Research Journal of Applied Sciences, Engineering and Technology 4(22): 4798-4804, 2012
ISSN: 2040-7467
© Maxwell Scientific Organization, 2012
Submitted: April 25, 2012 Accepted: May 16, 2012 Published: November 15, 2012

A Comparative Study of Several Hybrid Particle Swarm Algorithms for Function Optimization

[1]Yanhua Zhong and [2]Changqing Yuan
[1]Department of Electronics and Information Technology, Jiangmen Polytechnic, Jiangmen 529090, China
[2]Aviation University of Air Force, Changchun 130022, China

Abstract: Currently, the researchers have made a lot of hybrid particle swarm algorithm in order to solve the shortcomings that the Particle Swarm Algorithms is easy to converge to local extremum, these algorithms declare that there has been better than the standard particle swarm. This study selects three kinds of representative hybrid particle swarm optimizations (differential evolution particle swarm optimization, GA particle swarm optimization, quantum particle swarm optimization) and the standard particle swarm optimization to test with three objective functions. We compare evolutionary algorithm performance by a fixed number of iterations of the convergence speed and accuracy and the number of iterations under the fixed convergence precision; analyzing these types of hybrid particle swarm optimization results and practical performance. Test results show hybrid particle algorithm performance has improved significantly.

Keywords: Differential evolutionary particle swarm optimization algorithm, function optimization, particle swarm optimization with GA algorithm, Quantum Particle Swarm Optimization (QPSO)

INTRODUCTION

Particle Swarm Optimization (PSO) is a population based stochastic optimization technique developed by Kennedy and Eberhart (1995), inspired by social behavior of bird flocking or fish schooling. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called p_{best}. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called l_{best}. When a particle takes all the population as its topological neighbors, the best value is a global best and is called g_{best}. The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p_{best} and l_{best} locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward p_{best} and l_{best} locations. In past several years, PSO has been successfully applied in many research and application areas. However, due to the particles in PSO the best position of their own history together to form the rapid convergence of the particle effects of population, prone to fall into local minimum, premature convergence or stagnation (Kennedy and Eberhart, 1995; Langdon and Riccardo, 2005; Clerc, 2006). In response to these shortcomings, some researchers have proposed many hybrid PSO.

Hybrid PSO is to other evolutionary algorithm or traditional algorithms or other optimization techniques applied to the PSO, the aim is to improve the particle diversity and enhance the global exploration ability of the particle, or to improve local development capacity and enhance the convergence speed and accuracy. There are usually two ways of combination: First, use other adaptive optimization shrinkage factor, inertia weight and acceleration constants; second, combined with PSO and other evolutionary algorithm operators or other technology. Juang (2004) combined the ant algorithm and the PSO for solving discrete optimization problems; (Robinson *et al*., 2002; Juang, 2004) the combination of GA and PSO are used tooptimize the antenna design and recurrent neural network design; Documents (Krink and Løvbjerg, 2002) divided the population into more sub-population and then the different sub-populations using PSO or GA or independent evolution of hill-climbing; (Naka *et al*., 2003) have studied that the selection operation of GAs introduced into the PSO according to select a certain rate of reproduction to copy better individuals; (Angeline, 1998) introduced tournament

Corresponding Author: Yanhua Zhong, Department of Electronics and Information Technology, Jiangmen Polytechnic, Jiangmen 529090, China

selected into the PSO algorithm, according to the current location of individual fitness, to each individual and compared to several other individuals and then compare the results to the whole group based on sorting, the best half of the particle in the current group replace the worst half of the position and speed of the position and speed, while preserving the memory of individuals of each individual the best position; (El-Dib *et al*., 2004) proposed crossover operation on theparticle position and speed; (Higashi and Iba, 2003) introduced Gaussian mutation to PSO; (Miranda and Fonseca, 2002) used the variation, selection and breeding a variety of operating at the same time update the formula in the neighborhood of the best locations and inertia weight and acceleration constants; Zhang and Xie (2003) using differential evolution operator to select the speed of update formula in the best position of particles; And Kannan *et al*. (2003) using differential evolution to optimize the PSO inertia weight and acceleration constants. Document proposed a discrete quantum individual PSO; Document proposed the quantum PSO that update particle position based on behavior of quantum.

