Research Article

An Adaptive Fuzzy Chicken Swarm Optimization Algorithm

Zhenwu Wang ¹, Chao Qin ¹, Benting Wan ², William Wei Song ^{2,3}
and Guoqiang Yang ¹

¹Department of Computer Science and Technology, China University of Mining and Technology, Beijing 100083, China

²School of Software and IoT Engineering, Jiangxi University of Finance & Economics, Nanchang 330013, China

³Department of Information Systems, Dalarna University, Falun S-791 88, Sweden

Correspondence should be addressed to Zhenwu Wang; wangzhenwu@126.com and William Wei Song; wso@du.se

Received 22 September 2020; Revised 24 January 2021; Accepted 6 February 2021; Published 1 March 2021

Academic Editor: Carlos-Renato Vázquez

Copyright © 2021 Zhenwu Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The chicken swarm optimization (CSO) algorithm is a new swarm intelligence optimization (SIO) algorithm and has been widely used in many engineering domains. However, there are two apparent problems with the CSO algorithm, i.e., slow convergence speed and difficult to achieve global optimal solutions. Aiming at attacking these two problems of CSO, in this paper, we propose an adaptive fuzzy chicken swarm optimization (FCSO) algorithm. The proposed FCSO uses the fuzzy system to adaptively adjust the number of chickens and random factors of the CSO algorithm and achieves an optimal balance of exploitation and exploration capabilities of the algorithm. We integrate the cosine function into the FCSO to compute the position update of roosters and improve the convergence speed. We compare the FCSO with eight commonly used, state-of-the-art SIO algorithms in terms of performance in both low- and high-dimensional spaces. We also verify the FCSO algorithm with the nonparametric statistical Friedman test. The results of the experiments on the 30 black-box optimization benchmarking (BBOB) functions demonstrate that our FCSO outperforms the other SIO algorithms in both convergence speed and optimization accuracy. In order to further test the applicability of the FCSO algorithm, we apply it to four typical engineering problems with constraints on the optimization processes. The results show that the FCSO achieves better optimization accuracy over the standard CSO algorithm.

1. Introduction

There are many optimization problems in scientific and engineering domains [1–4], and traditional optimization methods (TOMs) [5], such as gradient descent method and Newton's method, cannot solve these optimization problems very well because of the following three reasons. (1) Most of the TOMs require that the objective function be convex and continuously differentiable. These strict requirements determine that the TOMs cannot be applied widely. (2) Due to their low calculation efficiency, it is difficult for the TOMs to deal with the NP-hard problem. (3) It is also difficult for the TOMs to avoid falling into the trap of local optimal solutions. Compared to the TOMs, the swarm intelligence optimization (SIO) algorithms (SIOAs) can handle the NP-hard problem beautifully owing to these obvious advantages: (1) flexibility: the whole population can adapt to the changing environment quickly. (2) Robustness: even if a few

individuals cannot work, the whole population can still work normally. (3) Self-organization: the whole population requires relatively little supervision or top-down control. (4) High efficiency: the SIOAs pay more attention to the improvement of the optimized efficiency, which can acquire the satisfied optimal or suboptimal solutions in an acceptable time.

In last decades, many classic SIOAs have been proposed, such as genetic algorithm (GA) [6], particle swarm optimization (PSO) algorithm [7], and artificial fish school (AFS) algorithm [8]. Among others, the chicken swarm optimization (CSO) algorithm [9] is a new SIO method proposed in 2014, which simulates the social relationship structure and foraging behaviour in a chicken swarm. Many variants of the CSO algorithm have been proposed recently aiming for bettering. Wu et al. [10] proved the convergence of CSO using Markov chain and gave an improved version of the CSO algorithm, which modifies

the update formula of chicks to learn from both their mother hens and rooster in their group. Qu et al. [11] used adaptive distribution to replace the Gaussian distribution in the update formula of rooster in order to balance the global and local searching abilities. Li et al. [12] introduced several improved factors learned from both the grey wolf optimizer (GWO) and the particle swarm optimization (PSO) to extend the searching capability. Li et al.'s method also integrates a duplicate remove operator to enhance the diversity of the chicken population. Torabi and Safi-Esfahani [13] combined CSO with an improved raven roosting optimization (RRO) algorithm in order to balance the global and local searching capabilities. Fu et al. [14] modified the update equation of roosters and introduced a mutation operator to solve the problem of easily falling into a trap of local optimal solutions. Furthermore, the CSO algorithm has also been extended to solve the constraint optimization problem [15] (see Section 6.1), 0-1 knapsack problem [16], and multiobjective optimization problem [17].

Although the research work of CSO has made some progress, however, for CSO, like the other SIOAs, these two obvious drawbacks still exist: easily falling into local optimum and slow convergence speed [11]. One important reason causing the aforementioned drawbacks is that the CSO algorithm cannot adaptively adjust the parameters according to the current optimization results. Generally speaking, it is almost impossible to precisely adjust parameters of the SIOAs following different population situations, but we can observe or detect a rough trend of how the parameters should be adjusted in different situations. For example, the more the times of iteration are, the smaller the parameters should become, and vice versa. Hence, it would be interesting if we can use relative fuzzy concepts such as “faster” iterations or “smaller” parameters to describe the parameter adjustment process of the CSO algorithm. It is known to us that the fuzzy system [18] is a powerful tool to imitate the thinking of the human brain to recognize and judge the fuzzy phenomena. In consequence, we propose a fuzzy chicken swarm optimization (FCSO) algorithm to incorporate the fuzzy system mechanism into the algorithm which is able to adaptively adjust the parameters of CSO and thus to overcome the problems of local optimum and slow convergence. Our contributions discussed in this paper are briefly summarized as follows:

- (1) To our knowledge, the fuzzy system has been introduced to and applied in the CSO algorithm for the first time, which dynamically adjusts the number of chickens and random factors of CSO according to optimization speed, chicken aggregation, and iteration times, in order to adaptively balance the exploitation and exploration abilities
- (2) We improve the position update strategy of roosters to overcome the randomness of the original, traditional strategy and make the local exploration more accurate with the increasing number of iteration times
- (3) Through modifying the design of the defuzzifier, we enhance the defuzzification method of the fuzzy system incorporated in the FCSO algorithm to make the algorithm more flexible
- (4) By evaluation of the key evaluation criteria, convergence speed, and optimization accuracy, the proposed algorithm outperforms the eight other state-of-the-art, prominent SIOAs in the high- and low-dimensional spaces, which provides a comprehensive and in-depth analysis of these prominent algorithms

The rest of this paper is organized as follows. The basic CSO algorithm is described in Section 2. In Section 3, we discuss the principle of the proposed FCSO algorithm, and the experiment process, the test data used, and evaluation criteria are discussed in Section 4. We also present comprehensive experimental results and their analysis. In Section 5, Friedman and Nemenyi tests are performed to validate the significance of the proposed method against other counterparts. Four classical engineering design problems are discussed in Section 6, which shows the optimization ability of the FCSO algorithm in practical engineering problems. We conclude the paper in Section 7, indicating our contributions to this research area and our future work in this direction.

2. Basic CSO Algorithm

In the basic CSO algorithm, there are three kinds of roles, roosters, hens, and chicks, each having different behaviour specifications. In the following, we give basic assumptions for the CSO algorithm:

- (1) The CSO algorithm divides a chicken swarm into a few groups, each of which has one rooster, several hens, and a small number of chicks.
- (2) The identities of roosters, hens, and chicks are determined by their fitness values, the best ones are selected as roosters, the worst ones are the chicks, and other individuals are the hens. Each hen randomly chooses one rooster as her mate and becomes a member of his group, and each chick also randomly selects one hen as its mother.
- (3) In the whole population, the individual identities, the spouse relationships, and the mother-children relationships remain unchanged for G generations (G is the iterative cycle), and the identities, the spouse relationships, and the mother-children relationships will be updated after G generations.
- (4) In each group of the whole population, hens follow their spouse rooster to find foods, and they will randomly compete for foods with other individuals within a group. The individuals with better fitness values are more likely to obtain foods.

Each chicken is described by its position. Let RN, HN, CN, and MN represent the number of roosters, hens, chicks, and mother hens, respectively, and $x_{i,j}^t$ is the position of the

i^{th} chicken in the j^{th} dimensional space on the t^{th} iteration, where $i \in \{1, \dots, N\}$, $j \in \{1, \dots, D\}$, and $t \in \{1, \dots, T\}$ and N , D , and T represent the total number of chickens, the dimension number, and the maximum iteration times, respectively. A rooster, a hen, and a chick have their specific position update formulas.

For a rooster, its recurrent position is defined as follows:

$$x_{i,j}^{t+1} = x_{i,j}^t * (1 + \text{Randn}(0, \sigma^2)), \quad (1)$$

$$\sigma^2 = \begin{cases} 1, & \text{if } f_i \leq f_k, \\ \exp\left(\frac{f_k - f_i}{|f_i| + \varepsilon}\right), & \text{otherwise } k \in [1, \text{RN}], \quad k \neq i. \end{cases} \quad (2)$$

Here, $\text{Randn}(0, \sigma^2)$ is a random number following Gaussian distribution with an expectation of zero and variance of σ^2 , ε is a very small constant, k is the number of another rooster which is chosen randomly, and f_i and f_k are the fitness values of the i^{th} and the k^{th} roosters, respectively.

The recurrent position of a hen is defined as follows:

$$x_{i,j}^{t+1} = x_{i,j}^t + C_1 * \text{Rand} * (x_{r_1,j}^t - x_{i,j}^t) + C_2 * \text{Rand} * (x_{r_2,j}^t - x_{i,j}^t), \quad (3)$$

$$C_1 = \exp\left(\frac{(f_i - f_{r_1})}{(\text{abs}(f_i) + \varepsilon)}\right), \quad (4)$$

$$C_2 = \exp(f_{r_2} - f_i). \quad (5)$$

Here, C_1 and C_2 are the learning factors, Rand is a random number following uniform distribution in the scope of $[0, 1]$, r_1 is the index of the rooster that is the spouse of the i^{th} hen, r_2 is the number of a rooster or a hen which is selected randomly, and $r_1 \neq r_2$.

The recurrent position of a chick is defined as follows:

$$x_{i,j}^{t+1} = x_{i,j}^t + \text{FL} * (x_{m,j}^t - x_{i,j}^t), \quad (6)$$

where $x_{m,j}^t$ represents the mother hen of chicks and FL is a random factor in the scope of $[0, 2]$. So, the basic CSO algorithm is shown in Algorithm 1.

3. The Principle of the FCSO Algorithm

A major principle of applying the fuzzy method is to apply the fuzzy concepts to the adjustment of the parameters of the CSO algorithm, which aims to intuitively reflect the human knowledge about the manipulation of the parameters to achieve a better convergence and accuracy of the algorithm. In other words, first, we fuzzify the parameter values to allow a knowledge-based judgement of how to modify the parameters. Then, we use the knowledge library of the parameter scales to determine which values to choose and execute the adjustment of the parameters. Finally, we defuzzify the fuzzy values for the parameters and continue

running the algorithm. The proposed FCSO algorithm adopts the fuzzy system to adaptively adjust parameters of CSO under different population situations. The parameters include random factors Rand in formula (3), FL in formula (6), and the number of chickens of the whole population N . This dynamic mechanism, applying the fuzzy method to adaptively adjust CSO's parameters in an iterative process, aims to overcome the defects (discussed in Section 1) of CSO to some extent. The key concepts that describe the proposed method and algorithm are presented in this section. After providing an overview of the framework, including the fuzzy system, of the algorithm in Section 3.1, we define and discuss in detail the key input variables for the fuzzy system in Section 3.2. In Sections 3.3 and 3.4, we describe two key mechanisms of the fuzzy system: one is the fuzzifiers and defuzzifiers, and the other is the rule base and inference engine. Finally, we present the proposed FCSO algorithm in Section 3.5.

