Applied Mathematics & Information Sciences
An International Journal

A self-guided Particle Swarm Optimization with Independent Dynamic Inertia Weights Setting on Each Particle

Huantong Geng[1,2], *Yanhong Huang*[1,2], *Jun Gao*[1,2] *and Haifeng Zhu*[1,2]

[1]School of Computer and Software, Nanjing University of Information Science and Technology, 210044 Nanjing, China
[2]Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, 210044 Nanjing, China

Received: 15 Aug. 2012; Revised 5 Sep. 2012; Accepted 16 Sep. 2012
Published online: 1 Mar. 2013

Abstract: In the standard PSO algorithm, each particle in swarm has the same inertia weight settings and its values decrease from generation to generation, which can induce the decreasing of population diversity. As a result, it may fall into the local optimum. Besides, the decreasing of weights values is restricted by the maximum evolutionary generation, which has an influence on the convergence speed and search performance. In order to prevent the algorithm from falling into the local optimum early, reduce the influence of the maximum evolutional generation to the decline rate of weights, A Self-guided Particle Swarm Optimization Algorithm with Independent Dynamic Inertia Weights Setting on Each Particle is proposed in the paper. It combines the changes of the evolution speed of each particle with the status information of current swarm. Its core idea is to set the inertia weight and accelerator learning factor dynamically and self-guided by considering the deviation between the objective value of each particle and that of the best particle in swarm and the difference of the objective value of each particle's best position in the two continuous generations. Our method can obtain a balance between the diversity and convergence speed, preventing the premature as well as improving the speed and accurateness. Finally,30independent experiments are made to demonstrate the performance of our method compared with the standard PSO algorithm based on 9 standard testing benchmark functions. The results show that convergence accurateness of our method is improved by 30%compared with the standard PSO, and there are 4 functions obtaining the optimal value. And convergence accurateness is improved by more than 20%for 5 functions at the same evolution generation.

Keywords: PSO, Linear inertia weight, SgDPSO, Self-guided, Dynamical Inertia Weight

1. Introduction

Particle swarm optimization (PSO) [1] is a population-based, self-adaptive search optimization technique, firstly introduced by Kennedy and Eberhart in 1995. The motivation for the development of this method was based on the simulation of simplified animal social behaviors such as fish schooling, bird flocking, etc. PSO algorithm has been widely used in various fields [2, 3] due to its few parameters to adjust, easy to understand, easy to implement, and computationally efficient. Despite of these advantages, PSO is easy to be trapped into local optima, and the convergence rate decreased considerably in the later period of evolution processing. To overcome these shortcomings, many researchers have proposed various modifications to the original PSO, including improved parameters [4, 5, 6], topology [7, 8] and hybrid algorithms [9, 10, 11]. Initially, the introduction of the linear inertia weight made a balance effectively to some extent between the global and local search, and formed the standard PSO algorithm. After that, a variety of inertia weight algorithms [15] were proposed such as random inertia weight algorithm, constriction factor inertia weight algorithm, etc. Those algorithms have their own advantages and disadvantages.

Compared with other evolutionary algorithms, standard PSO algorithm exist the following problems. Firstly, although standard PSO algorithm can converge faster, each

* Corresponding author: e-mail:htgeng@nuist.edu.cn

particle becomes more and more similar with the increasement of evolutionary generation, which can't make the optimization sustained but hover near the local optimal solution. Secondly, it is not proper that each particle uses the same inertia weight and the linear weights just depend on maximum generation not related to characteristic of problems, which may affect the balance between the global and local search ultimately. Finally, most early developments in PSO have been proven to be effective in static optimizing problems [12, 13, 14], which is not common. Based on the above problems, a Self-guided Particle Swarm Optimization with Independent Dynamic Inertia Weights Setting on Each Particle is proposed in the paper (SgDPSO), which considers the current situation and historical information of each particle, and the inertia weight of each particle is dynamic, self-guided setting to avoid the effectiveness of maximum generation to inertia weight.