These improved algorithms are declared that there good performance, it is necessary to test have these hybrid PSO algorithm using the same test functions that can compare those hybrid algorithm's actual performance. The following will select the standard Particle Swarm Optimization (bPSO) (Kennedy and Eberhart, 1995), differential evolution particle swarm optimization, Genetic Particle Swarm Optimization Algorithm (GAPSO) (Naka *et al*., 2003) and Quantum Particle Swarm Optimization algorithm (QPSO) to compare the algorithm performance. And select the Sphere function, Rosenbrock function and Rastrigin functions as test functions for testing.

METHODOLOGY

Several representative PSO algorithm:

Standard PSO (bPSO): The original PSO formulae defines each particle as potential solution to a problem in D-dimensional space. The position of particle i is represented as $x_i = \left[x_{i_1}, x_{i_2}, \cdots, x_{i_D}\right]$, Each particle also maintains a memory of its previous best position, represented as $p_i = \left[p_{i_1}, p_{i_2}, \cdots, p_{i_D}\right]$, A particle in a swarm is moving; hence, it has a velocity, which can be represented as $v_i = \left[v_{i_1}, v_{i_2}, v_{i_D}\right]$, the optimal position can be search in whole group which can be represented as $p_g = \left[p_{g_1}, p_{g_2}, \cdots, p_{g_D}\right]$,Updated the particle velocity and position as following:

$$v_{id}(t+1) = v_{id}(t) + c_1 r_1\left(p_{id} - x_{id}(t)\right) + c_2 r_2\left(p_{gd} - x_{id}(t)\right) \quad (1)$$

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

where, v_{id} - velocity of particle, $x_{i\,d}$ - current position of particle, c_1 - determine the relative influence of the cognitive component, c_2 - determine the relative influence of the social component, Usually take $c_1 = c_2 = 2$, p_{id} - pbest of particle i, p_{gd} - gbest of the group, r_1, r_2 - random numbers in Range [0, 1], t is the number of iterations, that is, particles flying steps.

limited the range of v, the particle velocity of each dimension are limited in between $[v_{min}, v_{max}]$ in order to prevent too fast and miss the optimal solution, where v_{max} is determined according to the actual situation. When the particles are small enough flying speed, or a preset number of iteration steps, the algorithm stop iteration and output the optimal solution.

Shi and Eberhart added the inertia weight in memory parts in order to better control algorithm optimization capability, so the formula (1) and (2) can be modified into the following:

$$v_{id}(t+1) = \omega(t) v_{ind}(t) + c_1 r_1\left(p_{id} - x_{id}(t)\right) + c_2 r_2\left(p_{gd} - x_{id}(t)\right) \quad (3)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (4)$$

This is called b PSO algorithm. ω- is non-negative, control over the previous impact on current speed. Control their size can be adjusted values of global and local optimization capability of PSO algorithm. When ω is larger, greater impact on former speed, Global search capability will be stronger, local search is weak; when ω is smaller, less impact on the previous speed, local search ability will be stronger, global search capability weakened. Formula ω (t) is expressed as follows:

$$\omega(t) = \left(w_{ini} - w_{end}\right)\left(MaxDT - t\right) / MaxDT + w_{end} \quad (5)$$

where,

MaxDT : Maximum number of iterations, w_{ini} the initial inertia weight.

w_{end} : The inertia weight when the particle evolve the maximum number of iterations, Typicaltake $w_{ini} = 0.9$, $w_{end} = 0.4$.

