Particle Swarm Optimization Algorithm Based on Divided-interval Chaotic Search

Qiushi Xu
Shenyang University of Technology
No.58 South Xinghua Street, Tiexi Dist.
Shenyang, 110023, P.R.China
wlss198089@sohu.com

Xiangdong Wang
Shenyang University of Technology,
No.58 South Xinghua Street, Tiexi Dist.
Shenyang, 110023, P.R.China
wangxd2007@163.com

ABSTRACT
Though chaotic particle swarm optimization algorithm lets particles search in the whole variable space, the search scale is too large and the high precision of solution is hard to achieve. This paper proposes a particle swarm optimization algorithm based on divide-interval chaotic search, it lets the particles search in the selected interval, reduces the scope of the search space, and makes the solution more approximate to the global optimum. And the comparable experiment shows that the algorithm has preferable results.

Keywords
Chaos; particle swarm optimization; divided-interval

1. INTRODUCTION
Due to the elicitation of the birds social behavior simulation in artificial intelligence, Kennedy and Eberhart presented a metaheuristic global optimization algorithm — particle swarm optimization(PSO)[1]. PSO algorithm is a stochastic optimiza -tion method based on swarm intelligence[2]. Its basic idea is that first initialize a flock of stochastic particles, each particle is a feasible solution of the optimization problem, and has a fitness value determined by the objective function, and also has a velocity vector to determine its flight direction and distance, then the particles follow the current best particle to search in solution space, the optimum is found through iteration. Based on the above work, Shi and Eberhart presented a particle swarm optimization with inertia weight[3], and introduced the inertia weight coefficient w, it made the exploration and exploitation ability reasonably allocated. They also found dynamic inertia weight has better search results than the fixed weight, and they used the linear decrease inertia weight.

If any particle finds a better solution in PSO algorithm, all particles update towards the best particle with the iteration formula of velocity and position. Once the PSO algorithm finds an optimum, it converges very fast, and is easy to be trapped in local optimum, even increase the iteration number, the precision of solution doesn't increase. The PSO system has no mechanism to definitely find the global optimum, because the particles search randomly, then other particles follow the best particle. Therefore, PSO needs improving and adding other factors to make the particles find the solution more approximate to global optimum Many authors presented methods to improve it, one of them is to add the chaos method[4].

Because of the pseudo-randomness and ergodicity of chaos, chaotic particle swarm optimization(CPSO) algorithm[5,6] should make the particles jump out of the local optimum, and find a better solution. However the existing CPSO algorithms are searching in the whole space of the variable, the search scale is too large, and the high precision of the solution is hard to achieve, so this paper presents a particle swarm optimization algorithm based on divided-interval chaotic search.

2 . ALGORITHM DESCRIPTION
2.1 Description of PSO
Each particle is treated as a point in a D-dimensional space, the position of the ith particle is represented as $X_I = (x_{i1}, x_{i2}, ..., x_{iD})$; The best previous position (the position giving the best fitness value) of any particle is represented as $P_I = (p_{i1}, p_{i2}, ..., p_{iD})$ .The index of the best particle among all the particles is represented by the symbol g. The velocity of particle i is represented as $V_I = (v_{i1}, v_{i2}, ..., v_{iD})$. The particles update their velocity and position according to the following equations[7]:

$$v_{id}^{t+1} = v_{id}^{t} + c_1 r_1 (P_{id} - x_{id}^{t}) + c_2 r_2 (P_{gd} - x_{id}^{t}) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

Where $c1$ and $c2$ are 2 positive constants, $r1$ and $r2$ are 2 random numbers in the range [0,1].

Velocity equation of PSO with inertia weight[3] is:

$$v_{id}^{t+1} = w v_{id}^{t} + c_1 r_1 (P_{id} - x_{id}^{t}) + c_2 r_2 (P_{gd} - x_{id}^{t}) \quad (3)$$

Where w is the inertia weight.

Bionetics'07, Dec.10–13, 2007, Budapest, Hungary.

This paper is supported by Science and Technology research key project of Chinese Ministry of Education (205032).

Linear decrease inertia weight formula[3] is:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{T_{\max}} T \quad (4)$$

Where $w_{\max}$, $w_{\min}$ is respectively the minimum , maximum of w ; T and $T_{\max}$ are the current iteration, the maximal iteration.

