International Journal of
Soft Computing And Software Engineering (JSCSE)
ISSN: 2251-7545
Vol.2,No.2, 2012
DOI: 10.7321/jscse.v2.n2.1

Empirical Study of the Inertia Weight Particle Swarm Optimization with Constraint Factor

*1YAN Chun-man, 2GUO Bao-long, 3WU Xian-xiang
1,2,3 *Institute of Intelligent Control and Image Engineering, Xidian University, China,*
yancha02@163.com, blguo@xidian.edu.cn, wuxianxiang@163.com
*1 *College of Physics and Electronic Engineering, Northwest Normal University, China,*
*1 *yancm@mail.xidian.edu.cn*

Abstract. **For improving the performance of the Particle Swarm Optimization (PSO), two major strategies are used, one is the parameter modifying method, and the other is the population diversity method. For these two methods, the first one obtains the balance between the local search ability and the global search ability of the PSO by using the parameter adjusting and the parameter adding or parameter reducing, in that it has less effect on the algorithm complexity and has attracted a great of attentions. One of the well-known improved PSO algorithms of the parameter modifying method is inertia weight PSO, by introducing the inertia weight, the performance of the original PSO is improved greatly. Experimentally, we find that the performance of the algorithm can be improved more when adding a constraint factor to the inertia weight. In this paper, we empirically study the effects of the constraint factor on the performance of the inertia weight PSO. Based on the experimental results, we obtain the optimal selection of the constraint factor and extend the ability of the inertia weight PSO.**

Keywords: *Swarm Intelligence, Particle Swarm Optimization (PSO), Inertia weight*

1. Introduction

The Particle Swarm Optimization (PSO) is an agent based search algorithm, which is first proposed by Kennedy et al [1]. The algorithm is motivated by the group organism behavior such as bee swarm, fish school, and the bird flock. The basic principle of PSO is to find the optimal solution through the cooperation and competition among particles. For the algorithm, the solution of the problem is treated as a volume-less particle with a special velocity in the search space. Through sharing the information with the individuals and companions, the particle dynamically adjusts its velocity and position and flies to an optimal orientation, and then makes the population evolve to an optimal solution through an iterative process.

Compared with evolutionary computation, the PSO is a more efficient parallel search algorithm. Because of the quick convergence speed and the fewer parameter settings, in the recent decade, the PSO has achieved a quick development, and a variety of proposals aiming at improving the performance of it had also been presented in many literatures. In generally, there are two main strategies for the improvements: (1) the parameter modifying methods, which revising the parameters of the original PSO to obtain the balance between the local and global search ability while making less effect on the algorithm complexity, these algorithms include the inertia weight PSO [2, 3], the constriction PSO [4], the self-organization PSO [5], etc.; (2) population diversity methods, for avoiding the premature convergence caused by the loss of the population diversity, these methods borrow the ideas from the natural selection and the social behavior to improve the performance of the algorithm, the family of them include different topology PSO [6,7,8], social division PSO [9], cooperative PSO [10,11], ecological selection PSO [12], etc.

A more efficient optimization algorithm must obtain a better balance between the local and global search ability, which means that the algorithm must has the ability to maintain a better local exploitation

International Journal of
Soft Computing And Software Engineering (JSCSE)
ISSN: 2251-7545
Vol.2,No.2, 2012
DOI: 10.7321/jscse.v2.n2.1

and global exploration ability. Exploration means that the particle departs from a solution and is able to find other promising candidates; while exploitation means that the particle is capable of going along the old path to obtain more optimal candidates. According to the "No free lunch theory" of optimization, the performance improvement of the algorithm is always not "free". The performance improvement of the algorithm must rely on much more additional computation and information, which may lead to the increase of the algorithm complexity. In other words, the increase of the locale search ability always means to the decrease of the global search ability, and vice versa. Compared with the population diversity methods, the parameter modifying methods develop the performance by revising the parameters of the original PSO, in that it has less effect on the algorithm complexity and has attracted a great of attentions.