The rest of this paper is organized as follows. In Sec. 2, a brief review of the updating process of the standard PSO algorithm is given, and some problems about the linear inertia weight method are analyzed. In Sec. 3, a novel method is proposed to set the inertia weight and accelerator learning factor of each particle dynamically and self-guided in accordance with problems in Sec. 2. Sec. 4 describes the procedure of SgDPSO algorithm. In Sec. 5, experimental settings for the benchmarks are explained and the results in comparison with the two algorithms are presented. Finally, conclusions are made in Sec. 6.

2. Problems of Setting Inertia Weight in Standard PSO

To improve the convergence performance of particle swarm optimization, Shi and Eberhart [3] published a paper in IEEE International Conference on Evolutionary Computation in 1998, which introduced the concept of inertia weight and used the linear inertia weight setting method. Their work improves the performance of the traditional PSO effectively, and forms the current standard particle swarm optimization algorithm.

In the standard PSO algorithm, the trajectory of each individual in the search space is adjusted according to (1) and (2) by dynamically altering the velocity of each particle based on its own flying experience and that of other particles in the search space.

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

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

Where x_{id} and v_{id} represent the position and the velocity of the ith particle $(i = 1, 2, \cdots, N)$in the d-dimension$(d = 1, 2, \cdots, D)$ search space,respectively.t is the current generation, r_1 and r_2 are two separately generated uniformly distributed random numbers in the range [0, 1], c_1 and c_2 are constants known as acceleration coefficients, p_{id} is the best position of the ith particle in the d-dimension search space, p_{gd} is the fittest particle found so far at generation t,w is the inertia weight, which determines the influence of the previous particle to the current particle velocity. Commonly,w is given by (3)

$$w = w_{max} - (w_{max} - w_{min}) \times \frac{t}{iter_{max}}. \quad (3)$$

where w_{max} and w_{min} are the initial and final values of the inertia weight, respectively, and $iter_{max}$is the maximum number of allowable generations.

In the standard PSO algorithm, the method of setting w is static effectively, which can't be adaptive to different problems and particles. Firstly, each particle in swarm has the same inertia weight settings and its value decreases from generation to generation, which can induce the decreasing of population diversity. As a result, it may fall into the local optimum, resulting in premature convergence. Secondly, a large w can make the neighborhood of particle skip the best optimal solution when it is in the vicinity of the particle during the early part of the search, thereby reducing search performance of the best optimal solution. Finally, we can see from (3), it is useful for particle to explore new search area with a large inertia weight, which has been few influenced by the best solution of particle and the best solution of swarm found so far. But, it is not helpful to the convergence of global optimal. If w with a small value, the result is opposite. Therefore, w has a great impact on the exploration and convergence of the algorithm. The value of w decreases slowly with the large number of generations while decreases rapidly with the small generations, and there is no doubt about the effectiveness on the exploration and convergence of the algorithm. To take into account the above problems, a method to dynamically set the inertia weight is introduced based on the history and current information of each particle.

3. A Method of Setting Inertia Weight Dynamically Based on Current and Historical Information

3.1. Motivation of "Historical + Current" Information

Two important messages are usually integrated when people make a decision, which is known by the Boy and Recharson [18] on the exploration of human decision-making process. One is the experience from themselves and their neighbors, which is called historical information. The other is the current situation. Inspired by this idea, we use it to set the inertia weight of each particle to guide them search in the search space.

3.2. Method of Setting Inertia Weight Dynamically

Inertia weight plays an important role on the balance between local optimum and global optimum. The appropriate inertia weight settings can improve the performance of PSO algorithm effectively. To solve the problems found in Sec. 2, we believe that the appropriate inertia weight settings should consider the following aspects at the same time. First, inertia weight settings should vary with the different problems. Second, inertia weight settings should vary with the environment changed. Finally, inertia weight settings should vary with the particle.