Hybrid DEPSO algorithm based on PSO and ED: The essence of PSODE algorithm thoughts is that the algorithm introduces new information exchange mechanism, so that information can pass in the two populations, to help the individual to avoid falling into local optimum due to the wrong information judgment. In order to improve the performance of hybrid optimization algorithm, documents proposed a non-linear inertia weight strategy. Inertia weight update status is as follows:

$$w = w_{\min} + \left(w_{\max} - w_{\min}\right) \cdot \exp\left[-\lambda \cdot \left(\frac{iter}{iter_{\max}}\right)^2\right] \quad (6)$$

where, λ is the controlling factor, control curve smoothness of w and *iter*, usually take λ = 3. In order to avoid mixing the PSO and DE algorithms in iterative optimization of the individual in the late stagnation, DEPSO introduced a mutation mechanism. If the individual (PSO and DE in either group) occurred in the evolution of stagnation within a predetermined maximum number of iterations, then the individual will be random variation, to achieve the following:

$$\begin{aligned} &if \\ &F\left(x_i^t\right) = F\left(x_i^{t-1}\right) = F\left(x_i^{t-2}\right) \\ &= F\left(x_i^{t-p}\right) and F\left(x_i^t \neq F^*\right) \\ &then \\ &x_i^{t-p+1} = X_{\min} + rand(0,1)\cdot\left(X_{\max} - X_{\min}\right) \end{aligned} \quad (7)$$

where, F^* is the global minimum of the fitness function, p is the maximum number of iterations to allow stagnation, $(X_{max} - X_{min})$ is defined to allow search.

The hybrid algorithm DEPSO implementation steps are as follows:

Step 1: Basic parameters, including: M-population size, T_{max}- maximum number of iterations, ε -solution accuracy, w_{max}-the maximum inertia weight, w_{min}-minimum inertia weight, λ -control factor, c_1,c_2-speed factor, F-scaling factor, CR-the mutation probability.

Step 2: Divided the group into two equal population POP^{PSO} and POP^{DE}, where POP^{PSO} and POP^{DE} be initialized the location of each in different regions.

Step 3: Set iteration counter t = 0.

Step 4: Update speed and location for all individuals in groups according to the Formula (2), (3) and (6).

Step 5: Performed on all individual variation, hybridization, selection based on differential evolution algorithm.

Step 6: Select the best individuals G_{BEST}^{PSO} in the group POP^{PSO}.

Step 7: Select the best individuals G_{BEST}^{DE} in the group POP^{DE}.

Step 8: Compares the Strengths and weaknesses between POP^{DE} and POP^{PSO} and choose the best individual as the basis for next-generation evolution.

Step 9: Determine whether there is stagnation of individuals, if any, according to Eq. (7) perform mutation operation.

Step 10: Update iteration counter and record the current best individual in thewhole group, if the precision to meet the requirements or the entire evolution has reached the maximum number of iterations, then terminate the algorithm, otherwise go to Step 4.

Hybrid optimization algorithm based on Genetic Algorithms and Particle Swarm (GAPSO): Genetic algorithms and particle swarm algorithm differs mainly in two aspects. First, the genetic algorithm is selected by a selection strategy for a number of individuals One on one match for breeding, while the PSO is the best for all individuals and groups of individuals for breeding, to generate the next generation of individuals; second, the genetic algorithm to be the mutation operations and PSO does not mutation operations, in the proposed hybrid algorithm based on particle swarm optimization as a template for these two different designs to get. Because the particle swarm algorithm in continuous space is a real-coded, so the genetic algorithm accordingly consider real-coded genetic algorithm.