In [2], Shi and Eberhart introduced the inertia weight to the velocity update equation of the original PSO. The present of the inertia weight increases the convergence speed greatly, and obtains a better balance between exploitation and exploration of the solution space, while having little increase of the algorithm complexity. Therefore, it is valuable to make deeper study on it. We find that, when adding a suitable constraint factor to the inertia weight, the algorithm appears to be more efficient. To validate the situation, in this paper, we empirically study the phenomena. Four different benchmark functions are selected as testing functions, and the convergence speed, the search accuracy, and the proportion of optimization successes are compared under different constraint factor values. Through the experimental results, we obtain the selection evidence for an optimal constraint factor, which is able to make the algorithm do better than the original inertia weight PSO.

2. PSO algorithm and its improvement

The PSO algorithm can be described as follows: in the D-dimensional search space, there are n particles forming a population $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_D\}$, each particle has a D-dimensional position vector $\boldsymbol{x}_i$ and a D-dimensional velocity vector $\boldsymbol{v}_i$. The vector $\boldsymbol{x}_i$ represents a promising solution of the optimization problem, and the velocity affects the convergence speed of the algorithm.

The velocity vector and the position vector of the i-th particle can be described as $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \ldots v_{iD})^T$ and $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{iD})^T$, respectively. Through the search procedure in the D-dimensional space, the particle "remembers" its own local best position $\boldsymbol{p}_i = (p_{i1}, p_{i2}, \ldots p_{iD})^T$. By comparing among the companions' information, the particle gets the global best position $\boldsymbol{p}_g = (p_{g1}, p_{g2}, \ldots p_{gD})^T$. At each generation, each particle updates the velocity by "following up" two experience values ($\boldsymbol{p}_i$ and $\boldsymbol{p}_g$) and its own velocity inertia, then updates its position, and "flies" to a promising position. Because of the convergence speed, Shi et al. [2] modified the velocity update equation by introducing the inertia weight w to the original PSO, and the improved PSO are manipulated according the following equation:

$$\boldsymbol{v}_i(t+1) = \boldsymbol{v}_i(t) + c_1 r_{1i} \left(\boldsymbol{p}_i(t) - \boldsymbol{x}_i(t)\right) + c_2 r_{2i} \left(\boldsymbol{p}_g(t) - \boldsymbol{x}_i(t)\right) \tag{1}$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{2}$$

where t represents the index of iteration, i indexes a particle, $i = 1, 2, \ldots, m$, and m represents the population size. $r_{1i}(\cdot)$ and $r_{2i}(\cdot)$ are two random functions in range $[0, 1]$. In addition, c_1 and c_2 are two positive constants, which can be looked as the personal cognitive factor and the social cognitive factor, respectively. This improved PSO is called the Standard Particle Swarm Optimization (SPSO), and notated as Inertia Weight PSO (IWPSO) in our paper.

By introducing the inertia weight, the performance of the original PSO has been significantly improved. By linearly decreasing the inertia weight from a relatively large value to a small value through the course of the PSO run [3], the PSO tends to have more global search ability at the beginning of the run while having more local search ability near the end of the run. The inertia weight update equation is described as Eq. (3):

$$w = w_{start} - \left(w_{start} - w_{end}\right)\frac{t}{T_{\max}} \tag{3}$$

where t represents the index of iteration, w_{start} and w_{end} are the beginning value and the end value of the inertia weight, respectively, and $T_{\max}$ is the maximum iteration times.

