Two Improvement Strategies for PSO

Quansheng Dou [1,2], Shasha Liu [2], Ping Jiang[2] , Xiuhua Zhou[2], Zhongzhi Shi [1]

[1]Key Laboratory of Intelligent Information Processing; Institute of Computing Technology; Chinese Academy of Sciences; Beijing; 100080, China
[2]School of Information and Electronics Engineering; Shandong Institute of Business and Technology, Yantai 264005,China

douqs@ics.ict.ac.cn

Abstract— This paper proposed an improved particle swarm optimization algorithm (IPSO) to solve continuous function optimization problems. Two improvement strategies named "Vector correction strategy" and "Jump out of local optimum strategy" were employed in our improved algorithm. The algorithm was tested using 25 newly proposed benchmark instances in Congress on Evolutionary Computation 2005 (CEC2005). The experimental results show that the search efficiency and the ability of jumping out from the local optimum of the IPSO have been significantly improved, and the improvement strategies are effective.

Key words — Particle Swarm Optimization; Convergence Property; Improved Strategy

1. Introduction

Particle Swarm Optimization (PSO) method was proposed by Kennedy and Eberhart in 1995 as an evolutionary computing technology [1] [2], which is widely used in various types of optimization problems [3]-[7]. In solving optimization problems, a potential solution of each optimization problem is seen as a "particle" in the search space. An ordered triple of numbers (x_i, v_i, p_i) corresponds to each particle; the location of particle's each iterations is decided by the following formula:

$$v_{t+1} = wv_t + c_1 r_1 (p_i - x_t) + c_2 r_2 (p_g - x_t)$$
$$x_{t+1} = x_t + v_{t+1}$$

Where x_i represents the current position of the particle, v_i represents the current speed of the particle, p_i is the best location of particle i (individual optimum), p_g is the best location of all the particles in the swarm have passed (swarm optimum); c1 and c2 are positive constants, r_1 and r_2 are random numbers which obey normal distribution in the interval [0,1] ; ω is an inertia parameter. The advantage of PSO is that it is easy to implement and there are few parameters to adjust. PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, etc.. However, for some complex problem, e.g. 25 benchmark problems in the CEC2005 (The 2005 IEEE Congress on Evolutionary Computation), the searching results of PSO method are not satisfactory. In this paper, two improvement strategies were proposed in order to improve the performance of PSO, the effectiveness of the improvement strategy is proved by 25 benchmark problem provided by CEC2005[8] [9], the following will describe the relevant content.

2. Two Improvement Strategies for PSO

The reasons for the algorithm to fall into local optimum are described in the following:

i). Because the velocity of particles in the swarm decays too fast, the step-length of particles revision decreases rapidly, which make the search efficiency too low or search stagnation;

ii). For some complex issues with a strong deceptive, once the search fall into local optimum, even if the particle's velocity does not completely decay, the probability of that particles hiting a better one than the existing optimal point is still smaller, and thus cause the search stagnation.

To overcome the above two points, we make the following improvements on the PSO method:

i). Vector revision strategy

In traditional PSO method, at each step t, particle's position is updated by the following program

for i=1 to Dims

$$v_{t+1}^{(i)} = wv_t^{(i)} + c_1 r_1 (p_i^{(i)} - x_t^{(i)}) + c_2 r_2 (p_g^{(i)} - x_t^{(i)})$$
$$x_{t+1}^{(i)} = x_t^{(i)} + v_{t+1}^{(i)}$$

end

Here, Dims is the dimensions of the search space, each components of vector is revised respectively. Yet if there is some relativity between the components of vector, the efficiency of such a revision method is low. To improve the search efficiency, this paper revises the particle as a whole vector according to a probability. For each step of the algorithm, particle's position is updated by the following program

if $Rand > \alpha$

for i=1 to Dims

$$v_{t+1}^{(i)} = wv_t^{(i)} + c_1 r_1 (p_i^{(i)} - x_t^{(i)}) + c_2 r_2 (p_{g\ g}^{(i)} - x_t^{(i)})$$

$$x_{t+1}^{(i)} = x_t^{(i)} + v_{t+1}^{(i)}$$

end

else

$$\vec{v}_{t+1} = w\vec{v}_t + c_1 r_1 (\vec{p}_i - \vec{v}_t) + c_2 r_2 (\vec{p}_g - \vec{v}_t)$$

$$\vec{x}_{t+1} = \vec{x}_t + \vec{v}_{t+1}$$

end if

Where $Rand$ is a random number between 0 and 1, α is a given threshold, such a strategy could speed up the efficiency of that individual converge to a swarm optimum. Especially when the various dimensional components of the issue are interrelated, the search efficiency improves clearly. Here, it is very important to set the threshold α. If the value of α is too small, the number of particles in the swarm revised as a vector is too large, while raising the efficiency , also increase the possibility of "premature". Otherwise, search efficiency will be affected .It will be better to set $\alpha = 0.945$ in our experiment.