For the second point is simple, just need to add a mutation operations in the particle swarm algorithm. On the first point is to increase a selectio numbers r in [0, 1] interval. With probability p_c perform the following cross:

$$\hat{x}^{gap} = r \cdot x_1^{gap} + (1-r)\cdot x_2^{gap} \quad (8)$$

$$\hat{x}_2^{gap} = r \cdot x_2^{gap} + (1-r)\cdot x_1^{gap} \quad (9)$$

$$\hat{v}^{gap} = r \cdot v_1^{gap} + (1-r)\cdot v_2^{gap} \quad (10)$$

$$\hat{v}_2^{gap} = r.v_2^{gap} + (1-r)\cdot v_1^{gap} \quad (11)$$

where,

gap : The number of iterations

x_1^{gap}, x_2^{gap} : Position vectors of parent individuals

v_1^{gap}, v_2^{gap} : Velocity vector of parent individuals

$\hat{x}_1^{gap}$, $\hat{x}_2^{gap}$: Position vectors of children Individual after crossover operation

$\hat{v}_1^{gap}$, $\hat{v}_2^{gap}$: Velocity vector of children Individual after crossover operation

Mutation operations: After the execution of crossover in p_m of the probability, individuals perform the following mutation operations:

$$\hat{x}_k^{gap+1} = \begin{cases} \hat{x}_k^{gap} + c_k \; if \; fitness\left(\hat{x}_k + c_k\right) > fitness\left(\hat{x}_k^{gap}\right) \\ \hat{x}_k^{gap} \qquad otherwise \end{cases} \quad (12)$$

$$\hat{v}_k^{gap+1} = v_k^{gap} \quad (13)$$

where, c_k -on Interval $\left[x^L - \hat{x}_k^{gap}, x^U - \hat{x}_k^{gap}\right]$ uniformly distributed random number, x^L, x^U-Search range of upper and lower limits, *fitness*(.) -Fitness function.

Here is the algorithm flow:

Step 1: Initialize the position and velocity of all individuals in the particle swarm, search the best individual position P_g in population, set the history of the individual best position P_i to the initial position, set the number of iterations gap = 0.

n and cross-breeding operation similar to the genetic algorithm process on the basis of the original breeding operation. These two steps are discussed below.

Selection, crossover process: First of all, selected M (even number) individuals according to some selection strategy, where used roulette wheel selection method. Then selected an individual to conduct a pair, the implementation of crossover, which randomly generates a real

Step 2: Update each particle's velocity and position according to Eq. 3 and (3) and calculate their fitness values.
Step 3: Terminate the program if the optimal solution to meet the output termination conditions, or continue to Step 4.
Step 4: Randomly selected M individuals according to the fitness value for crossover operations to get M new individuals.
Step 5: Execute mutation operation on all individuals, select N individuals in the high fitness into the next generation of M+N individuals, go to step 2

Quantum Particle Swarm Optimization (QPSO): Each particle with the quantum bits that is called qubit in QPSO proposed by Documents. A qubit, can be considered a superposition of two independent states |0> and |1>, denoted by |φ> = a|0>+b|1>, where a, b are complex numbers such that $|a|^2+|b|^2 = 1$, A composed system with N qubits is described using $N = 2^n$ independent states obtained through the tensor product of the Hilbert Space associated with each qubit. Quantum particles use probability amplitude or phase to represent. Qubit can also be expressed as follows, where θ is the phase of quantum bits:

$$|\varphi>=\sin\theta|0>+\cos\theta|1> \tag{14}$$

Initialize the first generation of particles, particle |0> state probability of α and particle |1> state probability of β, α is random number between [0, 1]. Mapping is:

$$S_{warm} = S_{warm} * \left(b_{\max} - b_{\min}\right) + b_{\min} \tag{15}$$

where, S_{warm} is a two-state information populations in the initialization, b_{max} and b_{min} are the upper and lower search. Particles in the algorithm are updated reference Particle swarm optimization; it can be expressed as follows:

$$\begin{aligned}\Delta\theta_{jd}^{t+1} &= \omega \times \Delta\theta_{jd}^{t} + C_1 \cdot R_1 \cdot \left(\theta_{GBEST} - \theta_{jd}\right) \\ &+ C_2^{jd} \cdot R_2 \cdot \left(\theta_{PBEST} - \theta_{jd}\right)\end{aligned} \tag{16}$$

