Proceedings of the 6th World Congress on Intelligent Control
and Automation, June 21 - 23, 2006, Dalian, China

Natural Exponential Inertia Weight Strategy in Particle Swarm Optimization *

Guimin Chen[1], Xinbo Huang[2], Jianyuan Jia[1] and Zhengfeng Min[1]

1. *School of Electronical and Mechanical Engineering*
Xidian University
Xi'an, China 710071
efoxxx@126.com

2. *School of Electromechanical Engineering*
Xi'an University of Engineering Science & Technology
Xi'an, China 710071
hxb1998@126.com

***Abstract* - Inertia weight is one of the most important parameters of particle swarm optimization (PSO) algorithm. Based on the basic idea of decreasing inertia weight (DIW), two strategies of natural exponential functions were proposed. Four different benchmark functions were used to evaluate the effects of these strategies on the PSO performance. The results of the experiments show that these two new strategies converge faster than linear one during the early stage of the search process. For most continuous optimization problems, these two strategies perform better than the linear one.**

Index Terms - Particle Swarm Optimization, Inertia Weight.

I. Introduction

Motivated by the prey behavior of bird group, Kennedy and Eberhart introduced a new population-based, self-adaptive search optimization technique—particle swarm optimization (PSO) [1]. Since then the researchers have engaged themselves to improve upon the original PSO variation to increase accuracy of solution without sacrificing the speed of solution significantly.

In PSO algorithm, inertia weight is one of the most important parameter among the few adjustable parameters. Great values of inertia weight could improve global exploration while small ones could improve local exploitation. Shi and Eberhart [2] have found a significant improvement in the performance of the PSO method with a linearly decreasing inertia weight (DIW) over the iterations. During the early part of the search, greater weight is adopted to allow use of the full range of the search space, so as to determine the location of the optimal solution. On the other hand, during the latter part of the search, when the algorithm is converging to the optimal solution, smaller weight is adopted to find the global optima efficiently. Shi *et al* [2] [3] [4] also proposed fuzzy inertia weight (FIW) strategy and random inertia weight (RIW) strategy. FIW needs expert knowledge on fuzzy set rules while RIW is used to solve dynamic system problem.

Based on the basic idea of DIW, the authors proposed two decreasing strategies of natural exponential functions.

II. Basic Particle Swarm Optimization Algorithm and LDIW

In basic PSO, a population of particles is generated with random positions, and then random velocities are assigned to each particle. The fitness of each particle is then evaluated according to a user defined objective function. At each iteration, the trajectory in the search space is adjusted by dynamically altering the velocity of each particle, according to its own flying experience and the flying experience of the other particles in the search space.

In a *d*-dimension solution space, the position vector and the velocity vector of the *i*th particle can be represented as $X_i = (x_{i1}, x_{i2}, \dots, x_{id})$ and $V_i = (v_{i1}, v_{i2}, \dots, v_{id})$ respectively. According to a user defined fitness function, let us say the best position of each particle (which corresponds to the best fitness value obtained by that particle at the present iteration) is $P_i = (p_{i1}, p_{i2}, \dots, p_{id})$, while the fitness particle found so far is $P_g = (p_{g1}, p_{g2}, \dots, p_{gd})$. Then, the new velocities and the positions of the particles for the next fitness evaluation are calculated using the following two equations:

$$V_i = V_i + c_1 \times \mathrm{R}_1 \times (P_i - X_i) + c_2 \times \mathrm{R}_2 \times (P_g - X_i) \quad (1)$$

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

where c_1 and c_2 are constants known as acceleration coefficients, which are fixed at 2 generally, R_1 and R_2 are two random numbers generated uniformly in the range [0, 1], which are independent with each other.