3.1. Overall Description of the FCSO Algorithm. The FCSO algorithm adopts the fuzzy system to adjust the number of chickens and random factors adaptively, and the adopted fuzzy system consists of four components [18], including a fuzzifier, a defuzzifier, a fuzzy rule base, and a fuzzy inference engine. The input variables (discussed in Section 3.2) to the fuzzy system are optimization speed, iteration times, and chicken aggregation, which are fuzzified with the Gaussian fuzzifier. The fuzzy rule base is composed of fuzzy IF-THEN rules, and the fuzzy inference engine (FIE) completes reasoning from input variables to output variables according to the fuzzy IF-THEN rules. The input variables should be fuzzified by the fuzzifier before reasoning, and the output variables need to be defuzzified by the defuzzifier after reasoning. The input variables (discussed further in Section 3.2) are extracted to monitor the running status of the proposed algorithm, including optimization speed, the aggregation of chickens, and iteration times. The output variables (discussed in Section 3.3) are the random factors Rand and FL and the number of chickens N . These output variables are transmitted to CSO algorithm to control its running, and the updated monitoring indicators (or input variables) are input into the fuzzy system iteratively to adaptively adjust the CSO parameters in order to overcome the shortcomings of CSO algorithm; the framework of FCSO is described in Figure 1. The $\diamond$ symbol in Figure 1 indicates that the input variables nv and nt can derive the output variable Rand and FL through the fuzzy system, and the $\circ$ symbol indicates that nv and $aggr$ can adjust the output variable N ; the details are described in Section 3.4.

3.2. Input Variables of the Fuzzy System. According to the principle of the CSO algorithm, random factors Rand and FL and the number of chickens N are important factors which can influence CSO's global searching ability and convergence speed. In order to adjust these parameters, some monitoring indicators of CSO algorithm need be extracted as the input variables of the fuzzy system. Generally speaking, if the fitness values (suppose minimum

Input: parameters N, T, RN, HN, CN, MN, G

Output: the best optimized solution

Steps:

- (1) Initialize positions for chickens randomly.
- (2) Compute the fitness value for each chicken, select the global best position of the population and the local best position of every chicken, and initialize iteration times $t = 1$.
- (3) If $t \% G$ equals zero ($\%$ is the remainder operator), sorting all the chickens in the descending order of fitness values, the best RN individuals are chosen as roosters, the worst CN individuals are chicks, and others are hens. Divide the population into several groups, each of which includes one rooster, several hens, and chicks that randomly select their spouse and mothers, respectively.
- (4) Update the positions for roosters, hens, and chicks according to formulas (1), (3), and (6), respectively, and compute their fitness values.
- (5) Update the global best position of the population and the local best position of each individual.
- (6) Iteration times $t = t + 1$; if t equals M or the solution satisfies accuracy requirement, CSO outputs the final result; else, go to Step 3.

ALGORITHM 1: The basic CSO algorithm.

optimization problem) of all the chickens are optimized too fast, it indicates that the whole population may fall into local optimum, so optimization speed is defined as one indicator, described in the following equations:

$$v^G = \frac{1}{G} \sum_{l=1}^G \frac{1}{N} \sum_{j=1}^N \frac{f_j^{l-1} - f_j^l}{|f_j^{l-1}| + \varepsilon}, \quad (7)$$

$$nv = \frac{1}{1 + e^{-(v^G + 10)}}, \quad (8)$$

where G is the iterative cycle (see basic assumption (3) of CSO in Section 2), N is the total number of chickens, v^G represents the average value of optimization speed in one iterative period G , f_j^{l-1} and f_j^l are the fitness values of the j^{th} chicken in the $(l-1)^{\text{th}}$ and l^{th} iterations among G , respectively, and nv is the normalization of v^G by sigmoid function. The purpose of adding a constant 10 to the sigmoid function [19] is to make nv evenly distributed in $[0, 1]$. So, too large optimization speed may make CSO fall into the local optimal solution, and too small speed means that CSO converges slowly. Another monitoring indicator is the aggregation of chickens, which is defined in equation (11):

$$\mu = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i, \quad (9)$$

$$\text{aggr} = \frac{1}{N} \sum_{i=1}^N \sqrt{(x_{i,1} - \mu_1)^2 + \dots + (x_{i,D} - \mu_D)^2}, \quad (10)$$

$$\text{naggr} = \begin{cases} \frac{\text{aggr}}{\text{AGGR}}, & \text{aggr} < \text{AGGR}, \\ 1, & \text{aggr} \geq \text{AGGR}, \end{cases} \quad (11)$$

where $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D})$ is the i^{th} chicken's position, $\mu = (\mu_1, \mu_2, \dots, \mu_D)$ is the centroid of positions of N chickens, aggr represents the average distance between $\mathbf{x}_i$ and μ , and AGGR is the initial average distance between $\mathbf{x}_i$ and μ in the initialization step. Generally speaking, the initial positions should cover the whole solution space as much as

possible, so AGGR should be larger than aggr , and naggr is the normalization of aggr , as described in equation (11). If aggr is too small, it indicates that the chicken swarm is too concentrated, which would weaken the global searching ability of the CSO algorithm. On the contrary, if aggr is too large, it means that the chicken population is too scattered, which would lead to slow convergence speed. The third indicator is the recurrent iteration times t . If t is large, it means chickens' positions need to be updated exquisitely at the late stage of the CSO algorithm. The normalization of t , nt , is defined as

$$nt = \frac{t}{T}. \quad (12)$$

3.3. Fuzzifiers and Defuzzifiers of the Fuzzy System. In this paper, all the three input variables are fuzzified by the Gaussian fuzzifier into three fuzzy sets, respectively, including high, medium, and low. For optimization speeds, the membership functions of low, medium, and high are defined as follows:

$$\mu_l(nv) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(nv)^2}{2\sigma^2}\right), & nv \leq p1, \\ 0, & p1 < nv \leq 1, \end{cases} \quad (13)$$

$$\mu_m(nv) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(nv - 0.5)^2}{2\sigma^2}\right), & p1 < nv \leq p2, \\ 0, & \text{otherwise}, \end{cases} \quad (14)$$

$$\mu_h(nv) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(nv - 1)^2}{2\sigma^2}\right), & nv > p2, \\ 0, & \text{otherwise}. \end{cases} \quad (15)$$

Here, σ^2 is the variance, and in this work, $\sigma = 0.04$. The parameters $p1$ and $p2$ are two intersecting points of the three Gaussian curves with μ being 0, 0.5, and 1, respectively, as

FIGURE 1: The framework of the FCSO algorithm.

described in Figure 2. The two parameters also apply to equations (16) to (21).