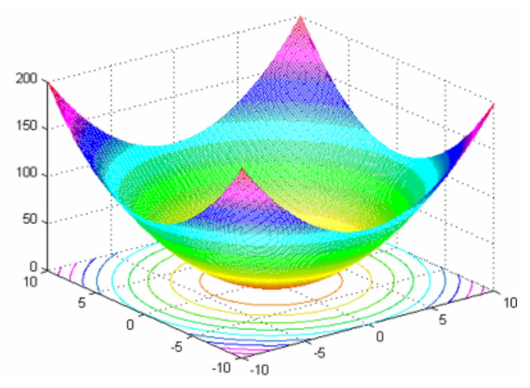
Through deeper experiments, we find that, when adding a suitable constraint factor f to the inertia weight updating equation such that:

$$w = w_{start} - \frac{1}{f}\cdot\left(w_{start} - w_{end}\right)\frac{t}{T_{\max}} \tag{4}$$

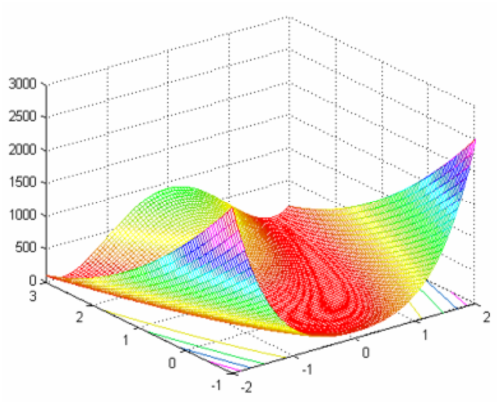
By Eq. (4), the algorithm appears to be more efficient. In the follows of this paper, we will empirically study the effects of the constraint factor f on the performance of the IWPSO.

3. Experimental settings

For comparison, four different benchmark functions, Sphere, Rosenbrock, Griewank and Rastrigrin, are used as testing functions. Their graphs are shown in Fig. 1, where (a) and (b) are the graph of function Sphere and Rosenbrock respectively, while (c) and (d) are the local graph of function Griewank and Rastrigrin respectively. For all these functions, the global optimal value is $f(x^*) = 0$. Among them, Sphere and Rosenbrock are single modal functions, they only have global optimal value in the search space and are used for testing the global convergence performance of an algorithm. While Griewank and Rastrigrin are multi-modal functions, they have a number of local optimal values, which are hard for optimization problem, and are used for testing the local exploitation ability and the global exploration ability. Table 1 shows the parameter settings for these testing functions, including dimension, search range, and the global error.



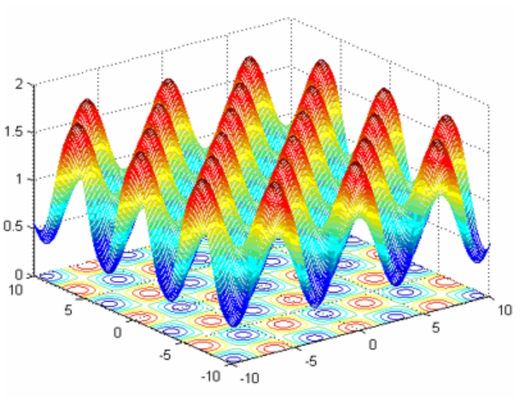
(a)



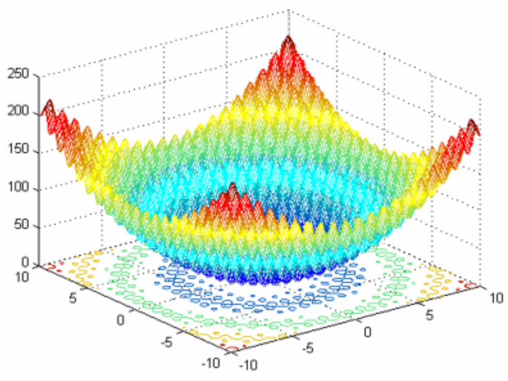
(b)

International Journal of
Soft Computing And Software Engineering (JSCSE)
ISSN: 2251-7545
Vol.2,No.2, 2012

DOI: 10.7321/jscse.v2.n2.1



(c)



(d)

Figure 1. Graph of four benchmark functions (D=2)

Following the suggestion in [3] and for the purpose of comparison, the method used in [11] is adopted here for population initialization, and the other parameters needed in the experiments are set as follows: the maximum number of generations is set as 1000; the dimension of each particle is 30; the population size is 40; the linearly decreasing inertia weight is used with $w_{start} = 0.9$ and $w_{end} = 0.4$; both c_1 and c_2 are set as 2.05; the maximum velocity $\boldsymbol{v}_{\max}$ and the maximum position $\boldsymbol{x}_{\max}$ are set to be equal, their value for each function are listed in Table 2.

