A Novel Particle Swarm Optimization with Non-linear Inertia Weight Based on Tangent Function

Li Li[1], Bing Xue[1], Ben Niu[1], Lijing Tan[2], and Jixian Wang[3]

[1] College of Management, Shenzhen University, Shenzhen 518060, China
[2] Measurement Specialties Inc, Shenzhen 518107, China
[3] School of Engineering, Anhui Agricultural University, Hefei 230036, China
drniuben@gmail.com

Abstract. Inertia weight is a most important parameter of particle swarm optimization (PSO), which can keep a right balance between the global search and local search. In this paper, a novel PSO with non-linear inertia weight based on the tangent function is provided. The paper also presents the method of determining a control parameter in our proposed method, saving the user from a tedious trial and error based approach to determine it for each specific problem. The performance of the proposed PSO model is amply demonstrated by applying it for four benchmark problems and comparing it with other three PSO algorithms. From experimental results, it can be concluded that using a non-linear dynamic inertia weight makes the rapidity of convergence rate with higher precision.

Keywords: Particle swarm optimization, inertia weight, tangent function.

1 Introduction

During the 1990's, the researchers paid their attentions on the group animals such as the birds, the ants, or the fishes, which is not very clever alone, but they can finished the high-performance cooperative work when they are together. Swarm intelligence algorithms were generated based on the investigations on these group animals. Particle swarm optimization (PSO) is one of the effective swarm intelligence algorithms firstly proposed by Eberhant and Kennedy [1, 2] in 1995.

As a relatively new swarm intelligence algorithm, PSO has shown some important advances, such as easy in implementation , few parameters to be adjusted, and has a faster convergence rate. It has been successfully applied in many areas [3, 4, 5, 6] However, in the original PSO, every particle searches the best solution based on the previous local best and the global best, and all the particles will be more and more familiar, which leads to the particles to be trapped in a local minima easily. To deal with these issues, many researchers provided different attempts, and adjusting the inertia weight is one of the most effective ways. The inertia weight can balance the global search and the local search to get better results: the global search is stronger when the inertia weight is larger, while the algorithm is good at the local search with a smaller inertia weight [7]. Therefore, the proper inertia weight can improve the performance with fewer generations. Many researchers have done lots of investigations

D.-S. Huang et al. (Eds.): ICIC 2009, LNAI 5755, pp. 785–793, 2009.

on it, such as Shi provided a linearly-decrease inertia weight [8] and the dynamic inertia weight based on fuzzy reasoning [9], Van offered the random inertia weight (RIW) [10], Han produced a novel PSO with an adapting inertia weight [11], Niu proposed a new hybrid global optimization algorithm PSODE combining particle swarm optimization (PSO) with differential evolution(DE) [12].

In the present paper, we proposed a novel non-linear inertia weight adjusting method based on tangent function. To illustrate the effectiveness and performance of the new strategy, a set of four representative benchmark functions was employed to evaluate it in comparison with other three different improved PSO with fixed inertia weight (FIW), a linearly-decrease inertia weight (LIW)[8], and Nonlinearly-decrease inertia weight (NIW)[13], respectively. The remaining contents of this paper are organized as follows: The next section introduces the standard PSO. A novel PSO with non-linear dynamic inertia weight is presented in Section 3. In Section 4, we describe the benchmark functions, experimental settings, and compare experimental results of the novel PSO with other three PSO with different inertia weight. Finally, Section 5 we end the paper with some conclusions and future work propositions.

2 Standard Particle Swarm Optimization Algorithm

PSO is a population-based searching algorithm and is initialized with a population of random solutions to search for the optima by updating generation. The particle i denoting one potential solution is flying in the *n*-dimensions; in the *t*th generation, it owns a position $x_i(t) = (x_{i1}, x_{i2} \cdots \cdots, x_{in})$, and velocity $v_i(t) = (v_{i1}, v_{i2} \cdots \cdots, v_{in})$. There is a fitness value evaluated with a predefined fitness function to appraise the particle's current position. At the same time, every particle can remember the best position they had ever been. Therefore, in every generation there are two "Best": The local best value is the best position that the particle has achieved in the current stage, which is called *Pbest;* the global best value is the overall best solution tracked by the particle swarm, which is called *Gbest*.

