Study on inertia weight decreasing strategy of particle swarm optimization based on inverse coseca

Huiming Liu¹, LIU Yong¹* and Jiaqi Zhen¹

Heilongjiang University, Haerbin, Heilongjiang, China,150080

¹Corresponding author’s e-mail: liuyong@hlju.edu.cn

Abstract The standard particle swarm optimization (pso), which introduces inertia weight w, is an effective method to find the extreme value of the function. However, particle swarm optimization (pso) has some disadvantages. When dealing with optimization problems, pso lacks effective parameter control and is prone to fall into local optimization, which leads to low convergence accuracy. In, this paper, put forward a new improved particle swarm optimization (pso) algorithm, The nonlinear decreasing inertia weight by the CSC function strategy, at the same time to join the beta distribution on random Numbers, thus to balance the global search and local search ability of the algorithm. The learning factor is changed asynchronously to improve the learning ability of the algorithm. By adopting Griewank, Rastrigrin, J.D. Schaffer three standard test functions to simulate experiment, at the same time and the basic particle swarm algorithm the inertia weigh in a fixed value, the linear regressive LDIW and nonlinear regressive comparison. The experimental results show that the nonlinear decreasing strategy with dynamic adjustment of inverse cosecant function can improve the convergence speed and stability.

1. Introduction

Particle Swarm Optimization (POS) is a global Optimization evolutionary algorithm proposed by Kennedy et al. [9] in 1995. The idea comes from the study on the predation behavior of birds. Particle swarm optimization (pso) is an emerging evolutionary technology based on swarm intelligence. It has fast iteration, strong robustness, good optimization effect, simple implementation, and is widely used to solve optimization problems in scientific research and engineering practice.

In particle swarm optimization (pso), the value of inertia weight and learning factor is very important. Larger weight is conducive to improving the global search ability of the algorithm, while smaller weight is conducive to improving the local search ability of the algorithm. Learning factors reflect the communication of information between particles. In order to make the pso algorithm converge quickly, robust and stable, researchers have done a lot of research work. Shi Y[4] proposed particle swarm optimization algorithm with inertia weight. Tong qiu juan [2] proposed a particle swarm optimization algorithm based on adaptive dynamic change. By using the idea of adaptive dynamic change of the behavior parameters of the algorithm, the fitness value of particles was introduced into the inertia weight coefficient and learning factor of the algorithm, and a new improved particle swarm optimization algorithm was proposed. Feng kang [7] et al focused on the research idea of chaos improved particle swarm optimization, proposed an embedded chaos particle swarm optimization algorithm, and demonstrated the superiority of the algorithm through simulation test. Li huirong [3] proposed a particle swarm optimization algorithm for nonlinear decreasing inertia weight strategy. In this paper, the inertia weight adopts the nonlinear decreasing of anti-cosecant function, and
the learning factor changes asynchronously. Meanwhile, it is compared and analyzed with other improved particle swarm optimization algorithms.

2. Elementary particle swarm optimization

For Standard Particle Swarm Optimization (SPSO), when solving Optimization problems, the algorithm first initializes a Swarm of random particles and then searches for the optimal solution according to the iterative formula.

Assuming that the target search space is d dimension, the position of the ith particle in the particle swarm can be expressed follows

\[ X_i = (x_{i1}, x_{i2}, ..., x_{id}), i = 1,2, ..., n \]  

(1)

In the particle swarm, the velocity of the ith particle swarm can be expressed as:

\[ V_i = (v_{i1}, v_{i2}, ..., v_{id}), i = 1,2, ..., n \]  

(2)

The best place to search for the ith particle in the swarm, is the individual extreme value, which can be expressed as:

\[ p_{best} = (p_{1i}, p_{2i}, ..., p_{ni}), i = 1,2, ..., n \]  

(3)

The best position searched by the whole particle swarm is the global extreme value, which can be expressed as:

\[ g_{best} = (g_1, g_2, ..., g_n), i = 1,2, ..., n \]  

(4)

When two optimal values are found, the velocity and position of the particle are updated as follows:

\[ V_{ij} = V_{ij} + c_1 r_1 [V_{ij} - X_{ij}] + c_2 r_2 [P_{ij} - X_{ij}] \]  

(5)

\[ X_{ij} = X_{ij} + V_{ij} \]  

(6)

In equation (5) and (6), i=1,2, ..., n, j=1,2, ..., d, n is the number of particles in the particle swarm, d is the dimension of the target space and \( c_1 \), \( c_2 \) is the learning factor. \( v_{ij} \) is the velocity of the particle, the magnitude of the \( v_{ij} \) is determined by the nature of the objective function, \( r_1, r_2 \) is a random number between [0,1].