where, $\Delta\theta_{jd}^{t+1}$ is phase shift of j particles of d dimensional in t+1 generation iteration; θ_{jd} is The current phase, θ_{GBEST} is global optimum particle phase; θ_{pbest} is the optimal phase of particle in history; ----is Inertia weight factor, taken as 0.6; C_1, C_2 is speed constant, the value is $C_1 = C_2 = 1.4$; R_1, R_2 is random number in [0, 1] . Updated revolving door according to the calculation of quantum phase for updating the particle, such as (17) shows:

$$\begin{bmatrix}\alpha_{jd}^{t+1}\\ \beta_{jd}^{t+1}\end{bmatrix} = \begin{bmatrix}\cos\Delta\theta_{jd}^{t+1} & -\sin\Delta\theta_{jd}^{t+1}\\ \sin\theta_{jd}^{t+1} & \cos\Delta\theta_{jd}^{t+1}\end{bmatrix}\begin{bmatrix}\alpha_{jd}^{t}\\ \beta_{jd}^{t}\end{bmatrix} \tag{17}$$

where, α_{jd}^{t+1}, β_{jd}^{t+1} is probability amplitude of j particles of d dimensional in t+1 generation iteration. This shows that particle update is based on phase changes and the phase difference between Global best particle and the best particle in the history, but when the population into a local optimum once after the particle update phase will soon be zero. To solve this problem, the introduction of adaptive probability and according to this probability to a random phase disturbance, to help the population out of the local optimum. Adaptive mutation probability is defined as:

$$P = \mu + Re * \delta \tag{18}$$

where, μ, δ Mutation rate of the adjustable parameters, Re-Optimal value is not updated continuously or updating the number of iterations is not obvious. Specifically:

$$\theta = rand(1, d) \tag{19}$$

where, θ is the Phase of j particles of d dimensional in t+1 generation iteration, algorithm flow is as follows:

Step 1: Initialize population; and the first generation population in the [0, 1] range.
Step 2: Express the quantum state according to (14) and first fitness assessment. If fitness meets the conditions, then go to Step 8, or turn to Step 3.
Step 3: Initialize the particle phase of the first generation in [0, 1] random number, calculate the optimal phase ofthe particle history, the global optimal particle phase, the particles fitness Expressed with the quantum states; Number of iterations plus 1; if the fitness meet the conditions then go to Step 8; otherwise transfer Step 4.
Step 4: Update the particle phase variation according to (16) and the use of Eq. (17) to update the particles.

Step 5: According to the formula (18) to determine the need for particle-phase transitions; if yes that process go to Step 6, Else directly go to Step 6.

Step 6: Observer and select particles according to the default probability of the particle state, make the particles collapse.

Step 7: Evaluate the fitness of the collapse of the particles, if conditions are satisfied, then go to Step 8.

Step 8: Jump out the algorithm and output the optimal value, the algorithm ends.

EXPERIMENTS AND RESULTS

Standard test functions:

De Jong's function 1: The simplest test function is De Jong's function 1. It is also known as sphere model. It is continuos, convex and unimodal. Function definition:

$$f_1(x)=\sum_{i=1}^{n} x_i^2, -5.12 \le x_i \le 5.12 \quad (20)$$

global minimum: $f_1(x) = 0$.

Rosenbrock's valley (De Jong's function 2): Rosenbrock's valley is a classic optimization problem, also known as Banana function. The global optimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult and hence this problem has been repeatedly used in assess the performance of optimization algorithms. Function definition:

$$f_3(x)=\sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) - 5.12 \le x_i \le 5.12 \quad (21)$$

Global minimum: $f_2(x) = 0$

Rastrigin's function: Rastrigin's function is based on function 1 with the addition of cosine modulation to produce many local minima. Thus, the test function is highly multimodal. However, the locations of the minima are regularly distributed. Function definition:

$$f_3(x)\sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) - 2.048 \le x_i \le 2.048 \quad (22)$$

