Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 5901258, 17 pages
https://doi.org/10.1155/2017/5901258

Research Article

Cloud Model-Based Artificial Immune Network for Complex Optimization Problem

Mingan Wang,[1] Shuo Feng,[2] Jianming Li,[3] Zhonghua Li,[3] Yu Xue,[4] and Dongliang Guo[5]

[1]*School of Information Science and Technology, Huizhou University, Huizhou 516007, China*
[2]*School of Electronic Information and Electrical Engineering, Huizhou University, Huizhou 516007, China*
[3]*School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China*
[4]*School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing 210044, China*
[5]*Public Experimental Teaching Center, Sun Yat-sen University, Guangzhou 510006, China*

Correspondence should be addressed to Shuo Feng; fs503@hzu.edu.cn and Zhonghua Li; lizhongh@mail.sysu.edu.cn

Received 4 December 2016; Revised 1 March 2017; Accepted 7 March 2017; Published 24 May 2017

Academic Editor: J. Alfredo Hernández-Pérez

This paper proposes an artificial immune network based on cloud model (AINet-CM) for complex function optimization problems. Three key immune operators—cloning, mutation, and suppression—are redesigned with the help of the cloud model. To be specific, an increasing half cloud-based cloning operator is used to adjust the dynamic clone multipliers of antibodies, an asymmetrical cloud-based mutation operator is used to control the adaptive evolution of antibodies, and a normal similarity cloud-based suppressor is used to keep the diversity of the antibody population. To quicken the searching convergence, a dynamic searching step length strategy is adopted. For comparative study, a series of numerical simulations are arranged between AINet-CM and the other three artificial immune systems, that is, opt-aiNet, IA-AIS, and AAIS-2S. Furthermore, two industrial applications—finite impulse response (FIR) filter design and proportional-integral-differential (PID) controller tuning—are investigated and the results demonstrate the potential searching capability and practical value of the proposed AINet-CM algorithm.

1. Introduction

Biological immune system (BIS), as one of the most complex body systems, plays a significant important role in protecting our bodies from the invasion of a large variety of external bacteria, viruses, and other pathogenic organisms. Inspired by BIS, artificial immune system (AIS) has been developed as an efficient optimization method, which has great computational potentials in solving scientific computing and engineering application problems [1, 2].

The variations of AIS mainly consist of four branches of theories/models: clone selection theory [3], negative selection theory [4], danger theory [5], and artificial immune network [6]. Specifically speaking, artificial immune network is widely used in a great number of applications, such as data analysis [7], function optimization [8], signal and image processing [9], process control [10], and Internet of Things [11]. De Castro and Timmis developed an earlier version of artificial immune network. Then, it was modified and called opt-aiNet [8], which is able to maintain stable local optima solutions in solving multimodal function optimization. Inspired by omni-optimization and immune evolution, omni-aiNet [12] was proposed with dynamical population size and low redundancy. By absorbing the elitist-learning strategy of particle swarm optimization (PSO), aiNet-EL [13] is able to discriminate the elitist antibodies and the other common antibodies during the mutation operation. By redesigning three major operators, the affinity-based cloning, the affinity-based mutation, and the concentration-based suppressor, IA-AIS [14] has more adaptability to field problems. Guided by elitist antibodies, AAIS-2S [15] divides the antibodies into two subpopulations: an elitist swarm with self-learning and a common swarm with elitist-learning. These achievements suggest that artificial immune network is becoming an active and hot research field.

Cloud model [16] is a conversion model with certainty between a qualitative concept and a quantitative number expression. Up to date, cloud model has been applied in many fields [17, 18] due to its randomness and stability. For example, in intelligent computation, several cloud model-based algorithms—the cloud-based adaptive genetic algorithm (CAGA) [19], asymmetrical cloud model-based genetic algorithm (ACGA) [20], and particle swarm optimization with normal cloud model (CPSO) [21]—have been developed. It is clearly shown that the combination of the cloud model and evolutionary algorithms is of interest to researchers and engineers.

The article [22] proposes an artificial immune network based on the cloud model (AINet-CM), where the cloud models are used to evaluate the candidate antibodies. Different cloud models are embedded into three major immune operators—clone, mutation, and suppression—to enhance the algorithmic convergence. As an extensive study of [22], this paper will systematically investigate the cloud-based operators of AINet-CM and examine the convergence and accuracy of AINet-CM by evaluating unimodal or multimodal functions whose dimension is 2D, 10D, and 30D. In addition, two kinds of typical applied experiments—band-pass FIR filter designing and industrial PID controller optimization—are arranged to demonstrate the effectiveness and high-performance of AINet-CM.

The remainder of this paper is organized as follows. Section 2 reviews the principles of three artificial immune network family members and a cloud model. Section 3 describes the technical details of the proposed AINet-CM algorithm. Section 4 makes some comparisons in solution accuracy and convergence speed between the proposed AINet-CM algorithm and the other three artificial immune networks by a series of numerical simulations. Section 5 describes and explains the experimental results obtained by four immune algorithms from FIR band-pass filter designing and PID controller parameter tuning. Finally, the conclusions are made in Section 6.

2. Reviews of Artificial Immune Network and Cloud Model

2.1. The Opt-aiNet Optimization. In opt-aiNet [8], all the antibodies experience five phases: clone, mutation, selection, suppression, and recruitment. At each generation, each individual parent antibody is cloned for a fixed number Nc and all the cloned offspring but the parent one must go through mutation operator. In this case, the mutated antibody with the highest fitness is selected to enter the next generation. If the average affinity among the current population is not significantly different from that among the previous population, the suppression process will be activated. If the Euclidean distance between any two antibodies is less than a threshold value Th_s, the antibody with lower affinity will be suppressed or abandoned. And then, a certain percentage of randomly generated antibodies are recruited. The iterative process is repeated until the stopping criterion is met. Many researches demonstrated that the opt-aiNet is good at exploration but weak in exploitation for accurate solutions.

2.2. The IA-AIS Optimization. To improve the adaptability of parameters in opt-aiNet, an improved adaptive AIS (IA-AIS) algorithm [14] is proposed. In IA-AIS, the cloning operator depends on the affinity measure. In other words, the number of cloned antibodies is nonlinearly determined by the normalized affinity. And a controlled Gaussian mutation operator is able to make the mutation level decrease sharply as the affinity increases. Moreover, a concentration-based suppressor can adjust dynamically the suppression threshold, because this threshold is proportional to the similarity of antibodies. However, IA-AIS pays little attention to the improvement in solution accuracy during the iterative process.

2.3. The AAIS-2S Optimization. In PSO, it is well known that the particle always learns from the best particle. Inspired by this elitist-learning strategy of PSO, AAIS-2S [15] separates the population into two subgroups: an elitist swarm (ES) and a common swarm (CS), where ES is able to go through self-learning mutation while CS is required to learn from the best antibody in ES. Meanwhile, a swarm updating mechanism is added to make those better antibodies in CS upgrade into ES. In addition, the searching step length is adjusted dynamically according to the Euclidean distance measure between antibodies. As a result, AAIS-2S can obtain the global optima quickly but has a potential risk in getting trapped into the local optimum.

2.4. The Cloud Model. The cloud model is a transforming model between qualitative concepts and their quantitative expressions. Assume that U is defined as a quantitative universe expressed by numerical value and C is defined as a qualitative concept in U. When the quantitative value $x \in U$ means a specific random realization of C, the certainty degree of x related to qualitative concept C can be expressed as $\mu(x) \in [0, 1]$. Then, $\mu(x) \in [0, 1]$ is a random number with the stable tendency:

$$\mu : U \longrightarrow [0, 1] \quad \forall x \in U,\ x \longrightarrow \mu(x). \tag{1}$$

The distribution of x in the universe U is called *cloud* in term, and each (x, μ) is called a *cloud drop* [16]. A simple normal cloud is illustrated in Figure 1. Known from the definition above and Figure 1, it is obvious that the mapping from U to interval $[0, 1]$ is equivalent to the one-point to multipoint transition with certainty by integrating fuzzy degree and randomness.

For a specific cloud, its characteristics can be measured by three parameters, that is, *expectation* (Ex), *entropy* (En), and *hyperentropy* (He), where Ex denotes the expectation value of the distribution of the cloud drops in the universe and is the typical swatch among the cloud. En is a measure of the coverage of the qualitative concept within the universe, which determines the range of the cloud. He is the entropy of En, which is decided by both randomness and fuzzification.

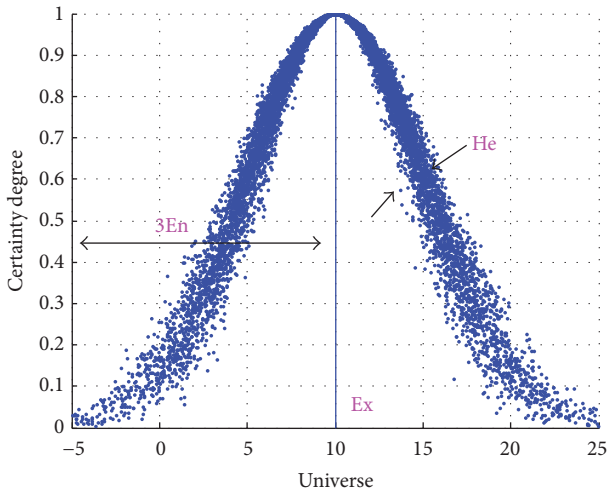


Figure 1: A normal cloud with (Ex, En, He) = (10, 5, 0.5).

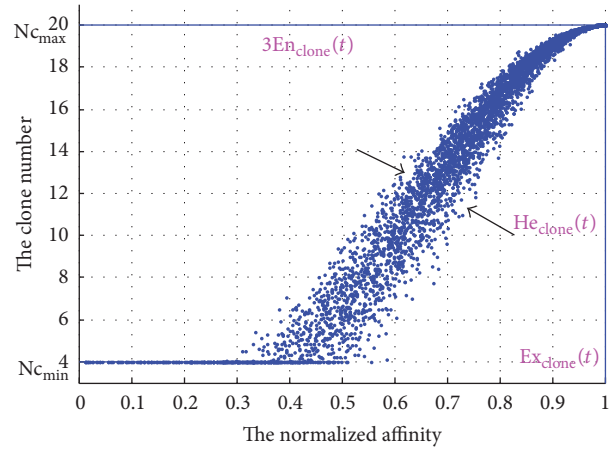


Figure 2: The increasing half cloud-based cloning operator (cloud chart) (see also Pseudocode 1).