2.2 Chaotic Search Description

1) Give the maximal chaotic variable iteration number M: give x an initial value $x_0 = (x_1^0, x_2^0, ..., x_n^0)$, $x_i \in [a_i, b_i]$, calculate the optimized function $f^* = f(x_0)$, and use $t_i^0 = (x_i^0 - a_i)/(b_i - a_i)$ to transfer x 's range from the original range to the chaotic variable's range [0,1], let k＝0,i= 1…n , k is the chaotic variable iteration number, n is the dimension of variable x ; 2)This paper takes Logistics chaos equation as an example to use chaos iteration $t_i^{k+1} = 4t_i^k(1 - t_i^k)$; 3) Use $x_i^{k+1} = a_i + (b_i - a_i)t_i^{k+1}$ to transfer x 's range from [0,1] to the original range; 4) While $k < M$, if $f(x^{k+1}) < f^*$, let $x^* = x^{k+1}$, $f^* = f(x^{k+1})$; (f is the corresponding best function value of chaotic variable); if $f(x^{k+1}) \geq f^*$, x^* and f^* aren't changed; and let k＝k+1, turn to step 2.

2.3 Description of CPSO

Cpso1 algorithm[5] introduces a chaotic variable x in PSO, using chaos iteration to update each particle's position. Compare the function value of chaos iteration $f^c(x^{k+1})$ with each particle's corresponding function value f^* , if $f^c(x^{k+1}) < f^*$, then turn the variable of chaos iteration to the variable range, take the variable value and function value of chaos iteration to replace x^* and f^* ; If $f^c(x^{k+1}) \geq f^*$, x^* and f^* don't change. Then continue the chaos iteration until reach the max loop number. And cpso2 algorithm[6] uses chaos iteration to optimize the global best particle after introducing the variable x .

Cpso1 uses chaos iteration to all particles rulelessly, only to find a better solution, while cpso2 uses chaotic search to update the global best position directly, then use the velocity iteration formula to make all particles move to the global best position, which could make the particles find the optimum more quickly. The chaotic search is used in the whole variable space, so it should make the algorithm jump out of the local optimum, and make the particles move towards the optimum.

2.4 Description of PSO Algorithm Based on Divided-interval Chaotic Search

As the CPSO algorithm is stochastic, the local optimum the particle swarm gets may already be near the optimum, but chaos makes the particles search in the whole space of variable, the condition that the particles may be near the optimum hasn't been used, the search scale is too large and also prolongs the convergence time, so the precision of the solution is too low. Therefore, this paper presents a particle swarm optimization algorithm based on divided-interval chaotic search: dacpso (particle swarm optimization based on divided-interval chaotic research to all particles' position) and dbcpso(particle swarm optimization based on divided-interval chaotic research to best particle's position). They can make CPSO algorithm search in the selected interval, exclude the intervals with no optimum, decrease the search scale, which makes the cpso algorithm find the solution more approximate to the global best position.

Description of dacpso(and dbcpso) algorithm:
1)Divide the range of the variable into several intervals. If the number of intervals is too big, the time will be increased. If the number of intervals is too small, the high precision of the solutions is hard to achieve. This paper chooses 11 intervals; 2) In each interval use the cpso1(cpso2) algorithm, loop 100 times to get the minimum of the optimized function; 3) In step 2's results, choose the interval in which the optimized function is minimal; 4) Use cpso1 (cpso2) algorithm to get minimum in the selected interval, loop 2000 times.

Pseudo-code of dbcpso:

```
initialize;
for i=1:iter   %loop of cpso
{calculate the fitness of each particle;
 update pbest and gbest according to postion of each particle;
    for j=1:D    {T(j,1)=(gbest(j)-xmin)/(xmax-xmin);}
    for k=1:o    %chaos begin
       {for j=1:D
           {T(j,k+1)=4*T(j,k)*(1-T(j,k));
           xc(j)=xmin+(xmax-xmin)*T(j,k+1);}
        fc(k+1)=Rosenbrock(xc); %call function Rosenbrock
        % to calculate the fitness of function
         if (fc(k+1)<gbestval) {gbest=xc;gbestval=fc(k+1);}
        } %chaos end
 update velocity and position with the fomula;}%end of cpso
```

Wherein, iter is the iteration number of particle swarm, o is the iteration number of chaos, D is the dimension of variable, T is the chaos variable, [xmin,xmax] is the range of the variable，xc is the chaos variable, fc is the fitness value of xc, gbest is the global best position, gbestval is the fitness value of gbest.

3. ALGORITHM COMPARISON

3.1 Experiment Design

This paper chooses 5 benchmark functions to optimize, the dimensions, search ranges, and theoretic minimums of these functions are shown in Table 1.