Similarly, the membership functions of low, medium, and high for the aggregation of chickens are defined as follows:

$$\mu_l(\text{naggr}) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(\text{naggr})^2}{2\sigma^2}\right), & \text{naggr} \leq p1, \\ 0, & p1 < \text{naggr} \leq 1, \end{cases} \quad (16)$$

$$\mu_m(\text{naggr}) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(\text{naggr} - 0.5)^2}{2\sigma^2}\right), & p1 < \text{naggr} \leq p2, \\ 0, & \text{otherwise}, \end{cases} \quad (17)$$

$$\mu_h(\text{naggr}) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(\text{naggr} - 1)^2}{2\sigma^2}\right), & \text{naggr} > p2, \\ 0, & \text{otherwise}. \end{cases} \quad (18)$$

Following are the membership functions of iteration times nt :

$$\mu_l(nt) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(nt)^2}{2\sigma^2}\right), & nt \leq p1, \\ 0, & p1 < nt \leq 1, \end{cases} \quad (19)$$

$$\mu_m(nt) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(nt - 0.5)^2}{2\sigma^2}\right), & p1 < nt \leq p2, \\ 0, & \text{otherwise}, \end{cases} \quad (20)$$

FIGURE 2: The membership functions of optimization speed nv .

$$\mu_h(nt) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(nt - 1)^2}{2\sigma^2}\right), & nt > p2, \\ 0, & \text{otherwise}. \end{cases} \quad (21)$$

For the output variables of the fuzzy system, in order to expand the position updating span, the random factors Rand and FL have the same value range [0, 3] in the FCSO algorithm, and the number of chickens N can select 0.7~2 times the initial number of chickens. For the aforementioned three output variables, too few fuzzy sets affect the accuracy of optimization, but too many sets would lead to complex rule mapping that would affect algorithm efficiency. We quantified them into 3, 5, and 9 fuzzy sets, respectively, and the experimented results showed that the effect of taking five fuzzy sets was a suitable solution, which include very low, low, medium, high, and very high. Rand is fuzzified into five fuzzy sets according to the following Gaussian membership functions, FL adopts the similar membership functions as Rand, and no more explanation in this article.

$$\mu_{vl}(\text{rand}) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{\text{rand}^2}{2\sigma^2}\right) & \text{rand} \leq p1, \\ 0 & \text{otherwise,} \end{cases} \quad (22)$$

$$\mu_l(\text{rand}) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(\text{rand} - 0.25)^2}{2\sigma^2}\right) & p1 < \text{rand} \leq p2, \\ 0 & \text{otherwise,} \end{cases} \quad (23)$$

$$\mu_m(\text{rand}) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(\text{rand} - 0.5)^2}{2\sigma^2}\right) & p2 < \text{rand} \leq p3, \\ 0 & \text{otherwise,} \end{cases} \quad (24)$$

$$\mu_h(\text{rand}) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(\text{rand} - 0.75)^2}{2\sigma^2}\right) & p3 < \text{rand} \leq p4, \\ 0 & \text{otherwise,} \end{cases} \quad (25)$$

$$\mu_{vh}(\text{rand}) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(\text{rand} - 1)^2}{2\sigma^2}\right) & \text{rand} \geq p4, \\ 0 & \text{otherwise.} \end{cases} \quad (26)$$

Here, σ^2 is the variance of rand with the value $\sigma = 0.04$, and $p1$, $p2$, $p3$, and $p4$ are the intersections among five Gaussian curves when μ is 0, 0.25, 0.5, 0.75, and 1, respectively, as described in Figure 3.

In this work, we fuzzified the times of the initial number of chickens into five fuzzy sets according to the following membership functions. The current number of chickens can be calculated through multiplying defuzzified n by the initial number of chickens.

$$\mu_{vl}(n) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{n^2}{2\sigma^2}\right) & n \leq p1, \\ 0 & \text{otherwise,} \end{cases} \quad (27)$$

$$\mu_l(n) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(n - 0.25)^2}{2\sigma^2}\right) & p1 < n \leq p2, \\ 0 & \text{otherwise,} \end{cases} \quad (28)$$

$$\mu_m(n) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(n - 0.5)^2}{2\sigma^2}\right) & p2 < n \leq p3, \\ 0 & \text{otherwise,} \end{cases} \quad (29)$$

FIGURE 3: Membership function of random factor Rand.

$$\mu_h(n) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(n - 0.75)^2}{2\sigma^2}\right) & p3 < n \leq p4, \\ 0 & \text{otherwise,} \end{cases} \quad (30)$$

$$\mu_{vh}(n) = \begin{cases} \frac{1}{\sqrt{2\pi} * \sigma} \exp\left(-\frac{(n - 1)^2}{2\sigma^2}\right) & n \geq p4, \\ 0 & \text{otherwise,} \end{cases} \quad (31)$$

where σ^2 is the variance of the chicken number and the parameters σ , μ , $p1$, $p2$, $p3$, and $p4$ have the same settings as Rand, as described in Figure 4.

