International Journal of Computational Intelligence Systems, Vol. 7, No. 2 (April 2014), 353-359

# Seeker Optimization Algorithm for Several Practical Applications

**Yunfang Zhu** *

*School of Electrical Engineering, National Engineering & Technology Research Center of Electrification and Automation in Rail Transit, Southwest Jiaotong University, Chengdu, 610031, China*

**Chaohua Dai, and Weirong Chen**

*School of Electrical Engineering, National Engineering & Technology Research Center of Electrification and Automation in Rail Transit, Southwest Jiaotong University, Chengdu, 610031, China*

*E-mail: daichaohua@swjtu.edu.cn, wrchen@swjtu.edu.cn*

Received 30 March 2012
Accepted 27 December 2012

**Abstract**

Optimization problems can often be simplified to the search for an optimal solution in the feasible search space. Based on the concept of simulating the act of human randomized search, a novel algorithm called seeker optimization algorithm (SOA) for real-parameter optimization is proposed in this paper. In the SOA, after given center point, search direction, search radius, and trust degree, every seeker moves to a new position (a candidate solution) from his current position based on his historical and social experiences. 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) greatly in terms of optimization quality, robustness and efficiency. At the same time, SOA greatly outperforms particle swarm optimization (PSO) in convergence speed. However, SOA needs more computation time. Simulations of designing both PID controller and IIR digital filter also show that SOA gets more satisfactory solutions with better evaluation values.

*Keywords*: Seeker optimization algorithm, PID controller design, IIR digital filter design.

## 1. Introduction

In the continuous variable space, 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 a narrower neighborhood of the point with higher fitness value, while he must find them in a wider neighborhood of the point with lower fitness value. Seeker optimization algorithm (SOA) was originally proposed in Ref. [1], which aims to mimic the above-mentioned searching rule of the search group and their means of information exchange to solve real-parameter optimization problems.

The study herein is the continuation and extension of work previously published in Ref. [1]. In comparison with [1], there are two main contributions in this paper: Firstly, some concepts in SOA are further clarified. Secondly, the applications are extented to the design of IIR digital filter and PID controller. Although a version of the SOA was applied in the design of IIR digital filter in Ref. [2], the algorithm herein is different, and the difference lies in: The search step/radius is assigned with cloud model [3] while Ref. [2] used Fuzzy reasoning to give this parameter. The preservation of the uncertainty in transition makes cloud model well meet the need of real life situation, and has already been successfully used in intelligent control [4], data mining [5], etc. Moreover, in Ref. [2], the calculation of search direction applied the proportional selection rule, which

* Address: School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 610031, China. Email: zyfdch@126.com

can improve the population diversity so as to boost global search ability of the algorithm. However, for the proportional selection rule, it is not easy to weigh the contributions from the different empirical gradients. Hence, this study still applies the same method as [1] to calculate the search direction.

This paper is organized as follows. Section 2 describes cloud model. In section 3, we introduce the SOA in details. The algorithm parameters are discussed in section 4. Convergence analysis is shown in section 5. Then, we compare the SOA with continuous genetic algorithm (CGA) and particle swarm optimization (PSO) using typical functions and the designs of PID controller and digital filter design in section 6. Finally, the conclusions and future work are presented in section 7.

## 2. Cloud Model

***DEFINITION 1*** Let $U$ be the set, $U=\{u\}$, as the universe of discourse, and $T$ is 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 (Figure 1) 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.

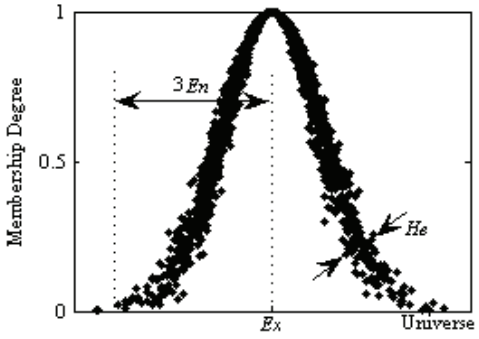


**Figure 1.** Illustration of a normal cloud

The cloud with $n$ drops is generated by the basic normal cloud generator [3].

*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($x_i$, $\mu_i$) is the $i$th cloud drop.