Based on the *Pbest* and *Gbest*, the PSO searches for the best solution by updating every particle's position and velocity until meeting the end conditions. And the following equations are used to update the generations:

$$V_{id}(t+1) = V_{id}(t) + c_1 \cdot rand() \cdot (p_{id} - x_{id}(t)) + c_2 \cdot rand() \cdot (p_{gd} - x_{id}(t)), \tag{1}$$

$$x_i(t+1) = x_i(t) + V_i(t+1). \tag{2}$$

Where the c_1 、 c_2 in the Eq(1) are the learning factors, they both are often taken as 2. $rand()$ is a random value, which is distributing during [0,1]. p_{id} presents the *Pbest* while p_{gd} presents the *Gbest*. In order to make sure the position $x_i(t)$ is in the feasible region, there is a maximum $v_i(t)$ denoted by $v_{\max}$ to confine the velocity:

$$V_{id}(t) = \begin{cases} V_{id}(t) \cdots\cdots\cdots\cdots\cdots \left|V_{id}(t)\right| < V_{\max} \\ V_{\max} \cdots\cdots\cdots\cdots\cdots \left|V_{id}(t)\right| \geq V_{\max} \end{cases}. \tag{3}$$

As when using the above equations to search the best solution, the convergence velocity in the earlier period is fast, but the local search is poor and the convergence velocity is slower along with generations, and the precise of the solution often can not meet the satisfactory level. For these problems, Shi [11] introduced the inertia weight w into the velocity equation:

$$V_{id}(t+1) = wV_{id}(t) + c_1 \cdot rand() \cdot (p_{id} - x_{id}(t)) + c_2 \cdot rand() \cdot (p_{gd} - x_{id}(t)) . \quad (4)$$

The Eq(4) also can be divided into three parts: the first part indicates the previous velocity of the particle, which can make sure the global search, the second and third part are the social reasons that lead to the change the velocity of the particle, and they can decide the local search of the algorithm. So the inertia weight w can control the influence of previous velocity on the new velocity, and it can make a balance between the global search and the local search: global search performance is fine with larger inertial weight while a smaller inertia weight facilitates a local search. The two searches with appropriate coordination can bring a better performance, so to find a proper w is one of the crucial ways to improve the capability of the algorithm.

3 Novel Non-linear Inertia Weight PSO Algorithms Based on Tangent Function

Based on lots of experiments, we found that the global search is strong when w is larger, and the algorithm owns faster convergence ability but it is hardly to get precise. On the other hand when w is smaller, the local search is stronger and the result is more precise, but the convergence ability is poorer, neither the ability to escape the local best. So the w should vary with the generations, and we try to find a proper variations to improve the performance of the PSO. During the initial experimental study, we tried to introduce a monotone increasing or decreasing strategy to update w. In our proposed method tangent function $y = \tan(x)$ is used to be the updating function, in which result y is increasing along with the independent variable x, and the speed of the increase also increases. When $x = 7/8$, $y = 1$. Based on this observation, we proposed a novel PSO with non-linear initial weight and the resulting inertia weight updating equation is as following:

$$w(t) = (w_{start} - w_{end}) * \tan(7/8 * (1 - (\frac{t}{t_{\max}})^k)) + w_{end} . \quad (5)$$

Where w_{start} is the initial value of the inertia weight, which is also the largest value and normally set w_{start}=0.9, w_{end} is the final value of the inertia weight, which also is the smallest one and normally set $w_{end} = 0.4$; $t_{\max}$ is the maximum number of iterations. According to the equation, w is nonlinearly-decrease along with the increase of the iterations. In the initial iteration, the PSO with the larger w is with stronger global search, so the particle can fly around the total search space quickly. The local research becomes stronger along with the w becoming smaller and smaller. The new strategy may enhance the capability of the algorithm to avoid premature convergence and

escapes the local optimal. There is a coefficient 7/8 in the Eq (5) to guarantee the w distributed in [0.4, 0.9]: When $t=1$, $w(t)=w_{start}=0.9$ and when $t=t_{\max}$, $w(t)=w_{end}=0.4$.

k is the control variable which can control the smoothness of the curve that reflects the relationship between the w and t. Figure 1-3 show the variation of inertia weight along with the generations when $k=0.2$、$k=1$、$k=3$, respectively. From these figures, it can be found that: when $k=0.2$, the functions between the w and t is convex function; when $k=1$, it is almost a linear one leaning to concave; and when $k=3$, it is a concave function.

