Seeker Optimization Algorithm

Chaohua Dai and Weirong Chen
School of Electrical Engineering
Southwest Jiaotong University
610031 Chengdu, China
dchzyf@126.com

Yunfang Zhu
Department of Computer Engineering
E'mei Campus, SWJTU
614202 E'mei, China
zyfdch@126.com

Abstract

A novel algorithm called seeker optimization algorithm (SOA) for the real-parameter optimization is proposed in this paper. SOA is based on the concept of simulating the act of human randomized search. In the SOA, after given center point, search direction, search radius, and trust degree, every seeker moves to a new position (next solution) from his current position based on his historical and social experience. In this process, the update formula is like Y-conditional cloud generator. The algorithm's performance was studied using several typically complex functions. In all cases studied, SOA is superior to continuous genetic algorithm (CGA) and particle swarm optimization (PSO) greatly in terms of optimization quality, robustness and efficiency.

1. Introduction

The evolutionary computation (EC) community has shown a significant interest in optimization for many years. In particular, there has been a focus on global optimization of numerical, real-valued 'black-box' problems for which exact and analytical methods do not apply. Recently, real-parameter genetic algorithm (GA) [1,2], particle swarm optimization (PSO) [3] and differential evolution (DE) [4] have been introduced and particularly PSO has received increased interest from the EC community. These techniques have shown great promise in several real-world applications. However, the diversity of algorithms is encouraged by the 'No Free Lunch' theorem [5,6], and it is valuable to propose new algorithms.

Optimization problems can often be simplified to the search for an optimal solution through a range of possible solutions. In the continuous decision variable spaces, there exists a neighborhood region close to the global extremum. In this region, the fitness values of the decision variables are inversely proportional to their distances from the global extremum. It can be believed that one must find the near optimal solutions in the narrower neighborhood of the point with higher fitness value, while he must find them in the wider neighborhood of the point with lower fitness value. In the processes of producing and living, human randomized searching behavior often occurs. Let's take an example. The black boxes are seen as the key to discovering the cause of the tragedy after air crash, and recovery operation for them is a kind of human randomized searching behavior. Regarding debris densities as the existential probability of black boxes in the crash scene, when finding the flight recorders, the behavior rule adopted is that the larger debris density is, the larger the existential probability is. At the same time, search group know how to follow the direction leading to a space with larger debris densities. Besides, any individual will mutually interact in the search group. The algorithm called seeker (or searcher) optimization algorithm (SOA) presented in this paper aims to mimic the behavior of the search group and their means of information exchange to solve real-parameter optimization problems.

Apparently, the behavior rule mentioned above is described by natural linguistic term. In order to exploit it, cloud theory [7], as a model of the uncertainty transition between a linguistic term of a qualitative concept and its quantitative data is introduced into new algorithm. The cloud theory is derived and advanced from fuzzy logic theory, but improves the weakness of rigid specification and too much certainty, which comes into conflict with the human recognition process, appearing in commonly used transition models. The preservation of the uncertainty in transition makes cloud theory well meet the need of real life situation, and has already been used successfully in intelligent control [8], data mining [9], etc..

Unlike GA, SOA searches the optimal solutions around the current positions until to converge to the optimum for as few generations as possible. In this way, it does not easily get lost and is able to locate the region in which the global optimum exists, and then to converge to the optimum.

This paper is organized as follows. Section 2 describes cloud theory. In section 3, we introduce the SOA in details. And the algorithm parameters are discussed in section 4. Convergence analysis is shown in section 5. Then, we compare the SOA with continuous GA and PSO by use of typical function optimization in section 6. Finally, the conclusions and future work are presented in section 7.

2. Cloud theory

The cloud theory is based on the cloud model defined as follows.

DEFINITION 1 [7,9] Let U be the set, $U=\{u\}$, as the universe of discourse, and T a linguistic term associated with U. The membership degree of u in U to the linguistic term T, $C_T(u)$, is a random number with a stable tendency. A cloud is a mapping from the universe of discourse U to the unit interval [0,1]. That is, $C_T(u)$: $U \rightarrow [0,1]$; $\forall u \in U$, $u \rightarrow C_T(u)$.

In the definition above, the mapping from U to the interval [0,1] is a one-point to multi-point transition, which shows the uncertainty. So the degree of membership of u to [0,1] is a probability distribution rather than a fixed value, which is different from the fuzzy logic.