Table 1. Parameter settings for the benchmark function

Function	Function Form	Dim.	Search Space	Global Error
Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100,100]^{30}$	10^{-2}
Rosenbrock	$f_2(x) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	30	$[-30,30]^{30}$	100
Griewank	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600,600]^{30}$	0.05
Rastrigrin	$f_4(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$	30	$[-5.12,5.12]^{30}$	100

Table 2. $\boldsymbol{v}_{\max}$ and $\boldsymbol{x}_{\max}$ settings for each testing function

Function	$\boldsymbol{v}_{\max} = \boldsymbol{x}_{\max}$
f_1	100
f_2	100
f_3	10
f_4	600

4. Experimental results and analysis

In this section, the effects of the constraint factor f on the performance of the inertia weight PSO (IWPSO) is investigated through empirical study, different values of the constraint factor are selected and three measures indices, convergence speed, search accuracy, and proportion of successes, are used to evaluate the optimization.

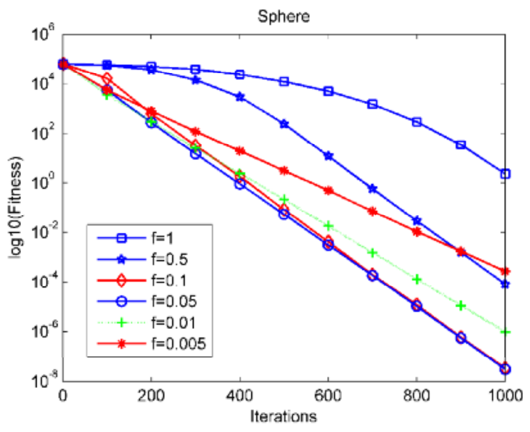
The convergence speed can be reflected by the iteration times to obtain optima. An algorithm would be thought to be not convergent if it dose not obtain the optima after a fixed number of search runs. The convergence speed can also be depicted by the convergence curve obviously, and a large gradient of the curve means a quick convergence speed. Fig. 2 depicts the average convergence curves with different constraints factor f for the four benchmark functions, in which, (a), (b), (c), and (d) are the curves for function Sphere, Rosenbrock, Rastrigrin and Griewank, respectively. Where the horizontal coordinate is the number of the iterations, and the vertical coordinate is the logarithm of the function values (fitness function values).

As shown in Fig. 2, for all testing functions, when $f = 1$, the convergence speed is the lowest, and when $f = 0.05$, the convergence speed is the fastest. While f decreasing from 1 to 0.05, the convergence speed increases, further decrease in f value beyond 0.05, the convergence speed increases not any more.

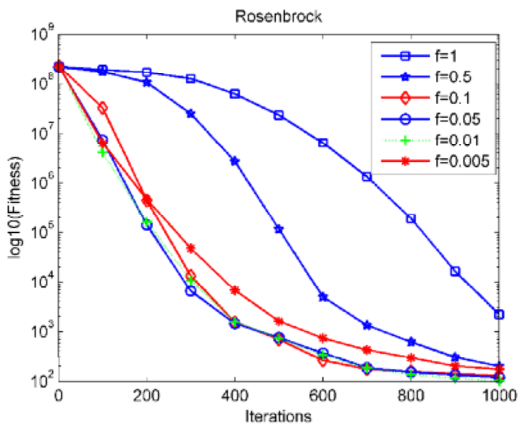
Table 3 lists the mean best fitness, which reflects the search accuracy under different constraint factor f. Search accuracy reflects the best search result attained after running a fixed number of function evaluations, and can also be represented by the average optimization error. For a testing function $f(x)$, describe the optimization error as Eq. (5):

