HYBRID LINEAR AND NONLINEAR WEIGHT PARTICLE SWARM OPTIMIZATION ALGORITHM

JIAN-RU ZHENG[1,2], GUO-LI ZHANG[1], HUA ZUO[1]

[1]Department of Mathematics and Physics, North China Electric Power University, Baoding 071003, China
[2]Baoding University, Baoding 071000, China
E-MAIL: bdxyzwx@163.com, zhangguoli@ncepu.edu.cn, zuohua_870317@126.com

Abstract:

The inertia weight is an important parameter in the Particle Swarm Optimization algorithm, which controls the degree of influence of the contemporary speed to the next generation and plays a role of balancing global search and local search. In the iteration process, the inertia weight will decrease nonlinearly at the early stage and decrease linearly at the later stage. The improved algorithm will effectively prevent premature convergence of the algorithm. The simulation results show that the improved algorithm is superior to the particle swarm optimization algorithm of the linear decreasing weight.

Keywords:

Particle Swarm Optimization algorithm; Inertia weight; Nonlinear

1. Introduction

Particle swarm optimization algorithm (PSO) is a stochastic, population-based global search algorithm. The PSO has drawn many researchers' attention due to its capability of finding the near global optimal solutions without putting restrictions on the characteristics of the target functions. But the PSO also suffers from premature problem. Some efforts have been given to solve this problem. Angeline introduced the concept of selection into PSO, which improved the global search capability [1]. Higashi et al. proposed the particle swarm optimization with mutation operator. The mutation operator increased the diversity of population and reduced the premature possibility [2]. Gao Ying introduced mechanism of the immune information processing into the particle swarm optimization to increase the diversity of the population, and improved particle algorithm's global search capability [3]. An adaptive particle swarm optimizer with group fitness variance was presented by Lv Zhensu, and the variation operation enhanced the ability of particle swarm optimization algorithm jump out of local optima [4]. Sivanandam et al. proposed hybrid particle swarm optimization with dynamically varying inertia. The new algorithm can balance the global exploration and local exploration[5]. Pooranian presented a hybrid PSO and GELS algorithm which increase the local search ability[6]. Mingzhi Chen et al. proposed the modified PSO based on expectations model and used the modified PSO to solve interval linear programming problem[7]. Jiao et al studied a nonlinearly increasing strategy of inertia weight w with the increasing of iterations[8]. Iranmanesh et al. studied RBF neural network and particle swarm algorithm and used it to predict energy demand consumption in Turkey and U.S. [9]. Inanloo et al. proposed a hybrid model of TOPSIS and PSO to find optimal solution of large-scale multi-objective optimization problems[10].

This paper will analyze the characteristics of particle swarm optimization algorithm and give a new particle swarm optimization which is not difficult to understand and does not increase computation burden. The new algorithm uses inertia weight of hybrid linear and nonlinear. Furthermore, this improved algorithm will increase global search capability, and maintain strong local search ability at the same time.

The rest of this paper is organized as follows. Section 2 describes briefly the base PSO algorithm, and gives hybrid linear and nonlinear weight particle swarm optimization algorithm. In Section 3, the experimental study on 4 typical benchmark functions is done. The comparison of the experimental results of the PSO algorithm proposed in literature [11] and the proposed new method are given. Finally, the conclusions are drawn in Section 4.