For the defuzzification of output variables Rand, FL, and N, this work adopts the center average defuzzifier (CADefz) [18], which is formally defined as follows:

$$\text{CADefz}(c, h) = \frac{\sum_{l=1}^M c_l * h_l}{\sum_{l=1}^M h_l}, \quad (32)$$

where M represents the number of fuzzy sets, $c_l \in c = \{c_1, c_2, \dots, c_M\}$, and $h_l \in h = \{h_1, h_2, \dots, h_M\}$, the center and the height of the l^{th} fuzzy set, respectively. We use the following equation (33) to identify the center of the fuzzy set in the standard CADefz:

$$\begin{cases} c = \bigcup_{i=1}^M \{c_i = \text{random}(i_\alpha)\}, \\ i_\alpha = \{ov | \mu_i(ov) \geq \alpha, ov \in \text{FIEout}_\alpha\}, \\ \text{FIEout}_\alpha = \{v | \mu_{\text{FIEout}}(v) \geq \alpha\}. \end{cases} \quad (33)$$

Here, $\mu_{\text{FIEout}}(v)$ is the membership function derived from the output of the fuzzy inference engine (FIE). The value of c_i is randomly selected from the set i_α , and we use the following equation (34) to identify h :

FIGURE 4: Membership of the number of chickens N .

$$h = \bigcup_{i=1}^M \{h_i = \max(\mu_i(ov_i)), ov_i \in i_\alpha\}. \quad (34)$$

Here, h_i is defined as the maximum value of $\mu_i(ov_i)$ where ov_i is within the scope of i_α . In this work, the threshold value α in equation (33) equals 0.5 for Rand and FL, and for the number of chickens N , it is set to 0.7, approximate to the membership values of $p1p2$, $p3$, and $p4$.

3.4. Fuzzy Rule Base and Fuzzy Inference Engine. There are two functions that the fuzzy system performs: the first (as discussed in Section 3.4.1) to use the optimization speed nv and iteration times nt to deduce random factors Rand and FL and the second (as discussed in Section 3.4.2) to use $naggr$ and nt to derive the number of chickens N .

3.4.1. Fuzzy System for Adjusting Random Factors. The random factors Rand and FL in the basic CSO algorithm are completely random. On the one hand, at the early stage of running the CSO algorithm, too small random factors indicate that each moving step of the position update (see equations (3) and (6)) is very tiny, which makes it easy for the algorithm to fall into the local optimum and damage the capability of the algorithm on global exploration. On the other hand, at the later stage, improper random factors (for example, large random factors) will lead to too large moving steps, which will hinder the local refinement and affect the convergence speed. Thus, we attempt to balance the global and local searching abilities through adaptively adjusting random factors, which change according to judging optimization speed nv and current iteration times nt , as described in Figure 1.

When the FCSO algorithm runs on the early stage of iteration (small iteration times), it should have a wide searching scope in order to find the global optimum as much as possible. Thus, according to equations (3) and (6), the

random factors should have a large range, which map to the fuzzy sets of very high, high, and medium correspondingly. On the contrary, when the FCSO algorithm runs on the later stage of iteration (large iteration times), it should focus on local refined exploration, so the scope of random factors should concentrate on a small range, which map to the fuzzy sets of very low, low, and medium. Similarly, according to equation (7), low optimization speed means the slow convergence, which generally indicates that the FCSO algorithm falls into the local optimal solution, enlarging the random factors in equations (3) and (6) can help FCSO jump to the local optimum, so Rand and FL map to the fuzzy sets of very high, high, and medium. On the contrary, high optimization speed means the rapid convergence, and narrowing random factors can help FCSO to exquisitely explore the global optimum, so they map to the fuzzy sets of very low, low, and medium. When iteration times and optimization speed are medium, respectively, random factors map to the fuzzy sets of low, medium, and high. According to the above discussion, we derive the following mapping rules in Table 1.

The fuzzy rule base is composed of fuzzy IF-THEN rules, each of which is formulated as IF $\langle FP1 \rangle$ THEN $\langle FP2 \rangle$, where $FP1$ and $FP2$ are fuzzy propositions. The rule base of adjusting random factors contains a set of rules as in the following equation:

$$R_{rf}^{(r)}: \text{IF } \langle FP_1^r \rangle \text{ THEN } \langle FP_2^r \rangle, \quad r = 1, 2, \dots, 9. \quad (35)$$

Here, $R_{rf}^{(r)}$ represents the whole rule set, and R_{rf}^r is the r^{th} rule, $r = 1, 2, \dots, 9$. For example, for the random factor Rand, the first rule R_{rf}^1 can be as follows: IF $\langle nt$ is high and nv is high $\rangle$ THEN $\langle \text{Rand is very low} \rangle$, as in Table 1.

3.4.2. Fuzzy System for Adjusting the Number of Chickens. The number of chickens in the basic CSO algorithm is changeless, while in this work, we adopt the fuzzy system to dynamically change the number of chickens according to optimization speed and chicken aggregation that are defined in Section 3.2. Low optimization speed means slow convergence. In order not to stuck in a local optimum, we should lift the population diversity through increasing the chicken number nt , so we map the chicken number to three subsets of the fuzzy set: very high, high, and medium. On the contrary, high optimization speed indicates rapid convergence. Reducing the chicken number would alleviate this trend, so it maps to the fuzzy sets of very low, low, and medium for the number of chickens. For the chicken aggregation, high aggregation means that individuals are too concentrated, which is easy to lead to the local optimal solution. It is necessary to increase the number of chickens to maintain the individual diversity, so when chicken density is high, the number of chickens maps to the fuzzy sets of very high, high, and medium. On the contrary, low aggregation indicates that we can decrease the number of chickens to enhance the local exploration, so the number of chickens maps to the fuzzy sets of very low, low, and medium. A medium optimization speed or aggregation maps to the fuzzy sets of the number of chickens, including high, medium, and low, respectively.

TABLE 1: Mappings of fuzzy rule base for adjusting random factors.

Iteration times nt	Optimization speed nv	Random factors Rand and FL
High	High	Very low
High	Medium	Low
Medium	High	Low
High	Low	Medium
Medium	Medium	Medium
Low	High	Medium
Medium	Low	High
Low	Medium	High
Low	Low	Very high

According to the above discussion, the fuzzy rule base of adjusting the number of chickens is defined as follows:

$$R_n^{(r)}: \text{IF } <FP_1^r > \text{ THEN } <FP_2^r >, \quad r = 1, 2, \dots, 9. \quad (36)$$

$R_n^{(r)}$ represents the whole rule set, and R_n^r is the r^{th} rule, $r = 1, 2, \dots, 9$. For example, R_n^1 can be defined as follows: IF $<nv$ is high and $naggr$ is high > THEN $<N$ is medium>, as described in Table 2. In the fuzzy inference engines FIE_{rf} and FIE_n , the fuzzy logic principle is used to combine the fuzzy IF-THEN rules in $R_{rf}^{(r)}$ or $R_n^{(r)}$ into a mapping from the input variables to the output variables. In the FCSO algorithm, the type of fuzzy inference engine is Mamdani [18], one of the most commonly used inference engine, which defines the fuzzy implication relation by calculating the Cartesian product (taking the smallest value) of two fuzzy sets.