Therefore, a cloud model can be built by these three parameters (Ex, En, He). Figure 1 illustrates a normal cloud with the numerical characteristics value (10, 5, 0.5).

3. The Proposed Artificial Immune Network Based on Cloud Model (AINet-CM)

With the help of the cloud model, it is possible that artificial immune network escapes from getting trapped into the local optimum by using the diversity and the stability of cloud model measure. Thus, it is advantageous for the candidate antibodies to evolve approximately to the global optimum. The technical details of the proposed artificial immune network based on cloud model (AINet-CM) optimization are described as follows.

3.1. Cloning Operator. Suppose that N antibodies with real number coding have been randomly generated in the initialization phase. Then, the ith antibody in the population at the tth generation can be defined as $\mathrm{Ab}(i,t)$ $(i = 1,\ldots,N)$. Generally speaking, in this cloning phase, the greater the parent antibody's affinity is, the more the cloning offspring antibodies are. It is clear that the cloned multiplier is positively related to the parent antibody's affinity.

To make sure that the antibodies with higher affinity should have larger clone multiplier, an increasing half cloud shown in Figure 2 is introduced. The characteristics of this cloud can be obtained by

$$\begin{aligned}
\mathrm{Ex}_{\mathrm{clone}}(t) &= \mathrm{Aff}^{*}_{\max}(t),\\
\mathrm{En}_{\mathrm{clone}}(t) &= \frac{\left(\mathrm{Aff}^{*}_{\max}(t) - \mathrm{Aff}^{*}_{\min}(t)\right)}{c^{1}_{\mathrm{clone}}},\\
\mathrm{He}_{\mathrm{clone}}(t) &= \frac{\mathrm{En}_{\mathrm{clone}}(t)}{c^{2}_{\mathrm{clone}}},
\end{aligned} \tag{2}$$

where $\mathrm{Aff}^{*}_{\max}(t)$ and $\mathrm{Aff}^{*}_{\min}(t)$ represent the maximum and minimum normalized affinity in the interval $[0,1]$ at the tth generation, respectively, and c^{1}_{clone} and c^{2}_{clone} are controlling parameters. While generating the increasing half cloud, $\mathrm{Ex}_{\mathrm{clone}}(t)$ should be equal to the maximum affinity value, which guarantees that the clone multiplier is larger for the antibodies with higher affinity. In addition, $\mathrm{En}_{\mathrm{clone}}(t)$—which is controlled by parameter c^{1}_{clone}—decides the range of the increasing half cloud. If $\mathrm{En}_{\mathrm{clone}}(t)$ is too small, the cloud will look narrow and most clone multipliers will be limited to the lower boundary. On the contrary, if $\mathrm{En}_{\mathrm{clone}}(t)$ is too large, the cloud will be wide and most clone multipliers will be close to the upper boundary. Based on the "3En" rule [16], $c^{1}_{\mathrm{clone}} = 3$ is chosen in this paper. $\mathrm{He}_{\mathrm{clone}}(t)$—another characteristic controlled by the parameter c^{2}_{clone}—decides the dispersion of the cloud. To be specific, too great $\mathrm{He}_{\mathrm{clone}}(t)$ will lose the stable tendency of cloud model to some extent, while too small $\mathrm{He}_{\mathrm{clone}}(t)$ will partly lose the randomness. According to the paper [20], the best range for c^{2}_{clone} equals $[6, 15]$. So, we choose $c^{2}_{\mathrm{clone}} = 10$ to evaluate $\mathrm{He}_{\mathrm{clone}}(t)$ in this paper.

After the increasing half cloud is formed, the certainty degree of individual antibody can be obtained. Then, in the proposed AINet-CM algorithm, an *increasing half cloud-based cloning operator (IHC-based cloning operator)*

$$\mathrm{Nc}(i,t) = \max\left(\mathrm{Nc}_{\max}\cdot\mu(i,t), \mathrm{Nc}_{\min}\right) \tag{3}$$

is used to determine the clone multiplier for each antibody individual. In (3), $\mathrm{Nc}_{\max}$ and $\mathrm{Nc}_{\min}$ are the upper and the lower bounds of the clone multiplier, respectively, and $\mu(i,t)$ represents the certainty degree of the ith antibody at the tth generation. For the sake of clarity, Pseudocode 1 shows pseudocode of IHC-based cloning operator and Figure 2 illustrates an increasing half cloud for $\mathrm{Nc}_{\max} = 20$ and $\mathrm{Nc}_{\min} = 4$.

3.2. Mutation Operator. To direct the mutation process, an *asymmetrical cloud-based mutation operator (AC-based mutation operator)* is well designed, which uses an asymmetrical cloud shown in Figure 3. The asymmetrical cloud consists of the left half and right half clouds, and their numerical

PROCEDURE *IHC_based_cloning_operator*(i,t)

Update cloud parameters as

$$\mathrm{Ex}_{\mathrm{clone}}(t) = \mathrm{Aff}^{*}_{\max}(t);$$

$$\mathrm{En}_{\mathrm{clone}}(t) = \frac{\left(\mathrm{Aff}^{*}_{\max}(t) - \mathrm{Aff}^{*}_{\min}(t)\right)}{c^{1}_{\mathrm{clone}}};$$

$$\mathrm{He}_{\mathrm{clone}}(t) = \frac{\mathrm{En}_{\mathrm{clone}}(t)}{c^{2}_{\mathrm{clone}}};$$

FOR $i = 1 : n$

$$\mathrm{Enn}_{\mathrm{clone}}(i,t) = \mathrm{randn}\left(\mathrm{En}_{\mathrm{clone}}(t), \mathrm{He}_{\mathrm{clone}}(t)\right);$$

$$x(i,t) = \mathrm{randn}\left(\mathrm{Ex}_{\mathrm{clone}}(t), \mathrm{Enn}_{\mathrm{clone}}(i,t)\right);$$

$$\mu(i,t) = \exp\left(\frac{-\left(x(i,t) - \mathrm{Ex}_{\mathrm{clone}}(t)\right)^{2}}{2\left(\mathrm{Enn}_{\mathrm{clone}}(i,t)\right)^{2}}\right);$$

Determine $\mathrm{Nc}(i,t)$ **by Eq.** (3);

END FOR

END PROCEDURE

Pseudocode 1

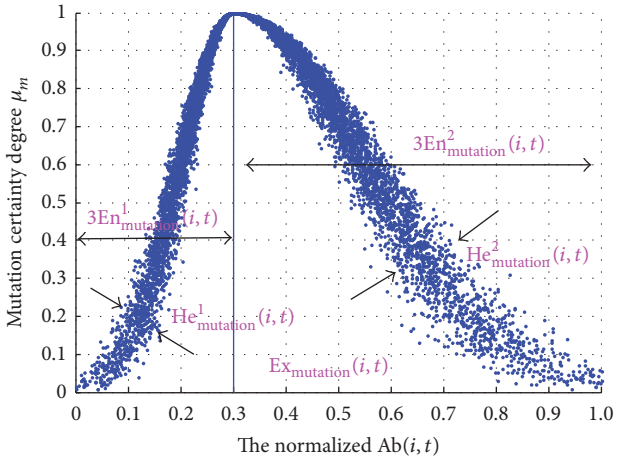


Figure 3: The asymmetrical cloud-based mutation operator (cloud chart) (see also Pseudocode 2).

characteristics can be marked as $(\mathrm{Ex}_{\mathrm{mutation}}(i,t), \mathrm{En}^{1}_{\mathrm{mutation}}(i,t), \mathrm{He}^{1}_{\mathrm{mutation}}(i,t))$ and $(\mathrm{Ex}_{\mathrm{mutation}}(i,t), \mathrm{En}^{2}_{\mathrm{mutation}}(i,t), \mathrm{He}^{2}_{\mathrm{mutation}}(i,t))$, respectively. The expectation $\mathrm{Ex}_{\mathrm{mutation}}(i,t)$ of the asymmetrical cloud denotes the mutation bit of the ith antibody at the tth generation. The entropy $\mathrm{En}_{\mathrm{mutation}}(i,t)$ represents the range of the searching step length for the ith antibody at the tth generation, and its left half and right half parts are marked as $\mathrm{En}^{1}_{\mathrm{mutation}}(i,t)$ and $\mathrm{En}^{2}_{\mathrm{mutation}}(i,t)$, respectively. Their expressions are given as

$$\begin{aligned}\mathrm{En}^{1}_{\mathrm{mutation}}(i,t) &= \frac{1}{3}\left(\mathrm{Ex}_{\mathrm{mutation}}(i,t) - V_{\min}\right)\cdot\beta(i,t),\\ \mathrm{En}^{2}_{\mathrm{mutation}}(i,t) &= \frac{1}{3}\left(V_{\max} - \mathrm{Ex}_{\mathrm{mutation}}(t)\right)\cdot\beta(i,t),\end{aligned} \tag{4}$$

where $V_{\min}$ and $V_{\max}$ represent the lower bound and upper bounds of mutation scale, respectively. According to the "3En" rule, (4) guarantees that the mutated antibody $\mathrm{Ab}(i,t+1)$ always lies within its range of domain. In addition, $\beta(i,t)$ is the controlled coefficient of searching step length, which can be obtained by

$$\begin{aligned}&\beta(i,t)\\ &= \begin{cases}\beta(i,t-1), & \text{if } \mathrm{Aff}(\mathrm{Ab}(i,t)) > \mathrm{Aff}(\mathrm{Ab}(i,t-1)),\\ \beta(i,t-1)\cdot K, & \text{elseif } \beta(i,t-1)\cdot K > \beta_{\min},\\ \beta(0,T+1), & \text{otherwise,}\end{cases}\\ &\beta(0,T)\\ &= \begin{cases}\beta(0,T-1)\cdot K, & \text{if } T>0,\ \beta(0,T)\cdot K \geq \beta_{\min},\\ \beta(0), & \text{otherwise,}\end{cases}\end{aligned} \tag{5}$$

where K is a controlled parameter in $(0,1)$, $\mathrm{Aff}(\mathrm{Ab}(i,t))$ represents the affinity of $\mathrm{Ab}(i,t)$, $\beta_{\min}$ denotes the minimum coefficient of the searching step length, and $\beta(0)$ is the initialized value in $(0,1)$. From (5), the searching step length will decrease if $\mathrm{Ab}(i,t)$ has no improvement in affinity after mutation. However, both $\beta(i,t)$ and $\beta(0,T)$ have to be greater than or equal to $\beta_{\min}$.