Based on the above considerations, a method is introduced that sets the inertia weight and accelerator learning factor dynamically and self-guided of each particle of swarm. The particles can be classified into three catalogues by observing their changing during the evolution: one is near the optimal solution, one is far from the optimal solution, and other is between the above two. We define that the particle is more far away from the best solution when its fitness value is larger than the fitness value of the best solution, so the increasing of the velocity of particle can enlarge its search range space. Otherwise, the decreasing of the velocity of the particle can make it search near the best solution. The particles between the above two are used to balance the global search and local search. Second, the inertia weight of particle should also be concerned with its history information, in this paper the evolutionary rate of particle are used to reflect its changing of historical information. We define that the velocity of particle should be increased if its evolutionary rate is slow. Otherwise its velocity is decreased. Based on the comments mentioned above and lots of experiments, the following setting methods are used in this paper. The inertia weight of each particle and evolutionary rate is calculated by (4) and (5), respectively.

$$w_i(t+1) = 1 - \frac{1}{\sqrt{(f_{p_g}(t) - f_i(t))^2 + hv_i(t)^2 + 1}}. \quad (4)$$

$$hv_i(t) = \frac{1}{\sqrt{(f_{p_i}(t-1) - f_{p_i}(t-2))^2 + 1}}. \quad (5)$$

Where w_i is the inertia weight of ith particle, $f_{p_i}$is the fitness value of the best solution of ith particle, which reflects the history information of ith particle.hv_i is the evolutionary rate of ith particle. f_i is the fitness value of ith particle, which reflects the current information, f_{p_g} is the fitness value of the best solution of swarm found so far.

The next generation inertia weight of each particle is dynamically decided by the difference between its current fitness value and the best fitness value of swarm and the revolution rate. So the inertia weight of each particle can vary with the generation, and the inertia weight of each particle is also different at the same generation. The inertia weight of each particle is different for various issues because different fitness functions are used.

3.3. Setting of other Parameters

Similar consideration, we define that the particle is more far away from the best solution when its fitness value is larger than the fitness value of the best solution, so increase the cognitive capacity of the particle. Otherwise, increase the social capacity of the particle. The particles between the above two are used to balance the cognitive and the social component of the particle. Based on the above considerations, according to the literature [16], after a number of experiments, the acceleration learning factor by (6) and (7), the velocity by (8) and the position for the next function evaluation is updated by (9).

$$c1_i = \frac{2}{3}\pi - \arcsin\left(\frac{1}{\sqrt{(f_{p_g}(t) - f_i(t))^2 + 1}}\right) \quad (6)$$

$$c2_i = \frac{1}{6}\pi - \arcsin\left(\frac{1}{\sqrt{(f_{p_g}(t) - f_i(t))^2 + 1}}\right) \quad (7)$$

$$\begin{aligned} v_{id}(t+1) &= w_i \times v_{id}(t) + c1_i \times (p_{id}(t) - x_{id}(t)) \\ &\quad + c2_i \times (p_g(t) - x_{id}(t)) \end{aligned} \quad (8)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (9)$$

Where $c1_i$ and $c2_i$ are accelerate learning factor of ith particle, respectively. The random number r_1 and r_2 are removed to reduce excessive random noise for each particle is set both $c1_i$ and $c2_i$.

4. Procedure of SgDPSO Algorithm

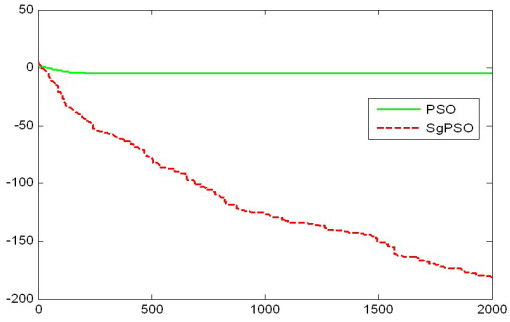
An approach is described in the above section, which is used to set the inertia weight and accelerator learning factor of each particle. On the basis of that the SgDPSO algorithm is proposed, and its main procedure can be stated as follows:

