10.1007/978-3-540-74377-4_18

# Seeker Optimization Algorithm

Chaohua Dai[1], Yunfang Zhu[2], and Weirong Chen[1]

[1] The School of Electrical Engineering, Southwest Jiaotong University, 610031 Chengdu, China
dchzyf@126.com
[2] Department of Computer & Communication Engineering, E' mei Campus, Southwest Jiaotong University, 614202 E' mei, China
zyfdch@126.com

**Abstract.** A novel swarm intelligence paradigm called seeker optimization algorithm (SOA) for the real-parameter optimization is proposed in this paper. The SOA is based on the concept of simulating the act of humans' intelligent search with their memory, experience, and uncertainty reasoning. In this sense, the individual of this population is called seeker or searcher just from which the new algorithm' name is derived. After given start point, search direction, search radius, and trust degree, every seeker moves to a new position (next solution) based on his social learning, cognitive learning, and uncertainty reasoning. The algorithm's performance was studied using several typically complex functions. In almost all cases studied, SOA is superior to continuous genetic algorithm (GA) and particle swarm optimization (PSO) in all 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 increasing 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 viewed as 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 based on the Intermediate Value Theorem. That is, better points are likely to be found in the neighbourhood of families of good points. Hence, search is intensified in regions containing good solutions [7]. It can be believed that one must find the near optimal solutions in the narrower

Y. Wang, Y. Cheung, and H. Liu (Eds.): CIS 2006, LNAI 4456, pp. 167–176, 2007.

neighborhood of the point with higher fitness value, while he must find them in the wider neighborhood of the point with lower fitness value. The algorithm called seeker (or searcher) optimization algorithm (SOA) presented in this paper aims to mimic the behavior of the search group mainly in terms of uncertainty reasoning, at this point where the new algorithm is intensively different from the existing search techniques.

Apparently, the behavior rules mentioned above are described by natural linguistic terms. In order to exploit the rules, cloud model [8], 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 [8] 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 [9], data mining [10], etc..

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

***DEFINITION* 1.** [8,10] 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 $He$ (Fig. 1).

The $Ex$ 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 $n$ cloud drops is generated by the following algorithm called basic normal cloud generator [10].

***Algorithm* 1.** Basic normal cloud generator

```
Input: Ex, En, He, n
Output: { (x1, μ1),..., (xn, μn) }
for i =1 to n
    En' =RANDN(En, He)
    xi =RANDN(Ex, En')
    μi = e^(-(xi-Ex)^2 / 2(En')^2)
    cloud(xi, μi)
end.
```

Here, the function *RANDN*(*a*,*b*) produces a normally distributed random number with mean *a* and standard deviation *b*. the *cloud*($x_i$, $\mu_i$) is the *i*th cloud drop in the universe. In my personal view, cloud models may be partly and originally similar to particle systems [11].

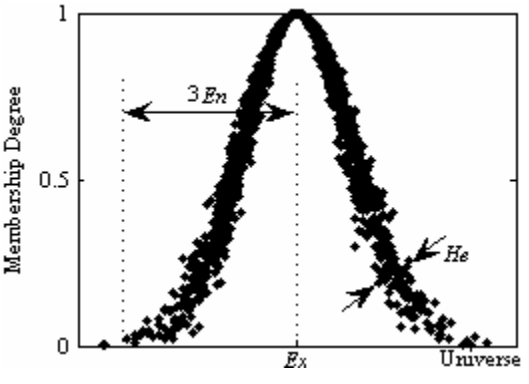


**Fig. 1.** Illustration of the three digital characteristics of a normal cloud

## 3 Seeker Optimization Algorithm

In the SOA, every seeker has a start position vector $\vec{c}$, which may be viewed as 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 start position is a process of uncertainty reasoning, and determined by a like Y-conditional cloud generator [10] as follows:

$$x_{ij}(t+1)=c_{ij}+d_{ij}r_{ij}(-\ln(\mu_{ij}))^{0.5}. \tag{1}$$

where "*i*" is the index of seekers, and "*j*" is the index of variable dimensions.

The pseudocode of the main algorithm is presented as follows.

```
begin
   t•0;
   generating S positions randomly and uniformly;
   repeat
       evaluating each seeker;
       giving search parameters:start position, search
         direction,search radius,and trust degreee;
       updating positions using (1);
       t•t+1;
   until t=T_max
end.
```

## 4 Algorithm Parameters

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

### 4.1 Start Point Vector

Intuitively, start 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. In this paper, the whole search group was classified into *k*=3 neighbourhoods according to the indexes of the seeker group. Then,

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

where $\phi_1$ and $\phi_2$ are real numbers chosen uniformly and randomly in the 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.

