A Dynamic Inertia Weight Particle Swarm Optimization Algorithm Based on Gaussian Disturbance

Fang Yiqiu[a], Cheng Yuan[b], and Ge Junwei[c]

College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China;
[a]fangyq@cqupt.edu.com, [b]498640643@qq.com, [c]gejw@cqupt.edu.com

Keywords: Dynamic inertia weight, Gaussian Disturbance, Particle Swarm Optimization

Abstract: As one of the representatives of intelligent algorithm, Particle Swarm Optimization (PSO) has been widely concerned and applied since it was proposed. However, the traditional Particle Swarm Optimization (PSO) algorithm has some disadvantages, such as premature convergence, local optimization and lo resolution accuracy. In order to solve the problems in the algorithm, this paper proposes a dynamic inertia weight Particle Swarm Optimization algorithm based on Gaussian Disturbance. Through testing experiments with 5 benchmark functions, the improved algorithm has significantly improved its global search ability and optimization accuracy, and also overcomes the shortcoming of traditional Particle swarm Optimization (PSO).

1. Introduction

Particle Swarm Optimization (PSO) is a bionic intelligent Optimization algorithm proposed by Kennedy et al[1] in 1995. It is simple and easy to implement. After being put forward, it has gained widespread attention and achieved great development. It has been applied in many fields [2-4].

The traditional particle swarm optimization algorithm can find the optimal solution quickly and accurately when it is used in the optimization of unimodal functions. It has good results. However it is easy to fall into the local optimum, and the convergence speed is fast when it is used in the optimization of multimodal functions. The traditional particle swarm optimization algorithm cannot jump out of the local optimal solution, so that the global optimal value cannot be found. In response to the problems, Li Rongyu et al [5] proposed an improved particle swarm optimization algorithm based on levy flight; Qi Xiaobo et al [6] proposed a chaotic particle swarm hybrid algorithm; Wang Wenyi et al [7] proposed a hybrid algorithm combining particle swarm optimization and genetic algorithm. The improved algorithms have certain improvement than the original algorithm, but there are still some shortcomings in the aspects of global optimization ability and optimization precision.

DOI: 10.23977/icmit.2018.017

2. Algorithm ideas and steps

2.1 Particle Swarm Optimization.

Particle swarm optimization is derived from the predation behavior of birds. The easiest and most effective strategy for birds looking for food is to search for the area around the bird closest to the food. In the algorithm is expressed as: Each particle represents a potential solution. The quality of particles is judged by fitness. Fitness is calculated from the objective function or specific optimization problem. The particle's velocity vector determines the direction and distance of the particle's movement. The traditional particle swarm algorithm steps are described as follows:

a) Initialization：The particle position and the velocity vector of particle are randomly generated within a prescribed range.

b) Iterative optimization: The fitness value of the particle is calculated and compared to find and record the position of the current optimal solution. According to the optimal position of the individual and the optimal position of the population, the speed and position of the particle are updated to search the entire solution space.

c) Algorithm termination：When the condition is met, the loop is terminated and the optimal solution is returned.

Among them，the velocity updating formula of particle is:

$$V_{id}^{k+1} = \omega V_{id}^{k} + c_1 r_1 \left(P_{id}^{k} - X_{id}^{k}\right) + c_2 r_2 (P_{gd}^{k} - X_{id}^{k}) \tag{1}$$

The position updating formula of particle is:

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \tag{2}$$

In the formula, ω is the inertia weight defaults to 1. Vid is the speed of the particle. X=(X1,X2,…,Xn) represents a population of n particles. Xi=(Xi1,Xi2,…,XiD)T indicates the position of the i particle in the D-dimensional search space. k is the number of iterations. c1 and c2 are non-negative constants called acceleration factors. R1 and r2 are random numbers in the interval of [0,1].

2.2 Dynamic inertia weight.

The dynamic inertia weight indicates that ω changes as the number of iterations k changes. By comparison, this paper use the inertia weight formula put forward in literature [8]. The specific formula:

$$\omega = \omega_s - (\omega_s - \omega_e)\left(\frac{k}{T_{max}}\right)^2 \tag{3}$$

Compared to fixed inertia weights, dynamic inertia weights can search for a larger range at the beginning of the search and a more precise search at the end of the search. Therefore, the global search ability and search accuracy of the original algorithm can be improved. However, when dealing with multimodal functions, the algorithm is still easy to fall into local optimum. So Gaussian Disturbance is added on the basis of dynamic inertia weight to further improve the search ability of the algorithm. This enables the algorithm to jump out when it is stuck in the local optimum and continue to search for the global optimum.