Equations (5) and (6) constitute the basic particle swarm optimization algorithm, experiments show that pso has a fast convergence rate, but its local search ability is poor and its stability is poor. In order to improve the local search capability of the algorithm, Shi Y[4] et al added the inertia coefficient \( w \) to formula (5) and modified formula (5) as follows:

\[ V_{ij} = w V_{ij} + c_1 r_1 [V_{ij} - X_{ij}] + c_2 r_2 [P_{ij} - X_{ij}] \]  

(7)

It can be seen from the updating formula of particle velocity: \( w \) represents inertia weight factor. Generally, the "inertia" part is the best in 0.4~0.9, which can balance global and local search, \( c_1 r_1 [V_{ij} - X_{ij}] \), it's the "cognitive" part, which suggests that the particle's flight comes from itself, enhancing the ability to search locally, \( c_2 r_2 [P_{ij} - X_{ij}] \), the "social" part, as it is called, indicates the ability of particles to communicate information to each other, enhancing global search.

Ratnaweera A[10] proposed the strategy of dynamically adjusting learning factors. Learning factor \( c_1 \), \( c_2 \), it's linearly increasing or linearly decreasing in some way. In this article, we will take \( c_1 \), \( c_2 \) the specific formula can be expressed as:

\[ c_1 = c_{1,ini} + (c_{1,fin} - c_{1,ini}) * t / t_{max} \]  

\[ c_2 = c_{2,ini} + (c_{2,fin} - c_{2,ini}) * t / t_{max} \]  

(8)

Type \( c_{1,ini}, c_{2,ini} \) represent the initial value of \( c_1, c_2 \), Among them, \( c_{1,fin}, c_{2,fin} \) is the final value of \( c_1, c_2 \), when \( c_{1,ini} = 2.5, c_{1,fin} = 0.5, c_{2,ini} = 0.5, c_{2,fin} = 2.5 \), the optimization effect of particle swarm is better.

Asynchronous change of learning factors can make particles tend to learn by themselves in the early stage of search and social learning in the late stage of search, which can ensure the coordinated development of global search and local search of particle swarm, which is conducive to convergence to the optimal solution.
In order to balance the ability of the algorithm in local search and global search, Shi et al. proposed the strategy of linear decrease (LDIW), which is, in the process of iteration, the value of linear decrease $w$ is:

$$w = (w_{\text{start}} - w_{\text{end}})(t_{\text{max}} - t)/t_{\text{max}} + w_{\text{end}} \ldots \ldots \ldots \ldots (9)$$

Among them, $t_{\text{max}}$ is the maximum number of iterations. It is the current number of iterations, $w_{\text{start}}$ is the initial inertia weight, and $w_{\text{end}}$ is the inertia weight after evolution. When $w_{\text{start}} = 0.9, w_{\text{end}} = 0.4$, the algorithm will perform better. Among them, the commonly used choice of inertia weight is as follows [6]:

$$w = w_{\text{start}} + (w_{\text{start}} - w_{\text{end}}) \left( \frac{2t}{t_{\text{max}}} - \left( \frac{t}{t_{\text{max}}} \right)^2 \right) \ldots \ldots \ldots \ldots (10)$$

This is the nonlinear progressive inertia weight. Compared with the fixed value and the linear decreasing inertia weight, the nonlinear decreasing inertia weight has faster convergence speed, higher stability, and better global search and local search.