The normal clouds are most useful in representing linguistic terms of vague concepts because normal distributions have been supported by the results in every branch of both social and natural sciences. A normal cloud is defined with three digital characteristics, expected value Ex, entropy En and hyperentropy He (Figure 1).

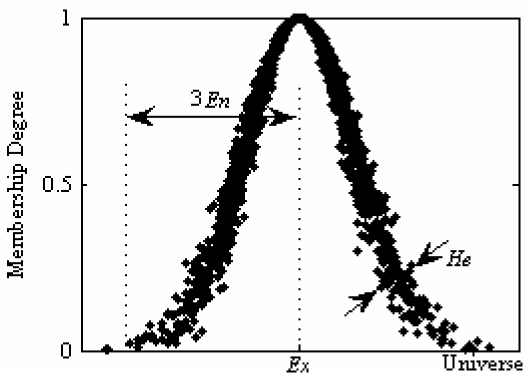

Figure 1. Illustration of the three digital characteristics of a normal cloud

Ex is the position at U corresponding to the center of gravity of the cloud. En is a measure of the coverage of the concept within the universe of discourse. He is the entropy of the entropy En, and is a measure of dispersion of the cloud drops.

Given the three parameters (Ex, En, He) of a normal cloud model, the cloud with n drops is generated by the following algorithm called basic normal cloud generator [9].

```
Algorithm 1 Basic normal cloud generator
    INPUT: $Ex$, $En$, $He$, $n$
    OUTPUT: $\{(x_1, \mu_1), \cdots, (x_n, \mu_n)\}$
    FOR $i$ =1 to $n$
        $En'$ =RANDN($En$, $He$)
        $x_i$ =RANDN($Ex$, $En'$)
        $\mu_i = e^{\frac{-(x_i - Ex)^2}{2(En')^2}}$
        point($x_i$, $\mu_i$)
```

Here, the function RANDN(a,b) produces a normally distributed random number with mean a and standard deviation b. point(x_i, μ_i) is the ith cloud drop in the universe.

For a certain element x_0 in the universe of discourse U, the membership degree, μ, to a linguistic term $T(Ex, En, He)$, is a random number calculated by X-conditional cloud generator [9]. On the other hand, for a certain membership degree, μ_0, to a linguistic term $T(Ex, En, He)$, the corresponding element in the universe is also a random number calculated by Y-conditional cloud generator [9].

3. Seeker optimization algorithm

In SOA, every seeker has a center position vector $\vec{c}$, which is described by expected value Ex of cloud model, as the start location to find next solution. Moreover, each seeker holds a search radius $\vec{r}$ which is equivalent to the En' of cloud model, a trust degree $\vec{\mu}$ described by membership degree of cloud model, and a search direction $\vec{d}$ showing him where to go.

At each time step t, the search decision-making is conducted to choice the four parameters and the seeker moves to a new position $\vec{x}(t+1)$. The update of the position from the center position is determined by a like Y-conditional cloud generator:

$$\vec{x}(t+1) = \vec{c} + \vec{d} \bullet \vec{r} \bullet \sqrt{-\log(\vec{\mu})} \tag{1}$$

The pseudocode of the main algorithm is presented as follows.

```
1.  $t \leftarrow 0$
2.  Initialization generating $S$ positions
```

```
   $\{x_i(t) | x_i(t) = (x_{i1}, x_{i2}, \cdots, x_{iD}), i = 1,2,\cdots,S, t = 0\}$
   randomly and uniformly in the parametric space.
3. **Evaluate** each seeker: Computing the fitness.
4. **Search strategy** giving search parameters
   including center position vector, search direction,
   search radius, and trust degree.
5. **Position update** new position of each seeker is
   simply calculated by (1).
6. $t \leftarrow t+1$
7. if $t<T_{\max}$, then Go to 3; Else Stop.
```

4. Algorithm parameters

In this section, we introduce how to decide the parameters in (1).

4.1. Center point vector

Intuitively, center position vector $\vec{c}$ is set to current position $\vec{x}(t)$. Inspired by PSO, Every seeker contains a memory storing its own best position so far $\vec{p}$ and a global best position $\vec{g}$ obtained through communication with its fellow neighbor seekers. Then,

$$\vec{c} = \vec{x}(t) + \phi_1(\vec{p}(t) - \vec{x}(t)) + \phi_2(\vec{g}(t) - \vec{x}(t)) \qquad (2)$$

where ϕ_1 and ϕ_2 are real numbers chosen uniformly and randomly in a given interval [0,1].