2.3 Dynamic inertia weight particle swarm optimization algorithm based on Gaussian Disturbance.

Aiming at the shortcomings of NPSO algorithm in searching ability, this paper proposes a dynamic inertia weight particle swarm optimization algorithm based on Gaussian Disturbance (NGPSO). The core idea of the algorithm is to enhance the search ability of the algorithm by adding Gaussian Disturbance to the position update formula. The improved algorithm not only has the ability to jump out of local optimum but also improves the search accuracy. The new location update formula:

$$X_{id}^{k+1} = X_{id}^{k}(1 + \varepsilon) \tag{4}$$

ε is a random number obeying a normal distribution. Different from most improved algorithms, this paper does not increase Gaussian Disturbance in each iteration of the loop. Gaussian Disturbance is only added if the original algorithm performs iterative update times more than 10 times and the global optimal value has not changed. This makes the algorithm not blindly disturbed during the process, which largely retains the advantages of the original algorithm. It not only reduces the time complexity of the algorithm but also improves the efficiency and accuracy of the algorithm. The specific steps of the NGPSO algorithm are as follows:

Step 1 Initializes the position and velocity of the particle

Step 2 Calculates the particle fitness value, and compares the individual extremum and the group extremum

Step 3 Update the position and velocity of the particles according to formula (1)(2)(3)

Step 4 Determines whether the global optimal value has not changed 10 times. If it changes, go directly to Step 6. If there is no change, go to Step 5

Step 5 Carries out Gaussian Disturbance to the current position of the particle according to formula (4)

Step 6 Determines whether the termination condition is satisfied, and if it is satisfied, ends the loop, and if not, returns to step 2

3. Simulation experiment

In order to verify the performance of the algorithm, this paper compares the three algorithms NPSO, GPSO and NGPSO. GPSO is a Gaussian Disturbance particle swarm optimization algorithm with default inertia weight of 1. In the experiment, the population size was set to 20, $\omega_s = 0.9$,$\omega_e = 0.4$, both c1 and c2 are set to 1.49445 and the number of iterations is 300. The test function is as follows:

$$\text{f1(x)} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad \text{x} \in [-600,600]$$

$$\text{f2(x)} = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \qquad \text{x} \in [-2.048,2.048]$$

$$\text{f3(x)} = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + \text{e} \qquad \text{x1, x2} \in [-5,5]$$

$$\text{f4(x1,x2)} = [x_1^2 - 10\cos(2\pi x_1)] + [x_2^2 - 10\cos(2\pi x_2)] + 20 \qquad \text{x1, x2} \in [-5.12,5.12]$$

$$\text{f5(x1,x2)} = \frac{sin\sqrt{x_1^2+x_2^2}}{\sqrt{x_1^2+x_2^2}} + e^{\frac{cos2\pi x_1 + cos2\pi x_2}{2}} - e \qquad \text{x1, x2} \in [-2,2]$$

Analysis of experimental results. The experimental results are shown in the figure, and Table 1

records the specific values of the experimental results.