### 3. Nonlinear decreasing weights in the form of anti-cosecant

Although the decreasing inertia weight gives the algorithm a faster speed in the early stage and a slower speed in the late stage. However, each generation of the population lacks diversity, which will lead to an increase in the number of iterations of the algorithm and a slower convergence speed. Based on random Numbers obeying beta distribution in literature (1), In this paper, an inverse cosecant form of dynamic distribution of nonlinear decreasing inertia weight strategy is proposed. Compared with fixed value, linear and nonlinear weight has faster convergence speed and better stability. Meanwhile, asynchronous learning factor is applied to enhance the local and global search ability. The expression is:

$$w = f_1 \ast (w_{\text{start}} - w_{\text{end}}) \ast \csc^{-1} \frac{2(t + T_{\text{max}})}{T_{\text{max}}} - \tau \ast \text{betarnd}() \ldots \ldots \ldots \ldots (11)$$

Among them, $f_1 = 3.6$, mainly control $w$ between the maximum and minimum value, and $\tau$ is the inertia deviation factor, $\tau \in [0.1, 0.9]$. Betarnd () is a random number generated by beta distribution, and its distribution probability includes from uniform distribution to normal distribution. Beta distribution [1] is used to make $w$ deviate, so as to make the value distribution of inertia weight more uniform and more flexible. The purpose of adding (inertia deviation factor) before betarnd () is to control the deviation degree of $w$, so as to make the deviation more reasonable. With the progress of iteration [8], the inertia weight $w$ has a relatively large value in the early stage and a fast speed, so as to avoid falling into local extreme value, and has a good convergence speed and a strong global search ability. Meanwhile, it has a relatively small value in the later stage and a slow speed, so as to enhance the local search ability.

### 4. Experimental simulation analysis

In this paper, four classical test functions (table 1) are adopted to test these algorithms. Table 2~ table 4 are respectively the mean and variance of the optimal solution obtained by the test function. Except that the global minimum of Schaffer J D test function is -1 and the global minimum of the other three test functions is all 0, and the number of particles in the algorithm is all 10, $c_1, c_2$ is the asynchronous learning factor. Formula in (4), $w_{\text{start}} = 0.9, w_{\text{end}} = 0.4, f_1 = 3.6$, the optimized end condition is that the number of iterations exceeds 500.

| function     | Functional expression                                                                 | dimension | A search scope | Maximum speed |
|--------------|--------------------------------------------------------------------------------------|-----------|----------------|--------------|
| Rastrigrin   | $\sum_{i=1}^{n} [x_i^2 - \cos(10(2\pi x_i)) + 10]$                                 | 10        | (-10,10)       | 10           |
Griewank
\[ \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1 \]

J.D Schaffer
\[ \sin \left( \sqrt{x_1^2 + x_2^2 - 0.5} \right) \frac{1}{(1 + 0.001(x_1^2 + x_2^2))^2} - 0.5 \]

Table 2 Rastrigrin test functions results

| Weight strategy selection | Fixed weight w=0.9 | A linear gradient | Nonlinear decline | Nonlinear decline in this paper |
|---------------------------|---------------------|-------------------|-------------------|---------------------------------|
| The mean                  | 0.7473              | 0.5516            | 0.1412            | 0.08886                         |
| The variance              | 0.6935              | 0.6896            | 0.4091            | 0.3567                          |

Table 3 Griewank test functions results

| Weight strategy selection | Fixed weight w=0.9 | A linear gradient | Nonlinear decline | Nonlinear decline in this paper |
|---------------------------|---------------------|-------------------|-------------------|---------------------------------|
| The mean                  | 1.628 \times 10^{-3} | 0.7707 \times 10^{-3} | 0.2052 \times 10^{-3} | 0.1156 \times 10^{-3} |
| The variance              | 3.092 \times 10^{-3} | 3.035 \times 10^{-3} | 1.578 \times 10^{-3} | 1.1482 \times 10^{-3} |

Table 4 J.D Schaffer t functions results

| Weight strategy selection | Fixed weight w=0.9 | A linear gradient | Nonlinear decline | Nonlinear decline in this paper |
|---------------------------|---------------------|-------------------|-------------------|---------------------------------|
| The mean                  | 0.452               | 0.323             | 0.1755            | 0.1725                          |
| The variance              | 0.8911              | 0.8657            | 0.6995            | 0.447                           |