2. Hybrid linear and nonlinear weight particle swarm optimization algorithm

2.1. PSO Algorithm

Particle swarm optimization algorithm proposed by

Kennedy and Eberhart [12-13] is a stochastic, population-based global search algorithm and it searches for the best solution by simulating the movement and flocking of birds. The algorithm works by randomly initializing a flock of birds over the searching space, where every bird is called as a "particle". At each iteration, each particle can adjust its velocity vector based on its momentum and the influence of its best position as well as the best position of the best individual. Then, the particle flies to a new position. Suppose that the search space is n -dimensional, and then the position and velocity of particle i are represented by $x_i=(x_{i1},x_{i2},\cdots,x_{in})$ and $v_i=(v_{i1},v_{i2},\cdots,v_{in})$ respectively. Velocity v_i is bounded between its lower and upper limits. The best position of the i -th particle is denoted by $p_l=\{p_{l1},p_{l2},\cdots,p_{ln}\}$, The best position achieved so far is described by $p_g=\{p_{g1},p_{g2},\cdots,p_{gn}\}$. Each particle's velocity and position are updated by using the following two formula.

$$v_{\mathrm{i}}^{\mathrm{t+1}}=v_{\mathrm{i}}^{\mathrm{t}}+c_1\times r_1(p_l^t-x_i^t)+c_2\times r_2(p_g^t-x_i^t) \qquad (1)$$

$$x_{\mathrm{i}}^{\mathrm{t+1}}=x_{\mathrm{i}}^{\mathrm{t}}+v_{\mathrm{i}}^{\mathrm{t}} \qquad (2)$$

where, $i=1,2,\cdots,m$, m is a number of particles in the particle swarm; t is the number of generations (iterations); v_i^t, v_i^{t+1} are the velocity of particle i at the t -th and the $t+1$ -th generation respectively; x_i^t, x_i^{t+1} are the position of particle i at the t -th and the $t+1$ -th generation respectively; p_l^t, x_g^t are personal best of agent i and group best; c_1 and c_2 non-negative constant are called as acceleration coefficients which control the maximum step size. r_1 and r_2 are independently uniformly distributed random variables in a range of $[0,1]$.

Shi. Y. and Eberhart. R [7] calculated the new velocity according to the following formula (3).

$$v_{\mathrm{i}}^{\mathrm{t+1}}=wv_{\mathrm{i}}^{\mathrm{t}}+c_1\times r_1(p_l^t-x_i^t)+c_2\times r_2(p_g^t-x_i^t) \qquad (3)$$

where w is called the inertia weight. The introduction of the inertia weight particle swarm optimization algorithm has been known as the standard particle swarm algorithm.

Obviously, the inertia weight controls the previous generation of particle speed on the speed of contemporary, and the inertia weight value directly affects the algorithm's global convergence and local convergence. At the beginning, the inertia weight is taken as a constant. Later, experiments show that the dynamic weight is able to obtain better optimization results than a fixed value. The introduction of the inertia weight adjusts algorithm capability for the local and global search from different sizes, and improves the performance of the particle swarm optimization algorithm. The larger inertia weight facilitates a global search; while the smaller inertia weight facilitates a local search. However, how to determine the inertia weight values is still an important issue.

In order to increase further the algorithm optimization capability, some scholars researched the inertia weight values and the range, and proposed some valuable inertia weight values method. Zhang Liping, etc. proposed a random inertia weight values strategy in literature [14], and algorithm optimization obtained a better balance, so it solved the multimodal function optimization. Wang Qifu [15] proposed a dynamic inertia weight's particle swarm optimization algorithm, the inertia weight calculation introduced engineering index e, namely $w(t)=e^{\frac{-\partial^t}{\partial^{t-1}}}$, which enhanced the heuristic of the search direction, but the algorithm was more difficult to achieve. Zhou Min [16] proposed particle swarm optimization algorithm based on the distance measure optimal K, and introduced distance concept into particle. According to distance of the particle and the optimal particle to adjust the inertia weight of the particle, its shortcoming was that parameter K value cannot be determined in practical application. Literature [11] presented a linear decreasing inertia value strategy algorithm which was better than the original algorithm. But it can not be well adapted to the complicated optimization problem in practical application. Literature [17] proposed a nonlinear weight particle swarm optimization algorithm, and the linearly decreasing inertia weight strategy formula had been improved, which enhanced the algorithm out of local optimal solution ability, and improved the global search ability.