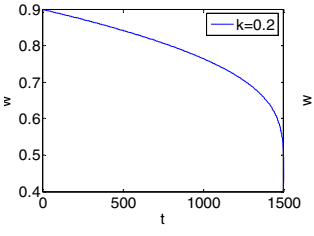

Fig. 1. $k=0.2$

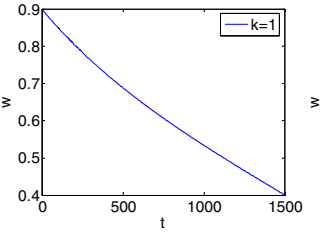

Fig. 2. $k=1$

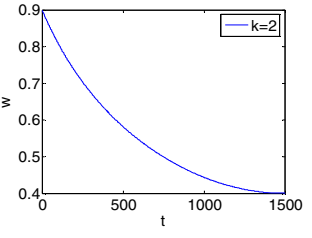

Fig. 3. $k=2$

In order to choose the best value of k, a multimode Griewank function is employed in the experiments. And k is confined in [0.1~2.0]. The experimental result (i.e., the mean and the standard deviations of the function values found in 20 runs) are listed in Table 1.

From Table 1, it shows that when k is within [0.4~0.6] and [1.4~1.7], the mean and the standard deviations of the function values are both stable. So we can choose the value which can take a faster convergence rate. Figure 4 and 5 show the variation of the logarithm (base 10) of the mean values along with the generations when $k=0.6$, $k=1.7$ respectively. It is easy to find it can get a faster convergence rate when $k=0.6$ than $k=1.7$. In the following experiments, we choose $k=0.6$.

Table 1. Experimental results on Griewank function using different k

k	mean value	standard deviation	k	mean value	standard deviation
0.1	0.0294	0.0276	1.1	0.0302	0.0230
0.2	0.0300	0.0223	1.2	0.0355	0.0194
0.3	0.0329	0.0248	1.3	0.0413	0.0291
0.4	0.0266	0.0191	1.4	0.0230	0.0212
0.5	0.0258	0.0278	1.5	0.0289	0.0273
0.6	0.0254	0.0207	1.6	0.0263	0.0200
0.7	0.0315	0.0307	1.7	0.0261	0.0198
0.8	0.0300	0.0343	1.8	0.0334	0.0219
0.9	0.0301	0.0235	1.9	0.0257	0.0253
1.0	0.0264	0.0267	2.0	0.0280	0.0256

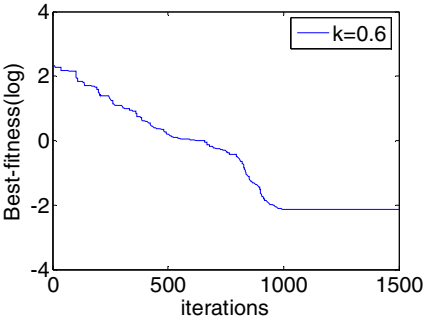

Fig. 4. $k = 0.6$

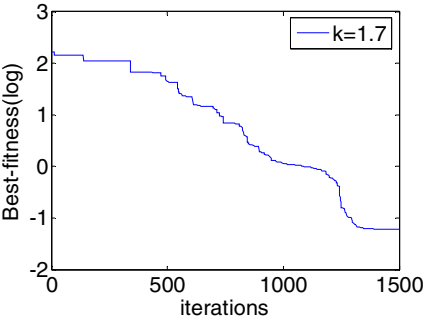

Fig. 5. $k = 1.7$

4 Experimental Studies

4.1 Test Functions and Parameters Settings

In order to test the effectiveness and performance of our proposed method four representative benchmark functions are used in comparison with other three different inertia weight strategies (FIW, LIW, and NIW). The parameter settings of these four functions are listed in Table 2.

Table 2. Test functions and parameters setting

Functions name	Function model	Dim	Search space	$V_{\max}$
Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	20	(-100,100)	100
Rosenbrock	$f_2(x) = \sum_{i=1}^{n} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right)$	20	(-30, 30)	30
Rastrigin	$f_3(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$	20	(-10, 10)	10
Griewank	$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	20	(-600, 600)	600

The four test functions can be classified as unimodal (Sphere function and Rosenbrock function) and multimodal functions (Rastrigin function and Griewank function). $w = 0.68$ is set in FIW. Eq (6) and Eq(7) are used to determine w in LIW and NIW , respectively.

$$w(t) = w_{start} - \frac{w_{start} - w_{end}}{t_{\max}} \times t \,, \tag{6}$$

$$w(t) = w_{start} - (w_{start} - w_{end}) \times \exp(-k * (\frac{t}{t_{\max}})^2) . \quad (7)$$

In our experimental studies, w used in w in the three methods (TANW□LIW and NIW) are all set within [0.9, 0.4], that is $w_{start} = 0.9$, $w_{end} = 0.4$. The other parameters $c_1 = c_2 = 2.0$, the swarm size is 40, and the allowable error $\sigma = 1e-80$, and $t_{\max} = 1500$. A total of 50 runs for each experimental setting are conducted.

