INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 8 (8): 1263-1268, 2009
ISSN 1812-5638
© 2009 Asian Network for Scientific Information

An Enhanced Particle Swarm Optimization Algorithm

[1]Xue-yao Gao, [1]Li-quan Sun and [2]Da-song Sun
[1]School of Computer Science and Technology, Harbin University of Science and Technology, Harbin, China
[2]Computer Center, Harbin University of Science and Technology, Harbin, China

Abstract: Particle Swarm Optimization (PSO) algorithm is often used for finding optimal solution, but it easily entraps into the local extremum in later evolution period. Based on improved chaos searching strategy, an enhanced particle swarm optimization algorithm is proposed in this study. When particles get into the local extremum, they are activated by chaos search strategy, where the chaos search area is controlled in the neighborhood of current optimal solution by reducing search area of variables. The new algorithm not only gets rid of the local extremum effectively but also enhances the precision of convergence significantly. Experiment results show that the proposed algorithm is better than standard PSO algorithm in both precision and stability.

Key words: Particle swarm, local extremum, chaos search, optimization algorithm

INTRODUCTION

The purpose of swarm intelligence method is to build a random optimization algorithm by simulating nature biology group behavior. Kennedy and Eberthart (1995) gained enlightenment from bird and fish group behavior and proposed the particle swarm optimization algorithm. The PSO algorithm has fewer parameters and can resolve the complicated optimization problems efficiently (Parsopoulos and Vrahatis, 2004). But PSO also can get into the local extremum and has slow search speed in later period, which is similar to other intellective algorithm.

The key to PSO algorithm is to deal with the problem of slow search speed and premature convergence. Kennedy and Eberhart (1997) used a discrete binary version of particle swarm optimization to resolve combinatorial optimization problems in engineering practice. TVAC was given to control local search and converged to global optimum solution efficiently (Ratnaweera *et al*., 2004). Zheng *et al*. (2007) proposed a method of changing velocity rate to enhance searching speed. Test experiments of domain topology were conducted (Kennedy, 1999), in which the best form of topology should be designed based on actual situation. Distance from the global best position to other position was calculated to adjust the velocity suitably of each particle (Kennedy, 2000), in order to improve the performance of PSO and maintain the diversities of particles. These methods adjust position and velocity of particles to enhance search speed. Dou *et al*. (2006) proposed the swarm-core evolutionary in dynamic optimization environments. Kwok *et al*. (2007) proposed an alternative social interaction scheme among swarm members. He *et al*. (2005) put forward an improved particle swarm optimization based on self-adaptive velocity to speed up the convergence, but the algorithm will still converge to local optimum. A new adaptive mutation particle swarm optimizer was proposed by Lv and Hou (2004), which is based on variance of the population's fitness. Meng *et al*. (2006) introduced chaotic series into PSO algorithm to avoid local optimization, but searching process is blind and slow. Ni *et al*. (2009) introduced multi-cluster structure into particle swarm optimization algorithm to escape from local optimal solutions, which enhanced global search ability.

Chaos is one of nonlinear phenomenon, which is stochastic and ergodic and can be used in optimization algorithm (Li and Jiang, 1997). Niu *et al*. (2009) proposed chaotic differential evolution algorithm for multi-objective optimization. In this study, an enhanced particle swarm optimization algorithm based on improved chaos search strategy is proposed (EPSO), in order to deal with premature convergence in later evolution. From testing resu lts of the benchmark functions, the results of EPSO are obviously better than that of standard particle swarm optimization algorithm (SPSO).

PARTICLE SWARM OPTIMIZATION ALTORITHM

Particle swarm algorithm starts from random initialization of a population of particles in the searching space and works on the social behavior of particles in the swarm. It finds the global best solution by adjusting simply the trajectory of each individual toward its own best location and toward the best particle of the entire swarm at each step.