4.2. Search direction

In our opinion, each seeker has four significative directions called local temporal direction $\vec{d}_{lt}$, local spacial direction $\vec{d}_{ls}$, global temporal direction $\vec{d}_{gt}$, global spacial direction $\vec{d}_{gs}$, respectively.

$$\vec{d}_{lt} = \begin{cases} sign(\vec{x}(t) - \vec{x}(t-1)) & \text{if } fit(\vec{x}(t)) \geq fit(\vec{x}(t-1)) \\ sign(\vec{x}(t-1) - \vec{x}(t)) & \text{if } fit(\vec{x}(t)) < fit(\vec{x}(t-1)) \end{cases} \qquad (3)$$

$$\vec{d}_{ls} = sign(\vec{x}'(t) - \vec{x}(t)) \qquad (4)$$

$$\vec{d}_{gt} = sign(\vec{p}(t) - \vec{x}(t)) \qquad (5)$$

$$\vec{d}_{gs} = sign(\vec{g}(t) - \vec{x}(t)) \qquad (6)$$

where $sign(\cdot)$ is signum function, $\vec{x}'(t)$ is the position of the seeker with the largest fitness in a given neighborhood region, $fit(\vec{x}(t))$ is the fitness function of $\vec{x}(t)$.

Then, search direction is assigned depending on the four directions. In our experiments in this paper, we give search direction as follows.

$$\vec{d} = sign(\omega(sign(fit(\vec{x}(t)) - fit(\vec{x}(t-1))))(\vec{x}(t) - \vec{x}(t-1)) + \varphi_1(\vec{p}(t) - \vec{x}(t)) + \varphi_2(\vec{g}(t) - \vec{x}(t))) \qquad (7)$$

where ω is the inertia weight, φ_1 and φ_2 are real numbers chosen uniformly and at random in a given interval [0,1].

4.3. Search radius

It is crucial but difficult how to rationally give search radius. For unimodal optimization problems, the performance of algorithm maybe is relatively insensitive to search radius within certain range. But for multimodal problems, different search radius may result to different performance of algorithm especially when dealing with different problems. In this paper, we introduce two methods to give search radius.

The first of two methods is as follows.

```
ALGORITHM 2  The first method of search radius
1. $c_1 = c_{1\_\max} - (c_{1\_\max} - c_{1\_\min})\frac{S - Sn}{S-1}$
2. $En = (\vec{v}_{\max} - \vec{v}_{\min}) / c_1$
3. $He = En/c_2$
4. $\vec{r}' = \text{RANDN}(En, He)$
5. $\vec{r} = RAND(0, \vec{r}')$
```

where $\vec{v}_{\max}$ and $\vec{v}_{\min}$ are the vector of upper bounds and the vector of lower bounds in the decision variable space, respectively. c_1 and c_2 are the controllable parameters. Based on "3*En*" rule [10] which shows that the elements beyond $Ex \pm 3En$ in the universe of discourse can be neglected for a linguistic atom [8], we adopt $6 \leqslant c_1 \leqslant 3S$ and let it adaptively tuned, where S is the neighbor search group size, and Sn is the sequence number of $\vec{x}(t)$ after sorting the fitnesses of neighbor seekers in ascending order. We suggest $5 \leqslant c_2 \leqslant 15$, and adopted $c_2=10$ in our experiments. $RAND(0, \vec{r}')$ is given as real numbers chosen uniformly and randomly in a given interval $[0, \vec{r}']$.

The second of two methods is as follows (adopted in our experiments).

```
ALGORITHM 3 The second method of search radius
1. $En = \vec{x}_{\max} - \vec{x}_{\min}$
2. $He = En/c_2$
3. $\vec{r}' = RANDN(En, He)$
4. $\vec{r} = RAND(0, \vec{r}')$
```

where $\vec{x}_{\max}$ and $\vec{x}_{\min}$ are the positions with the maximum fitness and the minimum fitness within its fellow neighbor, respectively. In this paper, we create k=3 neighbor clusters based on the indexes of the seeker group, and $\omega=(T_{\max}-t)/T_{\max}$.

