Optimization of PSO Algorithm Based on Adaptive Inertia Weight and Escape Strategy

Lili Gu¹, Yong Liu¹* and Jiaqi Zhen¹

Heilongjiang University, college of Electronical and Information Engineering, 000000 1415742975@qq.com

Abstract. For Particle Swarm Algorithm uses fixed inertia weights, weaken too slow and extremely easy to fall into local optimum so I propose an adaptive inertia weight method, this inertia weight has a larger weight in the early stage, can increase the global search capability and avoid algorithm falling into local optimum, in late period, this inertia weight has a smaller weight and can increase algorithm local search ability and overall increase convergence rate. But during the optimization process, There are some more complicated functions that will still fall into the local minimum even using this adaptive inertia weight. In order to avoid the occurrence of premature phenomenon so use reverse escape strategy, apply new optimization methods to the basic particle swarm algorithm, the application of several basic functions proves that the proposed optimization method has higher precision and convergence speed.

1. Introduction

Particle swarm optimization is a swarm intelligence optimization algorithm, the PSO algorithm have less parameters and high robustness. But the algorithm also has obvious flaws, it is extremely easy to fall into local minimum.

In order to improve the convergence speed and convergence precision of PSO algorithm, this paper projects an adaptive inertia weight and embeds the method of judging early maturity stagnation. Once the premature phenomenon is detected, the escape strategy is used to increase the diversity of particle swarm.

2. Basic particle swarm optimization

The PSO algorithm is to simulate the bird group foraging to achieve the optimal individual of the particle group. The PSO algorithm first initializes a group of particles in a solvable space. Each particle represents a potential optimal solution to the extreme value optimization problem. The PSO operator updates the speed and position information based on the previous generation particles, using position, velocity and fitness value to represent the characteristics of the particle and obtain a new population.

During each iteration, the particle updates its speed and position through individual extremum and global extremum. The formula is updated as follows:

\[ V_{id}^{k+1} = w V_{id}^k + c_1 r_1 ( P_{id}^k - X_{id}^k ) + c_2 r_2 ( P_{id}^k - X_{id}^k ) \]

\[ X_{id}^{k+1} = X_{id}^k + V_{id}^{k+1} \]

In this formula, \( w \) is the inertia weight; \( d = 1, 2, \cdots, D \); \( i = 1, 2, \cdots, n \); \( k \) is the current number of iterations, \( V_{id} \) is the speed of particle, \( c_1, c_2 \) are non-negative constant; \( r_1 \) and \( r_2 \) are
random number distributed between [0,1].

3. Improvement of PSO algorithm

3.1 Dynamic adaptive inertia weight

In the basic PSO algorithm, the inertia weight generally takes a fixed value between [0.4, 0.9]. In a large number of simulation experiments, it is found that the PSO algorithm has early convergence, that is, the algorithm converges slowly and is easy to fall into local optimum. In order to jump out of the local search optimal solution in time, the inertia weight should be adaptively changed according to the evolution speed and aggregation degree of the particle swarm to dynamically adjust the particle motion state to avoid falling into local optimum.

Using a dynamic adaptive inertia weight based on objective function:

\[ \lambda(k) = \frac{1}{N} \sum_{i=1}^{N} \left( f_{\text{gbest}}(PS_i(k)) - f_{\text{zbest}}(PS_i(k)) \right) \]

\( k \) is the number of times the particle is updated, \( f_{\text{gbest}}(PS_i(k)) \) is the corresponding individual extremum fitness value of the \( i \)-th particle at the \( k \)th update position, \( f_{\text{zbest}}(PS_i(k)) \) is the extreme value fitness value corresponding to the optimal particle at the \( k \)th update position, then the inertia weight can be obtained by the following formula:

\[ w = \exp\left(-\lambda(k)/\lambda(k-1)\right) \]