Moreover, $\mathrm{He}^{1}_{\mathrm{mutation}}(i,t)$ and $\mathrm{He}^{2}_{\mathrm{mutation}}(i,t)$—the left half part and right half part of hyperentropy—are determined by

$$\begin{aligned}\mathrm{He}^{1}_{\mathrm{mutation}}(i,t) &= \frac{\mathrm{En}^{1}_{\mathrm{mutation}}(i,t)}{c^{1}_{\mathrm{mutation}}},\\ \mathrm{He}^{2}_{\mathrm{mutation}}(i,t) &= \frac{\mathrm{En}^{2}_{\mathrm{mutation}}(i,t)}{c^{2}_{\mathrm{mutation}}},\end{aligned} \tag{6}$$

PROCEDURE *AC_based_mutation_operator*(i,t)

Update cloud parameters as

$\mathrm{Ex}_{\mathrm{mutation}}(i,t) = \mathrm{Ab}(i,t)$;

Decide $\mathrm{En}^1_{\mathrm{mutation}}(i,t)$ **and** $\mathrm{En}^2_{\mathrm{mutation}}(i,t)$ **by Eq.** (4)

$$\mathrm{He}^1_{\mathrm{mutation}}(i,t) = \frac{\mathrm{En}^1_{\mathrm{mutation}}(i,t)}{c^1_{\mathrm{mutation}}};$$

$$\mathrm{He}^2_{\mathrm{mutation}}(i,t) = \frac{\mathrm{En}^2_{\mathrm{mutation}}(i,t)}{c^2_{\mathrm{mutation}}};$$

$$\mathrm{Enn}^1_{\mathrm{mutation}}(i,t) = \mathrm{randn}\left(\mathrm{En}^1_{\mathrm{mutation}}(i,t), \mathrm{He}^1_{\mathrm{mutation}}(i,t)\right);$$

$$\mathrm{Enn}^2_{\mathrm{mutation}}(i,t) = \mathrm{randn}\left(\mathrm{En}^2_{\mathrm{mutation}}(i,t), \mathrm{He}^2_{\mathrm{mutation}}(i,t)\right);$$

Generate $\mu_m \in (0,1)$;

IF $\mathrm{rand}(0,1) < 0.5$ **THEN**

$$\mathrm{Ab}(i,t+1) = \mathrm{Ex}_{\mathrm{mutation}}(i,t) - \mathrm{Enn}^1_{\mathrm{mutation}}(i,t)\sqrt{-2\ln(\mu_m)};$$

ELSE

$$\mathrm{Ab}(i,t+1) = \mathrm{Ex}_{\mathrm{mutation}}(i,t) + \mathrm{Enn}^2_{\mathrm{mutation}}(i,t)\sqrt{-2\ln(\mu_m)};$$

END IF

END PROCEDURE

Pseudocode 2

where c^1_{mutation} and c^2_{mutation} are the controlling parameters. To keep the stability and the randomness, c^1_{mutation} and c^2_{mutation} should not be too large or too small. According to the paper [20], the best range for c^1_{mutation} and c^2_{mutation} is $[6, 15]$, so $c^1_{\mathrm{mutation}} = c^2_{\mathrm{mutation}} = 10$ is adopted in this paper.

According to the generated asymmetrical cloud, a mutation certainty degree μ_m is used to decide the mutated result of antibody. In the AC-based mutation operator, the searching step length adjustment mechanism can improve the solution accuracy, which is determined by the numerical characteristics of the asymmetrical cloud. The pseudocode of the AC-based mutation operator is shown in Pseudocode 2 and its corresponding asymmetrical cloud is depicted in Figure 3. After the mutation process, the antibody with the highest affinity is selected to enter the next generation.

3.3. Suppression Operator. Once the average affinity at the current generation is not significantly different from the previous one, a *normal similarity cloud-based suppression operator (NSC-based suppression operator)* will be activated. In the suppression phase, a normal similarity cloud shown in Figure 4 and Pseudocode 3 is determined by the Euclidean distance $D(i, j)$ between the ith and jth antibodies. According to the backward cloud generator [16], the numerical characteristics of the normal similarity cloud can be expressed as

$$\mathrm{Ex}_{\mathrm{suppress}} = \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{D(i,j)}{C_N^2},$$

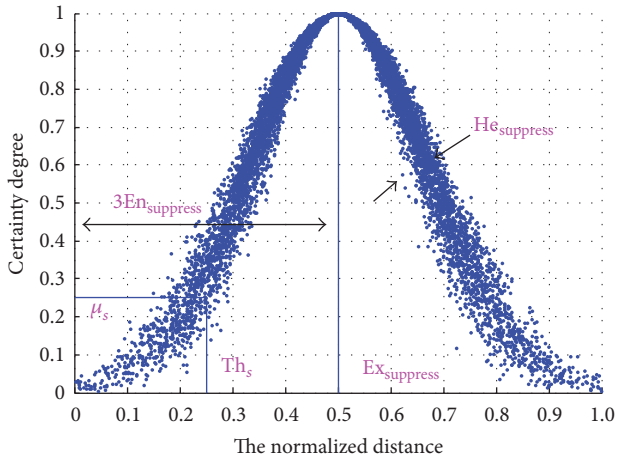


Figure 4: The normal similarity cloud-based suppression operator (cloud chart) (see also Pseudocode 3).

$$\mathrm{En}_{\mathrm{suppress}} = \sqrt{\frac{\pi}{2}} \times \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\left|D(i,j) - \mathrm{Ex}_{\mathrm{suppress}}\right|}{C_N^2},$$

$$\mathrm{He}_{\mathrm{suppress}} = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\left|D(i,j) - \mathrm{Ex}_{\mathrm{suppress}}\right|}{(C_N^2 - 1)}}. \tag{7}$$

In this normal similarity cloud, $\mathrm{Ex}_{\mathrm{suppress}}$ is the sample average which lies in the center of the cloud, $\mathrm{En}_{\mathrm{suppress}}$ is

PROCEDURE *NSC_based_suppression_operator*()

Update cloud parameters as

$$\mathrm{Ex}_{\mathrm{suppress}} = \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{D(i,j)}{C_N^2};$$

$$\mathrm{En}_{\mathrm{suppress}} = \sqrt{\frac{\pi}{2}} \times \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\left|D(i,j) - \mathrm{Ex}_{\mathrm{suppress}}\right|}{C_N^2};$$

$$\mathrm{He}_{\mathrm{suppress}} = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\left|D(i,j) - \mathrm{Ex}_{\mathrm{suppress}}\right|}{(C_N^2 - 1)}};$$

$$\mathrm{Enn}_{\mathrm{suppress}} = \mathrm{randn}\left(\mathrm{En}_{\mathrm{suppress}}, \mathrm{He}_{\mathrm{suppress}}\right);$$

Generate $\mu_s \in (0, 1)$;

Generate threshold Th_s **according to**

$$\mathrm{Th}_s = \mathrm{Ex}_{\mathrm{suppress}} - \mathrm{Enn}_{\mathrm{suppress}}\sqrt{-2\ln(\mu_s)};$$

END PROCEDURE

Pseudocode 3

derived from $\mathrm{Ex}_{\mathrm{suppress}}$ and denotes the range of the $D(i, j)$, and $\mathrm{He}_{\mathrm{suppress}}$ represents the dispersion of $D(i, j)$. According to the generated normal similarity cloud, a suppression certainty degree μ_s is used to determine the threshold Th_s. The pseudocode of NSC-based suppression operator is shown in Pseudocode 3 and its corresponding normal similarity cloud in suppression operator is illustrated in Figure 4.

For any couple of antibodies whose Euclidean distance is less than a threshold Th_s, the worse one in affinity will be suppressed or removed. After the suppression process, the number of the antibodies will decrease significantly. Thus, a number of randomly generated antibodies are recruited to keep the scale and the diversity of antibodies. Repeat the above iterative process until the termination condition is satisfied.

3.4. The Algorithmic Flowchart. The flowchart of the proposed AINet-CM algorithm is shown in Figure 5. Observed from Figure 5, the technical procedure of AINet-CM includes the following:

(1) Initializing the antibody population and parameter setting.

(2) Executing the increasing half cloud-based cloning operator.

(3) Executing the asymmetrical cloud-based mutation operator.

(4) Selecting the candidate antibodies with the highest affinity.

(5) Executing the normal similarity cloud-based suppression operator.

(6) While the stop criterion is met, the procedure is finished; otherwise turn to Step (2).

4. Numerical Simulations and Results

In this section, a series of numerical simulations are executed to examine the performance of the proposed AINet-CM algorithm. Comparisons are made among the proposed AINet-CM algorithms, opt-aiNet, IA-AIS, and AAIS-2S in solution accuracy and the convergence speed.

4.1. Benchmark Functions. For the sake of numerical evaluations, five benchmark functions are selected from *CEC2005* [23] and listed as follows.

(1) F_1 Shifted Sphere Function: unimodal

(2) F_2 Shifted Schwefel's Problem 1.2: unimodal

(3) F_6 Shifted Rosenbrock's Function: multimodal

(4) F_9 Shifted Rastrigin's Function: multimodal

(5) F_{12} Schwefel's Problem 2.13: multimodal

4.2. Parameter Settings. In the numerical simulations, the initialized population size of the four algorithms is set to 100, the maximum generation is 1000 for 2D or 5000 for 10D and 30D, and all the simulations are repeated for 50 trials. For the sake of clarity, all the parameters of four algorithms are listed in Table 1.

4.3. Simulation Results and Analyses

4.3.1. Performance Analyses in Solution Accuracy. The simulation results optimized by four algorithms in 2D, 10D, and 30D are presented in Tables 2–4, respectively. The resulting error indices derived from 50 replications consist of the best, the worst, the average, and the standard deviation (std) of the error. The best results among these algorithms are marked in bold.

