An Improved Particle Swarm Optimization Algorithm

Lin Lu, Qi Luo, Jun-yong Liu, Chuan Long
School of Electrical Information, Sichuan University, Chengdu, Sichuan, China, 610065
lvlin@email.scu.edu.cn

Abstract

A hierarchical structure poly-particle swarm optimization (HSPPSO) approach using the hierarchical structure concept of control theory is presented. In the bottom layer, parallel optimization calculation is performed on poly-particle swarms, which enlarges the particle searching domain. In the top layer, each particle swam in the bottom layer is treated as a particle of single particle swarm. The best position found by each particle swarm in the bottom layer is regard as the best position of single particle of the top layer particle swarm. The result of optimization on the top layer particle swarm is fed back to the bottom layer. If some particles trend to local extremum in particle swarm optimization (PSO) algorithm implementation, the particle velocity is updated and re-initialized. The test of proposed method on four typical functions shows that HSPPSO performance is better than PSO both on convergence rate and accuracy.

1. Introduction

Particle swarm optimization algorithm first proposed by Dr. Kennedy and Dr. Eberhart in 1995 is a new intelligent optimization algorithm developed in recent years, which simulates the migration and aggregation of bird flock when they seek for food[1]. Similar to evolution algorithm, PSO algorithm adopts a strategy based on particle swarm and parallel global random search. PSO algorithm determines search path according to the velocity and current position of particle without more complicated evolution operation. PSO algorithm has better performance than early intelligent algorithms on calculation speed and memory occupation, and has less parameters and is easier to realize.

At present, PSO algorithm is attracting more and more attention of researchers, and has been widely used in fields like function optimization, combination optimization, neural network training, robot path programming, pattern recognition, fuzzy system control and so on[2]. In addition, The study of PSO algorithm also has infiltrated into electricity, communications, and economic fields. Like other random search algorithms, there is also certain degree of premature phenomenon in PSO algorithm. So in order to improve the optimization efficiency, many scholars did improvement researches on basic PSO algorithm, such as modifying PSO algorithm with inertia weight[3], modifying PSO algorithm with contraction factor[4], and combined algorithm with other intelligent algorithm[5]. These modified algorithms have further improvement in aspects like calculation efficiency, convergence rate and so on.

Proper coordination between global search and local search is critical for algorithm finally converging to global optimal solution. Basic PSO algorithm has simple concept and is easy to control parameters, but it takes the entire optimization as a whole without detailed division, and it searches on the same intensity all along that to a certain extent leads to premature convergence. A hierarchical structure poly-particle swarm optimization approach utilizing hierarchy concept of control theory is presented, in which the parallel optimization calculation employs poly-particle swarm in the bottom layer, which is equivalent to increase particle number and enlarges the particle searching domain. To avoid algorithm getting in local optimum and turning into premature, disturbance strategy is introduced, in which the particle velocity is updated and re-initialized when the flying velocity is smaller than the minimum restrictions in the process of iteration. The best position found by each poly-particle swarm in the bottom layer is regard as best position of single particle in the top layer. The top layer performs PSO optimization and feeds the global optimal solution back to the bottom layer. Independent search of the poly-particle swarm on bottom layer can be used to ensure that the optimization to carry out in a wider area. And in top layer, particle swarm's tracking of current global optimal solution can be used to ensure the convergence of the algorithm. Several benchmark functions have been used to test the algorithm in this paper, and the results show that the new algorithm performance well in optimization result and convergence characteristic, and can avoid premature phenomenon effectively.

2. Basic PSO algorithm

PSO algorithm is swarm intelligence based evolutionary computation technique. The individual in swarm is a volume-less particle in multidimensional search space. The position in search space represents potential solution of optimization problem, and the flying velocity determines the direction and step of search. The particle flies in search space at definite velocity which is dynamically adjusted according to its own flying experience and its companions' flying experience, i.e., constantly adjusting its approach direction and velocity by tracing the best position found so far by particles themselves and that of the whole swarm, which forms positive feedback of swarm optimization. Particle swarm tracks the two best current positions, moves to better region gradually, and finally arrives to the best position of the whole search space.