4.2. Trust degree

The parameter μ is viewed as quality evaluation of different positions. It is directly proportional to the fitness of $\vec{x}(t)$ or the index of the ascensive sort order of the fitness of $\vec{x}(t)$ (we applied the latter in our experiments). That is, the global best position has the maximum μ=1.0, while other position has a μ<1.0. The expression by us is presented as (8).

$$\mu = \mu_{\max} - \frac{S - Sn}{S - 1}(\mu_{\max} - \mu_{\min}) \tag{8}$$

In (8), $\mu_{\max}$ and $\mu_{\min}$ are the maximum and the minimum μ given by user. We adopt $\mu_{\max}$=1.0, and $\mu_{\min}$=0.2.

5. Convergence analysis

From (2) and (8), when $\vec{x}_i(t) = \vec{g}(t)$, $1 \leqslant i \leqslant S$, it is apparently given that $\vec{c}_i(t) = \vec{g}(t)$ and $\mu_i(t) = 1.0$. Then (1) gives $\vec{x}_i(t+1) = \vec{g}(t)$ and $fit(\vec{x}_i(t+1)) = fit(\vec{x}_i(t)) = fit(\vec{g}(t))$. Hence, the maximum fitness of the t+1 step is larger than or, at least, equal to the maximum fitness of the t step. As a result, the SOA is convergent. But it is not determinate that the algorithm can be convergent to the global optimum.

6. Function optimization

In this section, we will discuss the experiments that we have conducted to compare the performance of the SOA, PSO and continuous genetic algorithm (CGA). We used the MATLAB codes of PSO with adaptive inertia weight and CGA presented by [2].

In this research, we have employed 6 typical functions with varying complexities and varying number of variables (NV). They are as follows.

F1 Goldstein-Price function

$$F1 = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)], -2 \le x_i \le 2, i = 1,2 \tag{9}$$

The function has only one minimum f(0,-1) =3 in its whole solution space.

F2 DeJong's $f2$

$$F2 = 100(x_1^2 - x_2)^2 + (1 - x_1)^2, -2.048 \le x_{1,2} \le 2.048 \tag{10}$$

This function has only one minimum in its whole solution space. But it is an abnormal function and is not optimized easily. Its global minimum is f(1,1) =0.

F3 DeJong's $f5$

$$F3 = 1/(0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}), \quad -65.536 \le x_{1,2} \le 65.536 \tag{11}$$

This is a spiky function (also known as Shekel's foxholes) with 25 sharp spikes of varying heights. Its global maximum is f(-32,-32)= 0.998004.

F4 DeJong's $f6$

$$F4 = 0.5 + \frac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{\left[1.0 + 0.001(x_1^2 + x_2^2)\right]^2}, -100 \le x_{1,2} \le 100 \tag{12}$$

This is a rapidly varying multimodal function with two variables, and is symmetric about the origin with the height of the barrier between innumerable adjacent minima increasing as the global optimum is approached. f(0,0)=0 is its global minimum.

F5 DeJong's $f7$

$$F5 = (x_1^2 + x_2^2)^{0.25}[\sin^2(50(x_1^2 + x_2^2)^{0.1}) + 1.0], \quad -100 \le x_i \le 100, i = 1,2 \tag{13}$$

This function is also similar to F_5, but has the barrier height between adjacent minima approaching zero as the global optimum is approached.

F6 Griewangk's function

$$F7 = \sum_{i=1}^{D} \frac{x_i^2}{4000} + \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1, -512 \le x_{1\sim D} \le 512 \tag{14}$$

This function has many local minima, making the global optimum very hard to find.

As a measure of performance, we consider the average number of generations (AN) that the algorithms require to generate a solution with a certain high fitness value. The average number of generations is obtained by performing the experiment repeatedly (in our case, 10 times) with different and randomly chosen initial populations. In order to compare the ability to prevent the convergence of the algorithms to a local optimum, we also evaluate the performance of the algorithms in terms of the number of runs (NR) (out of 10 trials) for which the algorithms get stuck at a local optimum. When the algorithm fails to achieve the near global optimum, that is, the absolute value of the best function value minus the ideal function value is larger than 0.0001, after a maximum number of generations (MNG), we conclude that it has gotten stuck at a local optimum and does not generate a solution with a certain high fitness value. Besides, the best function values (BV) of repetitious experiments, the average values of the best solutions (AV), and the standard deviations of the function values of the best solutions (STD) are also compared.

In all our experiments, we have used a population size of 100 for all functions. The results are presented in Table 1.