Global minimum: $f_3(x) = 0$

Experimental design: Experiment used three kinds of base functions, Sphere, Rosenbrock, Rastrigin function. Linear decrease of the inertia weight, its range is [0.4, 0.9]. Specific parameters are: population size is set to m = 15, particles of dimension Dim = 30, maximum iteration step number 150:

$v_{max} = 1$; $c_1 = c_2 = 2$; $T_0 = 3$; $T_g = 5$

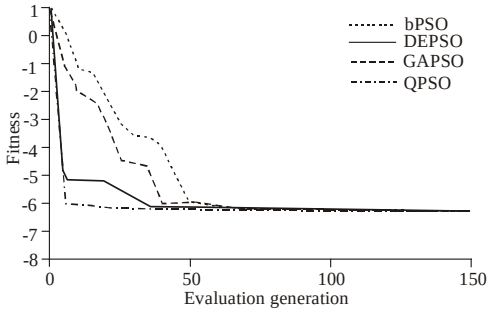

Fig. 1: Sphere test results

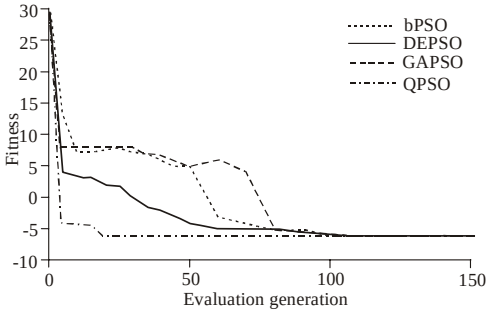

Fig. 2: Rosenbrock test results

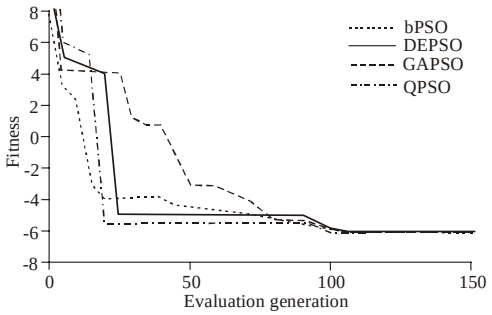

Fig. 3: Rastrigin test results

Appraised using the following methods:

- Assess the algorithm convergence speed and accuracy under fixed evolutionary iterations.
- Assess the algorithm convergence precision under a fixed convergence precision.

Experiments and analysis:

Convergence speed and accuracy under a fixed evolutionary iterations: bPSO, DEPSO, GAPSO, QPSO will be tested with three test functions, Evolution of the number of iterations fixed at 150 times, we Test the algorithm fitness of Sphere function, Rosenbrock function, Rastrigin function and test results are shown as Fig. 1, 2 and 3, respectively using sphere, Rosenbrock and Rastrigin function. Observe the effect of

Table 1: The experiments results

Experiment name		Iterations		
		Average	Min	Max
Sphere functionb	PSO	95	45	180
	DEPSO	15	10	30
	GAPSO	75	40	140
	QPSO	14	10	16
Rosenbrock functionb	PSO	350	250	600
	DEPSO	17	7	30
	GAPSO	240	100	400
	QPSO	15	7	30
Rastrigin functionb	PSO	410	100	700
	DEPSO	22	12	40
	GAPSO	220	400	140
	QPSO	18	9	40

Min: Minimum; Max: Maximum

these improved algorithm of DEPSO algorithm, GAPSO algorithm, QPSO evolutionary curve. (Note: In order to facilitate the display and observe the curve of evolution. Take 10 for the logarithm, on the fitness of function with a cutoff value of 10^{-6}.)