## 3. Seeker Optimization Algorithm

In SOA, every seeker has a center position vector $\vec{c}$, which is the start location to find next solution, and can be viewed as expected value $Ex$ of cloud model. Moreover, each seeker holds a search radius $\vec{r}$ viewed as the $En'$ of cloud model, a trust degree $\mu$ as membership degree of cloud model, and a search direction $\vec{d}$ showing him where to go. Then, the seeker with a certain trust degree follows a feasible direction and stochastically moves to next point (new candidate solution) in a certain search radius from his current position.

At each time step $t$, the search decision-making is conducted to calculate 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:

$$\vec{x}_{ij}(t+1) = \vec{c}_{ij}(t) + \vec{d}_{ij}(t) \times \vec{r}_{ij}(t) \times \sqrt{-\ln(\mu_i)} \quad (1)$$

where "$i$" is the subscript index of seeker, and "$j$" is the subscript index of variable dimensions.

The pseudocode of the algorithm is presented as follows.

1. $t \leftarrow 0$
2. **Initialization** generating $S$ positions $\{x_i(t) | x_i(t) = (x_{i1}, x_{i2}, \cdots, x_{iD}), i = 1, \cdots, S, t = 0\}$ 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

### 4.1. *Center Point Vector*

Intuitively, center position vector $\vec{c}$ is set to current position $\vec{x}(t)$. Like 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. All seekers are classified into $k$ classes in their subscript indexes, and the seekers in the same class belong to a virtual neighborhood. Thus, $\vec{g}$ is found just in the virtual neighborhood. Then,

$$\vec{c} = \vec{x}(t) + r_1\phi_1(\vec{p}(t) - \vec{x}(t)) + r_2\phi_2(\vec{g}(t) - \vec{x}(t)) \quad (2)$$

where $r_1$, $r_2$ are the cognitive and social learning rates, respectively. $\phi_1$ and $\phi_2$ are real numbers chosen uniformly and randomly in a given interval [0,1]. In all experiments conducted in this paper, $r_1$=1, $r_2$=1, and $k$=3.

### 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.

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

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

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

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

where $sign(\bullet)$ 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.

$$\vec{d} = sign(\omega(sign(fit(\vec{x}(t)) - fit(\vec{x}(t-1))))(\vec{x}(t) - \vec{x}(t-1)) + r_1\varphi_1(\vec{p}(t) - \vec{x}(t)) + r_2\varphi_2(\vec{g}(t) - \vec{x}(t))) \quad (7)$$

where $\omega$ is the inertia weight which is set to $\omega=(T_{max}-t)/T_{max}$ in this paper, $\varphi_1$ and $\varphi_2$ are real numbers chosen uniformly and randomly 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.

The methods adopted in our experiments is as follows.

*ALGORITHM* 3 Search radius
1. $En = \vec{x}_{max} - \vec{x}_{min}$
2. $He=En/10$
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. $RAND(0, \vec{r}')$ gives a real number uniformly and randomly in a given interval $[0, \vec{r}']$.

### 4.4. *Trust Degree*

The parameter $\mu$ is viewed as quality evaluation of different position. It is directly proportional to the fitness of $\vec{x}(t)$ or the index of the ascensive sort order of the fitness of $\vec{x}(t)$ (the latter used in this study). That is, the global best position has the maximum $\mu_{max}$=1.0,while other position has a $\mu$<1.0. The expression is presented as (8).

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

where $Sn$ is the sequence number of $\vec{x}(t)$ after sorting the fitnesses of neighbor seekers in ascending order, $\mu_{max}$ and $\mu_{min}$ are the maximum and the minimum $\mu$ given by user. We adopted $\mu_{max}$=1.0, and $\mu_{min}$=0.2.

## 5. Convergence Analysis

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 $\mu_i(t) = 1.0$ from (2) and (8). Then (1) gives $\vec{x}_i(t+1) = \vec{g}(t)$. Thus, $fit(\vec{x}_i(t+1)) = fit(\vec{x}_i(t)) = fit(\vec{g}(t))$. 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. Performance Test

Among the existing evolutionary algorithms, continuous genetic algorithm (CGA) [6] and particle swarm optimization (PSO) [7] have received increasing interest

from the evolutionary computation community as two well-known population-based optimization techniques. In addition, they both have been applied to function optimization [7, 8], PID controller [9-11] and digital filter design [12, 13]. So, the proposed method is compared mainly with the two algorithms.

### 6.1. *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).

In this research, we have used 8 typical functions with varying complexities. They are F1: Goldstein-Price function, F2: DeJong's f2, F3: DeJong's f5, F4: DeJong's f6, F5: DeJong's f7, F6: square sum function, F7: Multi-peak function (as Eq. 9) and F8: Yan and Ma's function.