Supposing in a D dimension objective search space, PSO algorithm randomly initializes a swarm formed by m particles, then the position X_i (potential solution of optimization problem)of ith particle can be presented as $\{x_{i1}, x_{i2}, \dots, x_{iD}\}$, substitute them into the object function and adaptive value will be come out, which can be used to evaluate the solution. Accordingly, flying velocity can be represented as $\{v_{i1}, v_{i2}, \dots, v_{iD}\}$. Individual extremum $P_i\{p_{i1}, p_{i2}, \dots, p_{iD}\}$ represents the best previous position of the ith particle, and global extremum $P_g\{p_{g1}, p_{g2}, \dots, p_{gD}\}$ represents the best previous position of the swarm. Velocity and position are updated each time according to the formulas below.

$$\begin{cases} v_{id}^{k+1} = wv_{id}^{k} + c_1 r_1 (p_{id}^{k} - x_{id}^{k}) + c_2 r_2 (p_{gd}^{k} - x_{id}^{k}) \\ \text{if } \ v_{id}^{k+1} > v_{\max}, \ v_{id}^{k+1} = v_{\max}; \\ \text{if } \ v_{id}^{k+1} < v_{\min}, \ v_{id}^{k+1} = v_{\min}; \\ x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \end{cases} \tag{1}$$

In the formula, k represents iteration number; w is inertia weight; c_1, c_2 is learning factor; r_1, r_2 is two random numbers in the range [0,1].

The end condition of iteration is that the greatest iteration number appears or the best previous position fits for the minimum adaptive value.

The first part of formula (1) is previous velocity of particle, reflecting the memory of particle. The second part is cognitive action of particle, reflecting particle's thinking. The third part is social action of particle, reflecting information sharing and mutual cooperation between particles.

3. Hierarchical structure poly-particle swarm optimization

During the search of PSO algorithm, particles always track the current global optimum or their own optimum, which is easy to get in local minimum [6]. Aiming at this problem in traditional PSO, this paper proposes a hierarchical structure poly-particle swarm optimization approach as the following.

(1) Based on hierarchical control concept of control theory, two-layer ploy-particle swarm optimization is introduced. There are L swarms on bottom layer and p_{ig} represents global optimum of ith swarm. There is a swarm on top layer and p_g represents global optimum of top layer. Figure 1 represents scheme of HSPPSO.

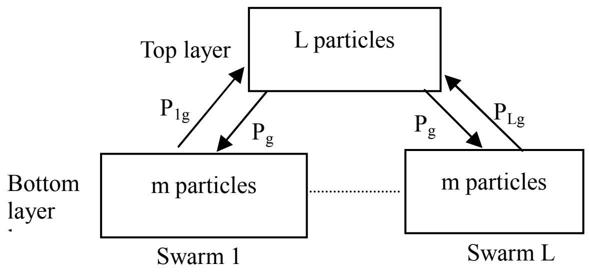

Figure 1. **Scheme of HSPPSO**

Supposing there are L swarms in bottom layer, m particles in each swarm, then parallel computation of L swarms is equivalent to increasing the number of particle to L*m, which expands search space. The parallel computation time of ploy-particle swarms doesn't increase with the number of particle. Besides individual and global extremum of the particle swarm, poly-particle swarm global extremum is also considered to adjust the velocity and position of particles in L swarms. Correction $c_3 r_3 (p_g^k - x_{ij}^k)$ is added to the algorithm, and the iteration formulas are as the following.

$$\begin{cases} v_{ij}^{k+1} = wv_{ij}^{k} + c_1 r_1 (p_{ij}^{k} - x_{ij}^{k}) + c_2 r_2 (p_{ig}^{k} - x_{ij}^{k}) + c_3 r_3 (p_{g}^{k} - x_{ij}^{k}) \\ \text{if } v_{ij}^{k+1} > v_{\max}, \quad v_{ij}^{k+1} = v_{\max}; \\ \text{if } v_{ij}^{k+1} < v_{\min}, \quad v_{ij}^{k+1} = v_{\min}; \\ x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} \end{cases} \tag{2}$$

In the formula, c_3 is learning factor; r_3 is random numbers in the range [0,1]. i represents swarm and i=1,…, L, j represents particle and j=1,…, m, x_{ij}

represents the position variable of jth particle in ith swarm; p_{ij} represents individual extremum of jth particle in ith swarm.

The fourth part of the formula (2) represents the influence of global experience to particle, reflecting information sharing and mutual cooperation between particles and global extremum.