$$E = \left| f(\boldsymbol{x}) - f(\boldsymbol{x}^*) \right|_{\min} \tag{5}$$

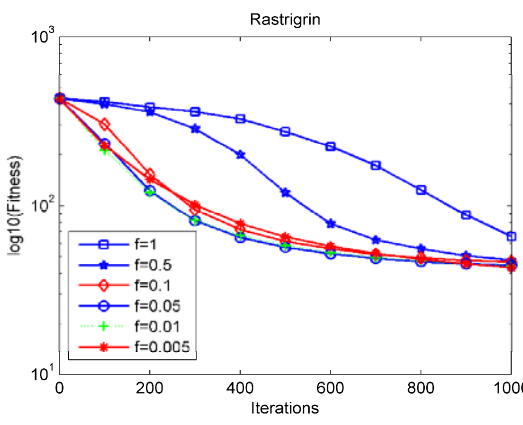
where $f(x^*)$ is the given optimal value of the function. As described in Section 3, for all benchmark functions, $f(x^*) = 0$, then the optimization error E is equal to the fitness function value, thus the search accuracy can be represented by the average global optima of the total search runs.



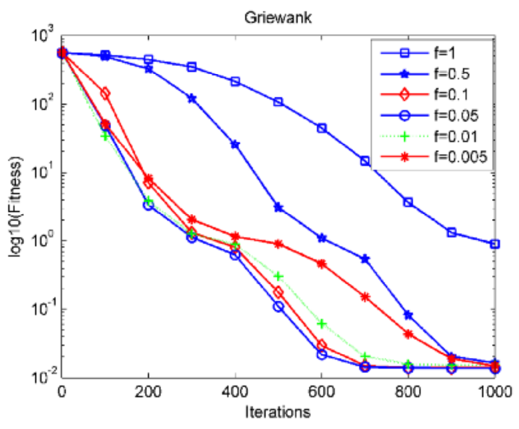
(a)



(b)



(c)



(d)

Figure 2. Convergence curves under different constraint factor

Table 3. Mean best fitness under different *f*

Function	*f=1*	*f=0.5*	*f=0.1*	*f=0.05*	*f=0.01*	*f=0.005*
Sphere	2.1838	0.0001	0.0000	0.0000	0.0000	0.0003
Rosenbrock	2386.5743	199.4966	129.0602	119.1987	102.5629	173.839
Griewank	67.1830	47.5886	46.5106	44.4015	43.9865	43.0154
Rastrigin	0.8765	0.01634	0.0139	0.0136	0.0151	0.01480

As seen in Table 3, for function Rastrigin, when $f = 0.05$, the best fitness can be obtained. For Sphere and Rosenbrock, when $f = 0.01$, the best fitness comes out, and for Griewank, when $f = 0.005$, the best fitness is obtained. Among all these testing functions, function Sphere is possessed of better search accuracy.

The proportion of optimization successes over 200 runs are listed in Table 4, which shows the percentage of times that the algorithm was able to reach the globally optimal region. For the optimization, while the fitness is less equal the global error, it is looked as optimization success. This measure reveals the expected probability of the algorithm reaching the criteria after a certain runs. If the probability is very high, it means that the algorithm is consistently capable of discovering the solution.

Table 4. Proportion of optimization successes (%)

Function	*f=1*	*f=0.5*	*f=0.1*	*f=0.05*	*f=0.01*	*f=0.005*
Sphere	0	100	100	100	100	100
Rosenbrock	0	51.5	70	76	68.5	52
Griewank	95	100	100	100	100	100
Rastrigin	0	93.5	93	95.5	95.5	98

As seen in Table 4, for all testing function, their proportion of optimization successes are the lowest when $f = 1$. For function Sphere, Rosenbrock, and Griewank, the better proportion of optimization successes can be obtained when $f = 0.05$, and for function Rastrigin, when $f = 0.005$, the proportion of optimization successes is the highest. We can also find from Table 4, for function Sphere, Rosenbrock, and Rastrigin, after 200 runs, the optimizations are not convergent when $f = 1$.