F7: Multi-peak function

$$F7 = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + x_2^2)x_2^2,$$
$$-3 \le x_i \le 3, i = 1,2 \quad (9)$$

As a measure of performance, we consider the average number of generations (*AG*) that the algorithms require to generate a solution with the absolute error between the actual function value and the ideal function value less than or equal to 0.0001. 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 with the absolute error between the actual function value and the ideal function value bigger than 0.0001. Besides, the best function values (*BV*), the average values (*AV*), and the standard deviations (*STD*) of the actual function values are also compared. Also compared is the average computation time (*AT*).

In all our experiments, we have used a population size of 100 for all functions, and the maximum number of generations is 3000 for F5 and F8, and 1000 for the rest. The results obtained by MATLAB 7.0 software run on Pentium IV of 2.4GHz are presented in Table 1.

As seen from Table 1, the SOA outperforms greatly the CGA and PSO in convergence speed with smaller AGs. For all the functions optimized here, the values of the *FN*s, *BV*s, *AV*s and *STD*s of SOA are much better than those of CGA, and better than or equivalent to those of PSO, which shows SOA has statistically better potential to get near global optimum. However, SOA needs more computation time.

**Table 1.** Comparison of performances of SOA, CGA and PSO

| Functions | | *F*1 | *F*2 | *F*3 | *F*4 | *F*5 | *F*6 | *F*7 | *F*8 |
|---|---|---|---|---|---|---|---|---|---|
| Ideal values | | 3 | 0 | 0.9980038 | 0 | 0 | 0 | -1.031628 | 1 |
| *AG* | CGA | 153.8 | 902.8 | 115.2 | 191 | 3000 | 25.5 | 29.9 | 2142.3 |
| | PSO | 183.8 | 116.4 | 143.5 | 122.2 | 984.3 | 76 | 104.1 | 1043 |
| | SOA | 20.6 | 49.3 | 52.4 | 15.1 | 93.1 | 8.7 | 10.4 | 487.6 |
| *NR* | CGA | 0 | 0 | 0 | 1 | 10 | 0 | 0 | 6 |
| | PSO | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | SOA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *BV* | CGA | 3.000000 | 5.2587e-05 | 0.998 | 0 | 0.00029844 | 2.236e-019 | -1.0316 | 1 |
| | PSO | 3.000000 | 0 | 0.9980038 | 0 | 3.3337e-81 | 7.889e-120 | -1.031628 | 1 |
| | SOA | 3.000000 | 0 | 0.9980038 | 0 | 0 | 0 | -1.031628 | 1 |
| *AV* | CGA | 3.000000 | 0.011178 | 0.998 | 1.4098e-5 | 0.0073763 | 3.052e-007 | -1.0316 | 1.0013 |
| | PSO | 3.000000 | 0 | 0.9980038 | 0 | 2.8934e-79 | 1.421e-114 | -1.031628 | 1 |
| | SOA | 3.000000 | 0 | 0.9980038 | 0 | 0 | 2.323e-242 | -1.031628 | 1 |
| *STD* | CGA | 1.6922e-05 | 0.014291 | 1.9642e-10 | 4.4572e-5 | 0.010015 | 5.273e-007 | 1.7246e-08 | 0.0012818 |
| | PSO | 1.4803e-16 | 0 | 1.6550e-16 | 0 | 4.5015e-79 | 3.144e-114 | 2.669e-016 | 0 |
| | SOA | 0 | 0 | 1.0467e-16 | 0 | 0 | 0 | 2.3406e-16 | 1.4237e-15 |
| *AT*(s) | CGA | 4.23 | 3.78 | 12.55 | 4.30 | 13.47 | 4.45 | 4.59 | 10.98 |
| | PSO | 3.78 | 3.38 | 13.78 | 3.97 | 12.63 | 3.85 | 3.63 | 9.39 |
| | SOA | 9.39 | 9.55 | 19.90 | 9.56 | 29.90 | 10.48 | 10.26 | 28.15 |

### 6.2. *Tuning of PID Parameters*

To verify the efficiency of the SOA-PID controllers, a marine control plant was taken as an example [10]. It is a second-order process with time delay described as follows.

$$G(s) = \frac{5}{(2s+1)(0.5s+1)} e^{-0.2s} \quad (10)$$

The evaluation function is given as (11)

$$J = \int_0^{+\infty} (\omega_1 t \mid e(t) \mid + \omega_2 u^2(t) + \omega_4 \mid e(t) \mid) dt + \omega_3 t_r \qquad (11)$$

where $e(t)$ is system error, $u(t)$ is controller output, $t_r$ rise time, $\omega_1$=0.999,$\omega_2$=0.001,$\omega_3$=2,$\omega_4$=20 (when $e(t)$<0) or $\omega_4$=0(when $e(t) \geqslant 0$).