The first part of (1) represents the previous velocity, the second part represents the personal thinking of each particle, and the third part represents the collaborative effect of the particles. It is accepted that proper control of global exploration and local exploitation is crucial in finding the optimum solution efficiently in population-based optimization methods. Therefore, Shi and Eberhart introduced the concept of inertia weight (w) to the basic PSO, in order to balance the local and global search during the optimization process. Then (1) is revised as follows:

$$V_i = w \times V_i + c_1 \times \mathrm{R}_1 \times (P_i - X_i) + c_2 \times \mathrm{R}_2 \times (P_g - X_i) \quad (3)$$

It is accepted that proper control of global exploration and local exploitation is crucial in finding the optimum solution efficiently in population-based optimization methods. We can choose an appropriate constant for w to compromise between global exploration and local exploitation. On the other hand, the value of w can be adjusted dynamically according to searching stage during iteration process. Shi and Eberhart proposed linearly decreasing inertia weight PSO (LPSO), in which the inertia weight w is decreased linearly over the

* This work was supported in part by the National 863 High Technique Projects under Grant No 2002AA862011.

iterations. The mathematical representation of w is as follows:

$$w_0 = (w_{\text{start}} - w_{\text{end}}) \times (\frac{T_{\max} - t}{T_{\max}}) + w_{\text{end}} \quad (4)$$

where w_{start} and w_{end} are the initial and final values of the inertia weight, respectively, $T_{\max}$ is the maximum number of allowable iterations, and t is the current iteration number. Through empirical studies, Shi and Eberhart have observed that the optimal solution can be improved by varying the value of inertia weight from 0.9 at the beginning of the search to 0.4 at the end of the search for most problems.

III. Natural Exponential Inertia Weight Strategy

In this paper, the authors proposed two natural exponential (base e) inertia weight strategies. The first strategy is expressed as:

$$w(t) = w_{\min} + (w_{\max} - w_{\min}) \cdot e^{-t/(\frac{\text{MAXITER}}{10})} \quad (5)$$

Hereafter, the PSO algorithm adopting this strategy is called e_1-PSO for short. The second one is expressed as:

$$w(t) = w_{\min} + (w_{\max} - w_{\min}) \cdot e^{-\left[t/(\frac{\text{MAXITER}}{4})\right]^2} \quad (6)$$

We call the PSO algorithm adopting this strategy e_2-PSO for short.

It is assumed that $w_{\text{start}} = 0.9$, $w_{\text{end}} = 0.4$, which is found to be very effective for LPSO. In addition, $T_{\max}$ is fixed at 3000. The variation curves of decreasing strategies function versus the number of iterations are shown in Fig. 1.

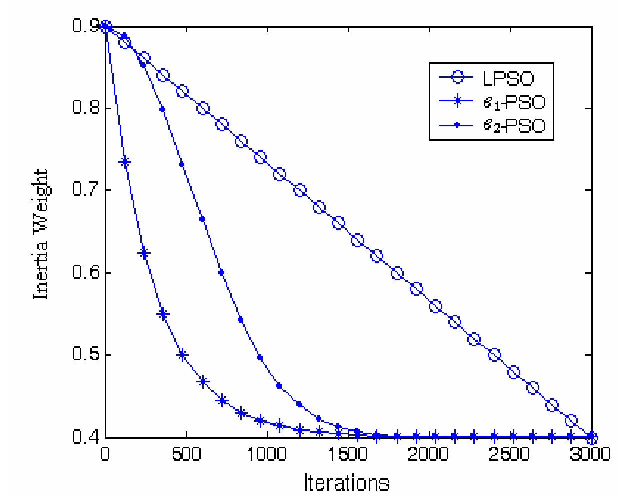

Fig.1. Comparison of the three decreasing strategies.

IV. Experimental Results

Four of the well-known benchmarks, given in TABLE I, were used to evaluate the performance in terms of the rate of convergence and the quality of the optimum solution of the three PSO versions. For each of the three PSO versions, fifty trials were carried out for each benchmark. In addition, the stopping criteria are set to 10^{-10}, 50, 0.02 and 50 for these benchmarks respectively. Other parameters in the algorithms are as follows: population size is 30 and acceleration coefficients $c_1 = c_2 = 2$.