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

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

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

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

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.

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

where $\omega$ is the inertia weight, and $\omega=(T_{\max}-t)/T_{\max}$. $\varphi_1$ and $\varphi_2$ are real numbers chosen uniformly and at random in a given interval [0,1].

The expression (2) and (7) are thought to adhere to the principle of self-organized aggregation behaviors [12].

### 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, the cloud generator based method is first introduced to give search radius.

***Algorithm* 2.** The cloud based method of search radius

$En_r = \vec{x}_{\max} - \vec{x}_{\min}$ ;
$He_r = En_r/10$;
$\vec{r}' = RANDN(En_r, He_r)$;
$\vec{r} = RAND(0, \vec{r}')$.

where $\vec{x}_{\max}$ and $\vec{x}_{\min}$ are the positions with the maximum fitness and the minimum fitness within its fellow neighbor, respectively. Such as, the *En* may be viewed as the "known" region of the problem domain, and the seekers from inside this region to outside this region are respectively kept under from a fine-grained search to a coarse-grained search. The function $RAND(0, \vec{r}')$ is given as real numbers chosen uniformly and randomly in a given interval $[0, \vec{r}']$.

The mathematic expected curve (MEC) of a membership cloud may be considered as its membership function from the fuzzy set theory point of view [9]. In order to decrease computing time, the simple method of search radius was expressed as $\vec{r} = RAND(0, En_r)$ where $En_r$ is presented as *ALGORITHM* 2. That is to say, fuzzy logic was used to deal with uncertainty reasoning.

### 4.4 Trust Degree

The parameter $\mu$ is, in fact, the grade of membership from cloud model and fuzzy set theory. According to the discussion in section 1, the uncertainty rule of intelligent search is described as "If {*fitness* is *large*}, Then {*search radius* is *small*}". The

linear membership function was used for "*large*" of "*fitness*". Namely, it is directly proportional to the fitness of $\bar{x}(t)$ or the index of the ascensive sort order of the fitness of $\bar{x}(t)$ (we applied the latter in our experiments). That is, the best position so far has the maximum $\mu_{\max}$=1.0, while other position has a $\mu$<1.0, and the worst position so far has the minimum $\mu_{\min}$. The expression is presented as (8) and (9).

$$\mu_i = \mu_{\max} - \frac{S - I_i}{S - 1}(\mu_{\max} - \mu_{\min}) . \tag{8}$$

$$\mu_{ij} = RAND(\mu_i, 1) \tag{9}$$

where $S$ is the neighbor search group size, and $I_i$ is the index (sequence number) of $\bar{x}_i(t)$ after sorting the fitnesses of neighbor seekers in ascending order.

Meanwhile, The Gaussian membership function $\mathrm{MEC}_A(x) = e^{-(x-Ex)^2/2En^2}$ was used for "*small*" of "*search radius*". Based on "3*En*" rule [13] which shows that the elements beyond *Ex*±3*En* in the universe of discourse can be neglected for a linguistic atom [9], $\mu_{\min}$=0.0111 is given at the point $x = Ex \pm 3En$ where $\mathrm{MEC}_A(Ex \pm 3En)$=0.0111.

## 5 Convergence Analysis

From (2) and (8), when $\bar{x}_i(t) = \bar{g}(t)$ , $1 \leqslant i \leqslant S$, it is apparently given that $\bar{c}_i(t) = \bar{g}(t)$ and $\mu_i(t) = 1.0$ . Then (1) gives $\bar{x}_i(t+1) = \bar{g}(t)$ and $fit(\bar{x}_i(t+1)) = fit(\bar{x}_i(t)) = fit(\bar{g}(t))$ . 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. The further mathematical convergence analysis will be still left for future.

## 6 Function Optimization

In this section, the experiments will be discussed to compare the performance of the SOA, PSO and continuous genetic algorithm (GA). We used the MATLAB codes of PSO with adaptive inertia weight and continuous GA presented by [2]. In the experiments, the parameters of the PSO are that: learning rate $c_1 = c_2$=2, inertia weight linearly decreased from 0.9 to 0.4 with run time increasing [14]. The parameters of the continuous GA are that: the fraction of population kept was set 0.8, the mutation rate was set 0.1.