4.2 Experimental Results

The results of the four functions are listed in Table 3, including the maximum value, the minimum value and the standard deviations. The graphs presented in Figs 6–9 illustrate the evolution of best fitness found by three algorithms, averaged for 50 funs for the four functions.

Table 3. Results for all algorithms on benchmark functions

Function	strategy	Max value	Min value	standard deviations	Mean value
Sphere	TANW	2.6552e-015	1.2535e-020	3.7727e-016	9.0940e-017
	NIW	6.3016e-012	1.1269e-015	1.3952e-012	7.4996e-013
	FIW	5.8000 e-002	1.000 e-003	1.4700e-002	1.0200e-002
	LIW	9.7600e-009	4.8377e-012	1.6531e-009	6.8240e-010
Rosenbrock	TANW	2.6385e+002	2.1250e-001	5.5068e+001	4.1048e+001
	NIW	4.4738e+002	1.1850e+001	7.7280e+001	5.0810e+001
	FIW	8.0398e+002	1.0502e+001	1.4272e+002	1.2477e+002
	LIW	5.6734e+002	4.4772e+000	1.0744e+002	7.0154e+001
Rastrigin	TANW	3.5819e+001	6.9647e+000	5.8282e+000	1.6916e+002
	NIW	3.1839e+001	5.9698e+000	5.3219e+000	1.8548e+001
	FIW	4.8523e+001	1.1067e+001	9.0932e+000	2.9706e+001
	LIW	3.3859e+001	6.9649e+000	5.3284e+000	1.8067e+001
Griewank	TANW	7.8700e-002	0.0000e+000	2.0800e-002	2.4000e-002
	NIW	1.3510e-001	2.6645e-015	2.6700e-002	2.6400e-002
	FIW	7.4520e-001	6.2000e-003	1.9350e-001	2.1550e-001
	LIW	1.0520e-002	9.9886e-011	2.5600e-002	3.2800e-002

The data in Table 3 show that new way of w (TANW) can obtain more precise results for all of the four functions than other three methods. As seen from the figures, TANW with the fastest convergence rate can get the best solution. Although for Rastrigin function, the standard deviations obtained by TANW is higher than the results obtained by LIW and NIW, while it still gets the faster convergence rate and more a promising end-results. FIW that owns fixed w cannot balance the global search and the local search. Therefore, it owns faster convergence rate in the earlier