2.2. Hybrid linear and nonlinear weight particle swarm optimization algorithm

From the optimization process of the particle swarm optimization algorithm, it is mainly to expand the search space in the first stage, which requires a larger inertia weight to make the particles have a large speed. The later stage is mainly searching near the optimal solution, which requires the smaller inertia weight to make the particles having smaller speed. This paper draws on the idea of reference [11]. In order to improve the search capability of the particle swarm algorithm in solving complex optimization problems of high-dimensional and multi-modal, we give hybrid linear and nonlinear weight particle swarm optimization algorithm (HLNWPSO). In the previous search stage, the inertia weight decreases nonlinearly from the initial value with the iterations. It guarantees the ability of the algorithm to search optimal solution in new areas in the initial stage. The inertia weight of the later stage decreases linearly with iterations, which makes the algorithm a fast convergence. Based on this idea, this paper presents the adaptive inertia weight adjustment formula as follows:

$$w(t)=\begin{cases} w_{\min}+(w_{\max}-w_{\min})\times h_1(t), & t<k \\ w_{\min}+(w^{'}-w_{\min})\times h_2(t), & t\geq k \end{cases} \qquad (4)$$

where t is iteration number, $h_1(t)$ is nonlinear function, $h_2(t)$ is linear function, $w_{\max}$, $w_{\min}$ are respectively the maximum inertia weight and the minimum inertia weight, and w' is the value of the inertia weight at the end of the previous stage .

2.3. Algorithm process

In the current research, the process of HLNWPSO algorithm technique can be summarized as follows:

Step 1: Initialize the particle swarm, set up the learning factors c_1 , c_2 and maximal iteration number M, and change iteration number K in the inertia weight formula. The current iteration number is set as $t=1$. m particles are randomly generated and named as $x_1^t, x_2^t, \cdots, x_m^t$, which composes the initial population $X(t)$. The initial update speed of particles $v_1^t, v_2^t, \cdots, v_m^t$ are also randomly generated.

Step 2: Evaluate the population and calculate the fitness of each particle. The optimal position of each particle is stored in p_l^t , and the best particle position of the entire population is stored in p_g^t .

Step 3: Calculate inertia weight according to formula (4).

Step 4: According to the formula (3) and (2), update the speed and position of the particle to generate the new population $X(t+1)$.

Step 5: Calculate the fitness value of each particle after the update. Compare the fitness value of each particle after updating with the fitness value of the optimal position previously. If the updated position is better, then take it as the optimal position p_l^t of the particle. Otherwise, p_l^t does not change.

Step 6: Compare the fitness value of every particle with the fitness value of the optimal position previously. If the former is better, then update p_g^t . Otherwise, p_g^t does not change.

Step 7: Check the termination condition. If it does not meet the termination condition, then $t=t+1$, and go to step (3). If it meets the termination condition, then stop the iteration, and output the optimal solution.

3. Experiment study

In order to test the HLNWPSO algorithm introduced in this paper, we make simulation experiments using the test function in reference [11]. And compare ' HLNWPSO ' with the particle swarm optimization algorithms (LDIW-PSO) with linearly decreasing inertia weight [7].

3.1. Test function and parameter selection

The two algorithms use the same calculation parameters: $c_1=c_2=2$, $w_{\max}=0.9$, $w_{\min}=0.4$, the initial inertia weight is 0.9, and particle swarm scale is $m=20$. The dimension of the optimization problem is $n=5$, and the maximum number of iterations is $M=2000$. For HLNWPSO algorithm, there is $h_1(t)=e^{\frac{-16t^3}{M^3}}$ ($t<k$), $h_2(t)=\frac{M-t}{t}$ ($t\geq k$), $k=\frac{M}{2}$ in inertia weight adjustment formula. The optimization interval and the optimal value of the test functions are shown in Table 1.

