Cloud Estimation of Distribution Particle Swarm Optimizer

Ying Gao, Xiao Hu, Huiliang Liu, Fufang Li
Department of Computer Science and Technology, Guangzhou University
Guangzhou, 510006, P.R. of China
falcongao@sina.com.cn

***Abstract*-Cloud estimation of distribution particle swarm optimizer combining PSO and cloud model is introduced. In the algorithm's offspring generation scheme, new particles are generated in the cloud estimation of distribution way or in the PSO way. The innovation of the algorithm is production of cloud particles according to the cloud model theory. The cognitive population obtained during optimization is used to estimate statistical characteristics of good solution regions by backward cloud generator. And then the estimated statistical characteristics are used to produce cloud particles by positive cloud generator. Both the global information from cloud particles and local information from PSO particles are used to guide the further search. The proposed algorithm is applied to some well-known benchmarks. The experimental results show that the algorithm has stronger global search ability than original version of PSO.**

***Keywords*-Cloud model; PSO; Backward cloud generator; Positive cloud generator**

I. INTRODUCTION

An optimization problem is the problem to find the optimal or near optimal solution from a specified set of feasible solutions using some measure for evaluating each individual solution. An algorithm to solve such problem is called an optimization algorithm. Over the last decades, there has been a growing interest in algorithms inspired by the behaviors of natural phenomena, for example genetic algorithm, simulated annealing, ant colony search algorithm, particle swarm optimization (PSO) and etc.

PSO works on the social behavior of particles in the swarm. It finds the best solution by simply adjusting the trajectory of each individual toward its own best location and toward the best particle of the entire swarm at generation. PSO is becoming very popular due to its simplicity of implementation and ability to quickly converge to a reasonably good solution[1-4]. However, as demonstrated by Van[5], PSO is not a global convergence guaranteed algorithm, and has less mechanism to extract and use global information about the search space[6].

Cloud model is the innovation and an effective tool in uncertain transforming between qualitative concepts and their quantitative expressions[7]. The cloud model-based optimization algorithm[8] (CMBOA) is a new optimization method. CMBOA directly extracts the global statistical information about the search space from the search so far and builds a cloud distribution model of promising solutions by backward cloud generator. New individuals are generated from the cloud distribution model thus built by forward cloud generator. CMBOA captures global information about promising areas of the search space that can be used to improve the search for better individuals.

The global information can guide the search for exploring promising areas, while the local information of individuals found so far can be helpful for exploitation. An efficient evolutionary algorithm should make use of both the global information about the search space and the local information of solutions found so far. In this paper, a cloud estimation of distribution particle swarm optimizer combining PSO and CMBOA is introduced. In the algorithm's offspring generation scheme, new individuals are generated in the cloud estimation of distribution way or in the PSO way. The algorithm uses information obtained during optimization to build cloud distribution model of good solution regions and use this cloud distribution model to produce part new individuals. In such way, both the global information from cloud particles and local information from PSO particles are used to guide the further search. The algorithm is compared with the version of the original PSO on some well-known benchmarks. The experimental results demonstrate that the algorithm is effective and outperforms the original PSO.

II. CLOUD MODEL

Cloud model is a conversion model with uncertainty between a quality concept which is expressed by natural language and its quantity number expression[7]. If U is a quantity domain expressed with accurate numbers, and C is a quality concept in U, if the quantity value , $x \in \mathrm{U}$ and x is a random realization of the quality concept C, $\mu(x)$ is the membership degree of x to C, $\mu(x) \in [0,1]$, it is the random number which has the steady tendency:

$$\mu : \mathrm{U} \to [0,1], \quad \forall x \in \mathrm{U}, \quad x \to \mu(x)$$

The distribution of x in domain is called cloud model, which is briefly called cloud, each x is called a cloud drop.

The statistical characteristics of cloud model are expressed with Expectation Ex, Entropy En and Super-entropy He, and they reflect the whole characteristics of the quality conception C. Expectation Ex of the cloud drops' distribution in domain reflects the cloud centre of gravity of cloud drops of the concept. Entropy En is the uncertainty measurement of the quality concept. The super-entropy He is the uncertain measurement of entropy, namely the entropy of the entropy.