Figure 1 iteration curve of fitness optimization of Rastrigrin function

Figure 2 iteration curve of fitness optimization of Griewank function
Figure 3 iteration curve of fitness optimization of J.D Schaffer function

As can be seen from table 2 and figure 1, when the fixed weight $w$ is 0.9, the premature convergence stops at the local optimal point, and when the iteration is 500 times, the convergence condition is not reached. The convergence performance proposed in this paper is the best, followed by the non-linear one, and the linear one is the worst. The table 3 and figure 2 shows that the four kinds of iteration algorithm to a certain number of times are close set the convergence precision of fixed weight convergence accuracy and convergence performance of the worst, the number of iterations of the linear need more, nonlinear need less number of iterations. this paper puts forward the CSC form the dynamic adjustment of nonlinear regressive strategy and requires the least number of iterations. According to figure 3 and table 4, the nonlinear decreasing inertia weight is still the best in this paper. To sum up, the anti-cosecant particle swarm optimization algorithm proposed in this paper, which dynamically adjusts the nonlinear decreasing inertia weight strategy, has better global and local search capabilities, can converge to the optimal solution faster, and needs the least number of iterations, which effectively improves the overall optimization ability of the algorithm.

5. conclusion
In this paper, the basic particle swarm algorithm and the inertia weight were analyzed, and the decreasing inertia of particle swarm optimization (PSO) algorithm is improved, puts forward a nonlinear weight change formula in the form of CSC, learning factor using the asynchronous strategy at the same time, better balance the global search and local search ability of the algorithm. And at the same time, join the beta distribution random number, so that the value of inertia weight more evenly and more flexible. By conducting simulation experiments on four representative test functions, the new algorithm can not only improve the convergence speed and accuracy, but also maintain good stability. In the next stage, the algorithm will be studied in related application fields.

Thanks
This subject is completed under the kind care and careful guidance of teacher liu in the process of topic selection and research. His serious scientific attitude, rigorous academic spirit and excelsior work style deeply inspired and inspired me. From the selection of the project to the final completion of the project, teacher liu has always given me careful guidance and unremitting support.

Fund project
1 National Natural Science Foundation of China (61501176), 2 Outstanding youth project of natural science foundation of heilongjiang province (YQ2019F015), 3 Natural science foundation of heilongjiang province(F2018025)

Reference
[1] Pei Yuhang, Liu Jingsen, Li Yu. An adaptive bat algorithm for dynamically adjusting inertia weight [J]. School of computer and information engineering, henan university, 2017.
[2] Tong Qiujuan, Zhao Qi, Li Meng. Particle swarm optimization algorithm based on adaptive dynamic change [J]. Xi’an university of posts and telecommunications, 2019.
[3] Li Huirong, Gao Xuelin, Li Jimin. A particle swarm optimization algorithm for nonlinear decreasing inertia weight strategy [J]. Journal of shangluo university, 2007(04): 16-20.

[4] Shi Y, Eberhart R C. A modified particle swarm optimizer. Proceedings of the IEEE Conference on Evolutionary Computation, 1998: 69-73.

[5] Chen Guimin, Jia Jianyuan, Han qi. Research on inertia weight decreasing strategy of particle swarm optimization algorithm [J]. Journal of xī ’an jiaotong university, 2006(01): 53-56+61.

[6] Wang Xiaochuan, Shi Feng, Yu Lei, Li Yang. MATLAB neural network. Beijing: Beijing university of aeronautics and astronautics press, 2013, 8.

[7] Feng Kang. Applied research on deformation prediction model based on chaotic particle swarm neural network [D]. North China university of water resources and hydropower, 2016.

[8] Guo lili, liu yong, wang weixi. Improvement of inertia weight decreasing strategy of particle swarm optimization algorithm [J]. Heilongjiang university, 2019.

[9] Kennedy J, Eberhart R C. Particle swarm optimization. IEEE International Conference on Neural Networks, 1995: 1942-1948.

[10] Ratnaweera A, Halgamuge S.K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 240-255.