In order to emphasize the advantages of the proposed SOA-PID controller, we also implement the PSO-PID (PSO with adaptive inertia weight) controller and CGA-PID controller and compare the step response performance of three controllers using the same evaluation function. For three algorithms, population size is set to 30 and maximum number of generations is set to 100. As to the CGA, roulette selection, arithmetic crossover with $p_c$=0.8, and uniform mutation with $p_m$=0.1 were used. For the PSO, inertia weight was decreased linearly from 0.9 to 0.4, and acceleration constant $c_1$ and $c_2$ were equal to 2.0.

When input is step signal ($r_{in}$=1.0), and sampling time is 0.005s, simulation research was conducted repeatedly (in our case, 10 times) with different and randomly chosen initial populations. Figure 2 shows average convergence curves of evaluation values in three different PID controllers. It is found that the SOA has higher convergence speed and better searching ability than PSO and CGA. Figure 3 presents the step response of control system with three PID controllers when the smallest evaluation values were obtained from several independent experiments. Table 2 gives the step response's characteristic of each method, including controller parameters $k_p$, $k_i$, $k_d$, overshoot $M_p$, rise time $t_r$, settling time $t_s$, static error $E_{ss}$, and minimum evaluation value $J_{min}$. As can be seen, the SOA-PID controller, as a whole, has better performance than the PSO-PID and CGA-PID controller.

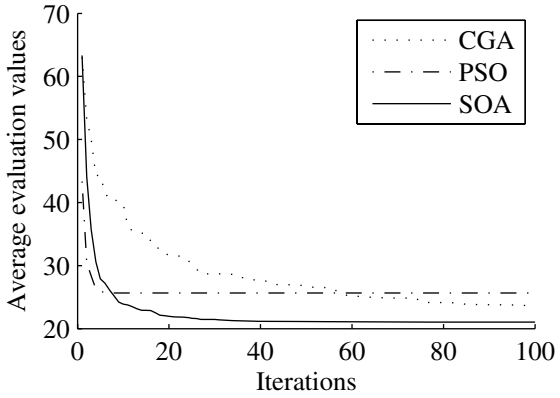


Figure 2. Average convergence curves of objective function

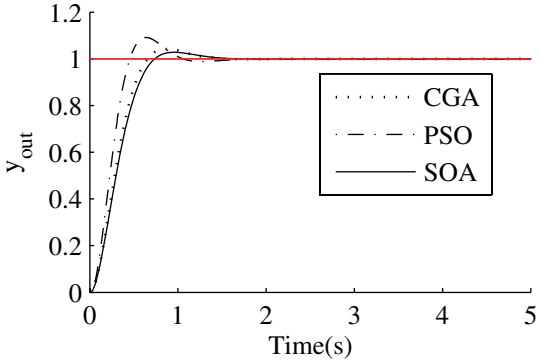


Figure 3. Step response curves of three PID controllers

Table 2. The parameters of three PID controllers

| Algorithms | $k_p$ | $k_i$ | $k_d$ | $M_p$ | *Ess* | $t_r$(s) | $t_s$(s) | $J_{min}$ |
|---|---|---|---|---|---|---|---|---|
| CGA-PID | 5.0779 | 0.96291 | 0.71946 | 4.94% | 3.3398e-05 | 0.5350 | 1.1100 | 21.2772 |
| PSO-PID | 5.6365 | 1 | 0.7735 | 4.19% | 1.3059e-06 | 0.5800 | 1.1600 | 21.2692 |
| SOA-PID | 4.7589 | 0.99089 | 0.6615 | 2.81% | 4.8230e-09 | 0.6300 | 1.1550 | 20.5605 |

### 6.3. *Designing IIR digital filter*

Because there are no analytical methods evidently when IIR digital filters with any frequency responses are considered, optimization methods are used generally [2, 12, 13]. In this research, SOA is used to optimize design of IIR digital filter in frequency domain directly.