(2) Jump out of local optimum strategy

In order to make particles jump out of local optimum effectively to avoid "premature" in PSO search process, this paper has developed the following strategies:

if Immovable$>\gamma$ or Sigm$<\delta$

Save global best into Bestgroup;

if $Rand > \theta$ and *Bestgroup* not Null

global best=Randomly select different particle from BestGroup

else

Initial population；

Initial global best;

end if;

end if

In the above program, Immovable is the iterative times of that swarm optimum p_g keep unchanged, γ、δ、θ are the specified thresholds respectively. *BestGroup* is an array which is used to save swarm optimum keeps unchanged or lower revision efficiency after Immovable iterations. Here Sigm is used to determine the revision efficiency of the swarm optimum p_g:

$$sigm = \frac{\sum_{i=1}^{M} rate_i}{M}$$

Where $rate$ is an M dimensional vector. Each component $rate_i$ i=1, M was given a relatively large value initially. When algorithm finds new swarm optimum, $rate$ will be updated as follows:

$$rate(2,....M) = rate(1,...M-1) \text{ and } rate_1 = \frac{g\_best_{i+1} - g\_best_i}{S(i+1) - S(i)}$$

In the above equation g_best_{i+1}、g_best_i are the swarm optimums what the algorithms have searched twice in succession. $S(i+1)$、$S(i)$ are the time for getting the two swarm optimum respectively.

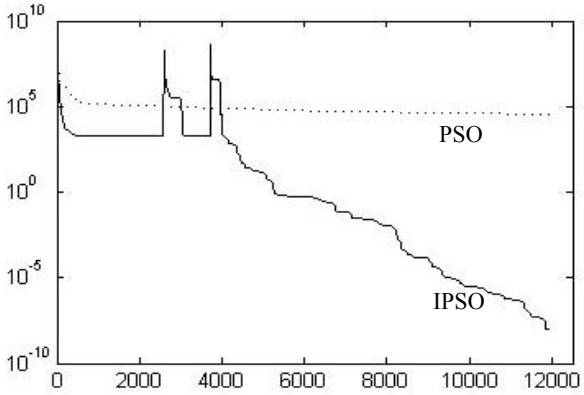
We can see from the above strategy that when the swarm optimum remains invariable or changes in value less than a certain threshold for a long time, the algorithm according to a probability choose a particle from the *BestGroup* as new swarm optimum or re-initialize the swarm and select a new swarm optimum from swarm. Because the particles in the *BestGroup* have been the "swarm optimum", when it is selected again, the original distribution pattern of the swarm is broken, the velocity of particles will be compensated indirectly, and the probability of the algorithm achieving the goal is still very high. For some complex multi-peak problems, search can be achieved possibly only when the particle jump out of a local optimum. Therefore, when the swarm optimum remains invariable or changes in value less than a certain threshold value for a long time, the algorithm according to a probability re-initializes the group and generates a new optimal group to jump out of the attraction of local optimum.

In this paper, the PSO with the above improved strategies is named Improved Particle Swarm Optimization(IPSO), in the following Fig.1a、b are examples of PSO method before and after improvement. Both f_3（Shifted Rotated High Conditioned Elliptic Function）and f_{12} （Schwefel's Problem 2.13）are two test functions in CEC2005. f_3 which is generated from moving and rotating Elliptic Function has a unique extreme point, f_{12} is obtained by moving the Schwefel's Problem, and has multiple local optimal point and one global optimal point. Dotted and solid lines are the search curve of PSO and IPSO respectively in the figure. In the Fig.1a, As a result of rotation and movement, although the test function has only one optimal point, it is still very difficult to find its global optimum point. Because f_3 is rotated by the following formula:

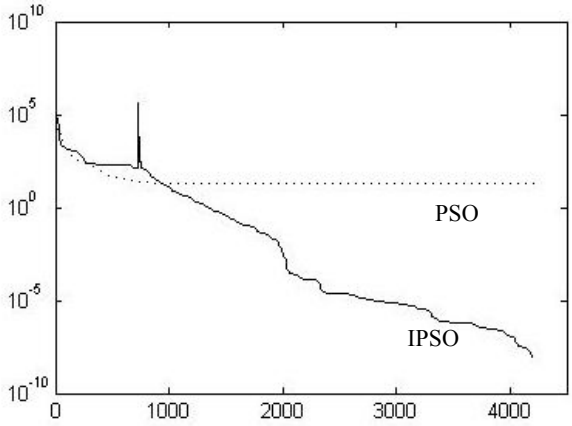
$$\vec{x} = \vec{x} \times M$$

Where M is the specified matrix, so the components of position vector x correlate between each other. It can be seen from the Fig1.a that the implementation curve of IPSO has two "mutations", it is the results triggered by "Jump out of local optimum strategy". After two jumps, the groups eventually converge to the global optimum point, however the PSO is trapped in "premature" state and the search fails. Before the "jump out of local optimum" strategy triggers, only the "Vector correction strategy" plays a role in the IPSO. It is not difficult to see

from Fig.4a that the convergence speed of IPSO is faster than PSO. Fig.4b experienced one time of jumping out, since f_{12} isn't rotated, every components of vector is independent, the role of "Vector correction strategy" does not seem great.



a: Shifted Rotated High Conditioned Elliptic Function (f_3)



b: Schwefel's Problem 2.13(f_{12})

Fig.1 Example of IPSO jumping out of local optimum point

Due to limited space, the content of the experiment is not described in this section, the details of experiment will be described in the following sections.

3. Experiment and Test Results

We adopted 25 benchmark problems [8]proposed in CEC 2005，where f_1 - f_6 are single-peak functions and f_7 - f_{25} are multi-peak functions, these functions are obtained through the shift, rotation and other operations based on the traditional benchmark functions. PSO method performed poorly to these 25 test problems. Literature [9] compared 11 algorithms which have been published in CEC2005.This paper used the same comparison method as the literature [9]: For all benchmark functions, the dimensions of search space was Dims=10 and Dims=30, the maximum number of iterations can be taken as $\mathrm{Dims} \times 10^5$, each test problem was carried out 25

times respectively, in order to measure the implementation of the algorithm results, let

$$FEs = \frac{\text{mean}(\#\text{fevals}) \times \#\text{all runs}}{\#\text{successful runs}}$$

Here, $\#\text{fevals}$ is the iteration times for each successful runs, $\#\text{all runs}$ is the total implementation times of the algorithm (25), $\#\text{successful runs}$ is the number of successful runs. The smaller the value of FEs is, the higher the search performance of the algorithm. The following Table 2 lists the comparison between the best results in literature [9] and the running results with IPSO in the conditions that the dimensions of search space are 10 and 30 respectively.

Table2. The results of IPSO method compared with literature [9] in the conditions that Dims=10 and Dims=30.The number in parentheses in the table is the number of algorithms which complete search at least once in literature [9]. The second sub-column of IPSO column represents the ratio of *FEs* of IPSO to the Best *FEs* of previous column. The number in square bracket is the value of the corresponding *FEs* in the corresponding row of f_{13}