Figure 1, 2 and 3 show that DEPSO, GAPSO, QPSO three hybrid algorithm is indeed better than bPSO in convergence accuracy with the very significant increase, The QPSO has significantly improved than the other three algorithm, include the late evolution convergence speed, convergence precision and Get rid of local extremum ability. This shows that quantum superposition operator has a good performance optimization; QPSO algorithm has highest convergence accuracy and fastest convergence rate, QPSO algorithm has highest convergence accuracy and fastest convergence rate, further, after 15~60 generation evolution iteration, to achieve the target accuracy

The convergence precision under a fixed number of iterations: Table 1 lists the maximum number of iterations, minimum number of iterations, the average number of iterations through 20 independent run, Fitness value of 10^{-6}, convergence precision

Data in the table shows, QPSO average number of iterations to achieve convergence targets are less than 50 times. This indicates that, with quantum superposition operator QPSO algorithm optimization has a very high performance and practicality, Followed DEPSO, GAPSO algorithm just better than bPSO.

CONCLUSION AND PROSPECTS

This study analyzes and compares the standard particle swarm algorithm, differential hybrid particle swarm algorithm, hybrid genetic algorithms and particle swarm hybrid quantum particle swarm algorithm convergence and to achieve the best goal accuracy, test results show that the hybrid algorithm relative to the standard particle swarm algorithm in all aspects of performance is really a significant improvement, which the quantum particle swarm optimization highest performance.

PSO attracted more and more the concern and research of researchers, but research is still in its infancy, there are still many problems, although has been applied in many areas, but the lack of strong theoretical foundation to support and rigorous mathematical proof, Therefore, in the following aspects to be further research and improvement: to strengthen the theoretical basis for research: particle swarm optimization and integration of other algorithms; expanded PSO applications.

ACKNOWLEDGMENT

This study was supported by the Natural Science Foundation of China under Grant No. 10902125

REFERENCES

Angeline, P.J., 1998. Using selection to improve particle swarm optimization. Proceeding of the IEEE Congress on Evolutionary Computation [C]. Anchorage, USA, pp: 84-89.

Clerc, M., 2006. Stagnation Analysis in Particle Swarm Optimization or what Happens when Nothing Happens. Retrieved from http:// clerc. maurice. free.fr/pso/.

El-Dib, A., H. Youssef, M. El-Metwally and Z. Osman., 2004. Load flow solution using hybrid particle swarm optimization. Proceeding of International Conference Elect., Electron. Comput. Eng. [C], pp: 742-746.

Higashi, N. and H. Iba, 2003. Particle swarm optimization with gaussian mutation. Proceeding of the 2003 Congress on Evolutionary Computation [C], pp: 72289.

Juang, C.F., 2004. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE T. Syst. Man Cyb. B, 34(2): 997-1006.

Kannan, S.K., D.J. Sharf and L.U. Ganu, 2003. Genration Planning using Differential Evolution.

Kennedy, J. and R. Eberhart, 1995. Particle swarm optimization. Proceedings of the 4th IEEE International Conference on Neural Networks, Piscataway, IEEE Service Center, pp: 1942 -1948.

Krink, T. and M. Løvbjerg, 2002. The lifecycle model: Combining particle swarm optimization. Genetic Algorithms and Hill Climbers, Proceeding of PPSN, pp: 621-630.

Langdon, W.B. and P. Riccardo, 2005. Evolving problems to learn about particle swarm and other optimizers. Proceeding of CEC-2005 [C], 1: 81-88.

Miranda, V. and N. Fonseca., 2002. EPSO: Evolutionary particle swarm optimization, a new algorithm with applications in power systems. IEEE/PES Transm. Distr. Conf. Exhibition Asia Pacific., 2: 745-750.

Naka, S., T. Genji, T. Yura and Y. Fukuyama, 2003. A hybrid particle swarm optimization for distribution state estimation. IEEE T. Power Syst., 18(1): 60-68.

Robinson, J., S. Sinton and Y. Rahmat-Samii, 2002. Particle swarm, genetic algorithm and their hybrids: optimization of a profiled corrugated horn antenna. Antennas and Propagation Society International Symposium [C], 1: 314-317.

Zhang W.J. and X.F. Xie, 2003. DEPSO: hy brib particle Swarm with differential evolution operator. IEE International Conference on System, Man and Cybernetics, 4: 3816-3821.