Table 1. Dimension, range and optimum of benchmark functions

Function	Dim	Range	Variable	Min
f1：Rosenbrock	30	[-100,100]	X_i=1	0
f2：Griewank	30	[-600,600]	X_i=0	0
f3：Rastrigin	30	[-5.12,5.12]	X_i=0	0
f4：Ackley	30	[-32,32]	X_i=0	0
f5：DeJong_f4	30	[-100,100]	X_i=0	0

Their equations are：

$$f1=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2) \quad (5)$$

$$f2=1+\sum_{i=1}^{n}(x_i^2/4000)-\prod_{i=1}^{n}\cos(x_i/\sqrt{i}) \quad (6)$$

$$f3=\sum_{i=1}^{n}(10+x_i^2-10\cos(2\pi x_i)) \quad (7)$$

$$f4=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) \quad (8)$$

$$+20+\exp(1)$$

$$f5=\sum_{i=1}^{n}ix_i^4 \quad (9)$$

3.2 Experiment Results and Analysis

We did 50 experiments for each algorithm on computer of P4(2.67GHz)， 256MB memory to obtain the results of table 2-6. Take maximal, minimal and mean value of every algorithm. Where functions f1-f3 use the data from references [8-12] to compare with the data this paper obtained, functions f4, f5 have no reference to compare. The simulation environment is matlab7[13].

Pso algorithm adopts the PSO method with inertia weight[14]. 8bpso denotes bpso[8], 9cpso denotes cpso[9], the rest may be deduced by analogy.

The experiment parameters in this paper are: 2000 times PSO iteration, population size is 30, 2000 times chaos iteration, the inertia weight is:

$$\omega=\begin{cases}((\omega_{\min}-\omega_{\max})/(\omega_e-1))*(i-1)+\omega_{\max}, & for \quad i\le\omega_e\\ \omega_{\min}, & for \quad i>\omega_e\end{cases}$$

where i is the iteration number, ω_e=1500, $\omega_{\max}$=0.9, $\omega_{\min}$=0.4, c1=c2=2, $v_{\max}=x_{\max}/25$, $x_{\max}$ is the maximum of the variable range.

Table 2. Results and comparison of f1

Algo	Time	Min	Max	Mean	Algo	Mean
pso#	2.017	6.4	156.8	47.50	8spso*	140
cpso1#	2506	7.5	80.8	54.07	8tspso*	48
cpso2#	169	1.3	127.4	37.25	9spso*	51
dacpso	3915	4.7	85	39.16	9cpso*	42
dbcpso	1753	9.3	84.9	37.75	10apso*	96

(* means the data are obtained from the references directly, # means the data are from the self-made programs according to the references. The unit of time is second.)

Table 3: Results and comparison of f2

Algo	Time	Min	Max	Mean	Algo	Mean
pso#	3.187	0	0.099	0.0164	8bpso*	150
cpso1#	2510	0	0.061	0.0188	8tspso*	12
cpso2#	192	0	0.064	0.0145	10spso*	0.0182
dacpso	4735	0	0.047	0.0134	10apso*	0.0232
dbcpso	1808	0	0.071	0.0182	11cpso*	0.0554

Table 4. Results and comparison of f3

Algo	Time	Min	Max	Mean	Algo	Mean
pso#	3.359	16.9	51.7	36.63	8bpso*	150
cpso1#	2344	19.9	42.8	29.64	8tpso*	24
cpso2#	180	17.9	64.7	33.64	8tspso*	18
dacpso	4006	0	2.13*10^-14	4.26*10^-15	10apso*	24
dbcpso	1780	0	1.6*10^-14	4.57*10^-15	12cpso*	0.00025

Table 5. Results and comparison of f4

Algo	Time	Min	Max	Mean
pso#	2.235	8.4*10^-12	4.34*10^-9	2.49*10^-10
cpso1#	2403	1.72*10^-11	5.7*10^-10	1.82*10^-10
cpso2#	181	3.41*10^-12	8.4*10^-10	1.19*10^-10
dacpso	3619	2.35*10^-11	4.78*10^-10	1.54*10^-10
dbcpso	1795	1.2*10^-11	1.1*10^-9	1.76*10^-10

Table 6. Results and comparison of f5

Algo	Time	Min	Max	Mean
pso#	2.332	2.19*10^-27	1.35*10^-21	6.03*10^-23
cpso1#	2483	1.02*10^-27	3.7*10^-22	3*10^-23
cpso2#	165	6.96*10^-27	1.01*10^-20	3.15*10^-22
dacpso	3927	6.15*10^-27	2*10^-22	2.64*10^-23
dbcpso	1826	2.1*10^-27	1.11*10^-21	4.12*10^-23