TABLE 1. THE BENCHMARK FUNCTION

Function name	*Function expression*	*Optimization interval*	*Optimal value*
f_1	$f_1(x)=\sum_{i=1}^{n-1}(100(x_{i+1}^2-x_i)^2+(1-x_i)^2)$	$[-10,10]$	0
f_2	$f_2(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\frac{x}{\sqrt{i}}+1$	$[-600,600]$	0
f_3	$f_3(x)=\sum_{i=1}^{n}x_i^2$	$[-100,100]$	0
f_4	$f_4(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	$[-20,20]$	0

f_1 is a non-convex function, which reaches its minimum in $x^*=(1,1,\ldots,1)$. f_2 is a multi-peak function with the global minimum point $x^*=(0,0,\ldots,0)$, and there are many local minima. f_3 is a typical spherical and unimodal function. f_4 is a multi-peak function with a large number of local extreme points, and it is difficult to get optimal solution by generaly algorithm.

With these four functions as the objective functions, the purpose of optimization algorithm is to find the minimum value of each function in the solution space. Determine the fitness value of every particle according to its function value. The smaller function value is, the higher fitness value is. Seek the global optimal particle through comparing the functions of particles. When the algorithm ends, the optimal solution is the particle owning the highest fitness value. The performance of the algorithm can be reflected by comparing the optimal results with minimum value of the text function.

3.2. Experimental results and analysis

In order to evaluate the performance of the HLNWPSO algorithm, we use the following evaluation index: the optimal result (OR), the average optimal solution (AOR), the success rate (SR), and the calculation time (CT). OR shows the best results in 50 experiments; the AOR shows the 50 experiments obtained the optimal value of the arithmetic average. SR shows the percentage of experiment numbers in 50 experiments, in which the error between optimal value and the minimum of the function is less than 10^{-4}. MCT shows the time used in the algorithm to search the minimum function value. MBF can reflect the accuracy of a given algorithm. SR reflects the reliability of the algorithm. The MCT can evaluate the complexity of the algorithm. For each test function, the algorithm is running repeatedly 50 times. Algorithm is programmed in Matlab7.0. The operation results of HLNWPSO algorithm and LDIW-PSO algorithm are given in Table 2.

TABLE 2. THE OPTIMIZATION RESULTS

Function name	*Optimization algorithm*	*OR*	*AOR*	*SR(%)*	*ST(S)*
f_1	LDIW-PSO	0.0124	0.0723	40	0.24
	HLNWPSO	$2.0104e^{-8}$	$2.5987e^{-7}$	100	0.36
f_2	LDIW-PSO	0.096	0.242	65	0.46
	HLNWPSO	$1.0276e^{-8}$	$1.1433e^{-7}$	100	0.76
f_3	LDIW-PSO	$8.4738e^{-6}$	$5.890e^{-6}$	100	0.02
	HLNWPSO	$1.1170e^{-9}$	$4.8811e^{-7}$	100	0.04
f_4	LDIW-PSO	0.0051	0.0519	48	0.08
	HLNWPSO	$2.7197e^{-5}$	$9.1433e^{-4}$	100	0.36

It can be seen from Table 2 that to the unimodal function, the algorithm presented in this paper is significantly higher than LDIW-PSO in the calculation accuracy and success rate. For multimodal function, they are all successful in searching the optimal solution, but LDIW-PSO prematurely gets into a local optimum. But compared to LDIW-PSO, the algorithm we give is time-consuming and has the huge amount of calculation.

4. Conclusion

This paper gives the weight of a combination of linear and nonlinear adaptive particle swarm optimization. In the early stage of the particle swarm search, the inertia weight decreases nonlinearly from the initial value with the iteration, to ensure the ability of exploring new areas to find the optimal solution in the previous stage. In the later stage of the algorithm, inertia weight decreases linearly with iterations number, and the algorithm has fast convergence. The new improved algorithm, to a certain extent, effectively avoids the algorithm from getting into local convergence. Simulation results show that the new algorithm has higher search precision and global search ability.