Corresponding Author: Xue-yao Gao, School of Computer Science and Technology, Harbin University of Science and Technology, Harbin, China

Each particle in the swarm flies with a velocity in the D-dimensional problem space, which is dynamically adjusted according to the flying experiences of its own and its colleagues. Let *m* represent the total number of particles. The location of the *i*th particle is represented as $X_i = (x_{i1}, x_{i2},..., x_{iD}; i = 1,2,...m)$. The best previous position of the ith particle is recorded and represented as $P_i = (p_{i1}, p_{i2},..., p_{iD}; i = 1,2,...m)$, which is also called pbest. The index of the best particle among all particles in the population is represented by symbol g and $Pg = (p_{g1}, p_{g,2},... p_{g,D})$, where $g \in \{1,2,...m\}$. The location P_g is also called gbest. The velocity for the ith particle is represented as $V_i = (v_{i1}, v_{i2},..., v_{iD}; i = 1,2,...m)$ and is clamped to a maximum velocity $V_{max} = (v_{max,1}, v_{max,2},..., v_{max,d})$, which is specified by the user. The concept of particle swarm optimization includes, at each step, changing the velocity and location of each particle toward its pbest and gbest locations according to Eq. 1 and 2, respectively:

$$v_{id}^{k+1} = w^k v_{id}^k + c_1 r_1 (p_{id}^k - x_{id}^k) + c_2 r_2 (p_{gd}^k - x_{id}^k) \tag{1}$$

$$x_{id}^k = x_{id}^k + v_{id}^k \tag{2}$$

where, $1 \le i \le m$, $1 \le d \le D$, c_1 and c_2 are acceleration constants, r_1 and r_2 are random functions in the range (0, 1). In order to control particles in the search area in iteration, take the boundary value as its value if position and velocity of particles go beyond its given area. *w* is inertia weight, which is reduced according to Eq. 3 in iterative change:

$$w^k = w_{max} - k\frac{(w_{max} - w_{min})}{N} \tag{3}$$

where, k is current iteration time, N is total iteration number, w_{min} and w_{max} are the lower and upper bounds for w.

CHAOS SEARCH STRATEGY

Chaos is one of the most popular phenomena that exist in nonlinear system and theory of chaos is one of the most important achievements of nonlinear system research. It is now widely recognized that chaos is almost a fundamental mode of motion in natural phenomena.

There has no strict definition of chaos and there are many chaos models now. Logistic mapping function is widely used to generate chaos, but research shows that its sequence isn't symmetrical, which affects capability of chaos search. Here logic self-mapping function is used to produce chaos variables because of its uniformity and ergodicity:

$$x_{n+1} = 1 - 2 \times x_n^2,\ n = 0,1,2,\ldots,\ -1 < x_n < 1 \tag{4}$$

Chaos will occur as long as initial iterative value does not equal to 0. This mapping is simple and easy to compute by computer.

If the target function f (x_i) is continuous, object problem to be optimized is shown in Eq. 5:

$$\min f(x_i),\ x_i \in [a_i, b_i],\ i = 1, 2, \ldots, D \tag{5}$$

The basic process of chaos optimization strategy can be described as follows:

- **Step 1:** Algorithm initialization. Let k = 0, create D different chaos variables x_i (k) randomly and $x_i (k) \neq 0$, I = 1,2,...D. k is the iterative symbol of chaos parameters. Let x_i denote the current best chaos variable, f is the current best solution and initialized as a biggish number
- **Step 2:** Map the chaos variable x_i (k) to optimization variable area, signed as mx_i (k):

$$mx_i(k) = \frac{(b_i - a_i)}{2} x_i(k) + \frac{(b_i + a_i)}{2} \tag{6}$$

- **Step 3:** Calculate f $(mx_i (k))$ and if $f (mx_i (k)) < f^*$, $f^* = f (mx_i (k))$, $x^*_i = x_i (k)$
- **Step 4:** Let k = k+1, $x_i (k) = 1-2 (x_i (k))^2$, repeat from step 2 to step 4 until f^* keep unchanged in certain steps or iterative time reaches the given one and x^*_i is the best chaos variable and f^* is the best solution

EPSO ALGORITHM

In later evolution, convergence rate of PSO algorithm will reduce much and entrap into local minimum most likely, which affects precision and efficiency of the algorithm. So, efficient variety should be adopted to improve capability of the PSO algorithm.

