guaranteed to be local optima. The modifications to the standard PSO involve replacing the velocity update equation 2 of only the best particle (P_g) with the following equation:

$$v_{\tau,d}(t+1) = wv_{\tau,d}(t) - x_{\tau,d}(t) + P_{gd}(t) + \rho(t)(1-2rand) \qquad (7)$$

where the index of the best particle with the best position in the population is represented by the symbol τ . The first sentence of equation 7 is the same as equation 1. The second and the third sentences ($-x_{\tau,d}(t) + P_{gd}(t)$) replace the particle τ with P_g. The last sentence performs a random search around P_g with a radius $\rho(t)$ where defined as follows:

$$\rho(0) = 1.0$$

$$\rho(t+1) = \begin{cases} 2\rho(t) & if\,\#\,successes > s_c \\ 0.5\rho(t) & if\,\#\,failures > f_c \\ \rho(t) & otherwise \end{cases} \qquad (4)$$

where s_c and f_c are tunable threshold parameters.

A failure happens when $P_g(t) = P_g(t-1)$ otherwise success will happen. Whenever the best particle improves its personal best position, the success count is incremented and the failure count is set to 0 and vice versa. The success and failure counters are both set to 0 whenever the best particle changes. These modifications cause the best particle to perform a directed random search in a non-zero volume around its best position in the search space.

The above modification can be applied to NBPSO to improve the algorithm. The obtained algorithm denoted as "Guaranteed Convergence BPSO" (GCNBPSO).

7.2 Improvement of NBPSO in the discrete space

To improve the NBPSO in the discrete space, the equation 5 can be replaced by the following equation:

$$\begin{aligned} S''(v_{id}) &= A + (1-A).S'(v_{id}) \\ &= A + (1-A) \times \left|\tanh(\alpha\, v_{id})\right| \end{aligned} \qquad (10)$$

where A is a factor that prevents the stagnation of the algorithm. The equation 10 is shown in Fig. 11. As explained before, when all particles are moving toward the best position (P_g), the velocity of the particles becomes zero. Then the probability of changing the position becoming zero based on equations 5. Now by considering equation 10, if the velocity of the particles becomes zero the probability of changing the position is A. This modification gives a chance to the algorithm not to trap in the local optima and search for a new solution. This modification is similar to the mutation operator in evolutionary algorithm. It may comes to mind that considering A as a constant parameter is not good and it should be changed properly based on the changes in the algorithm. It the other word, when the algorithm is trapped in local minima, the value of A should be increased otherwise A should be decreased. Therefore A can be a variable parameter by the following equation:

$$A = k(1 - e^{-\frac{F}{T}}) \tag{11}$$

Where k is a constant parameter and T is a time constant that defined based on the dimension of the algorithm. F is failure counter. A failure happens if the best particle does not improve its personal best position, or when $P_g(t) = P_g(t-1)$. Therefore, when failure happens then F is incremented and when success happens F is set to 0. when $F = 0$ then A becomes 0. That means that if the algorithm is not trapped in a local minima, the mutation dose not apply to the algorithm and in fact NBPSO algorithm finds the solution.

This improved version of NBPSO is denoted as “Improved NBPSO” (INBPSO).

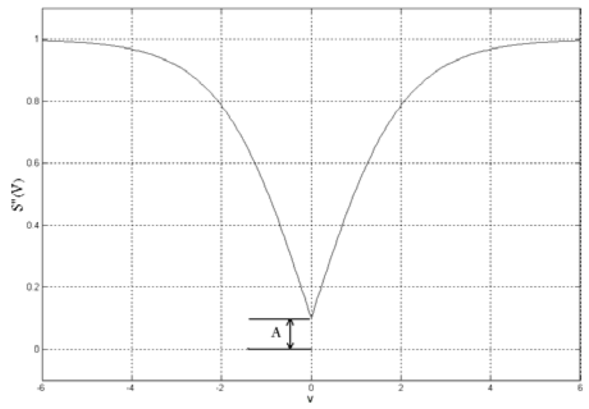


Fig 11. The proposed function in (10)