Table 1. Comparison of performance of SOA, CGA and PSO

Function		F1	F2	F3	F4	F5	F6
NV		2	2	2	2	2	10
MNG		1000	1000	1000	1000	3000	3000
Ideal		3	0	0.998004	0	0	0
AN	CGA	153.8	902.8	115.2	191	3000	3000
	PSO	261.2	230.2	327.2	223	1084.3	3000
	SOA	28.2	105	142.6	15.4	104.5	672.7
NR	CGA	0	0	0	1	10	10
	PSO	0	0	1	0	0	10
	SOA	0	0	0	0	0	0
BV	CGA	3.000000	5.2587e-005	0.998	0	0.00029844	0.53597
	PSO	3.000000	8.1130e-024	0.998004	0	2.8507e-69	0.036907
	SOA	3.000000	0	0.998004	0	0	0
AV	CGA	3.000000	0.011178	0.998	1.4098e-05	0.0073763	0.81229
	PSO	3.000000	4.0302e-13	1.097407	0	4.8940e-64	0.101883
	SOA	3.000000	0	0.998004	0	0	0
STD	CGA	1.6922e-05	0.014291	1.9642e-10	4.4572e-05	0.010015	0.17299
	PSO	3.5248e-15	1.1950e-12	0.314339	0	1.4540e-63	0.046745
	SOA	1.6813e-15	0	1.4803e-16	0	0	0

As seen from Table 1, the SOA outperforms the CGA and PSO for all the problems at all aspects. For all the functions optimized here, and the values of the BVs and AVs of SOA are better than that of CGA and PSO, especially the values of NRs of SOA are zeros, which shows SOA has better potential to get the global optimum. Besides, the values of ANs of SOA are greatly less than that of CGA and PSO, which shows SOA has higher convergent speed. Moreover, SOA has less STDs, which shows SOA is more robust.

7. Conclusions and future work

In this research, a novel optimization algorithm based on the concept of simulating the act of human randomized search is introduced whose performance in terms of robustness and efficiency is studied with a challenging set of benchmark problems. The SOA performed very well, converging to near global optimal solutions when solving different classes of problems with different degrees of complexities. In all cases studied, SOA was faster than CGA and PSO, and more robust and efficient in finding the global optimum.

Future research will include practical applications and theoretical analysis to better understand this algorithm's convergence properties and the effects of the parameters on its performance.

References

[1] Kalyanmoy Deb, Ashish Anand, and Dhiraj Joshi. A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization. *Evolutionary Computation*, 2002, 10(4), pp. 371-395.

[2] Randy L. Haupt, Sue Ellen Haupt. *Practical Genetic Algorithms*. 2nd edn. New Jersey: John Wiley & Sons, Inc., pp. 215-228 , 2004.

[3] J. Kennedy and R. C. Eberhart. Panicle Swarm Optimization. In: *Proceeding of the 1995 IEEE Internitional Conference on Neural Networks*, Vol. 4. pp. 1942-1948. IEEE Press ,1995.

[4] R. Storn and K. Price. *Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces.* Technical report. International Computer Science Institute, Berkley ,1995.

[5] D. W. Wolpen and W. G. Macready. No Free Lunch Theorem for Optimization. *IEEE Trans. Evol. Comp.*, 1997, vol. 1(1), pp. 67-82.

[6] Mario Köppen. No-Free-Lunch Theorems and the Diversity of Algorithms. In: *Proceedings of the 2004 Congress on Evolutionary Computation*, IEEE, Vol. 1, pp. 235- 241, 2004.

[7] Deyi Li,Haijun Meng,and Xuemei Shi. Membership Clouds and Membership Cloud Generators". *Journal of Computer Research and Development* (in Chinese), 1995,42(8) , pp. 32-41.

[8] Deyi Li, D.W. Cheung, Xuemei Shi, etc., Uncertainty Reasoning Based on Cloud Models in Controllers. *Computers and Mathematics with Applications*, Elsevier Science, 1998, 35(3), pp. 99-123.

[9] Deren Li, Kaichang Di, Deyi Li. Knowledge Representation and Uncertainty Reasoning in GIS Based on Cloud Models. In: *Proceeding of the 9th International Symposium on Spatial Data Handling*, Beijing, 2000, pp. 10-12.

[10] Liu Changyu, Li Deyi, Pan Lili. Uncertain Know-ledge Representation Based on Cloud Model. *Computer Engineering and Applications* (in Chinese), 2004, 40(2), pp. 32-35.