17 typical functions with varying complexities and varying number of variables (NV) were employed. They are as follows.

F1 Goldstein-Price function

$$\begin{aligned} F1 = {} & [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3{x_1}^2 - 14x_2 + 6x_1x_2 + 3{x_2}^2)] \times [30 + (2x_1 \\ & - 3x_2)^2 (18 - 32x_1 + 12{x_1}^2 + 48x_2 - 36x_1x_2 + 27{x_2}^2)], -2 \le x_i \le 2, i = 1,2 \end{aligned} \tag{10}$$

F2 DeJong's *f*2

$$F2 = 100(x_1{}^2 - x_2)^2 + (1 - x_1)^2, -2.048 \le x_i \le 2.048, i = 1,2 \tag{11}$$

F3 DeJong's *f*5

$$F3 = 1/(0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}), -65.536 \le x_i \le 65.536, i = 1,2 \tag{12}$$

F4 DeJong's *f*6

$$F4 = 0.5 + \frac{\sin^2\sqrt{x_1{}^2 + x_2{}^2} - 0.5}{\left[1.0 + 0.001(x_1{}^2 + x_2{}^2)\right]^2}, -100 \le x_i \le 100, i = 1,2 \tag{13}$$

F5 DeJong's *f*7

$$F5 = (x_1{}^2 + x_2{}^2)^{0.25}[\sin^2(50(x_1{}^2 + x_2{}^2)^{0.1}) + 1.0], -100 \le x_i \le 100, i = 1,2 \tag{14}$$

F6 Goldstein's function

$$F_6 = x^6 - 15x^4 + 27x^2 + 250; -10 \le x \le 10 \tag{15}$$

F7 Griewangk's function

$$F7 = \sum_{i=1}^{D} \frac{x_i{}^2}{4000} + \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1; D = 10, -512 \le x_i \le 512 \tag{16}$$

F8 Hyper-Ellipsoid function

$$F8 = \sum_{i=1}^{100} i^2 x_i^2; -1 \le x_i \le 1 \tag{17}$$

F9 Rastrigin's function

$$F9 = 10D + \sum_{i=1}^{D}(x_i{}^2 - 10\cos(2\pi x_i)); D = 15, -10 \le x_i \le 10 \tag{18}$$

F10 Schwefel's 2.21 function

$$F10 = \max_i \{|x_i|, i = 1, \cdots\}; -10 \le x_i \le 10 \tag{19}$$

F11 Schwefel's 2.22 function

$$F11 = \sum_{i=1}^{5}|x_i| + \prod_{i=1}^{5}|x_i|; -10 \le x_i \le 10 \tag{20}$$

F12 Schwefel's 2.23 function

$$F12 = \sum_{i=1}^{5} x_i{}^{10}; -10 \le x_i \le 10 \tag{21}$$

F13 Shubeurt's function

$$F13 = \left\{\sum_{i=1}^{5} i \bullet \cos[(i+1)x_1 + i]\right\}\left\{\sum_{i=1}^{5} i \bullet \cos[(i+1)x_2 + i]\right\} + 0.5\left\{(x_1 + 1.42513)^2 + (x_2 + 0.80032)^2\right\}; -10 \le x_i \le 10 \tag{22}$$

F14 Simple square sum function

$$F14 = x_1^2 + x_2^2; -5 \le x_{1,2} \le 5 \tag{23}$$

F15 Six-Peak function

$$F15 = (4 - 2.1{x_1}^2 + {x_1}^4 / 3){x_1}^2 + x_1 x_2 + (-4 + {x_2}^2){x_2}^2, \ -3 \le x_{1,2} \le 3 \tag{24}$$

F16 Yan and Ma's function

$$F16 = \frac{x_1 + x_2}{2} - \cos(20\pi x_1) \bullet \cos(20\pi x_2) + 2; \ -10 \le x_{1,2} \le 10 \tag{25}$$

F17 Yan and Ma's function

$$F17 = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2 \bullet \left[1 + \sin^2(3\pi x_{i+1})\right] + (x_D - 1)^2[1 + \sin^2(2\pi x_D)]\right\}; D = 5, -5 \le x_i \le 5 \tag{26}$$

As a measure of performance, we consider the average number of generations (AG) that the algorithms require to generate a solution with a certain high fitness value. 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 (IV) 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 given 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. Furthermore, the expression $En_r = \bar{x}_{\max} - \bar{x}_{avg}$ was used for search radius when functions F13, F16, and F7-11 were optimized. Here, $\bar{x}_{avg}$ is the average value of the points in a same neighbourhood. The results of experiments are presented in Table 1.