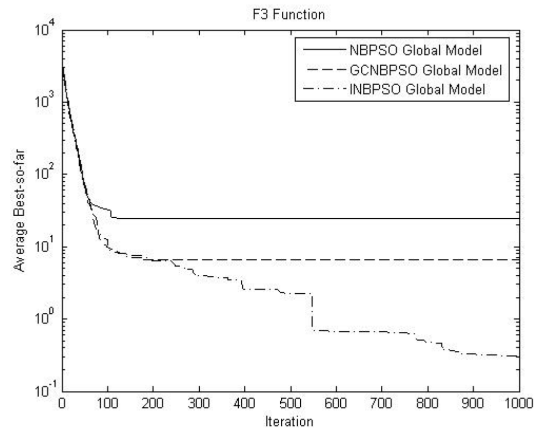
8. Comparing of Convergence Characteristic of NBPSO, GCNBPSO and INBPSO

Once again the benchmark functions listed in Table 1 are used to investigate the capability of the two improved version of NBPSO explained in Section 7. The same settings as before (Section 5) are used. Also, k and T are considered to be 1 and 1200, respectively. Both local and global model of PSO are applied. The obtained results are averaged over 50 runs. Table 3 summarizes the final results of GCNBPSO and INBPSO of global and local models in comparison with NBPSO.

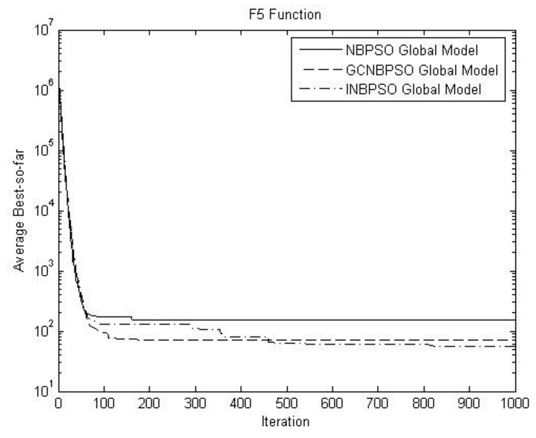
To have a better clarity, the average best-so-far of a few benchmark functions in finding the optimum solution over 50 independent runs are illustrated in Fig. 12.

Table 3. Comparison among NBPSO, GCNBPSO and INBPSO on the benchmark functions where “Ave.best” indicates the average best-so-far found in the last iteration, and “Ave.fit” stands for the average mean fitness function

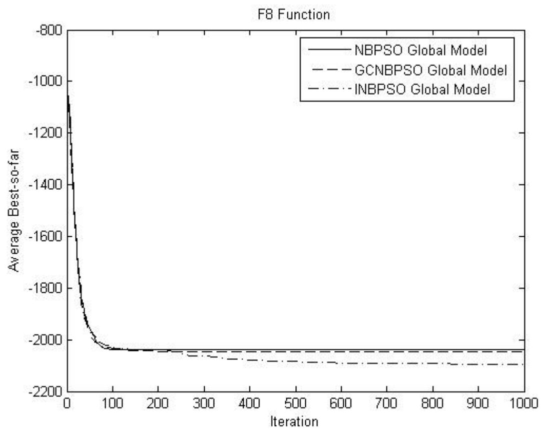
Local model						Global model						function
Median			Ave.best			Median			Ave.best			
INBPSO	GCNBPSO	NBPSO	INBPSO	GCNBPSO	NBPSO	INBPSO	GCNBPSO	NBPSO	INBPSO	GCNBPSO	NBPSO	
4.65E-5	44.14	0.0114	4.65E-5	58.93	0.3614	**4.65E-5**	4.65E-5	0.0035	**4.65E-5**	0.0145	0.0440	F_1
0.0015	1.0191	0.0076	0.0015	0.9942	0.0231	**0.0015**	0.0021	0.0082	**0.0015**	0.0039	0.0174	F_2
0.1966	110.56	5.205	1.6723	107.64	10.683	**0.0001**	0.3079	4.999	**0.3117**	6.5817	24.9435	F_3
0.0030	5.4933	0.4181	**0.0034**	5.6525	0.6893	**0.0030**	0.0335	0.1983	0.0065	0.1568	0.2885	F_4
3.7715	504.92	7.988	**3.8151**	1087.8	17.577	3.9958	3.9526	4.5084	55.924	72.282	155.80	F_5
0.2659	53.970	0.6541	**0.2944**	66.851	0.7335	0.5065	0.5663	0.5970	0.5721	0.6081	0.6819	F_6
0.4112	0.7519	0.4037	**0.4021**	0.7523	0.3990	0.4221	**0.3773**	0.3940	0.4114	0.3839	0.3947	F_7
-2094.8	-2094.8	-2094.8	**-2094.7**	-1937.0	-2094.4	-2094.7	-2064.3	-2060.2	-2094.6	-2048.0	-2040.3	F_8
0.9949	7.3436	2.2610	**0.8221**	7.5362	2.1356	1.2375	2.2358	3.2880	1.0343	2.7735	3.6922	F_9
0.0039	4.9097	0.1625	0.0039	4.3652	0.5734	**0.0039**	0.0064	0.0716	**0.0039**	0.1864	0.4821	F_{10}
100	90.5	100	100	89.66	100	**100**	100	99	**100**	99.66	99.3	F_{11}