Backward cloud generator is a conversion model which can convert quantity numbers to a quality concept. It can convert the accurate data $(x_1, x_2, \cdots, x_n)$ with the membership degree $(\mu_1, \mu_2, \cdots, \mu_n)$to the quality cloud concept expressed by statistical characteristic (Ex, En, He) . The algorithm of backward cloud generator can be summed as follows:

(1) According to the samples$(x_1, x_2, \cdots, x_n)$, calculate neam

978-0-7695-4281-2/10 $26.00 © 2010 IEEE
DOI 10.1109/ICGEC.2010.12

value $Ex = Mean(x_1, x_2, \cdots, x_n)$;

(2) According to the samples $(x_1, x_2, \cdots, x_n)$, calculate standard deviation value $En = Stdev(x_1, x_2, \cdots, x_n)$;

(3) calculate $En_i' = \sqrt{-(x_i - Ex)^2 / 2\ln \mu_i}$;

(4) According to the En_i', calculate standard deviation value $He = Stdev(En_1', En_2', \ldots, En_n')$;

According to the three statistical characteristics values (Ex, En, He), the positive cloud generator can generate the cloud drops (x, μ), x is the quantity values, μ is the membership degree of x. The algorithm of positive cloud generator can be summed as follows:

(1) Generate a normally distributed random number En' with mean En and standard deviation He;

(2) Generate a normally distributed random number x with mean Ex and standard deviation absolute value of En';

(3) Calculate $\mu = exp(-(x-Ex)^2 / 2(En')^2)$;

(4) μ is certain degree of x belongs to the qualitative concept, and the cloud drop (x, μ) reflects the whole contents of this qualitative quantitative transform;

(5) Repeat step (1)-(4) until N cloud drops are generated.

III. THE CLOUD MODEL-BASED OPTIMIZATION ALGORITHM

The cloud-based optimization algorithm uses backward cloud generator to estimate three statistical characteristics values of the selected solution from good solution regions during optimization. Then, the next population is produced by forward cloud generator according to three statistical characteristics values of cloud.

Step1 Initialize population $P(0)$ randomly.

Step2 Calculate $\mu_i = f_i \Big/ \sum_k^N f_k$,

where $f_i = fitness(\mathbf{x}_i), i = 1, \ldots, N$

Step3 Select M<N points with the best fitness value from $P(t)$ to form the parent set $P^s(t)$.

Step4 Estimate three statistical characteristics of $P^s(t)$, expected value Ex, entropy En, and hyper-entropy He according to the algorithm of backward cloud generator.

Step5 According to Ex, En and He by using the algorithm of forward cloud generator to generate N new points(cloud drops) to form the population $P'(t)$.

Step6 Select N points with the best fitness value from $P(t) \cup P'(t)$ to form the next generation population $P(t+1)$

Step7 If given stopping condition is not met, goto Step2.

IV. PARTICLE SWARM OPTIMIZATION

In each iteration of original PSO, the swarm is updated by the following equations:

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1 r_1(\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2 r_2(\mathbf{p}_g(t) - \mathbf{x}_i(t)) \quad (1)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (2)$$

Where $\mathbf{p}_i(t) = (p_{i,1}, p_{i,2}, \ldots, p_{i,m})(i = 1, 2, \cdots, N)$ and $\mathbf{p}_g(t)$ are given by the following equations, respectively:

$$\mathbf{p}_i(t+1) = \begin{cases} \mathbf{p}_i(t), & f(\mathbf{x}_i(t+1)) < f(\mathbf{p}_i(t)) \\ \mathbf{x}_i(t+1), & f(\mathbf{x}_i(t+1)) \geq f(\mathbf{p}_i(t)) \end{cases} \quad (3)$$

$$\begin{aligned} \mathbf{p}_g(t) \in \{&\mathbf{p}_1(t), \mathbf{p}_2(t), \cdots, \mathbf{p}_N(t) \mid f(\mathbf{p}_g(t)) \\ &= \min\{f(\mathbf{p}_1(t)), f(\mathbf{p}_2(t)), \cdots, f(\mathbf{p}_N(t))\}\} \end{aligned} \quad (4)$$

N is the number of particle. $\mathbf{x}_i (i = 1, 2, \cdots, N)$ and $\mathbf{v}_i (i = 1, 2, \cdots, N)$ is position vector and velocity vector of ith particle respectively in m-dimensional search space. w is called an inertia weight. c_1 and c_2 are acceleration coefficients which control how far a particle will move in a single iteration. r_1 and r_2 are elements from two uniform random sequences in the range $[0,1]$. $f(\mathbf{x})$ is the minimum objective function.