Every time you update, since the value of \( \lambda(k) \) changes with the value of the fitness function, the blindness of the inertia weight change is reduced, since the information of the fitness function value is fully applied in the inertia weight, the search direction of the algorithm is instructive and the individual particles move quickly to the high-quality area. At the same time, the inertia weight is not blindly reduced, but is determined according to the calculation result of the specific objective function. The faster \( k \) is reduced, the larger the value of \( w \) is, indicating that the adaptive function values of the individual particles differ greatly from each other. In order to search for the global optimal value more quickly, it is necessary to maintain the diversity of the particle group. Larger \( w \) values speed up search, when approaching the optimal value point, the value of \( w \) does not change much each time, that is, \( k \) is small, and the algorithm can be prevented from repeatedly oscillating near the extreme point. The dynamic adaptive inertia weight proposed above can effectively balance the exploration and development capabilities of the PSO algorithm and improve the efficiency of the algorithm.

3.2 Escape strategy

In the particle swarm iterative process, if the global optimal solution is invariant for successive \( M \) generations, it indicates that the algorithm is likely to have fallen into local minimum values. Therefore, the global optimal solution can be used as the sign of early maturity stagnation. In this optimization algorithm, by observing the simulation image, it is found that if the global optimal solution is unchanged for 30 consecutive generations, it is very likely that the phenomenon of "premature" has already occurred, so \( M \) is 30.

The escape strategy is used to greatly improve the convergence speed and optimization precision of the algorithm. The expression of particle escape is as follows:

\[ X_{id}^{k+1} = \text{rand} \times X_{id}^{k} \times (1 - X_{id}^{k})' \]

\text{Rand} is a random number between [0, 1].

The specific steps of the NCPSO algorithm implementation:

**Step1** Initialize the position and velocity of particles in a particle swarm

**Step2** Calculate the fitness value of each particle

**Step3** For each particle, compare its fitness value with the fitness value of the best position \( P_i \)


experienced. If it is better than the latter, use it as the best position of the current particle and adapt it to the global experience. The best fit value of position $P_g$ is compared, and if it is better than the latter, it is taken as the current global best position.

**Step 4** If the $P_g$ consecutive has not changed for $M$ generation then go to **Step 5**, otherwise go to **Step 6**.

**Step 5** Use the escape strategy to generate a new generation of particle swarms and turn to **Step 7**.

**Step 6** Update the velocity of the particle according to equation (1), where $w$ is according to an adaptive inertia weight

**Step 7** Determine whether the termination condition is satisfied. If it is satisfied, the algorithm ends. Otherwise, go to **Step 2**.

4. **Experimental process and analysis of results**

In order to verify the performance of the improved PSO algorithm based on adaptive inertia weight and escape strategy, this paper selects six test functions to test PSO (inertia weight $w$ takes fixed value 0.85), LPSO (inertia weight $w$ linear value), CPSO (inertia weight $w$ Non-linear value), and NCPSO (PSO algorithm based on adaptive inertia weight and inverse strategy proposed in this paper). Each test function runs 30 times. The specific experimental data settings are as follows: population size $N=20$, the number of iterations is $T=100$.

| Function name | Function expression | Search interval | Theoretical extremum |
|---------------|---------------------|-----------------|----------------------|
| Sphere $f_1$  | $f_1(x) = \sum_{i=1}^{D} x_i^2$ | $[-100,100]^D$ | 0 |
| Schwefel $f_2$ | $f_2(x) = \sum_{i=1}^{D} |x_i| \prod_{i=1}^{D} |x_i|$ | $[-10,10]^D$ | 0 |
| Ackley $f_3$  | $f_3(x) = -20\exp\left(-0.2\sqrt{\sum_{i=1}^{D} x_i^2 / D}\right) - \exp\left(\sum_{i=1}^{D} \cos(2\pi x_i / D)\right) + 20 + e [-32, 32]^D$ | $[-100, 100]^D$ | 0 |
| Shaffer $f_4$ | $f_4(x) = \sum_{i=1}^{D} \left( x_i^2 + \frac{1}{1 + 0.001(x_i^2 + y_i^2)} \right)$ | $[-100, 100]^D$ | 0 |
| Exponential $f_5$ | $f_5(x) = -\exp(-0.5 \sum_{i=1}^{D} x_i^2)$ | $[-1.0]^D$ | -1 |
| Duadric $f_6$ | $f_6(x) = \sum_{i=1}^{D} \left( \sum_{j=1}^{D} x_j \right)^2$ | $[-100,100]^D$ | 0 |