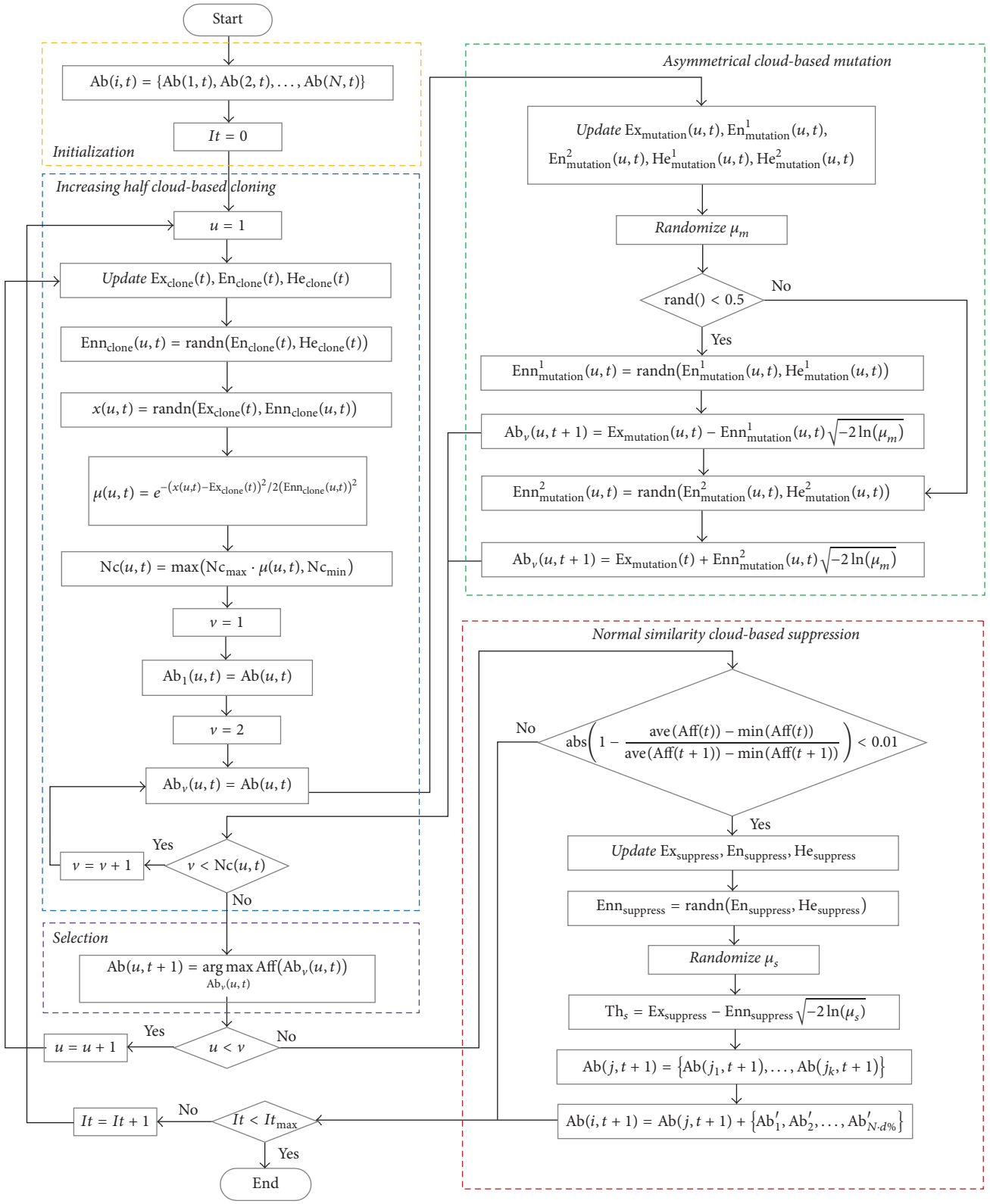


Figure 5: Flowchart of the proposed AINet-CM algorithm.

Table 1: Parameter settings of four algorithms.

Algorithm	Cloning operator	Mutation operator	Suppression operator	Others
Opt-aiNet	$\mathrm{Nc} = 20$	$\beta = 100$	$\sigma_s = 0.2$ $d = 40\%$	—
IA-AIS	$\mathrm{Nc}_{\max} = 20$ $\mathrm{Nc}_{\min} = 4$ $n = 2$	$\gamma = 2$ $\eta = 0.33$	$\xi = 0.1$	—
AAIS-2S	$\mathrm{Nc}_{\max} = 20$ $\mathrm{Nc}_{\min} = 4$ $n = 2$	$\eta = 0.33$	$\xi = 0.1$	$p_e = 0.25$ $N_{\mathrm{update}} = 10$
AINet-CM	$\mathrm{Nc}_{\max} = 20$ $\mathrm{Nc}_{\min} = 4$ $c^1_{\mathrm{clone}} = 3$ $c^2_{\mathrm{clone}} = 10$	$K = 0.5$ $\beta_{\min} = 10e-016$ $c^1_{\mathrm{mutation}} = 10$ $c^2_{\mathrm{mutation}} = 10$ $\beta_0 = 0.3$	$\mu_s = 0.3$	—

Table 2: The optimization simulation results in 2D.

Benchmark	Algorithm	Best	Worst	Average	Std
F_1	Opt-aiNet	$4.075673e-010$	$2.322654e+002$	$2.134278e+001$	$3.815297e+001$
	IA-AIS	$1.142894e-009$	$1.272159e-006$	$2.081269e-007$	$2.554790e-007$
	AAIS-2S	**0**	**0**	**0**	**0**
	AINet-CM	**0**	**0**	**0**	**0**
F_2	Opt-aiNet	$8.334382e-010$	$7.423637e+002$	$4.424344e+001$	$1.172033e+002$
	IA-AIS	$5.649667e-010$	$6.458520e-007$	$2.181431e-007$	$1.829096e-007$
	AAIS-2S	**0**	$5.684342e-014$	$1.136868e-015$	$8.038873e-015$
	AINet-CM	**0**	**0**	**0**	**0**
F_6	Opt-aiNet	$1.601273e-008$	$6.982493e+001$	$1.332500e+001$	$1.453700e+001$
	IA-AIS	$7.696849e-008$	**6.500257e − 006**	$1.330222e-006$	$1.461563e-006$
	AAIS-2S	**0**	$2.765851e-002$	$5.749097e-004$	$3.910412e-003$
	AINet-CM	**0**	$4.971701e-004$	**7.306988e − 007**	**1.300343e − 006**
F_9	Opt-aiNet	$6.197070e-010$	$1.484871e-006$	$3.183155e-007$	$3.302604e-007$
	IA-AIS	$2.181800e-007$	$9.205743e-005$	$2.840228e-005$	$2.386444e-005$
	AAIS-2S	**0**	$5.169959e-001$	$5.521124e-002$	$1.051350e-001$
	AINet-CM	**0**	**0**	**0**	**0**
F_{12}	Opt-aiNet	$7.541985e-010$	$5.799400e-007$	$6.917494e-008$	$9.323257e-008$
	IA-AIS	$8.398080e-007$	$1.160427e-004$	$2.146887e-005$	$2.347865e-005$
	AAIS-2S	**0**	$3.548364e-001$	$2.313225e-002$	$6.041159e-002$
	AINet-CM	**0**	**0**	**0**	**0**

For 2D optimization, as shown in Table 2, the proposed AINet-CM algorithm can capture the global optimum of each benchmark function except for F_6 in every trial. For instance, in optimizing functions F_1, F_2, F_9, and F_{12}, AINet-CM is always capable of finding their desired optima for every replication. As a result, the four statistical error indices are all equal to be 0. On the other hand, the other three algorithms except AAIS-2S almost do not reach the desired optima of these functions. The only exception is that AAIS-2S can get the desired optimum of F_1. For another instance, in optimizing function F_6, AINet-CM obtains the worst error $4.971701e-004$, which is greater than $6.500257e-006$ by IA-AIS, but the average and std of the error obtained by AINet-CM are much smaller.

For 10D optimization, as shown in Table 3, the proposed AINet-CM algorithm almost performs the best in all four statistical indices among four algorithms, whatever the optimized function is. For example, in optimizing functions F_1, F_6, F_9, and F_{12}, AINet-CM is always able to reach less index value in error than the other three algorithms. Specifically speaking, in optimizing F_1 and F_9, AINet-CM actually reaches their desired optima, that is, the best of error 0. For another example, in optimization function F_{12}, AINet-CM gets $1.271222e-001$ and IA-AIS does $6.819255e-002$ in terms of the worst error. Seen from Table 3, it is clear that AINet-CM loses to IA-AIS only in the worst error, but AINet-CM defeats IA-AIS in the other error indices.

Table 3: The optimization simulation results in 10D.

Benchmark	Algorithm	Best	Worst	Average	Std
F_1	Opt-aiNet	2.677496e + 003	1.447263e + 004	8.249127e + 003	3.058784e + 003
	IA-AIS	6.222499e − 004	2.049487e − 003	1.297094e − 003	3.459514e − 004
	AAIS-2S	6.107627e − 004	3.282865e − 001	3.606023e − 002	5.323906e − 002
	AINet-CM	**0**	**1.608669e − 011**	**9.518249e − 013**	**2.964222e − 012**
F_2	Opt-aiNet	2.332892e + 003	1.695573e + 004	9.691981e + 003	3.404834e + 003
	IA-AIS	6.819255e − 004	**6.819255e − 002**	1.439519e − 003	3.506100e − 004
	AAIS-2S	5.654689e + 000	7.858485e + 001	2.721496e + 001	1.389431e + 001
	AINet-CM	**5.488060e − 004**	1.271222e − 001	**1.024232e − 003**	**3.271727e − 004**
F_6	Opt-aiNet	4.059666e + 007	4.284734e + 009	1.364979e + 009	9.506303e + 008
	IA-AIS	1.919524e + 000	1.112680e + 003	1.404718e + 002	2.618577e + 002
	AAIS-2S	3.556545e + 003	9.837639e + 004	1.989282e + 004	1.866355e + 004
	AINet-CM	**2.994048e − 001**	**5.290081e + 001**	**1.066962e + 000**	**1.187079e + 001**
F_9	Opt-aiNet	1.592055e + 001	6.268373e + 001	3.993963e + 001	1.043980e + 001
	IA-AIS	7.114565e + 000	2.330663e + 001	1.634941e + 001	3.830072e + 000
	AAIS-2S	1.555034e + 001	2.822564e + 001	2.307381e + 001	2.660738e + 000
	AINet-CM	**0**	**1.215067e − 006**	**2.494085e − 008**	**1.718036e − 007**
F_{12}	Opt-aiNet	4.483347e − 002	1.999181e − 001	1.139036e − 001	3.514291e − 002
	IA-AIS	6.018482e + 000	1.337908e + 001	9.997727e + 000	1.814684e + 000
	AAIS-2S	9.189471e + 002	7.018537e + 003	3.657929e + 003	1.363907e + 003
	AINet-CM	**4.206413e − 012**	**2.696698e − 003**	**3.464392e − 004**	**7.601388e − 004**

Table 4: The optimization simulation results in 30D.