$\{\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_N\}$ is called as cognitive population. In the classical PSO, particles depend on their individual memory and peer influence to explore the search space. However, the swarm as a whole does not use its collective experience (represented by the array of previous best positions) to guide its search. This causes re-exploration of already known bad regions in the search space.

V. CLOUD ESTIMATION OF DISTRIBUTION PARTICLE SWARM OPTIMIZER

This paper proposes an approach in which cognitive population $\{\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_N\}$ is used to estimate good solution regions and generate new particles in the search space. The algorithm uses firstly backward cloud generator to estimate three statistical characteristics values of the cognitive population $\{\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_N\}$ during optimization. Then, the new particles are produced by positive cloud generator according to three statistical characteristics values of cloud. The implementation of the optimization algorithm as follows:

Step1 Initialize particles position, velocity and certain degree;

Step2 Update personal best $\mathbf{p}_i$ and global best $\mathbf{p}_g$;

Step3 Calculate velocity and update particle position, get PSO-particle;

Step4 Estimate three statistical characteristics of cognitive population $\{\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_N\}$, expected value **Ex**, entropy **En**, and hyper-entropy **He** according to the algorithm of backward cloud generator;

Step5 Generate N Cloud-particles according to **Ex**, **En** and **He** by using the algorithm of positive cloud generator;

Step6 Form the next generation population by comparing PSO-particle and Cloud-particle;

Step7 If the given stopping condition is not met, goto Step2;

The pseudocode for the algorithm is presented in Fig. 1.

```
FOR each particle i
  FOR each dimension d
    Initialize position x_{i,d} randomly within permissible range
    Initialize velocity v_{i,d} randomly within permissible range
  End FOR
    Initialize certain degree μ_i randomly within [0, 1] range
END FOR
Iteration t=1
DO
  FOR each particle i
    Calculate fitness value
    IF the fitness value is better than P_i in history
      Set current fitness value as the P_i
    END IF
  END FOR
  Choose the particle having the best fitness value as the P_g
  FOR each particle i
    Calculate velocity according to the equation
    v_i(t+1) = w v_i(t) + c_1 r_1 (p_i(t) - x_i(t)) + c_2 r_2 (p_g(t) - x_i(t))
    Update particle position according to the equation,
    x_i(t+1) = x_i(t) + v_i(t+1)  get PSO-particle
  END FOR
  Estimate three digital characteristics of cognitive population
  {p_1, p_2, ...p_N}, expected value Ex = (Ex_1, Ex_2, ..., Ex_m),
  entropy En = (En_1, En_2, ..., En_m), and hyper-entropy
  He = (He_1, He_2, ..., He_m) according to the algorithm of backward
  cloud generator:
    Ex_d = Mean(p_{1,d}, p_{2,d}, ..., p_{N,d}), (d = 1,..., m);
    En_d = Stdev(p_{1,d}, p_{2,d}, ..., p_{N,d}), (d = 1,..., m);
    En'_{i,d} = sqrt(-(p_{i,d} - Ex_d)^2 / 2 ln μ_i), (d = 1,...,m, i = 1,...,N);
    He_d = Stdev(En'_{1,d}, En'_{2,d}, ..., En'_{N,d}), (d = 1,..., m)
  Generate N Cloud-particle according to Ex, En and He by using the
  algorithm of positive cloud generator:
  FOR i=1 to N
    En'_d = Norm(En_d, He_d), (d = 1,..., m);
    x_{i,d} = Norm(Ex_d, |En'_d|), (d = 1,..., m);
    μ_i = exp(-Σ_{d=1}^{m} [(x_{i,d} - Ex_d)^2 / 2(En'_d)^2]);
  END FOR
  FOR each particle i
    Evaluate fitness of ith PSO-Particle
    Evaluate fitness of ith Cloud-Particle
    IF fitness (PSO-Particle) < fitness (Cloud-Particle)
      x_i(t+1) = PSO-Particle
    ELSE
      x_i(t+1) = Cloud-Particle
    ENDIF
  END FOR
t=t+1
WHILE maximum iterations or minimum error criteria are not attained
```