TABLE I
Four Benchmarks for Experiments

Function name	Mathematical representation	Dimension(d)	Range of search	$V_{\max}$
Sphere	$f_1 = \sum_{i=1}^{n} x_i^2$	30	$(-100, 100)^d$	100
Rosenbrock	$f_2 = \sum_{i=1}^{n} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	30	$(-100, 100)^d$	100
Griewank	$f_3 = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$(-600, 600)^d$	600
Rastrigrin	$f_4 = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	$(-10, 10)^d$	10

A. *Sphere Function*

TABLE II
MEAN VALUE AND STANDARD DEVIATION OF THE OPTIMAL VALUE OF SPHERE FUNCTION

	LPSO	e_1-PSO	e_2-PSO
Mean	9.3397×10^{-11}	9.3505×10^{-11}	9.3093×10^{-11}
Standard deviation	7.0417×10^{-12}	5.0294×10^{-12}	6.2098×10^{-12}

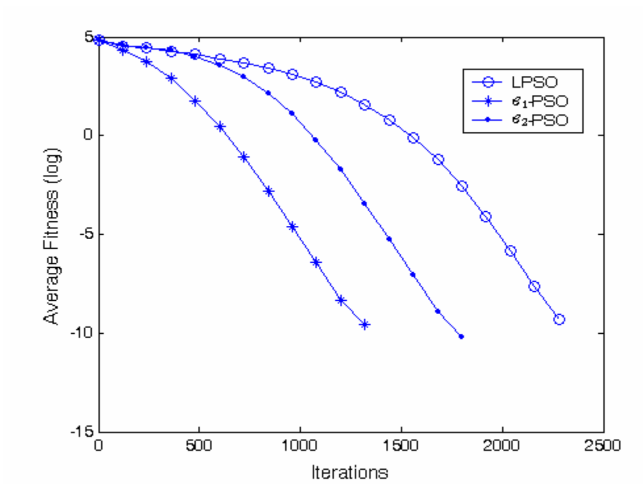

Fig.2. Variation of the mean best value of Sphere function versus the number of iterations

The average optimal value and the standard deviation for Sphere function are presented in TABLE II. Fig. 2 displays the variation of the average best solution over the iterations in the experiments. It is clear that the e_1-PSO method converges significantly faster than all of the other methods for Sphere function. It takes e_1-PSO 1300 iterations to reach the stopping criteria, while e_2-PSO 1760 iterations, and LPSO 2330 iterations.

B. *Ronsenbrock Function*

The average optimal value and the standard deviation for Ronsenbrock function are presented in TABLE III. Fig. 3 displays the variation of the average best solution over the iterations in the experiments. It is clear that e_1-PSO has the best searching capability over the given 3000 iterations; e_2-PSO takes the second place, and LPSO the third place. We can also see that e_1-PSO and e_1-PSO converge very fast during the early stage.

TABLE III
MEAN VALUE AND STANDARD DEVIATION OF THE OPTIMAL VALUE OF RONSENBROCK FUNCTION

	LPSO	e_1-PSO	e_2-PSO
Mean	73. 3659	57.3858	70.8313
Standard deviation	121.2056	86.4313	115.9860
Trials reaching stopping criteria	21	45	32

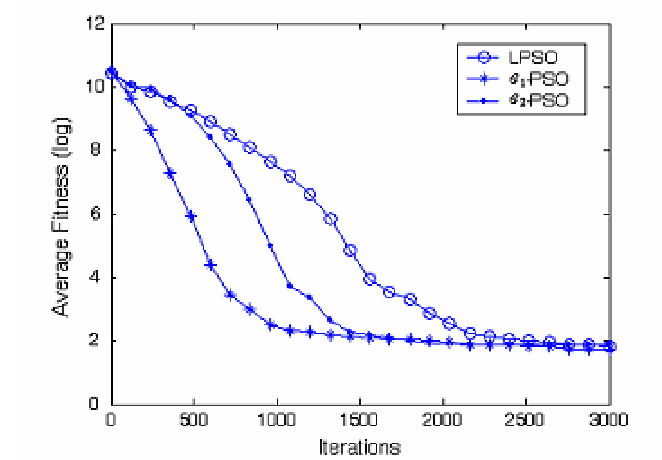

Fig.3. Variation of the mean best value of Rosenbrock function versus the number of iterations

C. *Griewank Function*

The average optimal value and the standard deviation for Griewank function are presented in Table IV. Fig. 4 displays the variation of the average best solution over the iterations in the experiments. Again, e_1-PSO and e_2-PSO show very fast convergence during the early stage of the search. However, e_1-PSO falls behind the other methods as the number of iterations increase. And finally, e_2-PSO has the best searching capability over the given 3000 iterations; LPSO takes the second place, and e_2-PSO the third place.