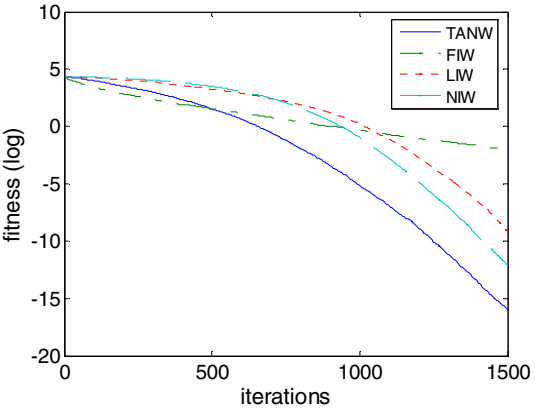

Fig. 6. Sphere function

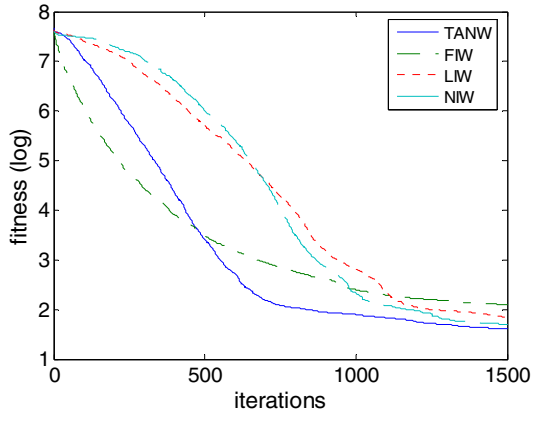

Fig. 7. Rosenbrock function

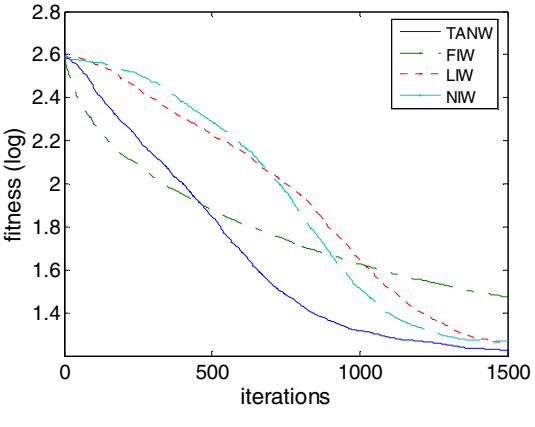

Fig. 8. Rastrigin function

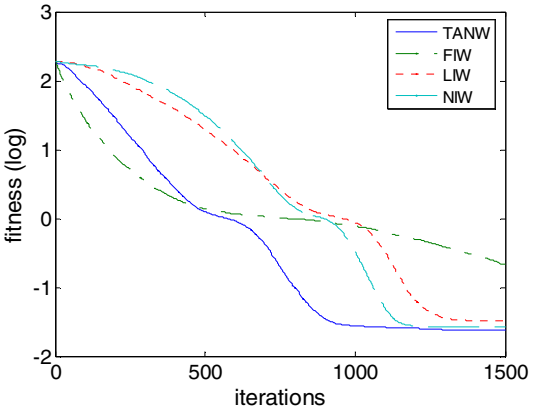

Fig. 9. Griewank function

period, but the final result is worst. NIW represented the weaker robustness (the larger standard deviations) in Griewank function and less precise (the larger mean of the function) in Rastrigin function than LIW, besides the two points, it produced better performance than LIW as a whole.

The results also indicated that in the most time, the fixed inertia weight （FIW） often cannot give the satisfactory result, and the non-liner inertia weight (NIW and TANW) performs better than linear one (LIW).

5 Conclusion and Future Work

Based on analyzing the effectiveness of the inertia weight and the features of the tangent function, a novel PSO with non-liner inertia weight is proposed in this paper. Its performance is evaluated by four benchmark functions compared with other three methods. The experimental results illustrated that our proposed TANW not only has a faster convergence rate but also yields high quality of the solution and robustness of the results.

Some further research to apply the proposed method to solve the actual problems should be carried out, such as the VRP and the portfolio optimization and other engineering problems.

Acknowledgment

This work is supported by Shenzhen-Hong Kong Innovative Circle project (Grant no.SG200810220137A) and Project 801-000021 supported by SZU R/D Fund.

References

1. Eberchart, R.C., Kennedy, J.: A New Optimizer Using Particle Swarm Theory. In: Proceedings of the 6th International Symposium on Micromachine and Human Science, Nagoya, Japan, pp. 39–43 (1995)

2. Eberchart, R.C., Kennedy, J.: Particle Swarm Optimization. In: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)
3. Mendes, R., Cortez, P., Rocha, M., Neves, J.: Particle Swarms for Feedforward Neural Network Training. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2002), pp. 1895–1899 (2002)
4. Venayagamoorthy, G.K., Doctor, S.: Navigation of Mobile Sensors Using PSO and Embedded PSO in a Fuzzy Logic Controller. In: Proceedings of the 39th IEEE IAS Annual Meeting on Industry Applications, Seattle, USA, pp. 1200–1206 (2004)
5. Parsopoulos, K.E., Papageorgiou, E.I., Groumpos, P.P.: A First Study of Fuzzy Cognitive Maps Learning Using Particle Swarm Optimization. In: Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003), Canbella, Australia, pp. 1440–1447 (2003)
6. Abido, M.A.: Optimal Power Flow Using Particle Swarm Optimization. Int. J. Elect. Power Energy Syst. 24, 563–571 (2002)
7. Shi, Y., Eberhart, R.C.: Parameter Selection in Particle Swarm Optimizations. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998)
8. Shi, Y., Eberhart, R.C.: A Modified Particle Swarm Optimizer. In: Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, AK, pp. 69–73 (1998)
9. Shi, Y., Eberhart, R.C.: Fuzzy Adaptive Particle Swarm Optimization. In: Proceedings of the Congress on Evolutionary Computation, Seoul, Korea, pp. 101–106 (2001)
10. Van den Bergh, F.: An Analysis of Particle Swarm Optimizer. Department of Computer Science. University of Pretoria, South Africa (2002)
11. Han, J.H., Li, Z.H.: An Adapting Particle Swarm Optimization and the Simulation Study. System simulation Journal (2006)
12. Niu, B., Li, L.: A novel PSO-DE-based hybrid algorithm for global optimization. In: Huang, D.-S., Wunsch II, D.C., Levine, D.S., Jo, K.-H. (eds.) ICIC 2008. LNCS (LNAI), vol. 5227, pp. 156–163. Springer, Heidelberg (2008)
13. Tan, L.J.: Particle Swarm Optimization and Its Application on the Control System of Continuous Conveyer of Disc-tube Assemble (Chinese). Master thesis, Liaoning Technical University (2007)