A Dynamical Particle Swarm Algorithm with Dimension Mutation

***Jingxuan Wei*[1], *Yuping Wang*[2]**

[1]*School of Science, Xidian University, Xi' an 710071, China*
[2]*School of Computer Science and Technology, Xidian University, Xi' an 710071, China*

Summary
In this paper, a dynamical particle swarm algorithm with dimension mutation is proposed. First, we design a dynamically changing inertia weight ω which can change dynamically based on the speed factor and the accumulation factor. The algorithm with dynamically changing inertia weight can solve complex and nonlinear optimization process that linearly decreasing weight algorithm (LDW) is not adapt. Second, in order to escape from the local optimum, a dimension mutation operator is designed. The degrees of convergence of every dimension are calculated from the beginning of mutation. The dimension of the minimal convergence degree is mutated according to some probability. Finally, the simulation experiments also prove its high efficiency

Key words:
Particle swarm optimization, dimension mutation, inertia weight

1. Introduction

Particle swarm optimization (PSO) originally developed by Kennedy and Eberhart [1] [2] is a population-based algorithm. PSO is initialized with a population of candidate solutions. Each candidate solution in PSO called particle, has associated a randomized velocity, moves through the search space, which is associated with the best solution (fitness) it has achieved so far, p_{best} . Another "best" value tracked by the global version of the particle swarm optimizer is the overall best value, g_{best} . The PSO has been found to be fast in solving nonlinear, non-differentiable, multimodal optimization problems

Comparing with GA, PSO's advantages lie on its easy implementation and few parameters to adjust. But, many problems need to be further researched, such as, how to adjust the parameter ω and how to overcome the original PSO's liability to convergence to the local optimum.

Inertia weight ω is a very important parameter in standard version [3], it can control algorithm's ability of exploitation and exploration. In standard version, ω reduce gradually as the generation increasing. In the searching process, the searching space will reduce gradually as the generation increasing. Recently, Berhart and Shi design a linearly decreasing weight PSO algorithm [3] (LDW), because the searching space reduces step by step, not linearly, so the linearly decreasing weight ω cannot exactly reflect the searching process. Recently, some new algorithms have been developed to improve the property of ω [4] [5].
Like other evolutionary algorithms, the difficulty to escape from local optimum is still existed in PSO. Some researchers use the mutation operator [6] [7] to make the PSO break away from the local optimum. The PSO algorithm with mutation operator not only has great advantages of convergence property, but also can avoid the premature convergence problem.
In this paper, a dynamical particle swarm algorithm with dimension mutation is proposed. Firstly, the speed factor and accumulation factor of the swarm are introduced in the new algorithm, and the inertia weight is formulated as the function of these factors. In each generation, the ω is changed dynamically according to the speed factor and accumulation factor. Secondly, in order to avoid the premature convergence, a dimension mutation operator is presented. The degree of convergence of every dimension is calculated in every generation from the beginning of mutation. Then the dimension of the minimal convergent degree is mutated according to some probability. Finally, simulation results show the efficiency of the new algorithm.

2. The Speed Factor and Accumulation Factor of the Swarm

PSO initialized the flock of birds randomly over the searching space, every bird is called a “particle”. At each generation, each particle adjusts its velocity vector, based on its best solution (p $_{best}$) and the best solution of its neighbors (g_{best}). The original PSO formulae are:

$$V_T = \omega V_{T-1} + c_1 r(.)(p_{best} - p_{present}) + c_2 r(.)(g_{best} - P_{present}) \qquad (1)$$

$$P_{present} = P_{present_{T-1}} + V_T \qquad (2)$$

Where V is the velocity vector, $p_{present}$ is the location vector, ω is the inertia weight in the range [0.1, 0.9]. c_1 and c_2 are positive constants.

If we need to search the global minimum, then $F(g_{best_T}) < F(g_{best_{T-1}})$, F is the objective function. We define $h = \dfrac{F(g_{best_T})}{F(g_{best_{T-1}})}$, $h \in (0,1)$, h is called speed factor, it reflects the evolution speed. When h is small, the evolution speed is fast. After some iterations, h is equal to 1 then we said the algorithm is stagnant or find the global optimum.

The other factor influences the property of algorithm is the accumulation degree of the swarm. We define $s = \dfrac{1}{N \cdot L} \cdot \sum_{i=1}^{N} \sqrt{\sum_{d=1}^{n} (p_{id} - \overline{p_d})^2} \in (0,1)$, where N is the population size, n is the number of variables, L is the length of the maximum diagonal in the search space, p_{id} indicates the dth coordinate of the ith particle, $\overline{p_d}$ indicates the average values of all particles in the dth coordinate. The smaller the value of s, the more centralized the swarm is. When the swarm is centralized, it becomes difficult for the algorithm to break away from the local optimum.

Based on above, we know that ω will change with the speed factor h and the accumulation factor s. When the evolution speed is fast, the algorithm will search in a large space. When the evolution speed is slow, the inertia weight ω will be decreased, so that the algorithm will search in a small space and find the optimum quickly.