Chaos search strategy is introduced into PSO, in order to enhance the global search capability and get rid of the attraction of local minimum. The main idea is that, when particles get into the local extremum, gbest in particle swarm is searched by chaos search strategy for a decided step, which guides the particle swarm toward the best solution.

In order to avoid particle swarm getting into local minimum and leading the algorithm stagnant, premature estimate mechanism is introduced into searching process, which uses swarm fitness variance to estimate whether particle swarm gets into local minimum. Swarm fitness variance is calculated as:

$$\sigma^2 = \sum_{i=1}^{N}[(f_i - \bar{f})/f]^2 \tag{7}$$

where, f_i is fitness of particle i, $\bar{f}$ is average fitness of particle swarm and f is unitary gene:

$$f = \begin{cases} \max|f_i - \bar{f}| & \max|f_i - \bar{f}| > 1 \\ 1 & \text{others} \end{cases}$$

when $\sigma^2 < m \times \beta$, chaos search strategy is used to deal with premature. Where, m is particle number, which is used to balance magnitude on both sides of the inequation. β is a given smallish constant in area of (0, 0.2).

When particles get into local extremum, chaos search strategy is used to arouse the particles. Logic self-mapping function is used to initialize particle swarm again. Because of the symmetrical characteristic of this function, it is used to initialize particle swarm at the beginning of the algorithm, to distribute particles more symmetrically. The best solution is searched blindly, because chaos search is random. In order to avoid particles searching loss some area, chaos search process is modified by reducing search area of variables around the current best solution. The chaos search area is controlled in the neighborhood of local extremum, to enhance precision and efficiency of chaos search. Following equations are used to reduce area:

$$a_i = x_i^* - C(b_i - a_i),\ b_i = x_i^* + C(b_i - a_i) \tag{8}$$

where, $C\in$ (0,05) is adjustment coefficient. The rest of particle search area remains invariably.

On the basis of above definitions and analysis, process of EPSO is described as follows:

- **Step 1:** Parameter initialization: acceleration constants c_1 and c_2, maximal inertia weight w_{max} and minimal inertia weight w_{min}, constant β, adjustment coefficient C, total iteration number N and chaos search time A
- **Step 2:** Particle swarm initialization: randomly take D variables in area (-1, 1) exclude 0, namely $X_i = (x_{i1}, x_{i2},... x_{iD})$, iterate m-1 times using Eq. 4 to produce position variables of particle swarm. Produce velocity variables by the same method. Map position variables to optimization variable area according to Eq. 6
- **Step 3:** Calculate fitness of every particle and take the best one among all particles as the current global best g
- **Step 4:** Estimate whether the terminate condition is satisfied, if true, turn to step 10 else turn to next step
- **Step 5:** Calculate w^k according to Eq. 3
- **Step 6:** Manipulate particles according to Eq. 1 and 2 respectively, calculate fitness of every particle and update p_i of every particle and g of the swarm
- **Step 7:** Estimate whether the terminate condition is satisfied, if true, turn to step 10 else turn to next step
- **Step 8:** Estimate whether $\sigma^2 < m\times\beta$, if true, turn to next step else turn to step 5
- **Step 9:** Guide particles to get away from the local extremum by improved chaos search strategy:

  - Set current iteration counter G = 1 and initialize the total iteration time A
  - Revert optimization variable m_{xi} (k) to chaos variable:

  $$x_i(k) = \frac{2}{b_i - a_i}(mx_i(k) - \frac{b_i + a_i}{2})$$

  - Create chaos variable according to Eq. 4
  - This chaos variable is mapped to optimization variable area according to Eq. 6. Calculate fitness and update f^* and x^*_i
  - Estimate whether the terminate condition is satisfied, if true, turn to step 6 else turn to next step
  - G = G+1, if G>A, turn to step 7, else reduce area according to Eq. 8. If value of a_i or b_i goes beyond the boundary value, take the boundary value as its value. Then turn to 3

- **Step 10:** Evolution is over and the global best solution is got

EXPERIMENT RESULTS AND ANALYSIS

In this section, three typical functions are used to evaluate the performance of EPSO. Actual minimum values of these functions are all zero.

Parameter initialization of the algorithm is as follows: parameters c_1 and c_2 are set to 1.49, w_{min} is set to 0.4,w_{max} is set to 0.95, total iteration number N is set to 500, adjustment coefficient C is set to 0.4, chaos search time A is set to 50, the number of particles m is set to 20, constant β is adjusted appreciably in area of (0, 0.2) for different problems and maximum velocity $v_{max,i}$ of each particle is set to twenty percent of optimization variable area. For each function, 50 trials are carried out. At the same time, minimum optimal value, maximum optimal value, average optimal value and average computing time are recorded.

Table 1: Compared results of f_1

		D = 2	D = 10
Minimum	SPSO	1.758049e-006	1.721806e+001
	EPSO	2.387099e-014	1.991971e-009
Maximum	SPSO	6.773652e-003	5.518307e+001
	EPSO	5.580938e-012	5.989847e-009
Average	SPSO	1.339305e-003	3.465803e+001
	EPSO	7.647534e-013	4.669285e-009
Computing time (m sec)	SPSO	10	15
	EPSO	70	175

Table 2: Compared results of f_2

Algorithm	SPSO	EPSO
Minimum	4.257702e-005	2.998769e-014
Maximum	1.242850e-001	1.027868e-010
Average	3.259531e-002	2.100050e-011
Computing time (m sec)	8	120

Table 3: Compared results of f_3

		D = 2	D = 10
Minimum	SPSO	5.112446e-006	7.585180e+001
	EPSO	1.958720e-015	7.404652e-003
Maximum	SPSO	1.546793e-001	2.140677e+002
	EPSO	4.475398e-010	8.196867e-001
Average	SPSO	5.035928e-002	1.521952e+002
	EPSO	6.316322e-011	1.660742e-001
Computing time (m sec)	SPSO	10	16
	EPSO	135	469

$$f_1 = \sum_{1}^{n} x_i^2,\ x_i \in [-5.12,\ 5.12]$$

The division of f_1 is set to 2 and 10, respectively. The experiment results are shown in Table 1. When D is set to 2, both algorithms can converge to optimal value. But precision of EPSO is much better than that of SPSO. When D is set to 10, in 50 trials, SPSO can not converge to optimal value. But EPSO can converge to optimal value with high precision.

Figure 1 is comparison of convergence results of SPSO and EPSO when D = 2. The horizontal ordinate indicates iteration number and vertical ordinate indicates logarithm of magnitude for average optimal value. From Fig. 1, it can be seen that SPSO entraps into the local extremum in later evolution, but EPSO can get rid of the attraction of local extremum and find the better solutions.

$$f_2 = x^2 - 0.4\cos(3\pi x) + 2y^2 - 0.6\cos(4\pi y) + 1,\ x, y \in [-10,\ 10]$$

The division of f_2 is 2. The experiment results are shown in Table 2. In 50 trials, SPSO can not converge to optimal value for the most, but EPSO can converge to optimal value with high precision. The maximal fitness of EPSO is much small than that of SPSO.