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

**Table 1.** Comparisons of performance of SOA, GA and PSO

| Functions | Algo. | AG | NR | BV | AV | STD |
|---|---|---|---|---|---|---|
| F1 | GA | 153.8 | 0 | 3 | 3 | 1.6922e-005 |
| (NV=2, IV=3, | PSO | 109 | 0 | 3 | 3 | 0 |
| MNG=1000) | SOA | 20.9 | 0 | 3 | 3 | 0 |
| F2 | GA | 902.8 | 0 | 5.2587e-005 | 0.011178 | 0.014291 |
| (NV=2, IV=0, | PSO | 77.1 | 0 | 7.3463e-030 | 2.7379e-026 | 7.3804e-026 |
| MNG=1000) | SOA | 42.9 | 0 | 0 | 0 | 0 |
| F3 | GA | 115.2 | 0 | 0.998 | 0.998 | 1.9642e-010 |
| (NV=2,IV=0.998, | PSO | 80.3 | 0 | 0.998 | 0.998 | 1.655e-016 |
| MNG=1000) | SOA | 70.7 | 0 | 0.998 | 0.998 | 1.4803e-016 |
| F4 | GA | 191 | 1 | 0 | 1.4098e-005 | 4.4572e-005 |
| (NV=2, IV=0, | PSO | 72.3 | 0 | 0 | 0 | 0 |
| MNG=1000) | SOA | 16.5 | 0 | 0 | 0 | 0 |
| F5 | GA | 3000 | 10 | 0.00029844 | 0.0073763 | 0.010015 |
| (NV=2, IV=0, | PSO | 853.5 | 0 | 8.7414e-060 | 2.8635e-058 | 5.1778e-058 |
| MNG=3000) | SOA | 97.8 | 0 | 0 | 0 | 0 |
| F6 | GA | 18.7 | 0 | 7 | 7 | 0 |
| (NV=0, IV=7, | PSO | 28.9 | 0 | 7 | 7 | 0 |
| MNG=1000) | SOA | 20.6 | 0 | 7 | 7 | 2.1843e-008 |
| F7 | GA | 3000 | 10 | 0.53597 | 0.81229 | 0.17299 |
| (NV=10, IV=0, | PSO | 3000 | 10 | 0.029509 | 0.068525 | 0.021366 |
| MNG=3000) | SOA | 476.8 | 0 | 0 | 1.9209e-007 | 3.9966e-007 |
| F8 | GA | 3000 | 10 | 13628 | 17236 | 2211.2 |
| (NV=100,IV=0, | PSO | 3000 | 10 | 0.004186196 | 348.9081 | 331.593447 |
| MNG=3000) | SOA | 78.6 | 0 | 1.9066e-103 | 5.0255e-097 | 1.4066e-096 |
| F9 | GA | 3000 | 10 | 13.188 | 24.484 | 7.2147 |
| (NV=15, IV=0, | PSO | 3000 | 10 | 1.989918 | 3.88034 | 1.720125 |
| MNG=3000) | SOA | 706.6 | 0 | 0 | 0 | 0 |
| F10 | GA | 3000 | 10 | 0.0058716 | 0.01661 | 0.007963 |
| (NV=5, IV=0, | PSO | 995.4 | 0 | 1.7342e-072 | 1.8576e-070 | 2.1241e-070 |
| MNG=3000) | SOA | 136.7 | 0 | 1.3200e-128 | 5.1487e-080 | 1.6181e-079 |
| F11 | GA | 3000 | 10 | 0.000801 | 0.0023801 | 0.0011435 |
| (NV=5, IV=0, | PSO | 971.1 | 0 | 9.02340e-087 | 4.7115e-084 | 5.6996e-084 |
| MNG=3000) | SOA | 137.3 | 0 | 3.5094e-127 | 3.6511e-125 | 5.5413e-125 |
| F12 | GA | 89.7 | 0 | 1.1076e-029 | 1.8645e-022 | 3.7167e-022 |
| (NV=5, IV=0, | PSO | 13.7 | 0 | 0 | 0 | 0 |
| MNG=3000) | SOA | 8.2 | 0 | 8.0952e-134 | 5.934e-057 | 1.8158e-056 |
| F13 | GA | 2086.4 | 6 | -186.73 | -186.5 | 0.20155 |
| (IV=-186.7309, | PSO | 446.9 | 5 | -186.7309088 | -186.7309088 | 2.9959e-014 |
| NV=2,MNG=3000) | SOA | 2247.1 | 3 | -186.730901 | -186.7309079 | 0.000120 |
| F14 | GA | 25.5 | 0 | 2.2357e-019 | 3.0517e-007 | 5.2728e-007 |
| (NV=2, IV=0, | PSO | 28.8 | 0 | 2.2862e-088 | 4.7246e-084 | 1.4408e-083 |
| MNG=1000) | SOA | 9.1 | 0 | 0 | 1.2337e-253 | 0 |
| F15 | GA | 29.9 | 0 | -1.0316 | -1.0316 | 1.7246e-008 |
| (IV=-1.031628, | PSO | 41.8 | 0 | -1.031628 | -1.031628 | 2.3406e-016 |
| NV=2,MNG=1000) | SOA | 13.7 | 0 | -1.0316 | -1.0316 | 2.3406e-016 |
| F16 | GA | 2142.3 | 6 | 1 | 1.0013 | 0.0012818 |
| (NV=2, IV=1, | PSO | 1169.3 | 0 | 1 | 1 | 0 |
| MNG=3000) | SOA | 117.4 | 0 | 1 | 1 | 4.1659e-009 |
| F17 | GA | 991 | 0 | 6.7894e-008 | 4.5504e-006 | 6.1197e-006 |
| (NV=5, IV=0, | PSO | 455.2 | 0 | 1.3498e-032 | 1.3498e-032 | 2.8850e-048 |
| MNG=3000) | SOA | 51.6 | 0 | 1.3450e-032 | 1.3500e-032 | 2.8850e-048 |