The top layer will commence secondary optimization after ploy-particle swarm optimization on bottom layer, which takes each swarm in L swarms for a particle and swarm optimum p_{ig} for individual optimum of current particle. The iteration formulas of particle velocity update are as the following.

$$\begin{cases} v_i^{k+1} = wv_i^k + c_1 r_1 (p_{ig}^k - x_i^k) + c_2 r_2 (p_g^k - x_i^k) \\ \text{if } v_i^{k+1} > v_{\max}, \quad v_i^{k+1} = v_{\max}; \\ \text{if } v_i^{k+1} < v_{\min}, \quad v_i^{k+1} = v_{\min}; \\ x_i^{k+1} = x_i^k + v_i^{k+1} \end{cases} \tag{3}$$

The top layer PSO algorithm adjusts particle velocity according to the global optimum of each swarm on bottom layer. Independent search of the L poly-particle swarm on the bottom layer can be used to ensure the optimization to be carried out in a wider area. On top layer, particle swarm's tracking of current global optimal solution can be used to ensure the convergence of the algorithm, in which both attentions are paid to the precision and efficiency of the optimization process.

(2) Introduce disturbance strategy. Optimization is guided by the cooperation and competition between particles in PSO algorithm. Once a particle finds a position which is currently optimum, other particles will quickly move to the spot, and gather around the point. Then the swarm will lose variety and there will be no commutative effect and influence between particles. If particles have no variability, the whole swarm will stagnate at the point. If the point is local optimum, particle swarm won't be able to search the other areas and will get in local optimum which is so-called premature phenomenon. To avoid premature, the particle velocity need to be updated and re-initialized without considering the former strategy when the velocity of particles on the bottom layer is less than boundary value and the position of particle can't be updated with velocity.

4. Algorithm flow

(1) Initialization. Set swarm number L, particle swarm scales m and algorithm parameters: inertia weight, learning factor, velocity boundary value, and the largest iterative number.

(2) Each swarm on bottom layer randomly generates m original solutions, which are regarded as current optimum solution p_{ij} of particles meanwhile. Adaptive value of all particles is computed. The optimum adaptive value of all particles is regarded as current optimum solution p_{ig} of swarm, which is transferred to top layer.

(3) Top layer accepts L p_{ig} from bottom layer as original value of particles, which are regarded as their own current optimum solution p_{ig} of particles meanwhile. The optimum adaptive value of all particles is regarded as current optimum solution p_g of swarm, which is transferred to bottom layer.

(4) Bottom layer accepts p_g form top layer, updates velocity and position of particle according to formula (2), if velocity is less than the boundary value, the particle velocity is updated and re-initialized.

(5) Bottom layer computes adaptive value of particles, and compares with current individual extremum. The optimum value is regarded as current optimum solution p_{ij} of particles. The minimum value of p_{ij} is compared with global extremum. The optimum value is regarded as current global extremum p_{ig} of swarm, which is transferred to top layer.

(6) Top layer accepts p_{ig} from bottom layer. The optimum adaptive value of p_{ig} is compared with current global extremum p_g. The optimum adaptive value is regarded as current global extremum p_g. Velocity and position of each particle are updated according to formula (3).

(7) Top layer computes adaptive value of particles, and compares with current individual extremum. The optimum value is regarded as current optimum solution p_{ig} of particles. The minimum value of p_{ig} is compared with global extremum p_g. The optimum value is regarded as current global extremum p_g of swarm, which is transferred to bottom layer.

(8) Evaluate end condition of iteration, if sustainable then output the result p_g, if unsustainable then turn to (4).

5. Example

In order to study algorithm performance, tests are done on four common benchmark functions: Spherical, Rosenbrock, Griewank and Rastrigin. The adaptive value of the four functions is zero. Spherical and Rosenbrock are unimodal function, while Griewank and Rastrigin are multimodal function which has a great of local minimum.

f_1: Spherical function

$$f_1(x) = \sum_{i=1}^{n} x_i^2, \quad -100 \le x_i \le 100$$

f_2: Rosenbrock function

$$f_2(x) = \sum_{i=1}^{n} [100 \times (x_{i+1} - x_i^2)^2 + (x_i - 1)^2], \ -100 \le x_i \le 100$$

f$_3$: Griewank function

$$f_3(x)=\frac{1}{4000}\sum_{i=1}^{n}(x_i-100)^2-\prod_{i=1}^{n}\cos(\frac{x_i-100}{\sqrt{i}})+1,\quad -100\le x_i\le 100$$

f$_4$: Rastrigin function

$$f_4(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10),\ \ -100\le x_i\le 100$$

Set the dimension of four benchmark function to 10, the corresponding maximum iteration number to 500, swarm scale m to 40, number of swarm to 5, inertia weight to 0.7298, c1, c2 and c3 to 1.4962. Each test is operated 10 times randomly.