The transfer function of IIR digital filter can be represented as the following expression.

$$H(z) = A \prod_{k=1}^{N} \frac{1 + a_k z^{-1} + b_k z^{-2}}{1 + c_k z^{-1} + d_k z^{-2}} \qquad (12)$$

Suppose that the frequency responses of the ideal filter and the designed filter are $H_d(e^{j\omega})$ and $H(e^{j\omega})$, respectively. Optimization design of IIR digital filter using the criterion of minimal mean square error (*MSE*) is that the *MSE* between the magnitude of frequency response of designed IIR filter and the magnitude of frequency response of the ideal filter in discrete frequency point $\{\omega_i \mid i=1,2,\ldots,M\}$ is made least, i.e.

$$\min E = \min \sum_{i=1}^{M} \left[ \left| H(e^{j\omega_i}) \right| - \left| H_d(e^{j\omega_i}) \right| \right]^2 \qquad (13)$$

***Example:*** design a lowpass IIR digital filter. The technological requirements are as follows.

$$\left| H_d(e^{j\omega}) \right| = \begin{cases} 1 & 0 \leq \omega \leq 0.4\pi \\ 0 & 0.5\pi \leq \omega \leq \pi \end{cases} \qquad (14)$$

The number of sample points, $M$, is set 46, and the order of the IIR digital filter is set 6, that is, $N$=3.

The maximum number of generations and population size is set 1000 and 80, respectively, and the best IIR

digital filter with the smallest MSE value is obtained by performing the experiment repeatedly (in our case, 10 times) with different and randomly chosen initial populations. The transfer function of IIR digital filter designed by SOA is

$$H(z) = 0.065316 \times \frac{1 + 0.074809z^{-1} + 0.99546z^{-2}}{1 - 0.60535z^{-1} + 0.20009z^{-2}} \times \frac{1 + 1.7115z^{-1} + 0.98661z^{-2}}{1 - 0.40977z^{-1} + 0.69138z^{-2}} \times \frac{1 + 0.54452z^{-1} + 0.99959z^{-2}}{1 - 0.089827z^{-1} + 0.77808z^{-2}} \quad (15)$$

Its corresponding magnitudes of frequency response are shown in Figure 4 (solid line). As comparison, CGA and PSO are also applied to design the lowpass IIR digital filter shown in Figure 4, too. Figure 5 shows the average convergence curves of MSE. Table 3 shows the performance parameters of three digital filters by different algorithms.

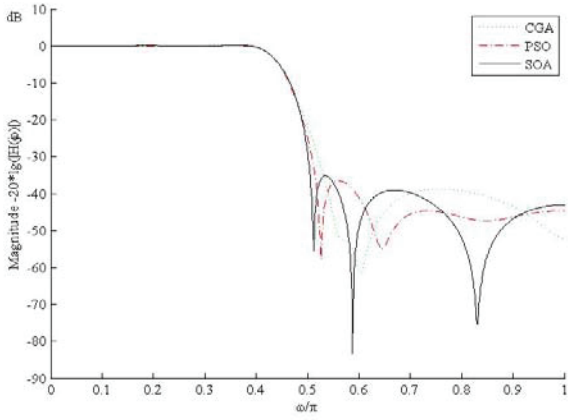


Figure 4. Frequency responses of IIR digital filters

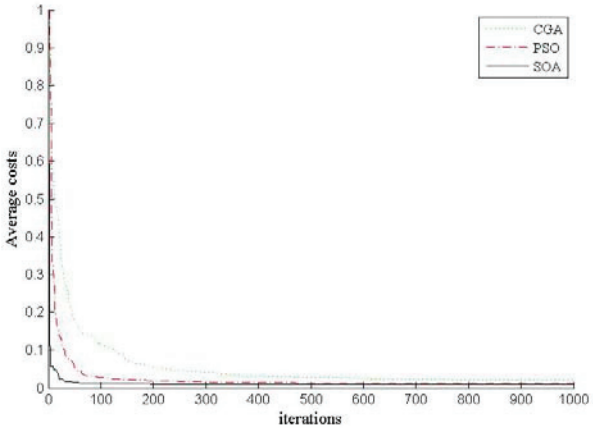


Figure 5. Average convergence curves of MSE

From figure 5, it is found that the SOA is superior to PSO and CGA in convergence speed and searching ability. From Figure 4 and Table 3, conclusions can be drawn that maximal ripple of the filter designed by SOA is smaller than that of the filter designed by CGA and PSO in the pass band. And in the stop band, magnitude of frequency response of the filter designed by SOA gets greater attenuation than that of CGA and PSO. According to the performance criterions that less ripple in the pass hand and greater attenuation in the stop hand correspond to higher performance, the proposed algorithm in this paper can get more satisfactory solution in the IIR digital filter design.