## 7 Conclusions and Future Work

In this research, a novel optimization algorithm based on the concept of simulating the act of human's intelligent search was introduced whose performance in terms of robustness and efficiency was 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, more robust and more efficient than GA and PSO 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. Deb, K., Anand, A., Joshi, D.: A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization. Evolutionary Computation 10(4), 371–395 (2002)
2. Randy, L., Haupt, S.E.: Practical Genetic Algorithms, 2nd edn. pp. 215–228. John Wiley & Sons, Inc, New Jersey (2004)
3. Kennedy, J., Eberhart, R.C.: Panicle Swarm Optimization. In: Proceeding of the 1995 IEEE International Conference on Neural Networks, Vol, pp. 1942–1948. IEEE Computer Society Press, Los Alamitos (1995)
4. Storn, R., Price, K.: Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. In: Technical report, International Computer Science Institute, Berkley (1995)
5. Wolpen, D.W., Macready, W.G.: No Free Lunch Theorem for Optimization. IEEE Trans. Evol. Comp. 1(1), 67–82 (1997)
6. Köppen, M.: No-Free-Lunch Theorems and the Diversity of Algorithms. In: Proceedings of the 2004 Congress on Evolutionary Computation, vol. 1, pp. 235–241. IEEE, Los Alamitos (2004)
7. Raphael, B., Smith, I.F.C., Direct, A.: Stochastic Algorithm for Global Search. Applied Mathematics and Computation 146, 729–758 (2003)
8. Li, D., Meng, H., Shi, X.: Membership Clouds and Membership Cloud Generators. Journal of Computer Research and Development 42(8), 32–41 (1995) (in Chinese)
9. Li, D., Cheung, D.W., Shi, X. et al.: Uncertainty Reasoning Based on Cloud Models In Controllers. Computers and Mathematics with Applications 35(3), 99–123 (1998)
10. Li, D.: Di, K., Li, D.: Knowledge Representation and Uncertainty Reasoning in GIS Based on Cloud Models. In: Proceeding of the 9th International Symposium on Spatial Data Handling, Beijing, 10-12 (2000)
11. Reeves, W.T.: Particle Systems - a Technique for Modeling a Class of Fuzzy Objects. ACM Transactions on Graphics 2(2), 91–108 (1983)
12. Camazine, S., Deneubourg, J.-L., Franks, N., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press, Princeton, NJ (2001)
13. Changyu, L., Deyi, L., Lili, P.: Uncertain Knowledge Representation Based on Cloud Model. Computer Engineering and Applications 40(2), 32–35 (2004) (in Chinese)
14. Shi, Y., Eberhart, R.: Empirical Study of Particle Swarm Optimization, In: Proceedings of the 1999 Congress.on Evolutionary Computation, Vol. 3, Washington, DC, USA, pp. 1945–1950 (1999)