Table 1 presents the results of four benchmark test function with PSO and HSPPSO.

Table1. **Compare of simulation results**

function	algorithm	minimum	maximum	average
f1	PSO	3.0083e-9	8.9688e-5	3.5601e-8
	HSPPSO	3.5860e-12	1.9550e-7	4.2709e-11
f2	PSO	5.7441	8.8759	7.65997
	HSPPSO	4.2580	7.8538	5.5342
f3	PSO	0	24.9412	7.36575
	HSPPSO	0	2.3861	0.23861
f4	PSO	4.9750	13.9392	10.15267
	HSPPSO	0.9950	7.9597	4.4806

Table 1 shows that HSPPSO performance is better than PSO in searching solution, and gets better results than PSO both in test for local searching and global searching. Though the difference of HSPPSO between PSO in searching solution of unimodal function is not obvious, HSPPSO performance is better than PSO for multimodal function, which indicates that HSPPSO has better application foreground in avoiding premature convergence and searching global optimum.

Figure 2 to 9 are adaptive value evolution curve after 10 times random iteration for four benchmark functions with HSPPSO and PSO, which indicates that HSPPSO performance is better than PSO both in initial convergence velocity and iteration time. PSO even cannot find optimum value in some tests.

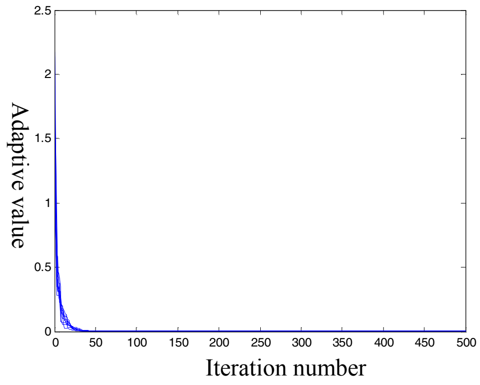

Figure 2. **HSPPSO test function f1**

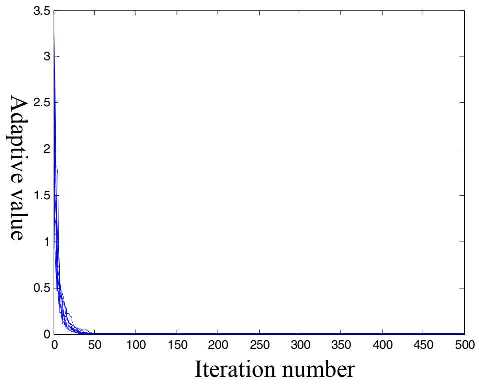

Figure 3. **PSO test function f1**

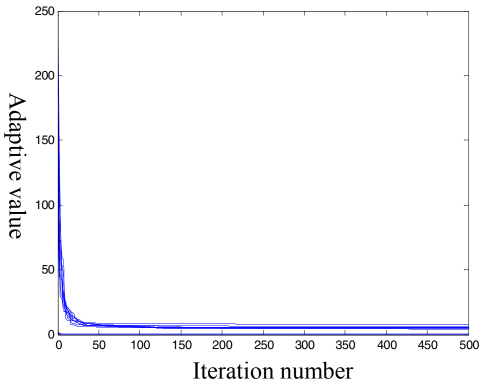

Figure 4. **HSPPSO test function f2**

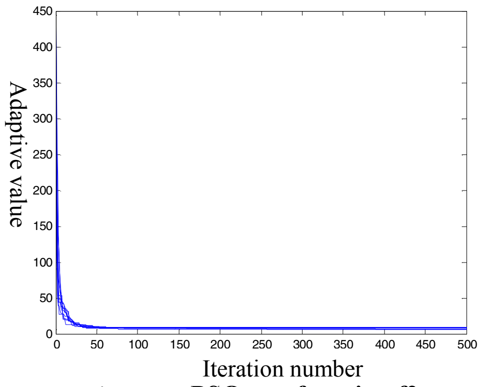

Figure 5. **PSO test function f2**

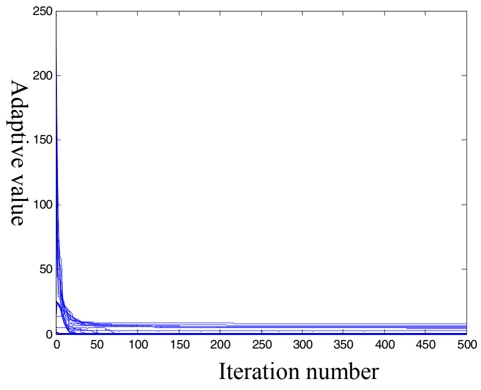