Figure 2 is comparison of convergence results under different iteration number. The horizontal ordinate indicates iteration number and vertical ordinate indicates logarithm of magnitude for average optimal value. From

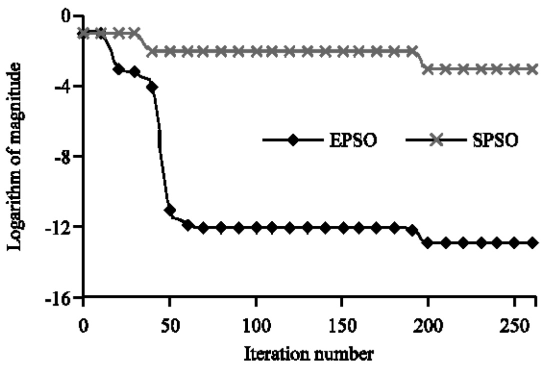

Fig. 1: Convergence results of f_1 when D is set to 2

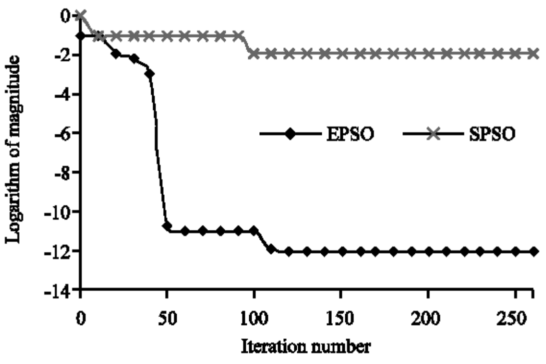

Fig. 2: Comparison of convergence results of f_2

Fig. 2, it can be seen that EPSO can get rid of local extremum and find the best solutions.

$$f_3 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,\ x_i \in [-600,\ 600]$$

The division of f_3 is set to 2 and 10 respectively. The experiment results are shown in Table 3. When D is set to 2, results of SPSO are closer to optimal value. But EPSO can converge to optimal value with high precision. When D is set to 10, SPSO can not converge to optimal value on the whole, while results of EPSO are closer to optimal value. Convergence of EPSO is improved than that of SPSO.

Figure 3 is comparison of convergence results under different iteration number when D is set to 2. The horizontal ordinate indicates iteration number and vertical ordinate indicates logarithm of magnitude for average optimal value. Figuer 3 shows that the capability of EPSO is obviously better than that of SPSO.

From the testing results of the benchmark functions, it can be seen that for these three benchmark functions, the results of the proposed algorithm is obviously better than that of the SPSO. EPSO is better than SPSO in

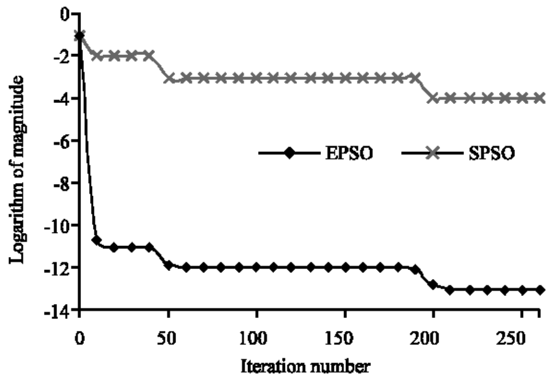

Fig. 3: Convergence results of f_3 when D is set to 2

aspects of convergence precision and stability. This is because that EPSO can find the better solutions in optimization process and local extremum will be avoided in searching process.

In compared experiments, it is found that parameter selections of the algorithm are very important. In order to control the algorithm to carry out chaos search process when particle swarm get into the local optimal, value of β should be proper. If value of β is set too small, the algorithm will implement chaos search prematurely. But the algorithm will be hard to carry out chaos search if value of β is set much bigger. From different comparison, it is found that capability of the algorithm is preferable when β is set to 0.07. For chaos search, selection of adjustment coefficient C is important, too. When C is smaller, search area reduces faster. From different experiments, capability of the algorithm is preferable when C is set to 0.4. Finally, from above tables, average computing time of EPSO is much more than that of SPSO, because chaos search strategy is introduced into PSO algorithm. But from above figures, when total iteration number is set to 200 and chaos search time is set to 40, the optimization results are actually so preferable in the proposed algorithm. But the searching time of EPSO will be decreased.