Table 3. The performances of three digital filters

| Algorithm | $A_p$ | $A_s$ | $MSE_{min}$ |
|---|---|---|---|
| CGA | 0.4437 | 21.5291 | 0.0142 |
| PSO | 0.3671 | 24.6270 | 0.0104 |
| SOA | 0.3171 | 28.6287 | 0.0070 |

## 7. Conclusions and Further Research

In this research, a novel real-parameter 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 optimum when solving different classes of problems with different degrees of complexities. In all cases studied, SOA was faster and more robust than CGA with far fewer generations to converge to more satisfactory global solutions and far fewer times to get stuck at a local optimum. Besides rapider convergence speed, SOA is also superior or, at least, equivalent to PSO in all other aspects. But, SOA still needs more computation time because of relatively complex computation and our non-optimized MATLAB codes. Furthermore, the performances of both PID controller and lowpass IIR digital filter designed using SOA are better than that of CGA and PSO. Therefore, the introduced method, SOA, is effective and practical algorithm.

Further research includes practical applications and theoretical analysis to better understand this algorithm's convergence properties and the effects of the parameters on its performance. Besides, computation complexity needs to be further analyzed and improved.

## Acknowledgements

This work was supported in part by the National Nature Science Foundation of China under contracts 51307144 and 51177138, and the Fundamental Research Funds for the Central Universities under Contracts SWJTU11ZT07.

## References


1. Chaohua Dai, Yunfang Zhu, and Weirong Chen. (2006). Seeker optimization algorithm. In: Proceedings of the 2006 International Conference on Computational Intelligence and Security, Guangzhou, China, Nov. 3-6.
2. Chaohua Dai, Weirong Chen, and Yunfang Zhu. (2010). Seeker optimization algorithm for digital IIR filter design, IEEE Transactions on Industrial Electronics, vol. 57, no. 5, 1710-1718.
3. Deyi Li, Changyu Liu, Wenyan Gan. (2009). A new cognitive model: Cloud model. International Journal of Intelligent Systems, vol. 24, no. 3, 357-375.
4. Sheng Liu, Xu-Cheng Chang. (2012). Synchro-control of twin-rudder with cloud model. International Journal of Automation and Computing, vol. 9, no. 1, 98-104.
5. Kun Qina, Min Xua, Yi Dub, etc. (2008). Cloud model and hierarchical clustering based spatial data mining method and application. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXVII, Part B2, 241-246.
6. C.F. Wang, Z.Y. Chen. (2012). Integration of growing self-organizing map and continuous genetic algorithm for grading lithium-ion battery cells. Applied Soft Computing, vol. 12, no. 8, 2012–2022.
7. Yu Liu, Xiaoxi Ling, Zhewen Shi,etc. (2011) A Survey on particle swarm optimization algorithms for multimodal function optimization. Journal of Software, vol. 6, no. 12, 2449-2455.
8. Yi-Tung Kao, Erwie Zahara. (2008). A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Applied Soft Computing, vol. 8, no. 2, 849–857.
9. Ayman A. Aly. (2011). PID parameters optimization using genetic algorithm technique for electrohydraulic servo control system. Intelligent Control and Automation, 2, 69-76.
10. Chen Junfeng, Ren Ziwu, and Fan Xinnan. (2006). Particle swarm optimization with adaptive mutation and its application research in tuning of PID parameters. In: Proceedings of the 1st International Symposium on Systems and Control in Aerospace and Astronautics, Harbin: 990-994.
11. Wei-Der Chang. (2009). PID control for chaotic synchronization using particle swarm optimization. Chaos, Solitons & Fractals, vol. 39, no. 2, 910–917.
12. Ranjit Singh, Sandeep K. Arya. (2012). Genetic algorithm for the design of optimal IIR digital filters. Journal of Signal and Information Processing, no. 3, 286-292.
13. Sheng Chen, Bing L. Luk. (2010). Digital IIR filter design using particle swarm optimisation. International Journal of Modelling, Identification and Control, vol. 9, no. 4, 327-335.