Benchmark	Algorithm	Best	Worst	Average	Std
F_1	Opt-aiNet	7.364028e + 004	9.043620e + 004	8.163091e + 004	6.326750e + 003
	IA-AIS	2.845131e − 002	3.355290e − 002	3.033896e − 002	1.764107e − 003
	AAIS-2S	3.594068e − 001	6.647420e + 003	5.031004e + 001	1.048068e + 002
	AINet-CM	**5.684342e − 012**	**5.684342e − 008**	**5.684342e − 009**	**3.54465e − 010**
F_2	Opt-aiNet	4.421657e + 004	1.179479e + 005	8.084211e + 004	2.078686e + 004
	IA-AIS	1.351844e − 001	3.844238e + 000	2.888746e + 001	1.203280e + 002
	AAIS-2S	8.063256e + 003	1.462887e + 004	1.114811e + 004	1.691137e + 003
	AINet-CM	**3.001361e + 000**	**2.047495e + 001**	**8.902851e + 000**	**3.844238e + 000**
F_6	Opt-aiNet	2.167594e + 010	8.408124e + 010	5.198204e + 010	1.449699e + 010
	IA-AIS	2.945045e + 001	1.243707e + 004	1.887542e + 003	3.299679e + 003
	AAIS-2S	1.759388e + 005	3.462921e + 008	2.484637e + 006	5.201979e + 007
	AINet-CM	**2.198607e + 000**	**2.419968e + 002**	**4.786265e + 001**	**6.821265e + 001**
F_9	Opt-aiNet	2.356387e + 002	2.895655e + 002	2.608678e + 002	1.794656e + 001
	IA-AIS	1.427379e + 002	2.238784e + 002	2.004291e + 002	2.165710e + 001
	AAIS-2S	8.164155e + 001	1.216505e + 002	1.042647e + 002	1.211470e + 001
	AINet-CM	**3.979836e + 000**	**2.089416e + 001**	**8.046239e + 000**	**5.393961e + 000**
F_{12}	Opt-aiNet	3.661858e + 001	**1.333772e + 002**	**8.587916e + 001**	**3.176119e + 001**
	IA-AIS	1.935926e + 003	2.898353e + 003	2.290360e + 003	3.430398e + 002
	AAIS-2S	8.728301e + 004	1.370337e + 005	1.149788e + 005	1.681238e + 004
	AINet-CM	**5.106115e + 000**	2.580782e + 003	9.023991e + 002	7.573875e + 002

For 30D optimization, as shown in Table 4, AINet-CM outperforms other algorithms except for F_{12}. To be specific, in optimizing F_1, F_2, F_6, and F_9, AINet-CM obtains the smallest index value in the best, worst, average, and std of error among four compared algorithms. Specially in optimizing F_1, AINet-CM gets relatively high-accuracy solution because it has the best error of 5.684342e − 012, the worst error of 5.684342e − 008, the average error of 5.684342e − 009, and the std of error

of 3.54465$e-010$. As an exception, in optimizing F_{12}, AINet-CM performs worse in the worst error, the average error, and the std error than opt-aiNet, but the best error captured by AINet-CM is better than that by other algorithms.

Seen from these results, it is obvious that the searching step length adjustment mechanism increases the solution accuracy of AIS. The improvements in the average error and the std of error prove that this mechanism is able to efficiently and dynamically adjust the searching range in order to guarantee the antibodies' evolution in affinity even when the error is rather small. However, according to the worst error in optimizing 10D function F_2 and the results in 30D function F_{12}, it is similar to other algorithms that AINet-CM still has small potential risk in getting trapped into some local optima.

4.3.2. Performance Analyses in Convergence Speed. Figures 6–8 present the average convergence processes for the five benchmark functions. It is clear that the proposed AINet-CM algorithm has much faster convergence speed than the other three algorithms in optimizing 2D, 10D, and 30D benchmark functions in most situations.

For 2D optimization, as shown in Figures 6(a)–6(e), the proposed AINet-CM algorithm can obtain the optima easily within only 100 generations in optimizing functions F_1, F_2, F_9, and F_{12}. For function F_6, although the AINet-CM algorithm cannot get the optima within 1000 generations, it still has much faster convergence speed than the other algorithms. For 10D optimization, as shown in Figures 7(a)–7(e), it is obvious that the proposed AINet-CM algorithm outperforms the other three algorithms in optimizing functions F_1, F_6, F_9, and F_{12}. Especially for function F_2, AINet-CM is a little slower convergence speed than IA-AIS and AAIS-2S in the earlier phase. However, after 2200 generations, AINet-CM still keeps faster speed than the other three algorithms. This is because that a tradeoff is made between the solution accuracy and the convergence speed. For 30D optimization, as shown in Figures 8(a)–8(e), the proposed AINet-CM algorithm has better convergence speed in optimizing functions F_1, F_2, F_6, and F_9. Especially for function F_{12}, seen from Figure 8(e), AINet-CM can get the best optima, although AINet-CM has a slower average convergence speed than opt-aiNet.

Observed from the convergence curves in Figures 6–8, the convergence speed of AINet-CM is improved significantly due to usage of cloud model measure. The results indicate that the diversity and stability of cloud model increase the probability of antibodies to evolve towards the global optimum and further decrease the risk in falling into the local optima.

5. Applications in FIR Filter Design and PID Parameter Tuning

5.1. Application in Designing FIR Filter. Finite impulse response (FIR) filter is known as a nonrecursive filter that the response due to an impulse input will decay within finite time [24]. Due to the lack of feedback and the symmetrical characteristics about the center tap position, FIR filter can be guaranteed to have strict linear phase at all frequencies. In addition, FIR filter has many desirable characteristics such as stability, robustness, and digital implementation. Therefore, FIR filter has broad applications in communications, image processing, pattern recognition, and so forth.

The z transform of an Q-point FIR filter is characterized by

$$H(z) = \sum_{q=0}^{Q-1} h(q) z^{-q}, \tag{8}$$

where $h(q)$ is the impulse response and is finite, that is, $0 \le q \le Q-1$. So the frequency response of the FIR filter can be calculated as

$$H\left(e^{j\omega_k}\right) = \sum_{q=0}^{M-1} h(q) e^{-j\omega_k q}, \tag{9}$$

where $\omega_k = 2\pi k/M$ and $H(e^{j\omega_k})$ is the Fourier transform complex vector and the frequency is sampled in $[0, \pi]$ with M points. Hence, the optimal filter design method is employed to minimize a particular error. The least squared error can be obtained by

$$\begin{aligned} E &= \sum_{k=1}^{M} \left(H_{\text{ideal}}\left(e^{j\omega_k}\right) - H_{\text{designed}}\left(e^{j\omega_k}\right)\right)^2 \\ &= \sum_{k=1}^{M} \left(\sum_{q=0}^{Q-1} h(q) e^{j\omega_k q} - H_{\text{designed}}\left(e^{j\omega_k}\right)\right)^2, \end{aligned} \tag{10}$$

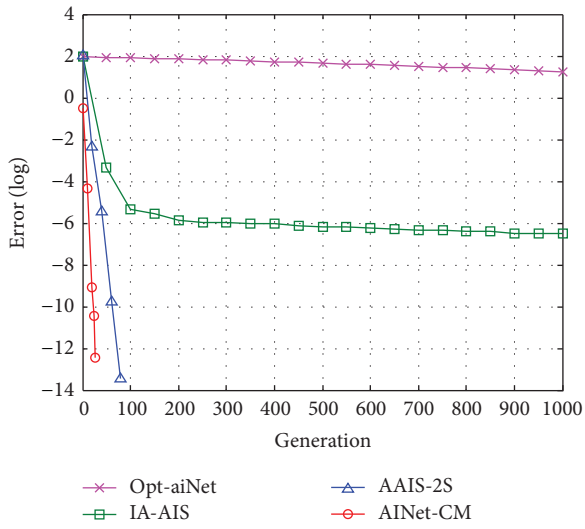
where $H_{\text{ideal}}(e^{j\omega_c})$ represents the magnitude response of the ideal filter and $H_{\text{designed}}(e^{j\omega_c})$ represents the filter to be designed. To design a FIR filter is focused on determining the set of $\{h(0), h(1), \ldots, h(Q-1)\}$ to minimize the least squared error.

In this application, an ideal band-pass FIR filter is to be designed in which the frequency response is expected as

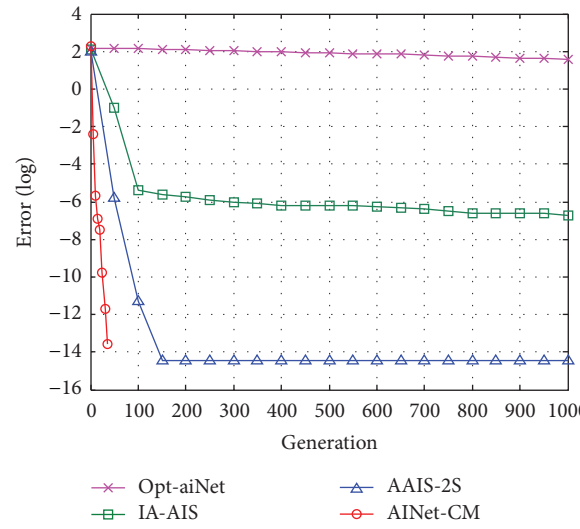
$$H_{\text{ideal}}\left(e^{j\omega}\right) = \begin{cases} 1, & 0.32\pi \le \omega \le 0.68\pi, \\ 0, & 0 \le \omega < 0.32\pi \text{ or } 0.68\pi < \omega \le \pi. \end{cases} \tag{11}$$

Moreover, M is set to be 100. The proposed AINet-CM algorithm is compared with opt-aiNet, IA-AIS, and AAIS-2S in searching the least square error, and the maximum of generation is 1000.