CONCLUSION

In this study, an enhanced particle swarm optimization algorithm is proposed. When particles get into local extremum, they are activated by improved chaos search strategy to search the best solution. From experiment results, it can be seen that the proposed algorithm can effectively solve premature convergence. Moreover, precision and stability of the algorithm are both obviously better than that of SPSO. In future study, the algorithm stability and searching speed will be improved further.

ACKNOWLEDGMENT

This study is supported by National Nature Science Foundation of China under Grant No. 60173055.

REFERENCES

Dou, Q.S., C.G. Hou, Z.Y. Xu and G.Y. Pan, 2006. Swarm-core evolutionary particle swarm optimization in dynamic optimization environments. J. Comput. Res. Dev., 43: 89-95.

He, R., Y.J. Wang, Q. Wang, J.H. Zhou and C.Y. Hu, 2005. An improved particle swarm optimization based on self-adaptive escape velocity. J. Software, 16: 2036-2044.

Kennedy, J. and R. Eberhart, 1995. Particle swarm optimization. Proceedings of the IEEE International Conference on Neural Networks, Nov. 27-Dec. 1, Piscataway, New Jersey, pp: 1942-1948.

Kennedy, J. and R.C. Eberhart, 1997. A discrete binary version of the particle swarm algorithm. Proceedings of the IEEE International Conference on Computational Cybernetics and Simulation, Dec. 12-15, New York, pp: 4104-4109.

Kennedy, J., 1999. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. Proceedings of the Congress on Evolutionary Computation, Jul. 6-9, Washington, USA., pp: 1931-1938.

Kennedy, J., 2000. Stereotyping: Improving particle swarm performance with cluster analysis. Proceedings of the IEEE Conference on Evolutionary Computation, July 16-19, California, USA., pp: 1507-1512.

Kwok, N.M., G. Fang, Q.P. Ha and D.K. Liu, 2007. An enhanced particle swarm optimization algorithm for multi-modal functions. Proceedings of the IEEE on Mechatronics and Automation, Aug. 5-8, Harbin, China, pp: 457-462.

Li, B. and W.S. Jiang, 1997. Chaos optimization method and its applications. Control Theor. Appl., 14: 613-615.

Lv, Z.S. and Z.R. Hou, 2004. Particle swarm optimization with adaptive mutation. Acta Electronica Sinica, 32: 416-420.

Meng, H.J., P. Zheng, G.H. Mei and Z. Xie, 2006. Particle swarm optimization algorithm based on chaotic series. Control Decision, 21: 263-266.

Ni, Q.J., Z.Z. Zhang, Z.Z. Wang and H.C. Xing, 2009. Dynamic probability particle swarm optimization based on varying multi-cluster structure. J. Software, 20: 339-349.

Niu, D.P., F.L. Wang, D.K. He and M.X. Jia, 2009. Chaotic differential evolution for multiobjective optimization. Control Decision, 24: 361-364.

Parsopoulos, K.E. and M.N. Vrahatis, 2004. On the computation of all global minimizers through particle swam optimization. IEEE Trans. Evol. Comput., 8: 211-224.

Ratnaweera, A., S.K. Halgamuge and H.C. Watson, 2004. Self-organizing hierarchical particle swarm optimizer with time varying acceleration coefficients. IEEE. Trans. Evol. Comput., 8: 240-255.

Zheng, S.F., S.L. Hu, S.X. Su, C.F. Lin and X.W. Lai, 2007. A modified particle optimization algorithm and application. Proceedings of the 6th International Conference on Machine Learning and Cybernetics, Aug. 19-22, Hong Kong, China, pp: 945-951.