TABLE IV
MEAN VALUE AND STANDARD DEVIATION OF THE OPTIMAL VALUE OF GRIEWANK FUNCTION

	LPSO	e_1-PSO	e_2-PSO
Mean	0.0141	0.0186	0.0101
Standard deviation	0.0155	0.0183	0.0126
Trials reaching stopping criteria	34	31	39

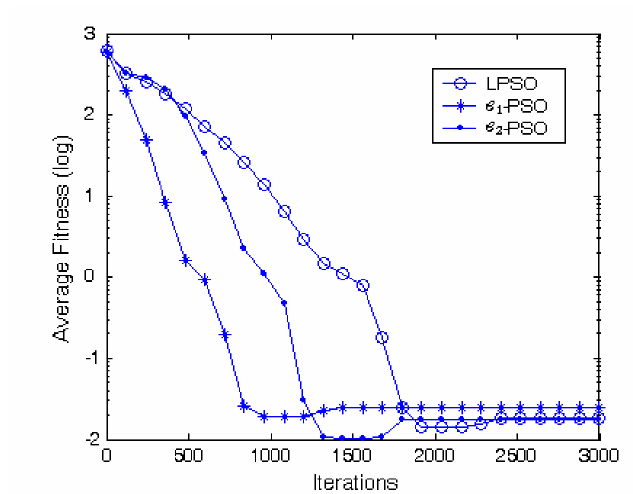

Fig.4. Variation of the mean best value of Griewank function versus the number of iterations

D. Rastrigrin Function

Fig. 5 displays the variation of the average best solution over the iterations in the experiments. The average optimal value and the standard deviation for Rastrigrin function are presented in TABLE V. Although e_1-PSO and e_2-PSO converge fast during the early stage of the search, they slightly fall behind LPSO after about 2000 iterations. Further, the final solutions found by e_1-PSO and e_2-PSO after 3000 iterations are almost the same.

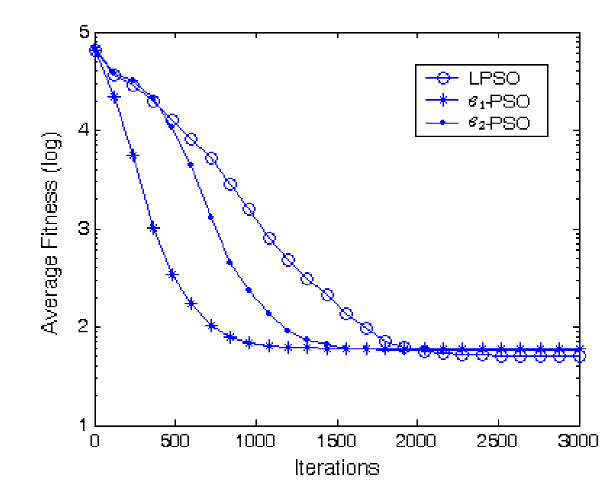

Fig.4. Variation of the mean best value of Rastrigrin function versus the number of iterations

TABLE V
MEAN VALUE AND STANDARD DEVIATION OF THE OPTIMAL VALUE OF RASTRIGRIN FUNCTION

	LPSO	e_1-PSO	e_2-PSO
Mean	49.2375	52.5772	53.3436
Standard deviation	12.7724	11.1903	15.9680
Trials reaching stopping criteria	23	19	17

IV. CONCLUSIONS

Based on the basic idea of decreasing inertia weight, the authors proposed two natural exponential inertia weight strategies. The results of the experiments show that these two new strategies converge faster than linear one during the early stage of the search process. For most continuous optimization problems, these two strategies perform better than the linear one.

REFERENCES

[1] Kennedy J. and Eberhart R.. Particle swarm optimization. Proceeding of IEEE International Conference on Neural Networks, 1995, 1942-1948.

[2] Shi Y. and Eberhart R.. Empirical study of particle swarm optimization. Proceeding of IEEE International Conference on Evolutionary Computation, 1999, 1945-1950.

[3] Shi Y. and Eberhart R.. Fuzzy adaptive particle swarm optimization. Proceeding of Congress on Evolutionary Computation, 2001, 101-106.

[4] Eberhart R. and Shi Y.. Tracking and optimizing dynamic systems with particle swarms. Proceeding of Congress on Evolutionary Computation, 2001, 94-100.