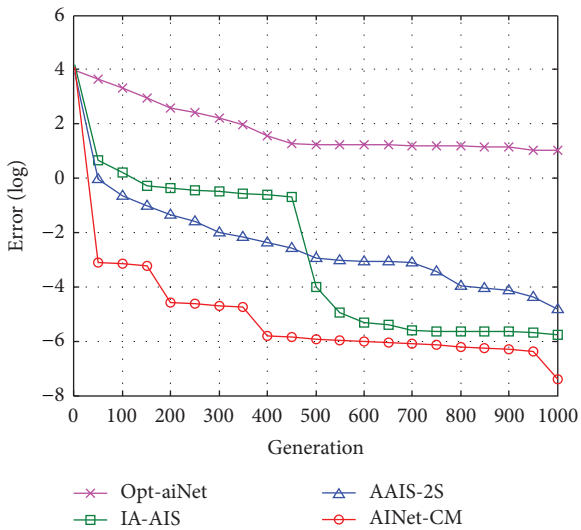
The simulation results reached by four algorithms with $Q = 10$ and $Q = 30$ are presented in Tables 5-6, respectively. The results include the best, worst, average, and standard deviation (std) of the error, and the best results are typed in bold. Seen from Tables 5-6, the results obtained by AINet-CM are smaller than the other three algorithms. To be specific, with $Q = 10$, AINet-CM harvests the best error of 3.436117, the worst error of 3.438717, the average error of 3.436537, and the std of error of 7.202963$e-004$. With $Q = 30$, AINet-CM reaches the best error of 1.122552, the worst error of



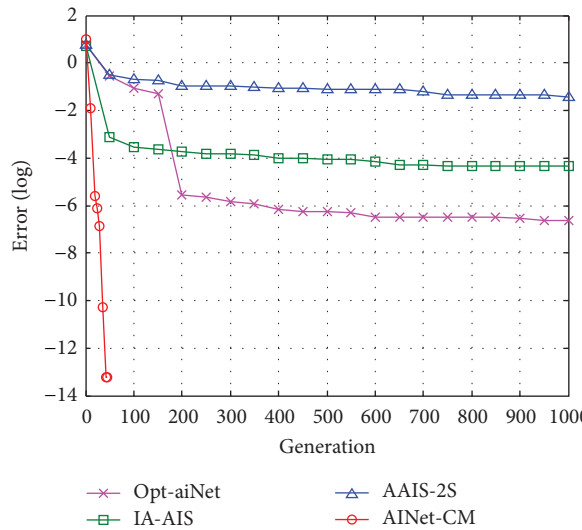
(a) Average convergence of F_1 in 2D



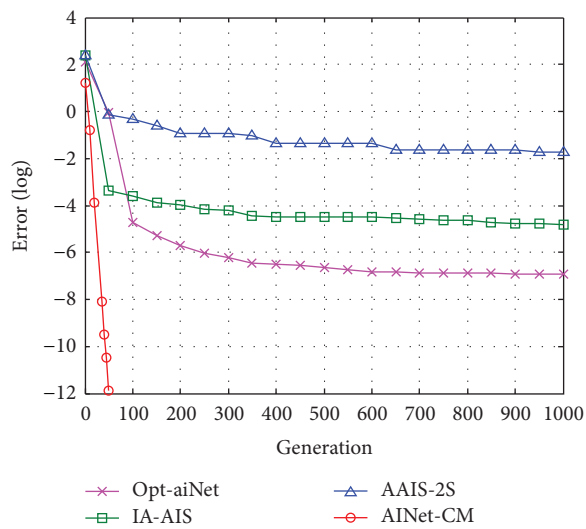
(b) Average convergence of F_2 in 2D



(c) Average convergence of F_6 in 2D

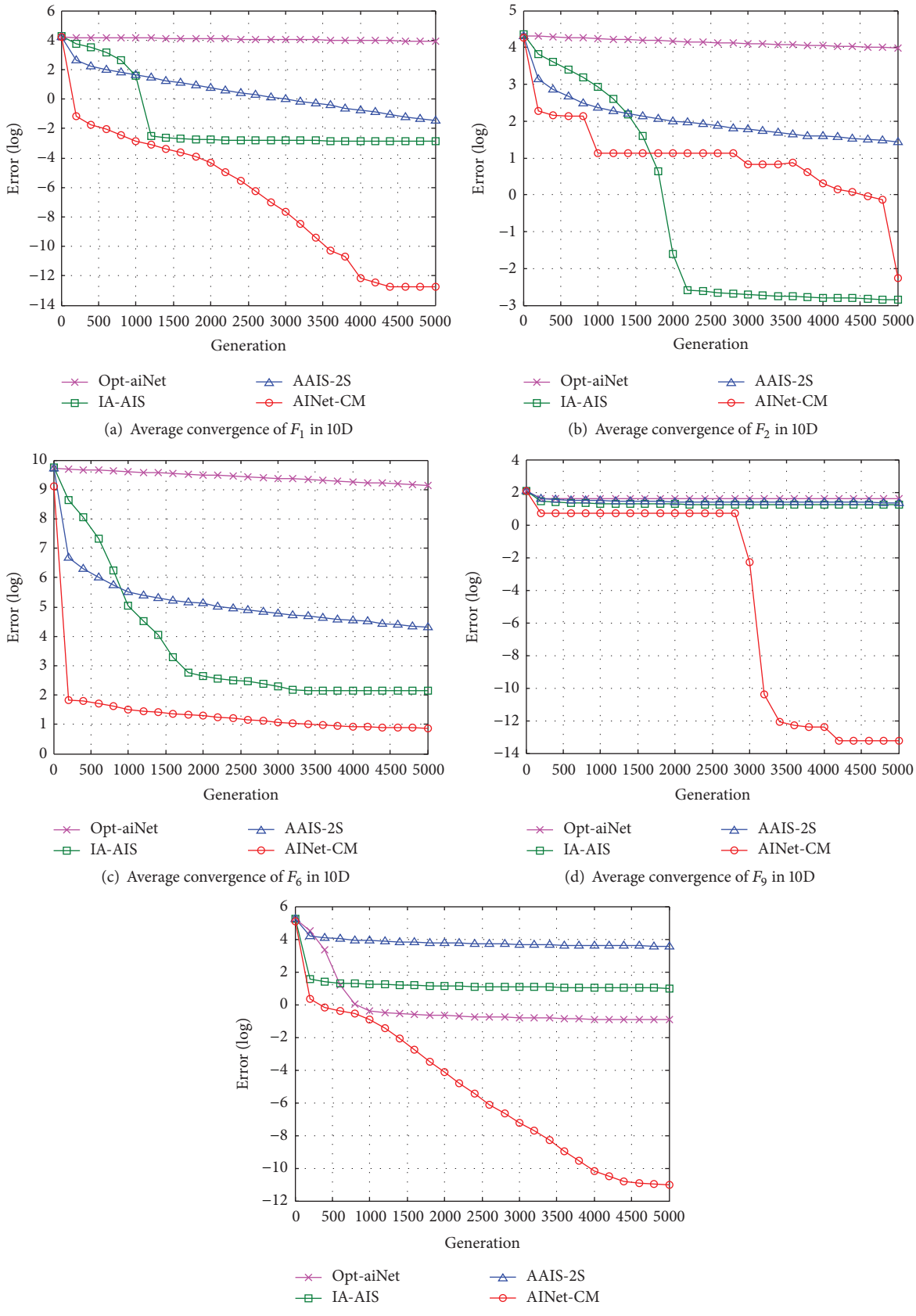


(d) Average convergence of F_9 in 2D



(e) Average convergence of F_{12} in 2D

Figure 6: Average convergence processes of five 2D benchmark functions.