| Test function | PSO | LPSO | CPSO | NCPSO |
|---------------|-----|------|------|-------|
| $f_1$ | 1.005e-11 | 0.0851 | 3.909e-30 | 0.0502 | 4.95e-33 | 0.0259 | 3.99e-40 | 0.0028 |
| $f_2$ | 3.249e-05 | 0.0291 | 2.108e-14 | 0.0432 | 1.135e-16 | 0.0534 | 5.07e-21 | 0.0256 |
| $f_3$ | 1.255e-04 | 0.2282 | 4.441e-15 | 0.1683 | 8.882e-16 | 0.1224 | 8.882e-16 | 0.0774 |
| $f_4$ | 3.127e-03 | 3.138e-03 | 3.127e-03 | 3.229e-03 | 3.127e-03 | 3.144e-03 | 2.467e-10 | 5.496e-04 |
Analysis Table 1 shows that for the single peak function \( f_1 \) (Sphere functions), compared with the previous three PSO algorithms, the NCPSO algorithm converges faster and the convergence accuracy reaches \( 10^{-40} \); For the function \( f_2 \) (a function), this function is proposed by Schwefel and is considered to be a more classical test function. The function independent variable has epistasis, so its gradient direction does not change along the axis direction, and it has higher difficulty in searching. It turns out that the optimization precision of this function is low, but based on the NCPSO algorithm proposed in this paper, the order of magnitude of the optimization precision is reached, which is a certain improvement compared with the first three PSO algorithms. For the function \( f_3 \) (Ackley function), it can be seen that the minimum values searched by the CPSO and NCPSO algorithms are the same and the accuracy is not high; For the function \( f_4 \) (shaffer function), the function only has a local minimum point, but the optimization precision is not high. Among them, the optimization precision of PSO, LPSO and CPSO is \( 10^{-8} \), and the optimization of the optimized NCPSO algorithm is performed, the accuracy is \( 10^{-10} \); For the function \( f_5 \) (Exponential function), the function is a two-dimensional function, the local minimum value is -1, and the optimization precisions of the four algorithms PSO, LPSO, CPSO, and NCPSO all reach -1; For the function \( f_6 \) (Duadric function), the local minimum value of the function is 0, the optimization precision of PSO, LPSO, CPSO, NCPSO is high, and the optimization precision of NCPSO reaches \( 10^{-83} \).

### Table 1

| \( f_5 \) | -1 | -0.997 | -1 | -0.9843 | -1 | -0.9996 | -1 | -0.9972 |
|---|---|---|---|---|---|---|---|---|
| \( f_6 \) | 1.004e-21 | 2.425e-02 | 2.017e-59 | 7.41e-03 | 5.469e-66 | 1.697e-03 | 1.534e-83 | 3.668e-05 |
Conclusion: In order to improve the search ability of PSO algorithm, two improvements are made in this paper. One is to refer to the fitness function value of the particle, take the adaptive inertia weight, and the second is to use the escape strategy to guide the particle swarm search for more regions. The test is performed using six basic functions. The experimental results show that the NCPSO algorithm can enhance the optimization ability and the convergence speed is faster.

References
[1] DANG, T. (2019) Optimization algorithm of spider monkey with dynamic adaptive inertia weight. Computer Engineering and Applications, 55(14): 40-47.
[2] Zheng, Q. (2018) Computer Engineering and Applications. NCI Publishing, Beijing.
[3] Wang, X. (2011) Particle Swarm Optimization. In: shoupeng, C. (Eds.), Analysis of 43 cases of MATLAB neural network. Beihang University Press, Beijing. pp. 306-312.
[4] Ling, J., (2019) Economic Optimization Operation Analysis of Microgrid Based on Improved Particle Swarm Optimization. In: Proceedings of the 2019 International Conference on Modeling. Sanya pp. 165-178.
[5] Kaiyou Lei. (2006) Particle swarm optimization and its application. http://www.wanfangdata.com.cn/details/detail.do?_type=degree&id=Y1015940