If the particles are dispersive, the swarm is not easy to plunge into the local optimum. But when particles are centralized, it becomes easy to plunge into the local optimum.

From above, we know that ω will decrease when the evolution speed is slow and increase when particles are centralized, namely, s is small. So ω can be described as follows:

$$\omega = \omega_0 - h\omega_h - s\omega_s$$

Where, $\omega_0 = 1$, $\omega_h \in (0.4,0.6)$, $\omega_s \in (0.1,0.2)$

3. Dimension Mutation Operator

When swarm is centralized, it is difficult for PSO to break away from the local optimum. In order to overcome the disadvantage, we use the dimension mutation operator. Let $l(d) = \dfrac{\sum_{i=1}^{N} | p_{id} - \overline{p_d} |}{N}, d \in (1,n)$, which indicates the convergence degree in the dth dimension. When the value of $l(d)$ is small, it indicates the particles are centralized in the dth dimension. For every $d \in (1,n)$, we calculate $l(d)$ and find the $d_{\min}$, let $d_{\min} = \arg\min_{d \in [1,n]} l(d)$. Then all particles in the d $_{\min}$ th dimension are mutated according to some probability, the positions of all particles in this dimension are distributed in the range [$l_{\min}, u_{\min}$]

4. The Proposed Algorithm

Step1: Using orthogonal design method [8] to initialize a group of particles, including position and velocity.

Step2: Calculate inertia weight ω according to the expression (3)

Step3: for each particle, compare its fitness and its personal best position p_{best}, if its fitness is better, replace p_{best} with its fitness.

Step4: for each particle, compares its fitness and the global best position g_{best}, if its fitness is better, replace g_{best} with its fitness.

Step5: transform each particle’s velocity and position according to the expression (1) and (2). The new swarm is defined as O_1.

Step6: given constant t_0 , if $t \ge t_0$ (t is the current generation), go to step 7,else go to step2

Step7: calculate $d_{\min}$ and all particles in O_1 are dimension mutated according to the probability p_{rate} .

$x_{id_{\min}} = rand(l_{\min}, u_{\min})$, $x_i \in O_1, i \in (1, N)$

Step8: loop to step2 until a stopping criterion is met, usually a given maximum generations.

5. Simulation and discussion

5.1 Test Function

To evaluate the efficiency of the new algorithm (DPSO), we choose five benchmark minimization problems [9] as follows.

(1) Branin Function (n=2)

$$F_1(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$$

$$-5 \le x_1 \le 10, 0 \le x_2 \le 15$$

The best solution obtained by DPSO is (9.4249, 2.4701), the corresponding function value is 0.3979.

(2) Goldstein-Price Function (n=2)

$$F_2(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

$$-2 \le x_i \le 2, i = 1,2$$

The best solution obtained by DPSO is (0.0001, -1), the corresponding function value is 3.

(3) Six-Hump Camel-back Function (n=2)

$$F_3(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

$$-5 \le x_i \le 5,$$

The best solution obtained by APSO is (-0.08987, 0.712760), the corresponding function value is -1.031626.

(4) Generalized Rastrigin's Function (n=5)

$$F_4(x) = \sum_{i=1}^{5}[x_i^2 - 10\cos(2\pi x_i) + 10],$$

$$-5.12 \le x_i \le 5.12,$$

The best solution obtained by APSO is (1.0e-003 * -0.4291, -0.4277, 0.9562, 0.4816, -0.0400), the corresponding function value is 3.0055e-004.

(5) Sphere Model (n=30)

$$F_5(x) = \sum_{i=1}^{10} x_i^2$$

$$-100 \le x_i \le 100$$

The best solution obtained by APSO is (-0.0361, 0.0125, -0.0069, -0.0006, -0.0025, 0.0140, -0.0108, 0.0146, -0.0021, -0.0223), the corresponding function value is 0.0025.

5.2 Simulation Results

DPSO, FEP [9], CEP [9] and LDW [3] are applied to five testing function optimization problems and then the results are compared, which are given in table1. The changing charts of the best results with increasing of the generations are plotted respectively with LDW in figure 1-5. The DPSO is implemented using MATLAB. The parameters of DPSO are given as follows:

$\omega_h = 0.5, \omega_s = 0.02$, $P_{rate} = 0.01$, $t_0 = 10$, $c_1 = c_2 = 2$.