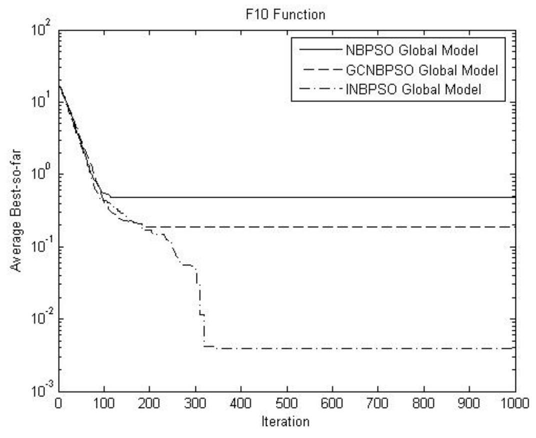
(a)



(b)



(c)



(d)

Fig. 12. Average best-so-far of NBPSO, GCNBPSO and INBPSO on a few benchmark functions a: f_3; b: f_5; c: f_8 and d: f_{10} .

Fig. 12 shows that the GCNBPSO and INBPSO improved the performance more than NBPSO. Also, it is quite clear that the performance of INBPSO improved much more than GCNBPSO. Furthermore, Fig. 12 reveal that while GCNBPSO algorithm as well as NBPSO algorithm are trapped in the local minima and converged prematurely, INBPSO algorithm still is searching for the new solutions.

Furthermore, Table 3 show that not only the local model of GCNBPSO does not improve the performance in comparing with NBPSO but also it is worsening the performance. Both local and global models of INBPSO have improved NBPSO’s performance significantly. Also, both local and global models of INBPSO, reach to similar solution. Therefore, the global model of INBPSO can be used instead of the local model due to the simplicity of the global model.

9. Conclusion

This paper addressed the PSO algorithm in depth and explained why the standard PSO and binary version of PSO cannot perform well in solving some problems. There are some disadvantages with the original BPSO that makes the algorithm not to converge well. In view of these difficulties associated with BPSO, this paper proposed an improved version of BPSO denoted as NBPSO. 11 benchmark problems used to evaluate the proposed algorithm (NBPSO). The obtained results show that, the NBPSO performing much better than BPSO.

Unfortunately, NBPSO as well as BPSO appeared to become trapped in a poor local optimum and unable to escape from it on some optimization function. This leads the algorithms to premature convergence. In view of this, two different methods are suggested to prevent the stagnation of the algorithms. One of these methods improves NBPSO through the concept of GCPSO. The improved version denoted as GCBPSO. The other improvement is achieved through a small change to the NBPSO. The new resulted version of INBPSO denoted as INBPSO. To validate GCPSO and INBPSO, a few experiments are carried out. It is very encouraging that INBPSO for both local and global models is capable of performing better than GCPSO and NBPSO in view of better convergence rate.