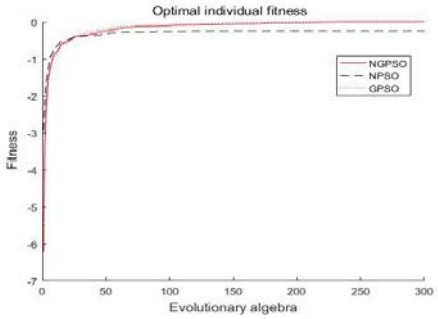

Fig.1 f1 function

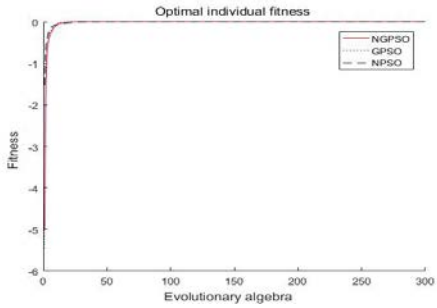

Fig.2 f2 function

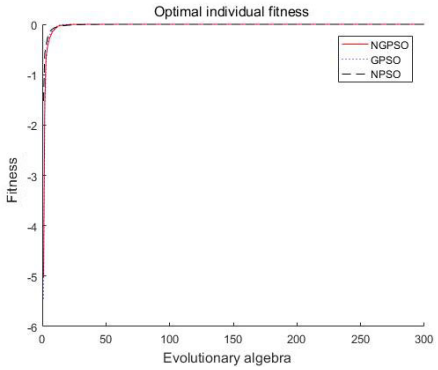

Fig.3 f3 function

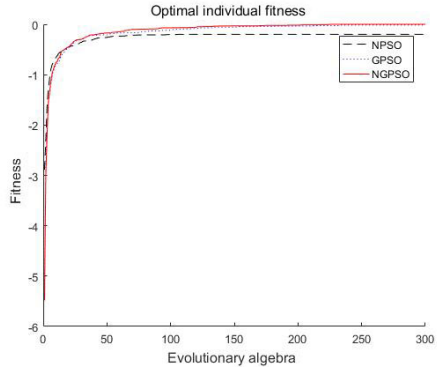

Fig.4 f4 function

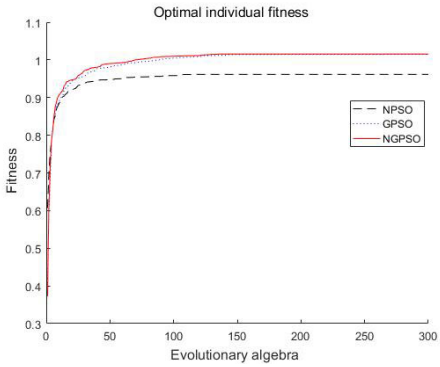

Fig.5 f5 function

Table 1 Experimental result table

Function	Optimal fitness	Algorithm	Fitness
f1	0	NPSO	-0.2487
		GPSO	-0.0029
		NGPSO	-2.6864e-05
f2	0	NPSO	-7.0875e-29
		GPSO	-5.1835e-31
		NGPSO	-2.1187e-33
f3	0	NPSO	-2.2027e-15
		GPSO	-0.0034
		NGPSO	-3.3867e-05
f4	0	NPSO	-0.1990
		GPSO	-0.0132
		NGPSO	-1.3208e-04
f5	1.0156	NPSO	0.9615
		GPSO	1.0148
		NGPSO	1.0155

The experimental data is averaged from 100 experiments. It can be seen from the experimental results that the improved NGPSO algorithm has stronger global search ability and is less likely to fall into local optimum when dealing with multimodal function (f1, f4, and f5) whose local maximum values are close to global maximum value. It also has better stability. When dealing with the unimodal function, although the Gaussian Disturbance has an impact on accuracy, NGPSO also has a good ability to find the result. NPSO is easy to fall into local optimum when dealing with multimodal functions whose local maximum value tends to the global maximum value, but it has better performance in the unimodal function (f3). GPSO has improved in search ability, but GPSO is still not as good as NGPSO in optimizing accuracy. In summary, the NGPSO algorithm improves the PSO algorithm, which is prone to premature convergence, falls into local extremum, and has low accuracy. It has achieved better results in experiments.

4. Summary

This paper adds Gaussian Disturbance to the dynamic inertia weight particle swarm optimization algorithm. Let the algorithm have the ability to jump out of the local optimal solution when it falls into the local optimum, which greatly improves the global search ability of the algorithm. The improved algorithm has better performance in multimodal function. Different from most improved algorithms, the Gaussian Disturbance added in this paper is a conditional disturbance in the optimization process, not a blind disturbance. This makes the algorithm both improved in search abilities and search accuracy.

References

[1] Kennedy J, Eberhart R C. Particle swarm optimization [C]// Proceedings of the IEEE Conference on Neural Networks, IV. Perth, Australia: IEEE Press, 1995: 1942-1948.

[2] Long Xue,Jun Cai,Jing Li,Muhua Liu. Application of Particle Swarm Optimization (PSO) Algorithm to Determine Dichlorvos Residue on the Surface of Navel Orange with Vis-NIR Spectroscopy [J]. Procedia Engineering, 2012,29.

[3] Javad Sadeghi,Saeid Sadeghi,Seyed Taghi Akhavan Niaki. Optimizing a hybrid vendor-managed inventory and transportation problem with fuzzy demand: An improved particle swarm optimization algorithm [J]. Information Sciences, 2014,272.

[4] V.K. Patel,R.V. Rao. Design optimization of shell-and-tube heat exchanger using particle swarm optimization

technique [J]. Applied Thermal Engineering, 2010, 30(11).
[5] Li Yurong, Wang Ying. Improved Particle Swarm Optimization Based on Lévy Flights [J]. Journal of System Simulation, 2017, 29(08):1685-1691+1701.
[6] Xu Xiaobo, Zheng Kangfeng, Li Dan, Wu Bin, Yang Yixian. New chaos-particle swarm optimization algorithm [J]. Journal on Communications, 2012, 33(01):24-30+37.
[7] Wang Wenyi, Qin Guangjun, Wang Ruoyu . Research on Genetic Algorithm Based on Particle Swarm Algorithm [J]. Computer Science, 2007(08):145-147.
[8] Sun Linyan. A new improved particle swarm Optimization [D]. Dalian Maritime University, 2008.