Known from the above results: (1)About computational time, although the time of dacpso and dbcpso increases with regards to the previous algorithms, they get better optimization results in Tables 2-6. Dacpso uses chaos iteration to all particles in the population, while dbcpso only uses chaos iteration to the particle with global best position, so dbcpso is faster than dacpso; (2)About optimization effect, for function f1, the results of dacpso and dbcpso algorithms are respectively 39.16 and 37.75, which are better than those of the references; for f2, the mean value of dacpso—0.0134 is smaller than the data of the references; for f3, dacpso and dbcpso have notable solutions, their maximums and minimums are notably reduced from pso, cpso1 and cpso2. Their minimums reach theoretical optimum 0, and the order of magnitude of their mean value reaches 10^-15, which is much smaller than the best result of the references—0.00025; for f4, all the algorithms in tables are approximate; for f5, dacspo has better solutions than those of all the citing references. (3)About stability, for f1, the mean of cpso2 is approximate to dacpso and dbcpso, but the optimum range of cpso2 [1.3,127.4] is broader than those of dacpso and dbcpso, which indicates cpso2 algorithm has bigger

randomness, and dacpso and dbcpso have better stability; for f2, the maximum of dacpso is smaller than those of cpso1 and cpso2, which means dacpso has higher stability; for f3, since the optimization effect of dacpso and dbcpso are much better than pso, cpso1 and cpso2, there is no need to compare the stability; for f4 and f5, all the results of the tables are good. In short, for f1-f5, , there is at least one of dacpso and dbcpso better than the data of the citing references.

4. CONCLUSIONS

This paper analyzed the reason why pso can't find the global best optimum and why the high precision of solution of cpso algorithm is hard to achieve, then proposes PSO algorithm based on divided-interval chaotic search—dacpso and dbcpso. They search purposely by dividing variable range to several intervals, and increase the precision of the solution. The experiment proves this algorithm is a more effective algorithm.

5. REFERENCES

[1] Kennedy, J., Eberhart, R. *Particle Swarm Optimization* [C]. In： IEEE Int Conf on Neural Network. Perth.Australia. 1995：1942-1948.

[2] Lu Kezhong,Wang Ruchuan,Shuai Xiaoying. *Improving Particle Swarm Optimization by keeping particles activity.* Computer Engineering and Applications[J], 2007,43(11): 35-38.

[3] Shi Y, Eberhart RC. *A modified particle swarm optimizer.* In:Proc. Of the IEEE Int Conf of Evolutionary Computation. Piscataway:IEEE Press,1998:69-73.

[4] Li Bing, Jiang Weisun, *Chaos optimization method and its application,* Control Theory and Applications[J], 1997,14 (4): 613-615.

[5] Dai Dongxue, Wang Qi, *Ruan Yongshun etc. Chaos-based particle swarm optimiztion algorithm and its application,* Huazhong Univ. of Sci.&Tech.(Nature Science Edition)[J], 2005,33(10):53-55,82.

[6] Zhang Jinsong, Li Qiqiang, Wang Zhaoxia. *Hybrid particle swarm optimization based on chaos search,* Journal of Shandong University (Engineering Science) [J],2007,37(1): 47-50.

[7]Yuhui Shi, Russell Eberhart, *A modified particle swarm optimizer*, Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, 1998 :69-73

[8] Hu Wang, Li Zhishu. *A more simple and effective particle swarm optimization algorithm,* Journal of Software[J], 2006,18(4):861-868.

[9] Meng Hongji, Zheng Peng, *Mei Guohui. Particle swarm optimization algorithm based on chaotic series,* Control and Decision[J], 2006,21(3):263-266.

[10] Xiao-Feng Xie,Wen-Jun Zhang, Zhi-Lian Yang. *Adaptive Particle Swarm Optimization on Individual Level,* Proceedings of IEEE ICSP'02 2002:1215-1218

[11] Keiji Tatsumi, Syuhei Sasaki, Tetsuzo Tanino.*Chaotic Particle Swarm Optimization Method Exploiting Sinusoidal Perturbation*s, SICE-ICASE International Joint Conference,2006:6013-6016.

[12] Yang Junjie, Zhou Jianzhong, Yu Jing. *Particle swarm optimization algorithm based on chaos searching,* Computer Engineering and Applications[J],2005(16): 69-71.

[13] Zhang Zhiyong etc.,*Master matlab6.5[M]*, Bei Hang University press, 2003.

[14] Brian Birge, *PSOt-a Particle Swarm Optimization Toolbox for use with Matlab*.IEEE Swarm Intelligence Symposium Proceedings, 2003:182-186.