References

[1]. Kennedy J. and Eberhart R.C (1995) Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks(4).

[2]. Kennedy J. and Eberhart R.C (1997) A Discrete Binary version of the particle swarm algorithm. IEEE International Conference on Computational Cybernetics and Simulation(5).

[3] Zhao B, Guo C.X, and Cao Y.J (2005) A multi-agent-based particle swarm optimization approach for optimal reactive power dispatch. IEEE Transactions on Power Systems, 20(2), 1070 – 1078.

[4] Huang C-M, Huang C-J, and Wang M-L (2005) A particle swarm optimization to identifying the ARMAX model for short-term load forecasting. IEEE Transactions on Power Systems, 20(2), 1126 – 1133.

[5] Zhao B, Guo C.X, and Cao Y.J (2005) Correction to “A Multi-agent-Based Particle Swarm Optimization Approach for Optimal Reactive Power Dispatch. IEEE Transactions on Power Systems, 20(3), 1663 – 1663.

[6]. Lip H.B, Tang Y.Y, Meng J, and Jp Y (2004) Neural networks learning using vbest model particle swarm optimization. Proceedings of the 3rd International Conference on Machine Learning and Cybernetics, Shanghai, china.

[7]. Al-kazemi B, and Mohan C.K (2002) Training feed forward neural networks using multi-phase particle swarm optimization. Proceedings of the 9th International conference on Neural Information(5).

[8]. Merwe D, and Engelbrecht A (2003) Data clustering using particle swarm optimization. http://cirg.cs.up.ac.za/ publications/ CEC2003d.pdf

[9]. Gudise V.G, and Venayagamoorthy G.K (2004) FPGA placement and routing using particle swarm optimization. Proceedings of the IEEE Computer Society Annual Symposium on VLSI Emerging trends in VLSI Systems Design (ISVLSI’04).

[10]. Machado T.R, and Lopes H.S (2005) A hiybrid particle swarm optimization model for the traveling salesman problem. in: Ribeiro B et al., Adaptive and Natural Computing Algorithms. Springer, 255-258.

[11]. Firip H.A, and Goodman E (2004) Swarmed feature selection. in IEEE Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop, AIPR’04.

[12]. Yang S (2002) Adaptive Crossover in Genetic Algorithms using statistics mechanism. in Artificial Life VIII, 182-185.

[13]. Raymer M.L, Punch W.F, Gooddman E, Kuhn L.A, and Jain A.K (2000) Dimensionality reduction using genetic algorithm. IEEE Transaction on Evolutionary Computation, 4(2), 164- 171.

[14]. Jain A, and Zongker D (1997) Feature selection: evaluation, application, and small sample performance. IEEE Transaction on Pattern Recognition and Machine Intelligence, 19(2).

[15]. Manjunath B.S, and Ma W.Y (1996) Texture feature for browsing and retrieval of image data. IEEE PAMI, 18(8), 837-842.

[16]. Szummer M, and Picard R.W (1998) Indoor-outdoor image classification. IEEE Int. Workshop on Content-Based Access of Image and Video Database, Bombay.

[17]. Vailaya v, Jain A.K, and Zhang H.J (1998) On image classification: city vs. landscape. Pattern Recognition, 31, 1921-1935.

[18]. Nezamabadi-pour H and Saryazdi S (2004) Object-based image indexing and retrieval in DCT domain using clustering techniques, International Conference on Information Technology, ICIT2004, Turkey, 207-210.

[19]. Rubner Y, Puzicha J, Tomasi C, and Buhmann J.M (2001) Empirical evaluation of dissimilarity measures for color and texture. Computer Vision and Image Understanding, 84, 25-43.

[20]. Yao X, Liu Y, and Lin G (1999) Evolutionary programming made faster. IEEE Transaction on Evolutionary Computation, 3(2), 82-102.

[21]. Digalakis J.G, and Margaritis K.G (2002) An experimental study of benchmarking functions for genetic algorithms. International Journal of Computer Mathematics, 79(4), 403-416.