Acknowledgement

This work is supported by Hebei Province Science Foundation (A2012502061).

References

[1] Peter J. Angeline, “Using selection to improve particle swarm optimization”, IEEE Congress on Evolutionary Computation – CEC, pp .84-89, 1998.

[2] Higashi. N. H., “Particle Swarm Optimization with Gaussian Mutation”. Institute Of Eleetrieal and Eleetronies Engineers, No.4, pp.72-79, 2003.

[3] Ying Gao, and Shengli Xie, “Immune particle swarm optimization algorithm”, Computer Engineering and Applications, Vol.40, No.6, pp.4-6, 2004.

[4] Zhensu Lv, and Zhirong Hou, “Adaptive mutation particle swarm optimization”, Journal of Electronics, Vol. 32, No.3, pp. 416-420, 2004.

[5] Sivanandam S. N., and Visalaks P., “Multiprocessor Scheduling Using Hybrid Particle Swarm Optimization with Dynamically Varying Inertia”, International Journal of Computer Science and Applications, Vol. 4 , pp. 95-106, 2007.

[6] Pooranian Z., Harounabadi A., Shojafar M., and Mirabedini J., “Hybrid PSO for Independent Task scheduling in Grid Computing to Decrease Make span”， 2011 International Conference on Future Information Technology，Vol. 13，pp. 327-331, 2011.

[7] Mingzhi Chen, Shuili Chen, and Guorong Cai, “Interval Linear Programming Based on Modified PSO and Variation of Intervalvalued Coefficient”, 2011 IEEE International Conference on Intelligent Computing and Intelligent Systems, Vol. 3, pp. 613-617, 2011.

[8] Jiao B., Lian Z., and Gu X., “A dynamic inertia weight particle swarm optimization algorithm”, Chaos, Solitons and Fractals, Vol.37, pp. 698-705, 2008.

[9] Iranmanesh H., Abdollahzade M., and Samiee S., “RBF Neural Networks and PSO Algorithm for Time Series Prediction Application to Energy Demand Prediction”, 2011 IEEE International Conference on Intelligent Computing and Intelligent Systems, Vol.03, pp. 978-982, 2011.

[10] Inanloo B., Zahraie B., and Roozbahani A.， “TPSO: Hybrid Model of TOPSIS and PSO for Solving Multi-Objective Optimization Problems”，2011 IEEE International Conference on Intelligent Computing and Intelligent Systems, Vol.3, pp. 224-229, 2011.

[11] Shi. Y., and Eberhart. R, “A Modified Particle Swarm Optimizer”, Proceedings of IEEE International Conferences Evolution Computation, Anchorage, UK, pp. 69-73, 1998.

[12] Liping Zhang, Huanjun Yu, Dezhao Chen, and Shangxu Hu, “Particle swarm optimization analysis and improvement of the algorithm”, Information and Control, Vol.33, No.5, pp. 513-517, 2004.

[13] Eberhart. R, and Kennedy. J., “A new optimizer using particle swarm theory”, The 6th Int’1 Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995.

[14] Kennedy. J., and Eberhart. R., “Particle Swarm Optimization”, Proceedings of IEEE International Conferences on Neural Networks, Australia, pp. 1942 -1948, 1995.

[15] Qifu Wang, Zhanjiang Wang, and Shuting Wang, “A dynamic inertia weight particle swarm optimization algorithm”, China Mechanical Engineering, Vol. 16, No. 11, pp. 945-948, 2005.

[16] Min Zhou, “Based on the distance K optimization particle swarm optimization algorithm”, Computer Engineering and Application, Vol. 47, No. 15, pp. 43-45, 2011.

[17] Gang Xu, Yuqun Yang, and Huang Xianjiu, “A nonlinear weighting adaptive particle swarm optimization algorithm”, Computer engineering and applications, Vol. 46, No. 35, pp. 49-51, 2010.