(a) Average convergence of F_1 in 10D

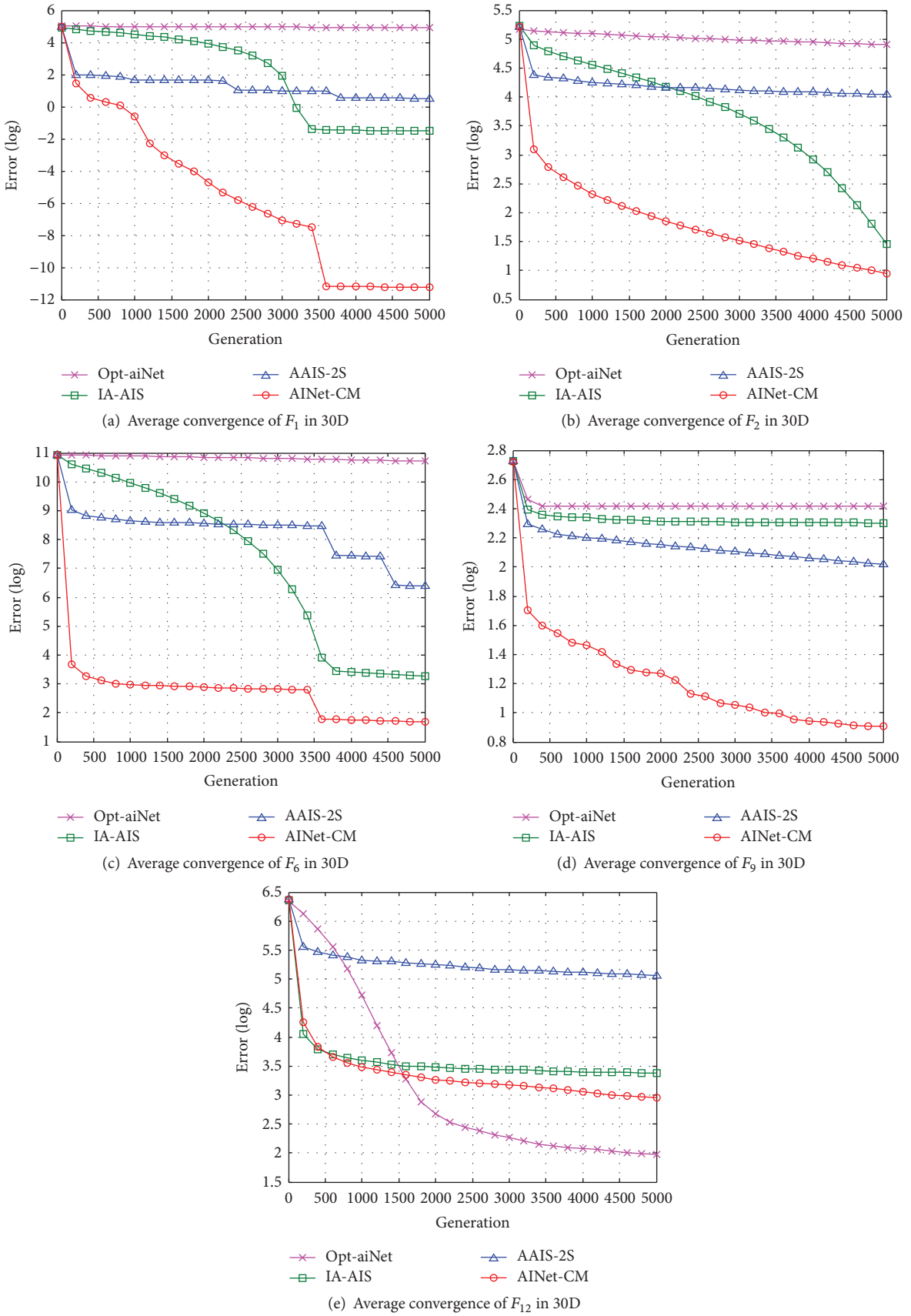
(b) Average convergence of F_2 in 10D

(c) Average convergence of F_6 in 10D

(d) Average convergence of F_9 in 10D

(e) Average convergence of F_{12} in 10D

Figure 7: Average convergence processes of five 10D benchmark functions.



(a) Average convergence of F_1 in 30D

(b) Average convergence of F_2 in 30D

(c) Average convergence of F_6 in 30D

(d) Average convergence of F_9 in 30D

(e) Average convergence of F_{12} in 30D

Figure 8: Average convergence processes of five 30D benchmark functions.

Table 5: Simulation results in FIR filter designing ($Q = 10$).

Algorithm	Best	Worst	Average	Std
Opt-aiNet	3.436566e + 000	3.439407e + 000	3.436971e + 000	2.458096e − 003
IA-AIS	3.460142e + 000	3.542927e + 000	3.510426e + 000	2.657726e − 002
AAIS-2S	3.436307e + 000	3.471676e + 000	3.447269e + 000	9.706593e − 003
AINet-CM	**3.436117e + 000**	**3.438717e + 000**	**3.436537e + 000**	**7.202963e − 004**

Table 6: Simulation results in FIR filter designing ($Q = 30$).

Algorithm	Best	Worst	Average	Std
Opt-aiNet	1.131956e + 000	1.137890e + 000	1.135113e + 000	1.818352e − 003
IA-AIS	2.716332e + 000	3.592075e + 000	3.112379e + 000	2.250955e − 001
AAIS-2S	1.132001e + 000	1.134296e + 000	1.133744e + 000	1.525802e − 003
AINet-CM	**1.122552e + 000**	**1.122616e + 000**	**1.122571e + 000**	**1.661895e − 005**

1.122616, the average error of 1.122571, and the std of error of 1.661895e − 005. In particular, whatever Q equals 10 or 30, the std of error obtained by AINet-CM is always the smallest, which means the most consistent solution among replications. It is clear that AINet-CM is capable of capturing better solutions than opt-aiNet, IA-AIS, and AAIS-2S.

Figure 9 illustrates the frequency responses by using the best solutions in Tables 5-6. Seen from Figure 9(a) when Q equals 10, the amplitude responses produced by four algorithms are similar in the pass band, while the frequency responses produced by opt-aiNet and AINet-CM outperform the other algorithms in the stop band because there exist smaller amplitudes in the figure. Observed from Figure 9(b) when Q equals 30, all the algorithms except IA-AIS produce similar frequency responses to the ideal one in the pass band. However, AINet-CM produces much smaller amplitude response (in red) than the other three algorithms in the stop band. Hence, it would be concluded that the band-pass filter designed by AINet-CM is superior to the other three algorithms.

5.2. Application in Tuning PID Controller Parameters. The proportional-integral-derivative (PID) controllers are the most popular controllers in the process industries [25]. Although PID controllers are characteristic of effectiveness, strong robustness, and simple implementation, they are poorly tuned. Therefore, parameter tuning s is crucial to PID controller design. The closed-loop diagram of the PID control system is illustrated in Figure 10.

In Figure 10, $R(s)$, $E(s)$, $U(s)$, and $Y(s)$ are the reference input, the error input, the controller output, and the system output (also the feedback variable), respectively, $C(s)$ is the transfer function of PID controller and $G(s)$ is the controlled object transfer function, and K_p, K_i, and K_d are the proportional, integral, and derivative parameters of the PID controller, respectively. So the transfer function $C(s)$ is expressed as

$$C(s) = K_p + K_i \frac{1}{s} + K_d s. \tag{12}$$

In this application, the object transfer function $G(s)$ is modeled as

$$G(s) = \frac{1}{s^4 + 6s^3 + 11s^2 + 6s}. \tag{13}$$

Activated by a step pulse stimulus, the parameters K_p, K_i, and K_d of PID controller are required to be tuned to meet the desired performance criteria of the entire control system. In this paper, two indices, the integral of absolute magnitude of the error (IAE) and the integral of time-weight absolute error (ITAE), are selected as the performance criteria. They are mathematically expressed by

$$\begin{aligned} \text{IAE} &= \int_0^\infty |e(t)|\, dt, \\ \text{ITAE} &= \int_0^\infty t\, |e(t)|\, dt, \end{aligned} \tag{14}$$

where $e(t)$ is the negative feedback control system error. In addition, the size of initialized population is 80 and the maximum number of generation is 100 in the simulations.

Tables 7-8 present the simulation results optimized by the four algorithms. Seen from these tables, the IAE value and the ITAE value optimized by AINet-CM are 1.3204049e+002 and 1.5664397e + 001, respectively, which are smaller than those by the other three algorithms. It means that AINet-CM obtains better PID parameters than the other algorithms.

For the sake of clear observations, Figure 11 shows the step response curves corresponding to four algorithms, which are marked in different colors such as pink (opt-aiNet), green (IA-AIS), blue (AAIS-2S) and red (AINet-CM). Compared to the other three algorithms, seen from Figure 11, the proposed AINet-CM algorithm has smaller overshoot or less settle time in both IAE and ITAE measures. Specifically speaking, PID controller obtains a better ideal step response curve which is tuned by AINet-CM when ITAE is used. To be specific, in Figure 11(a), AINet-CM harvests the least settle time of 5.5 seconds even if its overshoot and rise time are not the smallest. In Figure 11(b), the proposed AINet-CM algorithm

Table 7: Simulation results in PID parameter tuning (IAE).

Algorithm	PID parameters			IAE
	K_p	K_i	K_d	
Opt-aiNet	1.4629331e + 001	6.4300879e + 000	5.1237192e − 003	1.3618251e + 002
IA-AIS	1.2500524e + 001	5.4443793e + 000	2.6231018e − 003	1.3625645e + 002
AAIS-2S	1.6155824e + 001	1.0210501e + 001	2.6938200e + 000	1.7934799e + 002
AINet-CM	**1.1785053e + 001**	**5.9124436e + 000**	**6.2479890e − 004**	**1.3204049e + 002**

Table 8: Simulation results in PID parameter tuning (ITAE).

Algorithm	PID parameters			ITAE
	K_p	K_i	K_d	
Opt-aiNet	1.5915666e + 001	7.0229236e + 000	2.9382817e − 003	2.2933311e + 001
IA-AIS	1.2001105e + 001	1.0006757e + 001	2.7380186e + 000	3.8046764e + 001
AAIS-2S	1.2471515e + 001	6.4891158e + 000	9.2537522e − 005	1.8178800e + 001
AINet-CM	**8.7795676e + 000**	**5.4957820e + 000**	**1.1497691e − 004**	**1.5664397e + 001**

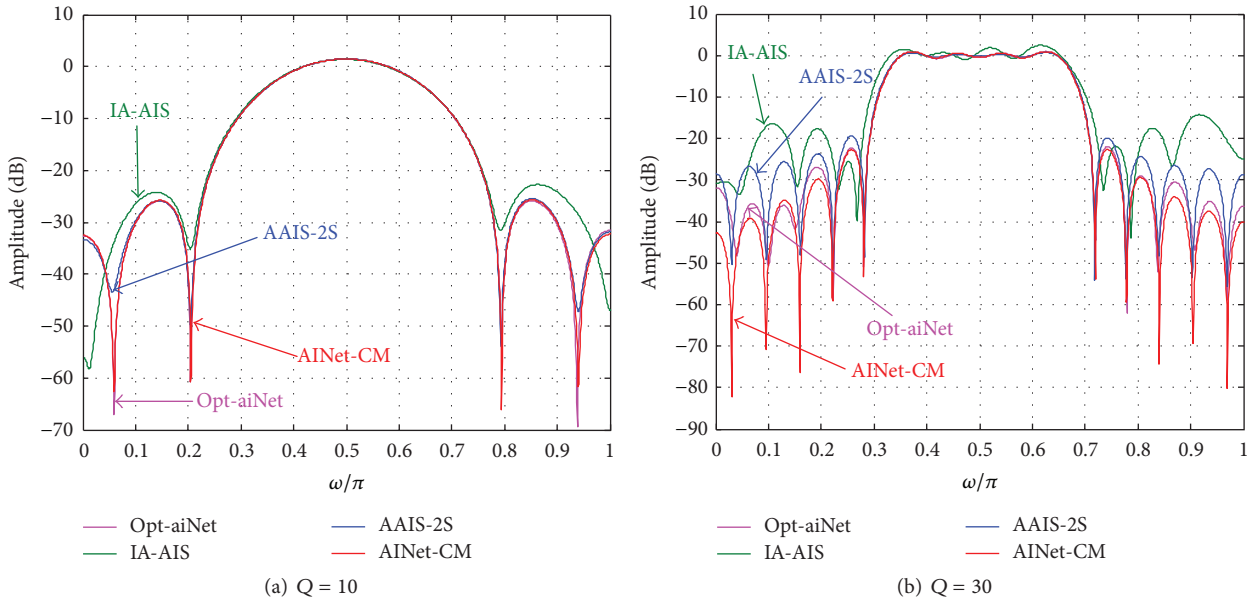


Figure 9: The frequency response of the band-pass filter.