Fig. 1. Pseudocode for the proposed algorithm

VI. RESULTS FROM SIMULATIONS

In the section, the performance of the proposed algorithm is compared with that of the original particle swarm optimization with weight factor. The following some well-known benchmark functions have been used to test. For each of these functions, the goal is to find the global minimizer, formally defined as

Given $f : \mathrm{R}^M \to \mathrm{R}$

find $\mathbf{x}^* \in \mathrm{R}^M$ such that $f(\mathbf{x}^*) \le f(\mathbf{x}), \forall \mathbf{x} \in \mathrm{R}^M$

A. Sphere function, defined as

$$f_1(\mathbf{x}) = \sum_{i=1}^{M} x_i^2 \text{ , where } \mathbf{x}^* = \mathbf{0} \text{ and } f(\mathbf{x}^*) = 0 \text{ for}$$

$-100 \le x_i \le 100$

B. Rastrigin function, defined as

$$f_2(\mathbf{x}) = \sum_{i=1}^{M} (x_i^2 - 10\cos(2\pi x_i) + 10), \text{ where } \mathbf{x}^* = \mathbf{0}$$

and $f(\mathbf{x}^*) = 0$ for $-5.12 \le x_i \le 5.12$

C. Griewank function, defined as

$$f_3(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{M} x_i^2 - \prod_{i=1}^{M} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \text{ where } \mathbf{x}^* = \mathbf{0}$$

and $f(\mathbf{x}^*) = 0$ for $-600 \le x_i \le 600$

D. Rotated hyper-ellipsoid function, defined as

$$f_4(\mathbf{x}) = \sum_{i=1}^{M} \left(\sum_{j=1}^{i} x_j \right)^2 \text{ , where } \mathbf{x}^* = \mathbf{0} \text{ and } f(\mathbf{x}^*) = 0 \text{ for}$$

$-100 \le x_i \le 100$

E. Schwefel's Problem 2.22 (Yao et al., 1999), defined as

$$f_5(\mathbf{x}) = \sum_{i=1}^{M} |x_i| + \prod_{i=1}^{M} |x_i| \text{, where } \mathbf{x}^* = \mathbf{0} \text{ and } f(\mathbf{x}^*) = 0$$

for $-10 \le x_i \le 10$

F. Schwefel's function 6, defined as

$$f_6(\mathbf{x}) = \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2} + 0.5, \text{ where } \mathbf{x}^* = \mathbf{0}$$

and $f(\mathbf{x}^*) = 0$ for $-100 \le x_i \le 100$

G. Schwefel's function 2.2.1, defined as

$$f_7(\mathbf{x}) = \max_{i=1}^{M} \{|x_i|\}, \quad \text{where } \mathbf{x}^* = \mathbf{0}$$

and $f(\mathbf{x}^*) = 0$ for $-100 \le x_i \le 100$

H. Ackley's function, defined as

$$f_8(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{M} x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{M}\cos(2\pi x_i)\right) + 20 + e$$

, where $\mathbf{x}^* = \mathbf{0}$ and $f(\mathbf{x}^*) = 0$ for $-32 \le x_i \le 32$

I. Rosenbrock function, defined as

$$f_9(\mathbf{x}) = \sum_{i=1}^{M-1} \left(100\left(x_i - x_{i-1}^2\right)^2 + \left(x_{i-1} - 1\right)^2\right),$$

Where $\mathbf{x}^* = (1, \ 1, \ \cdots \ ,1)$ and $f(\mathbf{x}^*) = 0$ for $-30 \le x_i \le 30$

Table 1 Statistical Analyses for The Proposed Algorithm

	Dim	Average	Standard Deviation
f_1	2	2.9724e-319	0
	30	1.8668e-53	3.7816e-53
f_2	2	0	0
	30	160.9742	259.0169
f_3	2	0	0
	30	0	0
f_4	2	4.3771e-195	0
	30	2.2019e-4	3.5104e-4
f_5	2	4.3151e-158	3.2689e-158
	30	2.3966e-24	3.7542e-24
f_6	2	0	0
f_7	2	1.4504e-155	2.6624e-155
	30	8.4478	15.5123
f_8	2	8.8818e-16	1.4043e-15
	30	20.3972	32.2994
f_9	2	0	0
	30	36.3735	69.6862