Figure 6. **HSPPSO test function f3**

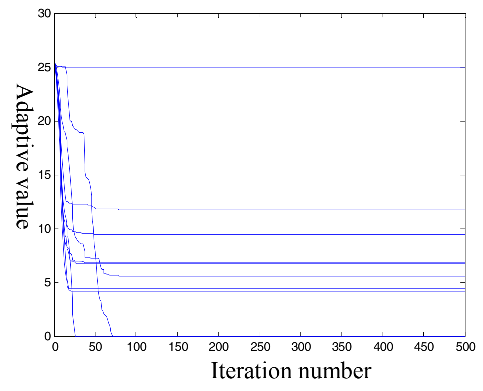

Figure 7. **PSO test function f3**

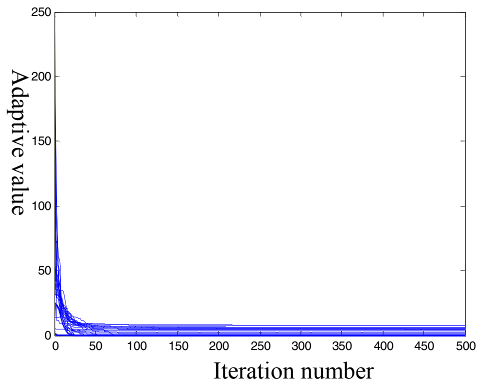

Figure 8. **HSPPSO test function f4**

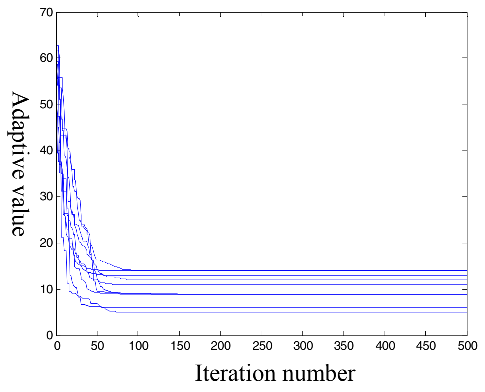

Figure 9. **PSO test function f4**

6. Conclusion

Comparing with basic PSO, HSPPSO proposed in this paper can get better adaptive value and have faster convergence rate on condition of the same iteration step. HSPPSO provides a new idea for large scale system optimization problem.

References

[1] J Kennedy, R Eberhart. Particle Swarm Optimization[C].In：Proceedings of IEEE International Conference on Neural Networks, 1995, Vol 4, 1942-1948.

[2] Vanden Bergh F, Engelbrecht A P. Training Product Unit Networks Using Cooperative Particle Swarm Optimization [C]. IEEE International Joint Conference on Neural Networks, Washington DC, USA.2001,126-131.

[3] Shi Y, Eberhart R C. A modified particle swarm optimizer [C] IEEE World Congress on Computational Intelligence, 1998: 69- 73

[4] Eberhart R C, Shi Y.Comparing inertia weights and constriction factors in particle swarm optimization[C] Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, 2001: 84- 88

[5] Lφvbjerg M, Rasmussen T K, Krink T.Hybrid particle swarm optimizer with breeding and subpopulations[C] Third Genetic and Evolutionary Computation Conference. Piscataway, NJ: IEEE Press, 2001

[6] WANG Ling. Intelligent Optimization Algorithms with Applications. Beijing: Tsinghua University Press, 2001.