Step1: *To generate the initial population. The population is generated randomly in the search space, population size is P_{size} and the dimension of problem is D.*
Step2: *To evaluate the population. Here, the objective function value is considered as fitness.*
Step3: *To update p_{best} and p_g. If the best solution is found, the algorithm terminates, or goes to Step4.*
Step4: *To update inertia weight and acceleration learning factors of particles. The evolutionary rate of each particle is calculated by (5), the inertia weight is calculated by (4), and the acceleration learning factors is calculated by (6) and (7).*
Step5: *To update the velocity and position of particles. The velocity and position of each particle is updated by (8) and (9), and then go to Step2.*

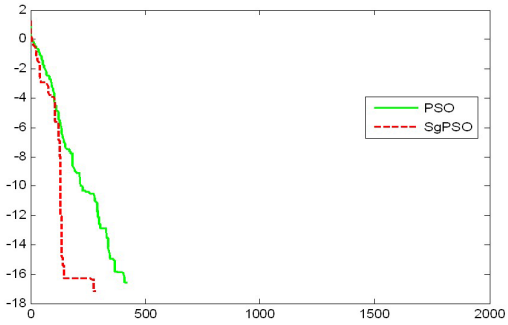
Table 1 Nine benchmarks used in SgDPSO algorithm in this paper

Function Names	Mathematical representation	Range of search	Global optimal fmin	Global optimal position
Sphere	$f_1(x)=\sum_{i=1}^{D} x_i^2$	$[-100,100]^D$	0	0.0^D
Griewank	$f_2(x)=\frac{1}{4000}\sum_{i=1}^{D} x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]^D$	0	0.0^D
Rastrigin	$f_3(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	$[-5.12,5.12]^D$	0	0.0^D
Rosenbrock	$f_4(x)=\sum_{i=1}^{D-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	$[-30,30]^D$	0	1.0^D
Ackley	$f_5(x)=-20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}}-e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}+20+e$	$[-32,32]^D$	0	0.0^D
Step	$f_6(x)=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]^D$	0	0.0^D
Schwefel(P2.22)	$f_7(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}x_i$	$[-10,10]^D$	0	0.0^D
Quadric	$f_8(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2$	$[-100,100]^D$	0	0.0^D
Schwefel(P2.21)	$f_9(x)=\max_i\{\lvert x_i\rvert,1\le i\le D\}$	$[-100,100]^D$	0	0.0^D

The characteristic of our algorithm is that the inertia weight and acceleration learning factors of each particle is calculated dynamically during the evolution, and the setting of inertia weight takes both the current and historical information into consideration to avoid the effectiveness by maximum evolution generation. Compared with the linear inertia weight method, our method is more proper because the inertia weight can vary with different problems and particles.



f1: Sphere



f2: Griewank

5. Experimental Design and Analysis

5.1. Experimental Design

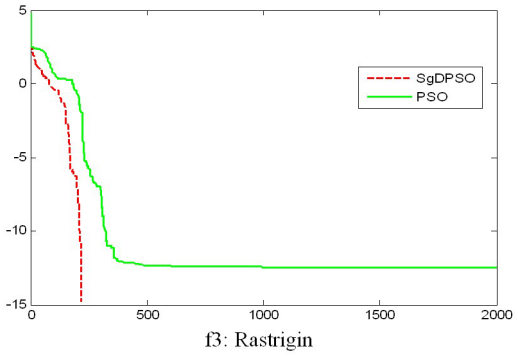
To demonstrate the performance of SgDPSO algorithm, 9 standard testing benchmarks are selected to make a comparison experiment listed in table I, and the dimension is set to 30. To ensure the fairness of the tests, both testing algorithms are run based on SzAPSO [17] algorithm. In the experiment, population size is 15, $w\in[0,1]$. In the standard PSO algorithm,$c_1=c_2=1.49445$ $r_1,r_2\in U(0,1)$, In the SgDPSO algorithm, c_1 and c_2 are calculated by (6) and (7), and the range is referenced as [16]. Two algorithms run 30 times independently.