Table 2 Statistical Analyses for The original PSO Algorithm

	Dim	Average	Standard Deviation
f_1	2	9.0085e-97	2.4474e-96
	30	1.5086e3	2.7751e3
f_2	2	0	0
	30	2.0259e3	3.1501e3
f_3	2	0.0044	0.0062
	30	1.3233	2.0579
f_4	2	1.6843e-97	3.1357e-97
	30	2.2078e3	3.8512e3
f_5	2	7.1798e-49	8.6944e-49
	30	409.8087	674.8927
f_6	2	0.0058	0.0111
f_7	2	2.0667e-48	3.5654e-48
	30	16.9113	27.1379
f_8	2	8.8818e-16	1.4043e-15
	30	20.9893	33.1823
f_9	2	0	0
	30	2.9432e7	5.2212e7

For original PSO algorithms, $w=0.72$. and $c_1 = 1.49$ and $c_2 = 1.49$. These values have been shown to provide very good results[2-4]. All test functions have a global minimum with a fitness value of 0. Population size N=50. All functions were implemented in 2, 30 dimensions except for the two-dimensional Schwefel's function 6. The initial population was generated from a uniform distribution in the ranges specified below. The initial certain degree μ_i was generated from a uniform distribution within [0, 1] range. All experiments were repeated for 50 runs. The maximum number of iterations is set to 1000 in each running. Tables 1 listed the optimal fitness values and standard deviation of the new algorithm averaged over 50 trails on f_1 - f_9 functions. Tables 2 listed the optimal fitness values and standard deviation of the original PSO algorithm averaged over 50 trails on f_1 - f_9 functions. As shown in the Table1 and Table2, the performance of the proposed algorithm is better that of the original particle swarm optimization with weight factor for all test functions.

VII. CONCLUSION

In this paper, we proposed a cloud estimation of distribution particle swarm optimizer combining PSO and cloud model which is an effective tool in uncertain transforming between qualitative concepts and their quantitative expressions. In the algorithm's offspring generation scheme, new particles are generated in the PSO way or in the cloud model way. Both the local information from PSO particles and global information from cloud particles are used to guide the further search. The backward cloud generator is used to estimate three statistical characteristics of the cognitive population obtained during optimization. And, the positive cloud generator is used to produce cloud particles according to three statistical characteristics of the cognitive population. The proposed algorithm is applied to some well-known benchmarks. The relative experimental results show that the algorithm is effective and has stronger global search ability than original version of PSO. However, there are many issues which require further attention and experiments such as analysis of the algorithm's behaviors and the comparison with other optimization algorithm.

REFERENCES

[1] Van den Bergh, F., Engelbrecht, A study of particle swarm optimization particle trajectories. Information Sciences 176 (8), 937–971, 2006.

[2] Chao-Hsing Hsu, Wen-Jye Shyr, and Kun-Huang Kuo. Optimizing Multiple Interference Cancellations of Linear Phase Array Based on Particle Swarm Optimization. Journal of Information Hiding and Multimedia Signal Processing, Vol. 1, No. 4, 292-300, October 2010.

[3] Chao-Li Sun, Jian-Chao, Jeng Shyang Pan, An Improved Particle Swarm Optimization with Feasibility Based Rules for Constrained Optimization Problems, IEA/AIE 2009, pp. 202-211, 2009

[4] Jui-Fang Chang, Shu-Chuan Chu, John F. Roddick and Jeng-Shyang Pan, A Parallel Particle Swarm Optimization Algorithm with Co mmunication Strategies, Journal of Information Science and Engineering, Vol. 21, No. 4, pp. 809-818, 2005

[5] Van den Bergh, F.: An Analysis of Particle Swarm Optimizers. PhD Thesis. University of Pretoria, South Africa (2001)

[6] Ying Gao, Zhaohui LI, Hui Zheng Extended Particle Swarm Optimiser with Adaptive Acceleration Coefficients and Its Application in Nonlinear Blind Source Separation, ICONIP 2006, Part II, LNCS 4233, 2006.10, 1174–1182.

[7] Deyi, L., Yi, D.. Artificial intelligence with uncertainty. Chapman & Hall, 376pp, 2005.

[8] Ying Gao, An Optimization Algorithm Based on Cloud Model， IEEE International Conference on Computational Intelligence and Security (CIS09)，Vol.2,84-87.