3.5. The FCSO Algorithm. Roosters play an important role in the CSO algorithm that will affect the optimization results of hens and chicks. In the basic CSO algorithm, the position update of a rooster is of great randomness, which has a strong influence on the convergence speed of the CSO algorithm. CSO is easy to fall into a local optimum in the late iterations [15]. In this paper, we adopt the cosine function to adjust a rooster position, and the periodic variation of the cosine function will affect the updating step length of a rooster's position in the later iteration stage, which can help CSO jump out of the local optimal solutions. The modified method is defined as follows:

$$x_{i,j}^{t+1} = |\cos(t)| * x_{i,j}^t * (1 + \text{Randn}(0, \sigma^2)). \quad (37)$$

The update range of a rooster position will be narrowed with the increase of iterations according to the trend of the cosine function, which is conducive to the convergence of the CSO algorithm in the later running stage. The FCSO algorithm is given in Algorithm 2.

4. Experimental Results and Analysis

In this section, we first introduce the experimental environment and then make a comprehensive comparison on optimization accuracy and convergence speed. We consider a comprehensive comparison of the proposed FCSO with other eight related algorithms along two directions: fundamental biointelligence algorithms and biointelligent

algorithms equipped with fuzzy systems. One direction is to compare the FCSO with four fundamental biointelligence algorithms, including the basic CSO algorithm [4], genetic algorithm (GA) [1], particle swarm optimization (PSO) algorithm [2], and artificial fish school (AFS) algorithm [3]. The other direction is to compare the FCSO with four biointelligence algorithms which are equipped with the fuzzy logic mechanism, including the fuzzy GA (FGA) algorithm [20], fuzzy artificial fish school (FAF) algorithm [21], fuzzy particle swarm optimization (FPSO) algorithm [22], and the improved CSO (ICSO) algorithm [10].

4.1. Description of the Experimental Environment. For the fairness of comparison, each algorithm will run 10 times independently, and the number of particles of each algorithm is set to 50, and iteration times are set to 500 for 10-dimensional space and 1000 for 100-dimensional space, respectively. The parameters of the compared algorithms are consistent with the corresponding original work [1–4, 10, 19–21] in order to ensure the best performance of each algorithm; they are described in Table 3. In the experiments, the algorithms are compared from two qualities: accuracy and convergence speed. In terms of accuracy, we compute the fitness values of minimum (MIN), maximum (MAX), mean (MEAN), and standard deviation (STD) on 10 times independent experiments owing to the randomness of single experiment's result. For convergence speed, we also count the average value of the best fitness under the corresponding iterative number in 10 independent experiments. All the 9 compared algorithms are implemented by Python programming language, and the software and hardware environments are described as follows: CPU is Intel (R) Core (TM) i7-4800MQ @2.70 GHz, memory is 16.00 GB, the operating system is Windows 10 64 bit, the development platform is PyCharm, and the interpreter version of Python is Python3.7.

4.2. Description of Benchmark Functions. In this work, 30 black-box optimization benchmarking (BBOB) functions are adopted as fitness functions which are proposed in the 2017 IEEE Congress on Evolutionary Computation (CEC 2017) [23]. The optimum values of the 30 BBOB functions (F0~F29) are from 100 to 3000 with the step of 100, and they include unimodal functions, multimodal functions, hybrid functions, and composition functions. These BBOB functions are adopted to compare the accuracy and convergence

TABLE 2: Mappings of fuzzy rule base for adjusting the number of chickens.

Optimization speed nv	Chicken aggregation $naggr$	Number of chickens N
High	Low	Very low
Medium	Low	Low
High	Medium	Low
High	High	Medium
Low	Low	Medium
Medium	Medium	Medium
Medium	High	High
Low	Medium	High
Low	High	Very high

Input: parameters N, T, RN, HN, CN, MN, G , and α

Output: the best optimized solution

Steps:

- (1) Initialize positions for chickens randomly.
- (2) Compute the fitness value for each chicken, select the global best position of the population and the local best position of every chicken, and initialize iteration times $t = 1$.
- (3) If $t \% G$ equals zero ($\%$ is the remainder operator), run the fuzzy systems to adjust random factors Rand, FL, and the number of chickens N .
 - (1) Compute nv , nt , and $naggr$ according to equations (8), (11), and (12)
 - (2) Operate fuzzification on nv , nt , $naggr$, Rand, FL, and N according to equations (13)~(31)
 - (3) Operate fuzzy inference according to fuzzy rule bases $R_{rf}^{(r)}$ in equation (35) and $R_n^{(r)}$ in equation (36)
 - (4) Operate defuzzification on Rand, FL, and N according to equation (32); the results are regarded as the new parameter in the next generation;
- Sorting all the chickens in the descending order of fitness values, the best RN individuals are chosen as roosters, the worst CN individuals are chicks, and others are hens. Divide the population into several groups, each of which includes one rooster, several hens, and chicks that randomly select their spouse and mothers, respectively.
- (4) Update the positions for roosters, hens, and chicks according to formulas (1), (3), and (6) respectively, and compute their fitness values.
- (5) Update the global best position of the population and the local best position of each individual.
- (6) The iteration times $t = t + 1$; if t equals T or the solution satisfies accuracy requirement, CSO outputs the final result; else, go to Step 3.

ALGORITHM 2: The steps of the FCSO algorithm.

TABLE 3: Parameters of compared algorithms.