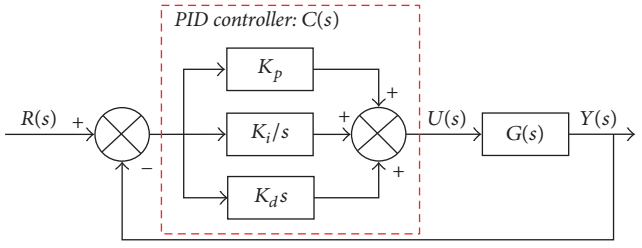


Figure 10: The block diagram of the PID control system.

obtains the overshoot of 10% and the settle time of 6 seconds while the other algorithms get larger overshoot and settle time. These results indicate that the proposed AINet-CM algorithm is more capable of finding the optimal parameters in PID controller tuning.

6. Conclusion

In this paper, an artificial immune network based on cloud model (AINet-CM) is proposed for complex optimization problems. By introducing the cloud model, the proposed AINet-CM algorithm is formed by redesigning three immune operators, that is, the increasing half cloud-based cloning operator, the asymmetrical cloud-based mutation operator, and the normal similarity cloud-based suppression operator.

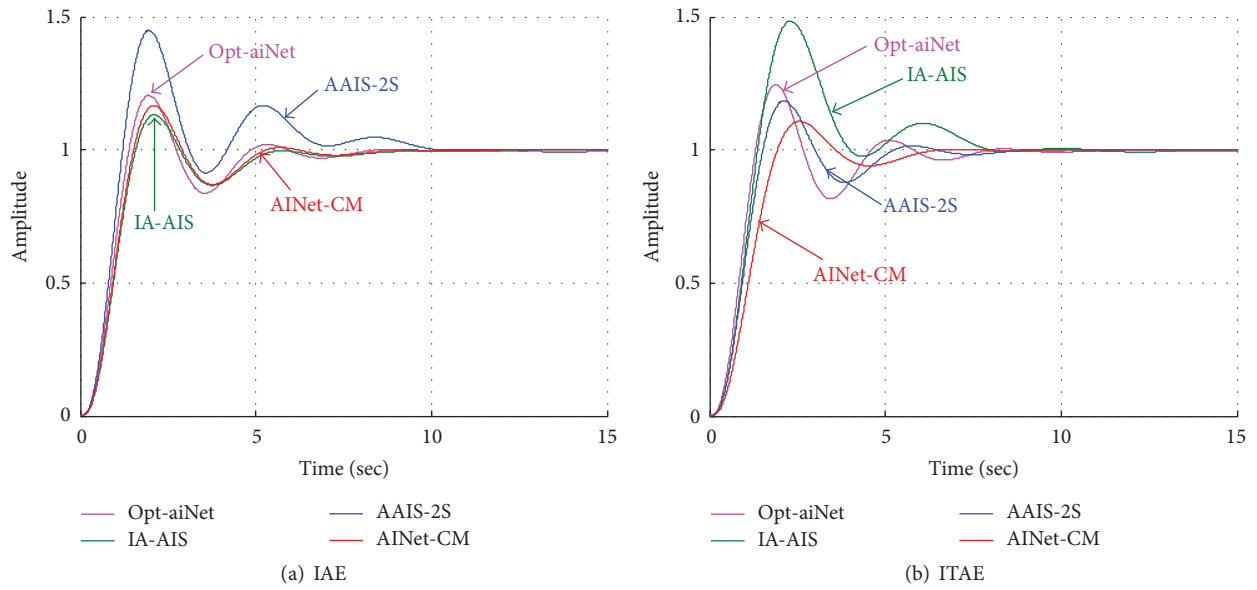


Figure 11: Step response of PID controller.

A series of numerical simulations are executed, and the resulting data indicate that the proposed AINet-CM algorithm has much less error in solution accuracy and much faster convergence speed in most situations by comparison with opt-aiNet, IA-AIS, and AAIS-2S in optimizing 2D, 10D, and 30D functions. Further, the simulation results in FIR filter design show that the proposed AINet-CM algorithm provides an efficient and superior approach for digital filter design. The simulation results in tuning PID parameters prove that PID controller optimized by AINet-CM is superior to those by opt-aiNet, IA-AIS, and AAIS-2S.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is partially supported by the National Science Foundation of China (no. 61471122) and the Science and Technology Program of Huizhou (nos. 2013B020015006 and 2014B020004025). Also, it is funded by the Priority Academic Program Development (PAPD) of Jiangsu Higher Education Institutions, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology (CICAEET).

References

[1] D. Dasgupta, S. Yu, and F. Nino, "Recent advances in artificial immune systems: models and applications," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1574–1587, 2011.

[2] A. Askarzadeh and A. Rezazadeh, "Artificial neural network training using a new efficient optimization algorithm," *Applied Soft Computing Journal*, vol. 13, no. 2, pp. 1206–1213, 2013.

[3] W. Tian, L. Ai, Y. Geng, and J. Liu, "A new fuzzy identification approach using support vector regression and immune clone selection algorithm," in *Chinese Control and Decision Conference, CCDC 2009*, pp. 1234–1239, June 2009.

[4] O. S. Soliman and A. Adly, "Quantum-negative selection algorithm for associative classification," in *Proceedings of IEEE International Conference on Granular Computing, GrC 2012*, pp. 418–423, August 2012.

[5] M. Yin, T. Zhang, and Y. Shu, "An artificial immune model with danger theory based on changes," in *Prooceedings of International Conference on Computer Science and Service System, CSSS 2012*, pp. 672–676, August 2012.

[6] M. Gong, X. Chen, L. Ma, Q. Zhang, and L. Jiao, "Identification of multi-resolution network structures with multi-objective immune algorithm," *Applied Soft Computing Journal*, vol. 13, no. 4, pp. 1705–1717, 2013.

[7] Z. H. Li and H.-Z. Tan, "A combinational clustering method based on artificial immune system and support vector machine," in *Proceedings of International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pp. 153–162, 2006.

[8] L. N. De Castro and J. Timmis, "An artificial immune network for multimodal function optimization," in *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2002*, pp. 699–704, May 2002.

[9] Z. Li, C. He, and H.-Z. Tan, "AINet-SL: artificial immune network with social learning and its application in FIR filter designing," *Applied Soft Computing Journal*, vol. 13, no. 12, pp. 4557–4569, 2013.

[10] M. Wang, S. Feng, C. He, Z. Li, and Y. Xue, "An artificial immune system algorithm with social learning and its application in industrial pid controller design," *Mathematical Problems in Engineering*, vol. 2017, 13 pages, 2017.

[11] Z. Li, J. Li, S. Wang, C. Tang, and M. Xia, "Comparative performance analysis of various artificial immune networks applied to RFID reader-to-reader collision avoidance," *Applied Soft Computing Journal*, vol. 32, pp. 553–567, 2015.

[12] G. P. Coelho and F. J. Von Zuben, "omni-aiNet: an immune-inspired approach for omni optimization," in *Prooceedings of the 5th International Conference on Artificial Immune Systems*, pp. 294–308, 2006.

[13] Z. Li, Y. Zhang, and H.-Z. Tan, "An efficient artificial immune network with elite-learning," in *Proceedings of the 3rd International Conference on Natural Computation, ICNC 2007*, pp. 213–217, August 2007.

[14] Z. Li, Y. Zhang, and H.-Z. Tan, "IA-AIS: An improved adaptive artificial immune system applied to complex optimization problems," *Applied Soft Computing Journal*, vol. 11, no. 8, pp. 4692–4700, 2011.

[15] Z. H. Li, C. H. He, and H. Z Tan, "The AAIS-2S optimization: an efficient adaptive artificial immune system with two swarms," in *Proceedings of the the International Conference on Computational Intelligence and Software Engineering*, pp. 5–8, 2012.

[16] D. Y. Li, H. J. Meng, and X. M. Shi, "Membership clouds and membership cloud generators," *Journal of Computer Research and Development*, vol. 32, pp. 15–20, 1995.

[17] H. Wang, S. He, X. Liu et al., "Simulating urban expansion using a cloud-based cellular automata model: a case study of Jiangxia, Wuhan, China," *Landscape and Urban Planning*, vol. 110, no. 1, pp. 99–112, 2013.

[18] Y. Jiang, X. Wang, and F. Lin, "Voice communication network quality of service estimation and forecast based on cloud model," *Applied Mechanics and Materials*, vol. 284–287, pp. 3463–3467, 2013.

[19] Y. Jiang, J. Jiang, and Y. Zhang, "A novel fuzzy multiobjective model using adaptive genetic algorithm based on cloud theory for service restoration of shipboard power systems," *IEEE Transactions on Power Systems*, vol. 27, no. 2, pp. 612–620, 2012.

[20] Q. Fu, Z.-H. Cai, and Y.-Q. Wu, "A novel hybrid method: genetic algorithm based on asymmetrical cloud model," in *Proceedings of the 2010 International Conference on Artificial Intelligence and Computational Intelligence, AICI 2010*, pp. 445–449, October 2010.

[21] M.-W. Li, W.-C. Hong, and H.-G. Kang, "Urban traffic flow forecasting using Gauss-SVR with cat mapping, cloud model and PSO hybrid algorithm," *Neurocomputing*, vol. 99, pp. 230–240, 2013.

[22] Z. Li, J. Li, D. Guo, and Z. Yang, "A cloud-based artificial immune network for optimization," in *Proceedings of the 9th International Conference on Natural Computation, ICNC 2013*, pp. 628–633, July 2013.

[23] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Tech. Rep., Nanyang Technological University, Singapore, 2005.

[24] C. Wu, D. Gao, and K. Lay Teo, "A direct optimization method for low group delay FIR filter design," *Signal Processing*, vol. 93, no. 7, pp. 1764–1772, 2013.

[25] O. Karasakal, M. Guzelkaya, I. Eksin, E. Yesil, and T. Kumbasar, "Online tuning of fuzzy PID controllers via rule weighing based on normalized acceleration," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 184–197, 2013.