5.2. Experimental Results

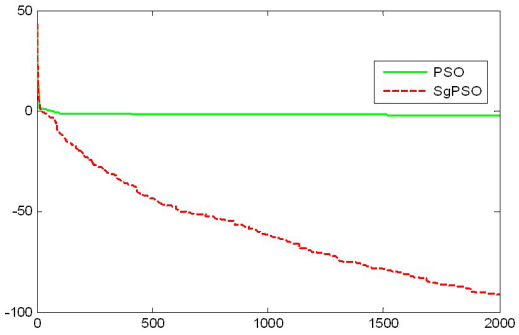
Figure 1 shows the average convergence curve of PSO and SgDPSO separately on 9 standard benchmarks, the results of average convergence accuracy on 9 standard benchmarks is listed in Table II.

5.3. Analysis and Discussions

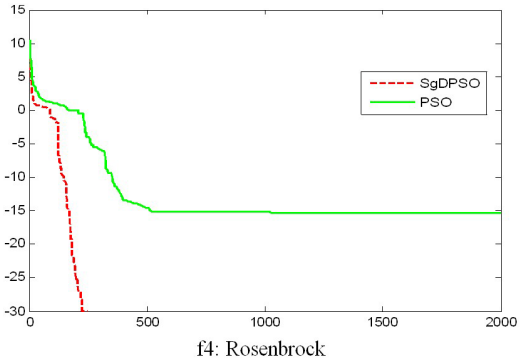
The convergence curve of the two test algorithms is showed in nine standard test functions in Figure 1. From



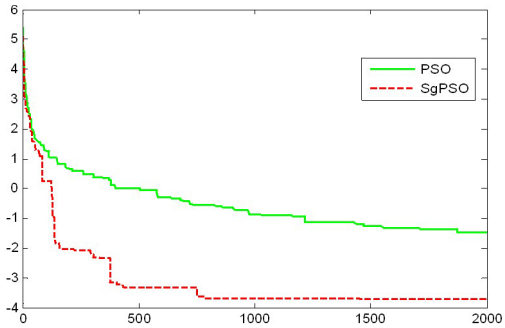
f3: Rastrigin



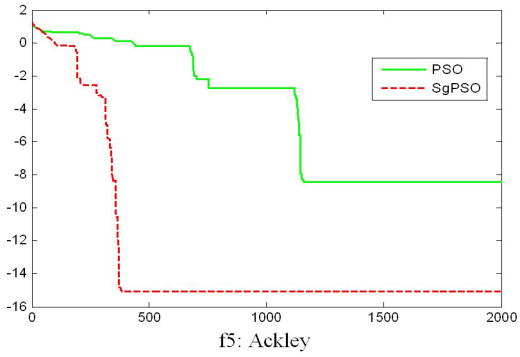
f7: Schwefel (P2.22)



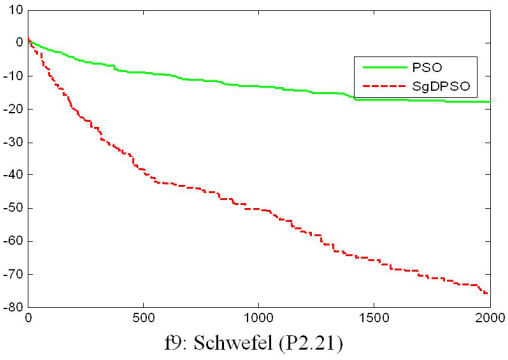
f4: Rosenbrock



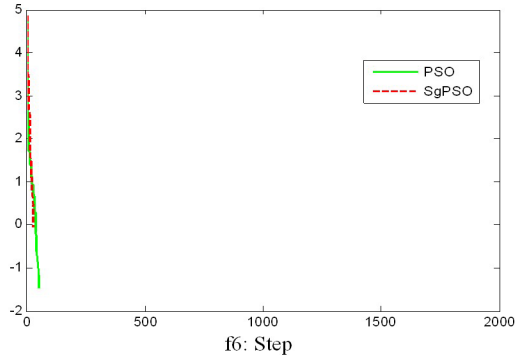
f8: Quadric



f5: Ackley



f9: Schwefel (P2.21)



f6: Step

Figure 1 Comparison among standard PSO and SgDPSO on the numerical optimization problems, the vertical axis is the function value of common logarithm function $\log_{10} f(x)$, and the horizontal axis is the number of generation. f1, f2, f3, f4, f5, f6, f7, f8 and f9 show the optimization result of functions in table I. All are averaged over 30 independent runs.