Algorithms	Parameters
CSO	$RN = N * 0.2, HN = N * 0.6, CN = N - RN - HN, MN = HN * 0.1, G = 10, FL \in (0.4, 1)$
FCSO	$RN = N * 0.1, HN = N * 0.4, CN = N - RN - HN, MN = HN * 0.6, G = 3, \alpha = 0.5$ for Rand and FL, $\alpha = 0.7$ for N
FGA	$P_c = 0.8, P_m = 0.1, Pop = 100, Maxgen = 3000$
FPSO	$C_1 = 2, C_2 = 2$
FAFS	Visual = 2.5, try_number = 20, step = 0.3
AFS	Visual = 2.5, try_number = 20, step = 0.3, $\lambda = 0.618$
GA	$\theta_1 = 1, \theta_2 = 0.8, \theta_3 = 0.2$
PSO	$C_1 = 2, C_2 = 2$
ICSO	$RN = N * 0.2, HN = N * 0.6, CN = N - RN - HN, MN = HN * 0.1, G = 10, FL \in (0.4, 1), C = 0.4$

between FCSO and the other eight SIOAs. The compared algorithms can be divided into two groups: the fuzzy-based SIO algorithms, including FGA, FPSO, and FAFS, and the standard or other method-based SIOAs, including CSO, GA, PSO, AFS, and ICSO. Generally speaking, the higher the dimension, the harder to find global optimum and the slower the convergent speed. In literature [11], 30 dimensions are considered to be high dimensional. However, in order to evaluate the convergence speed and optimization accuracy, in FCSO, we take 100 and 10 as the high and low dimensions for the 30 BBOB functions, respectively.

4.3. Analysis of Experimental Results. As briefed in the beginning of this section, we compare the proposed FCSO with other eight SIOAs (with two directions) on the 30 BBOB functions for the low- and high-dimensional spaces, respectively, and get the comparison results of accuracy and convergence speed. Supplementary Materials A and B give the detailed experimental results of 9 algorithms' accuracy and convergence speed. Specifically, Tables A1 and A2 in Supplementary Material A give the accuracy comparison on 10- and 100-dimensional spaces, respectively; the bold font represents the optimal values of 9 algorithms. In

Supplementary Material B, Figures S1~F30 and F31~F60 show the comparison results of convergence speed on low- and high-dimensional spaces, respectively.

4.3.1. Results' Comparison on Accuracy and Stability. The results in Tables A1 and A2 of Supplementary Material A are concluded in Table 4; the bold numbers represent the winners of SIOAs according to the specific criterion, and the corresponding winning functions are shown in the following brackets. As described in Table 4, for the 10-dimensional space, FCSO achieves remarkably better results than the other eight SIOAs. Specifically, FCSO obtained 15, 17, 14, and 9 best results on the MAX, MIN, MEAN, and STD criteria, respectively. The second best algorithm is FAFS, and the corresponding results are 7, 6, 8, and 4, respectively; those values of FGA are 5, 4, 4, and 1, respectively, and the other six algorithms have no advantages in the 10-dimensional space. In the 100-dimensional space, FCSO performs better than other eight algorithms. It has obtained 12, 11, 12, and 10 best results on the MAX, MIN, MEAN, and STD criteria, respectively. The corresponding results of FGA are 7, 4, 6, and 5, respectively, and the best results of FAFS are 4, 8, and 5 on the MAX, MIN, and MEAN criteria, respectively; other six algorithms still have no advantages on the 100-dimensional space. In addition, it seems that fuzzy-based SIOAs obtained better results than other compared SIOAs, which indicated that the mechanism of fuzzy control can adaptively adjust parameters for complex optimization functions.

In detail, in the 10-dimensional space, FCSO simultaneously obtained the best results on 12 functions (F1, F4, F6, F7, F13, F14, F18, F20, F22, F23, F24, and F26) on the MAX, MIN, and MEAN criteria. For the second best algorithm FAFS, it only obtained the corresponding best results on 4 functions (F0, F12, F17, and F29), and those of FGA are 3 functions (F3, F8, and F19). The above results indicate that FCSO has the best accuracy compared with other eight algorithms in the low-dimensional space. In addition, in the 10-dimensional space, FCSO and CSO have obvious advantages on the stability than other seven algorithms because they obtained 9 and 6 best results on the STD criterion, respectively. In the 100-dimensional space, FCSO simultaneously obtained the best results on 11 functions (F2, F5, F6, F7, F8, F13, F15, F19, F20, F24, and F27) on the MAX, MIN, and MEAN criteria, the second best algorithm is FAFS, and it only obtained best results on 3 functions (F0, F9, and F21) on the MAX, MIN, and MEAN criteria simultaneously. FCSO also obtained the best results of 10 functions on the STD criterion; it has obvious advantage than other eight algorithms, which indicates that, in the 100-dimensional space, FCSO has the best stability than other eight algorithms. Based on the above analysis of experimental results, we can conclude that FCSO has obvious advantages on accuracy compared with other eight state-of-the-art SIOAs.

4.3.2. Efficiency Comparison and Analysis. In terms of optimization efficiency, the convergence curves of nine SIOAs in the 10- and 100-dimensional spaces are described in

Figures S1~S30 and S31~S60 of Supplementary Material B, respectively. In the 10-dimensional space, FCSO acquired the fast optimization efficiency except functions F23 and F27, which has obvious advantages than other eight SIOAs. GA, AFS, and CSO have slow convergence speed on most functions. In the 100-dimensional space, FCSO achieved the fast optimization efficiency on all the functions, while GA, FGA, AFS, and FAFS have the slow speed on almost all the functions. In addition, the convergence speed of CSO in the high-dimensional space is more faster than that in the low-dimensional space, which indicates that CSO-based algorithms are suitable to solve high-dimensional problems. Although FGA and FAFS perform well in the 10-dimensional space, they have slow convergence speed in the 100-dimensional space. The proposed FCSO algorithm has the fast optimization efficiency in both 10- and 100-dimensional spaces on almost all the functions.