Fun	Dims=10					Dims=30				
	Best in [9]			IPSO		Best in [9]			IPSO	
	Method Name	Succ. Rate	Best FEs	Succ. Rate	$\frac{FEs}{B\_FEs}$	Method Name	Succ. Rate	Best FEs	Succ. Rate	$\frac{FEs}{B\_FEs}$
1	K-PCX	100%(11)	1000	100%	0.45	K-PCX	100%(7)	2700	100%	0.30
2	K-PCX	100%(11)	2400	100%	0.33	K-PCX	100%(9)	12000	100%	0.30
3	G-CMA-ES	100%(7)	6500	100%	2.6	G-CMA-ES	100%(4)	43000	100%	5.9
4	G-CMA-ES	100%(10)	2900	100%	0.41	G-CMA-ES	40%(4)	59000	100%	0.44
5	G-CMA-ES	100%(7)	5900	100%	0.40	G-CMA-ES	100%(2)	66000	100%	0.36
6	K-PCX	100%(8)	7100	100%	0.34	G-CMA-ES	100%(6)	60000	100%	0.25
7	G-CMA-ES	4700(9)	4700	100%	1.22	G-CMA-ES	100%(10)	6100	100%	1.75
9	L-SaDE	100%(7)	17000	100%	0.76	L-SaDE	100%(3)	99000	100%	0.85
10	K-PCX	92%(2)	55000	80%	1.4	K-PCX	56%(2)	450000	44%	2.2
11	DE	48%(3)	190000	28%	3.79	G-CMA-ES	4%(1)	500000	-	-
12	K-PCX	56%(3)	8200	100%	0.95	K-PCX	20%(3)	180000	100%	0.70
13	-	-	-	8%	[1.07e+6]	-				
15	L-SaDE	92%(3)	33000	16%	8.16	-	-	-	-	-

For all the algorithms in literature [9], function f_8、f_{13}、f_{14}、f_{16}-f_{25} were not completely searched within the allotted time. IPSO completed the search of f_{13} twice in 25 times implementation when Dims=10, and the corresponding *FEs* was listed in the table. For the comparable 13 functions, IPSO gets 8 best outcomes for the function in the cases when dimension is 10. When Dims=30, There are seven functions obtain the best search results, otherwise the PSO without the improvement have not completed the search except *f1*、*f2*、*f4* . It sufficiently proved the effectiveness of the improved strategies.

4. Conclusion

This paper proposes an improved particle swarm optimization algorithm (IPSO) to solve continuous function optimization problems. Two improvement strategies named "Vector correction strategy" and "Jump out of local optimum strategy" were employed in our improved algorithm. The algorithm was tested using 25 newly proposed benchmark instances in Congress on Evolutionary Computation 2005 (CEC2005). For these benchmark problems, the problem definition files, codes and evaluation criteria are available in http://www.ntu.edu.sg/home/EPNSugan. The performance of IPSO was compared with the 11 algorithms published in CEC2005. The experimental results show that the search efficiency and the ability of jumping out from the local optimum of the IPSO have been significantly improved, and the improvement strategies are effective.

Acknowledgments. This work was partially supported by the National Natural Science Foundation of China (No.60970088,60775035), the National High-Tech Research and Development Plan of China (No. 2007AA01Z132), National Basic Research Priorities Programme(No. 2007CB311004) and National Science and Technology Support Plan (No.2006BAC08B06), Dean Foundation of Graduate University of Chinese Academy of Sciences(O85101JM03).

References

[1] R C Eberhart, J Kennedy. A new optimizer using particle swarm theory. Proceedings of the sixth International Symposium on Micro Machine and Human Science. Nagoya Japan, 1995. 39-43

[2] J Kennedy, R C Eberhart. Particle Swarm Optimization. Proc IEEE International Conference on Neural Networks. IEEE Service Center, Piscataway, NJ, IV: 1942-1948.

[3] M Clerc. TRIBES-Aparameter Free Particle Swarm Optimizer. http://clerc.maurice.free.fr/PSO 2002-08-10/2003-10-08

[4] A Salman. Discrete Particle Swarm Optimization for Heterogeneous Task Assignment Problem. Proceedings of World Multiconference on Systemics, Cybernetics and Informatics(SCI 2001). 2001

[5]M Clerc. Discrete Particle Swarm Optimization: A Fuzzy Combinatorial Black Box. http://clerc.maurice.free.fr/PSO/Fuzzy_Discrete_PSO/Fuzzy_DPSO.htm. 2000-04-01/2003-10-08

[6] Hirotaka, Yoshida, Kenichi. A particle Swarm Optimization
for Reactive Power and Voltage Control Considering Voltage Stability. IEEE International Conference on Intelligent System Applications to Power Systems, Rio de Janeiro, 1999

[7] M S Voss, Xin Feng. Arma Model Selection Using Particle Swarm Optimization and Aic Criteria. 15th Triennial World Congress, Barcelona Spain, 2002

[8] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization,2006 available at http://www3.ntu.edu.sg/home/EPNSugan/index_files/CEC-05/CEC05.htm

[9]N. Hansen. Compilation of Results on the 2005 CEC Benchmark Function Set, 2006, available at http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC-05/compareresults.pdf