which we can see that SgDPSO algorithm is better than standard PSO algorithm both in convergence speed and convergence accuracy, and prevents the premature convergence phenomenon effectively. Table II lists the standard deviation of various functions. Compared with the standard PSO algorithm, SgDPSO algorithm is more stable. For the single peak function like f1, f7, f9, the algorithm

shows a good continuous optimization performance, improves the accuracy of the results obviously, about by 20-30%. For the multi-peak function like f5,f8, the convergence accurateness is improved by more than 20%.For the multi-peak function like f2,f3, the pathological quadratic function which is extremely difficult to minimize like f4, and the discontinuous single-peak function like f6, the algorithm can quickly find their optimum, and avoid the defect of premature convergence.

Our algorithm is able to obtain good results for several reasons. Firstly, each particle can use the information of current generation and historical information to adjust flight speed, so algorithm allows the particles which is far away from the optimum in each generation to explore in the global scope , and particles which is close to the optimum in each generation to search in the vicinity, and the particles located between the above two particles can balance the global search and local search, which makes the new algorithm not easily to miss the optimum found earlier as well as the function not to converge in the local optimum. Secondly, our setting method can change the inertia weight depending on the problem, just as shown in Figure 2. Take the functions f1 and f3 for example, they are nonlinear symmetrical function and a large number of local minima of the complex multi-peak function, respectively. We can see from the Figure 2, the standard PSO algorithm is exactly the same inertia weight for different functions which will not vary with different problems, but in the SgDPSO algorithm, there has different inertia weight in different problems. To take the function f1 for example for the same problem, shown in Figure 3,w is different in the same generation of different particles in SgDPSO algorithm, but the inertia weight is the same for all particles in the same generation in the standard PSO algorithm, where the particles can't search according to their characteristics in the search space, resulting in the population diversity decreases during the evolution, and the algorithm can't jump out of local optimum.

From the convergence curve of f1 function, we can see that the optimal value of the swarm continues reducing

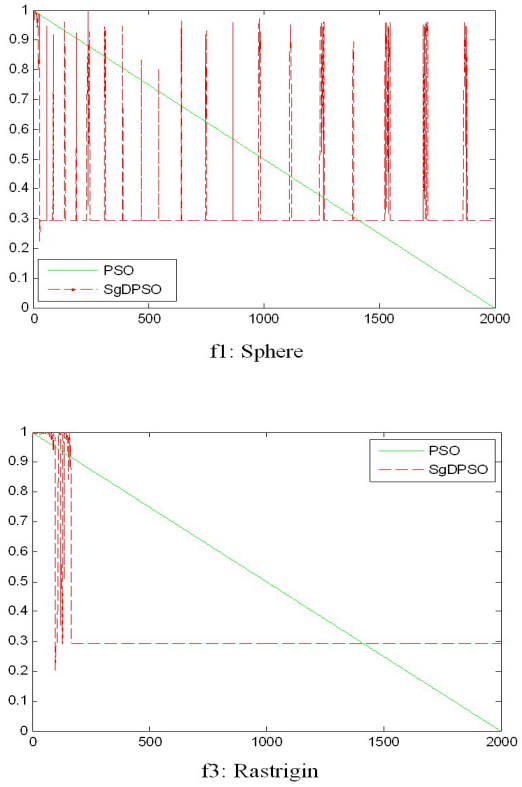


Figure 2 Comparison of the w changing curve of ith particle between PSO and SgDPSO on functions of f1 and f3, the vertical axis is the value of w, and the horizontal axis is the number of generation.

in the SgDPSO algorithm, which makes the difference between the optimal value of the swarm and the fitness value of ith particle constantly vary as well as the evolution rate of ith particle.From the convergence curve of f3 function, we can see that the algorithm converges to the global optimum over 200 generations , which makes the difference between the best optimal values and fitness value of ith particle maintain at zero at the later of evolution, and the evolution rate maintain at one.So the w of ith particle is maintained at 0.3. In Figure 3, there are different in evolutionary rate and fitness of ith particle and jth particle on SgDPSO for f1 during the optimization, the reason is that their w changing curve is different.