For each test function, we execute the proposed algorithm DPSO 10 independent runs. We record the mean of the objective functions of the best solutions in every run, denoted by mean and the standard deviation, denoted by Std. The beeline indicates DPSO and the dashed indicates LDW in figure1 to figure5

Table 1

function	Global optimal		DPSO	FEP	CEP	LDW
F_1	0.398	Mean Std	0.3979 0	0.398 1.5e-7	0.398 1.5e-7	0.3979 0
F_2	3.000	Mean Std	3.000 0	3.02 0.11	3.0 0	3.000 0
F_3	-1.0316	Mean Std	-1.0316 0	-1.03 4.9e-7	-1.03 4.9e-7	-1.0316 0
F_4	0	Mean Std	0.0003 0	0.14 0.40	4.08 3.08	0 0
F_5	0	Mean Std	0.0025 0	5.7e-4 1.3e-4	52.2e-4 15.9e-4	0.0002 0

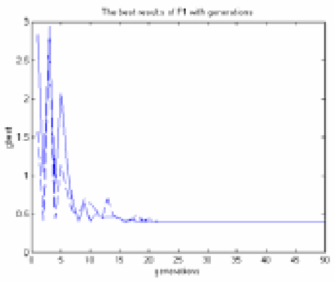

Figure1

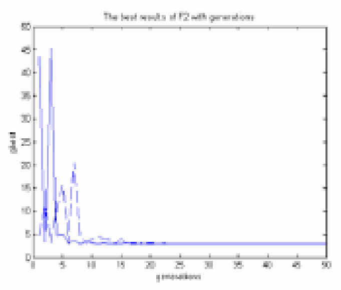

Figure2.

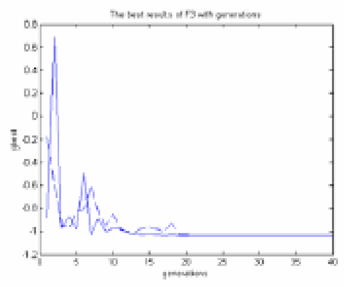

Figure3

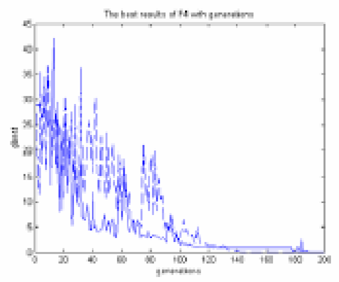

Figure4.

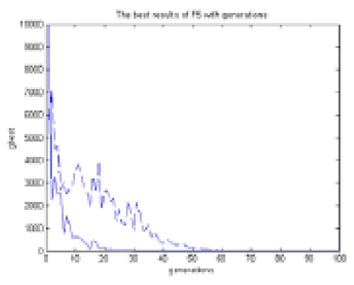

Figure5

It can be seen from table1.that DPSO can get more exact solutions than FEP and CEP for all five benchmark functions. It also can be seen that the DPSO shows considerably better stability than other three algorithms.
From figure1 to figure5, it can be observed that DPSO constringency is faster and better than LDW, especially in earlier stage, the constringency of DPSO is more evident.
The results presented are still preliminary, but show clearly that DPSO is effective to find the global optimum and it is faster than other three algorithms. Work in future is to investigate the influence of P_{rate} to the property of DPSO

6. Conclusions

In this paper, a dynamical particle swarm algorithm with dimension mutation is proposed. Our main contributions: firstly, a dynamically changing inertia weight ω which can change dynamically based on the speed factor and the accumulation factor is designed. The dynamical ω can improve the property of PSO, such as DPSO constringency is faster and better than LDW, especially in earlier stage.

Secondly, in order to escape from the local optimum, a dimension mutation operator is designed. The experiment results indicate that the proposed algorithm is superior to other compared algorithms, especially the evolution speed.

Acknowledgment

This research is supported by National Natural Science Foundation of China (No.60374063).The first author is now pursuing a PhD at Xidian University

References

[1] J. Kennedy, and R. Eberhart. "Particle swarm optimization", in Proceedings of the IEEE International Conference on Neural Networks.IEEE Service Center, Piscataway, NJ, IV: 1995, pp.1941-1948.
[2] J. Kennedy, R. Eberhart, and Y. Shi, "Swarm Intelligence", SanFrancisco: Morgan Kaufmann Publishers, 2001.
[3] Shi Y, Eberhart R, "A Modified Particle Swarm Optimizer" [C]. IEEE Int. Conf. on Evolutionary Computation, Piscataway: NJ, IEEE Service Center, 1998, 69-73.
[4] Clerc M, Kennedy J. "The particle swarm: Explosion, stability, and convergence in a multi-dimensional complex space" [J]. IEEE Transactions on Evolutionary Computation,6(1) 2002: 58-73.
[5] "Adaptive Particle Swarm Algorithm with Dynamically Changing Inertia weight", Journal of Xi'an Jiaotong University. Vol. 39, Oct. 2005.
[6] L. S. Coelho, and R. A. Krohling, "Predictive controller tuning using modified particle swarm optimization based on Cauchy and Gaussian distributions", in Proceedings of the 8th On-Line World Conference on Soft Computing in Industrial Applications. WSC8, 2003.
[7] Guojiang Fu, Shaomei Wang, "A PSO with Dimension Mutation Operator", Engineering Journal of Wuhan University. 4(2005): 79-83.
[8] Y. W. Leung and Yuping Wang, "An Orthogonal Genetic Algotithm with Quantization for Global numerical optimization." IEEE transactions on evolutionary Computation, Vol, 5, No.1, February 2001,pp. 41-53.
[9] Xin Yao, Yong Liu, "Evolutionary programming made faster", IEEE Trans. On Evolutionary Computation, vol.3, no3, July 1999:82-102.