From the experimental results, it can be seen that the constraint factor f formulated in Eq. (4) has a great effects on the performance of the IWPSO, thinking about the measurement indices, $f = 0.05$ is a preferred one.

5. Conclusion and discussion

Through the comparative experiments, parameter modifying methods to improve the performance of the PSO is demonstrated the better balance between the global search ability and the local search ability, while making less effect on the algorithm complexity. One of the well-known parameter based improving PSO is the inertia weight PSO (IWPSO), with inertia weight linearly decreasing strategy, the performance of the original PSO is improved obviously. In this paper, we empirically studied the IWPSO with a constraint factor, and the experimental results shown that the factor can obviously affect the performance of the original IWPSO. Under a certain conditions (40 populations, 1000 generations $w_{start} = 0.9$, $w_{end} = 0.4$, and 200 runs), and synthetically thinking about the measurement indices, constraint factor $f = 0.05$ is a preferred selection, which can provide better convergence speed, search accuracy, and proportion of optimization successes for the benchmark testing functions.

Although the performance of the improved IWPSO by the constraint factor *f* is superior to the original IWPSO, it still can not solve the problems of the premature convergence and falling into a local optimum value. For open research, under the enlightenment of the population diversity methods, a variety group structure PSO can be constructed by the improved IWPSO to overcome the above problems, and in the future works, these improved PSO can be applied for further applications based on PSO algorithm, such as image retrieval, mobile robot path planning, etc.

6. Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grants No. 61003196 & No. 61105066 and the Fundamental Research Funds for the Central Universities under Grants K50510040007 & K50510040004.

The authors would like to thank all editors and reviewers for their constructive comments and suggestions. Moreover, they would also want to thank Dr. ZHANG Xu and coworkers for their helps for the preparations of our manuscript.

References

[1] J. Kennedy, R. C. Eberhart, "Particle Swarm Optimization", In Proceedings of IEEE International Conference on Neural Networks, pp. 1942-1948, 1995.
[2] Y. Shi, R. C. Eberhart, "A modified particle swarm Optimizer", In Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway, pp. 69-73, 1998.
[3] R. C. Eberhart. Y. Shi, "Comparing inertia weights and constriction factors in Particle Swarm Optimization", In Proceedings of the Congress on Evolutionary Computation, pp.84-88, 2000.
[4] M. Clerc, J. Kennedy, "The particle swarm-explosion, stability and convergence in a multidimensional complex space ", IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp.58-73, 2002.
[5] A. Ratnaweera, S. K. Halgamuge, H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients ", IEEE Transactions on Evolutionary Computation, vol. 3, no. 8, pp.240-255, 2004.
[6] J. Kennedy. "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance", In Proceedings of IEEE Congress on Evolutionary Computation Piscataway, pp.1931-1938, 1999.
[7] J. Kennedy, R. Mendes, "Population Structure and Particle Swarm Performance", In Proceedings of IEEE Congress on Computational Intelligence, pp.1671-1676, 2002.
[8] R. Mendes, J. Kennedy, J. Neves, "The Fully Informed Particle Swarm: Simpler, Maybe Better ", IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp.204-210, 2004.
[9] J. S. Vesterstrom, J. Riget, T. Krink, "Division of labor in particle swarm optimization", In Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, pp.1570-1575.

[10] F. van den Bergh, A. P. Engelbrecht, "A Cooperative approach to particle swarm optimization", IEEE Transactions on Evolutionary Computation, vol. 8 , no. 3, pp.225-239, 2004 .
[11] WU Xian-xiang, GUO Bao-long, WANG Juan, "Lotka-Volterra model based particle swarm optimization", Control and Decision, vol. 25, no. 11, pp.1619-1624, 2010.
[12] Y. Y. Yan, B. L. Guo, "Particle Swarm Optimization Inspired by r- and K-Selection in Ecology", In Proceedings of IEEE Congress on Evolutionary Computation, pp.1117-1123, 2008.