Table 2 Experimental Results of the Numerical Optimization Functions, Mean, Min and Stdev Represent the Mean Best; Min Best and Standard Deviation. All are averaged over 30 Independent runs

Function Names	PSO			SgDPSO		
	mean	min	std	mean	min	std
Sphere	6.05E-6	2.58E-156	3.31E-5	1.07E-181	2.14E-217	0
Griewank	0	0	0	0	0	0
Rastrigin	3.16E-13	0	1.36E-12	0	0	0
Rosenbrock	5.72E-16	0	3.13E-15	0	0	0
Ackley	3.60E-9	8.88E-16	1.97E-8	8.88E-16	8.88E-16	0
Step	0	0	0	0	0	0
Schwefel(P2.22)	0.01	8.58E-74	0.05	4.05E-92	7.55E-110	1.69E-91
Qudric	0.04	5.26E-73	0.07	1.99E-4	1.72E-206	7.78E-4
Schwefel(P2.21)	2.66E-65	1.28E-16	1.28E-16	2.40E-112	8.46E-76	8.46E-76

6. Experimental Design and Analysis

To solve the existing problems in standard PSO algorithm, we proposed SgDPSO algorithm that uses the existing information of particle swarm during evolution in this paper. The algorithm is simple and easy to implement. Both the current and historical information of particle are taken into consideration to set the inertia weight and accelerator learning factor dynamically and self-guided to

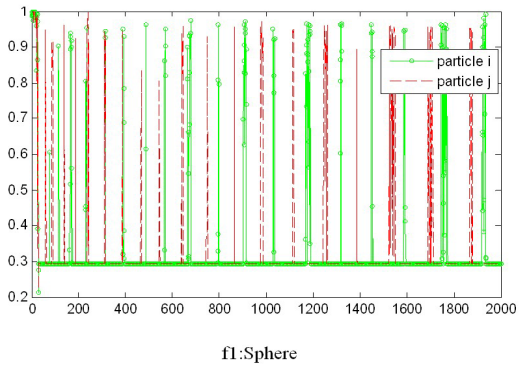


Figure 3 Comparison of the w changing curve of ith and jth particle on SgDPSO for f1, the vertical axis is the value of w, and the horizontal axis is the number of generation.

obtain a balance between the diversity and convergence speed, preventing the premature as well as improving the speed and accurateness. The experiment result shows that our algorithm performance is superior to standard PSO algorithm, and has a good robustness. In the future, we will continue to study different dynamic inertia weight setting methods based on the "historical + current" information ideal to find a better method to set the inertia weight.

Acknowledgement

This research is supported by grant 20080431114 and 20100471350 from the China Postdoctoral Science Foundation funded project and grant 61103235 from the Natural Science Foundation of China and Project Funded by the Priority Academic Program Development of Jiangsu Higer Education Institutions. The authors thank the reviewers for their helpful comments.

References

[1] R.C. Eberhart, J. Kennedy. New optimizer using particle swarm theory. *The 6th Int'l Symposium on Micro Machine and Human Science, Nagoya, Japan* (1995), 39-43.

[2] R.C. Eberhart and Y. Shi. Particle Swarm Optimization: Developments, Applications and Resources. *Proceedings of Congress on Evolutionary Computation, Seoul, Korea, Piscataway, NJ: IEEE Service Center*, Vol.**1**, (2001), 81-86.

[3] K. Kameyama. Particle Swarm Optimizatio - A Survey. *IEICE Trans. Inf. & Syst.*, Vol. E92-D, No.7, July (2009), 1354-1361.

[4] Y. Shi, R.C. Eberhart. A modified particle swarm optimizer. *Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage*, (1998), 67-73..

[5] Y. Shi, R.C. Eberhart. Fuzzy adaptive particle swarm optimizer. *Proceedings of Congress on Evolutionary Computation, Korea: IEEE Service Center*, (2001), 101-106.

[6] R. C. Eberhart, Y. H. Shi. Tracking and Optimizing Dynamic Systems with Particle Swarms. *Proceedings of the 2001 Congress on Evolutionary Computation. Piscataway, NJ: IEEE Press*, (2001), 94-00.

[7] J. Kennedy. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. *In Proc. IEEE Transactions on Evolutionary Computation*, Vol. **3**, Jul. (1999), 1931-1938.

[8] J. Kennedy and R. Mendes. Population structure and particle swarm performance. *IEEE Transactions on Evolutionary Computation*, Vol.**2**, May (2002), 1671-1676.

[9] T. Hendtlass, M. Randall. A Survey of Ant Colony and Particle Swarm Meta-Heuristics and Their Application to Discrete Optimization Problems. *In Proceedings of the Inaugural Workshop on Artificial Life*, (2001), 15-25.

[10] J. Robinson, S. Sinton and Y. Rahmat-Samii. Particle Swarm, Genetic Algorithm, and Their Hybrids: Optimization of a Profiled Corrugated Horn Antenna. *In Antennas and Propagation Society International Symposium, IEEE Press, New York*, Vol.**1**, (2002), 314-317.

[11] C.F. Juang. A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design. *IEEE Transactions on Systems, Man and Cybernetics (Part B-Cybernetics)*, Vol.**34**, No.2, (2004), 997-1006.

[12] Y. Fukuyama and H.A. Yoshida, Particle swarm optimization for reactive power and voltage control in electric power systems. *IEEE Transactions on Evolutionary Computation (2001) (CEC 2001)*, Vol.**1**, May (2001), 87-93.

[13] V. Tandon, H. E. Mounayri and H. Kishawy. NC end milling optimization using evolutionary computation. *Int. J. Mach. Tools Manuf.*, Vol.**42**, No.5, (2002), 595-605.

[14] G. Ciuprina, D. Ioan and I. Munteanu. Use of intelligent-particle swarm optimization in electromagnetic. *IEEE Trans. Magn.*, Vol. **38**, March (2002), 1037-1040

[15] R. Poli, J. Kennedy and T. Blackwell. Particle swarm optimization an overview. *Swarm Intelligence*, Vol. **1**, No.1, (2007), 33-57.

[16] A. Ratnaweera, S. K. Halgamuge. Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients. *IEEE Transactions on Evolutionary Computation*, Vol.**8**, No.3, June (2004), 240-255.

[17] Y. H. Chi, F. C. Sun. An improved particle swarm optimization algorithm with search space zoomed factor and attractor. *Chinese journal of computers*, Vol.**34**, No.1, (2011), 115-130.

[18] R. Boy, P. Recharson. Culture and the evolutionary process. Chicago: University of Chicago Press, (1985).

Huantong Geng received the PhD degree in computer science from University of Science and Technology of China in June 2006. He is currently a professor in the School of Computer and Software, Nanjing University of Information Science and Technology, China. He has published over 50 articles in journals and conferences. His research interests are in the areas of Computational Intelligence and Data Assimilation.
Email: htgeng@nuist.edu.cn

Jun Gao is a graduate student in the School of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests are in the areas of Computational Intelligence and Software Design.
Email: 474527653@163.com

Yanhong Huang received the BS degree in software engineering from Nanjing University of Information Science and Technology in June 2009. She is currently a master student in the School of Computer and Software, Nanjing University of Information Science and Technology, China. Her research interests are in the areas of Computational Intelligence and Machine Learning.
Email: huangyanhong-1006@163.com

Haifeng Zhu received the BS degree in network engineering from Nanjing University of Information Science and Technology in June 2009. He is currently a master student in the School of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests are in the areas of Artificial Intelligence and Evolutionary Computational.
Email: rim611@163.com