4.3.3. Empirical Analysis of Experimental Results. According to the analysis in Sections 4.3.1 and 4.3.2, we can conclude that the FCSO algorithm has obvious advantages in the criteria of accuracy, stability, and optimization efficiency compared with other eight SIOAs. For the accuracy, the basic CSO algorithm is no better than other SIOAs, while the proposed FCSO performs very well because the fuzzy system (see Section 3) has been incorporated into the CSO algorithm to dynamically tune the parameters in order to adaptively balance the exploitation and exploration abilities. Compared with the parameters of PSO, AFS, CSO, GA, and ICSO, adopting the fixed empirical constants, the accuracy of FCSO, FGA, and FAFS has obvious advantages owing to that the fuzzy system can dynamically adjust the parameters according to optimization problems. FCSO adopts the grouping and regrouping mechanisms, where roosters, hens, and chicks have different evaluation strategies, and the identities of individuals will be changed periodically, so it is easier to obtain better solutions than FGA and FAFS that have the relatively single and static evolutionary strategies. For the stability, FCSO obtained the best results on the STD criterion in both 10- and 100-dimensional spaces; part of the reason lies in intrinsic learning mechanisms of CSO (CSO obtained the best stability except FCSO), including the diverse learning strategies of chicken swarm and the grouping and regrouping mechanism. Another important reason is the mechanism of adaptive parameter tuning in FCSO that can avoid the influence of random position initialization. For the optimization efficiency, as described in Section 3.5, the cosine function has been adopted to adjust rooster positions that play an important influence on the evolutionary process of CSO. The update step length of a rooster position will be narrowed with the increase of iterations according to the trend of the cosine function, which is conducive to the convergence of the CSO algorithm in the later running stage.

4.3.4. Time Complexity Analysis for Compared Algorithms. As described in Section 2, N , D , and T represent the number of chickens, the dimension number, and the maximum iteration

TABLE 4: The number of wins and corresponding functions of each criterion for 9 SIOAs.

SIOAs		Criteria			
		MAX	MIN	MEAN	STD
FCSO	$D = 10$	15 (F1, F4, F5, F6, F7, F13, F14, F15, F16, F18, F20, F22, F23, F24, F26)	17 (F1, F2, F4, F6, F7, F10, F13, F14, F18, F20, F21, F22, F23, F24, F25, F26, F27)	14 (F1, F4, F5, F6, F7, F13, F14, F16, F18, F20, F22, F23, F24, F26)	9 (F11, F13, F14, F15, F16, F18, F19, F24, F26)
	$D = 100$	12 (F2, F4, F5, F6, F7, F8, F13, F15, F19, F20, F24, F27)	11 (F2, F5, F6, F7, F8, F13, F15, F19, F20, F24, F27)	12 (F2, F4, F5, F6, F7, F8, F13, F15, F19, F20, F24, F27)	10 (F0, F1, F2, F3, F4, F8, F11, F15, F20, F25)
FAFS	$D = 10$	7 (F0, F10, F12, F17, F21, F27, F29)	6 (F0, F9, F11, F12, F17, F29)	8 (F0, F9, F10, F12, F17, F21, F27, F29)	4 (F0, F17, F27, F29)
	$D = 100$	4 (F0, F9, F18, F21)	8 (F0, F9, F14, F17, F18, F21, F28, F29)	5 (F0, F9, F21, F23, F28)	
FGA	$D = 10$	5 (F3, F8, F11, F19, F28)	4 (F3, F8, F15, F19)	4 (F3, F8, F19, F28)	1 (F3)
	$D = 100$	7 (F12, F16, F22, F25, F26, F28, F29)	4 (F1, F3, F10, F16)	6 (F3, F16, F22, F25, F26, F29)	5 (F12, F16, F26, F28, F29)
FPSO	$D = 10$	1 (F25)		1 (F25)	1 (F25)
PSO	$D = 100$				1 (F22)
	$D = 10$		1 (F28)	1 (F15)	2 (F4, F23)
AFS	$D = 100$		1 (F22)		
	$D = 10$	1 (F9)		1 (F11)	5 (F1, F2, F10, F12, F21)
CSO	$D = 100$	2 (F11, F23)	2 (F11, F12)	3 (F11, F12, F18)	1 (F23)
	$D = 10$				6 (F5, F6, F7, F20, F22, F28)
GA	$D = 100$	1 (F10)	1 (F4)	1 (F10)	5 (F7, F9, F19, F21, F24)
	$D = 10$		2 (F5, F16)		
ICSO	$D = 100$	3 (F1, F3, F14)	3 (F23, F25, F26)	2 (F1, F14)	2 (F14, F18)
	$D = 10$	1 (F2)		1 (F2)	2 (F8, F9)
	$D = 100$	1 (F17)		1 (F17)	6 (F5, F6, F10, F13, F17, F27)

times, respectively. Suppose T_{FZ} represents the time complexity of the fuzzy system for all the compared fuzzy-based algorithms, the time complexity of each compared algorithm is concluded in Table 5. From Table 5, we can see that the time complexities of AFS and FAFS are $O(D \cdot N \cdot T \cdot \text{Try}_{\text{number}})$, the time complexities of other compared algorithms are $O(D \cdot N \cdot T)$, so FCSO is a competitive algorithm.

5. Statistical Tests for Algorithm Comparison

In this work, we consider two statistical tests, Friedman test [24] and Nemenyi test [24]. A Friedman test is constructed to analyse the performance of the compared SIOAs. Table 6 provides the Friedman statistics F_F and the corresponding critical value in terms of each evaluation criterion. As shown in Table 6, the null hypothesis (that all of the compared algorithms will perform equivalently) was clearly rejected for each evaluation criterion at a significance level of $\alpha = 0.05$ for the experimental results in both 10- and 100-dimensional spaces. Consequently, we proceed to conduct a post hoc test [24] in order to analyse the relative performance among the compared SIOAs.

The Nemenyi test [24] is used to test whether each of the SIOAs performed competitively against the other compared SIOAs in both 10- and 100-dimensional spaces, including the proposed FCSO. In the test, the performance between two SIOAs was considered to be significantly different if the

corresponding average ranks differed by at least the critical difference $CD = q_\alpha \sqrt{k(k+1)/6N}$. For the test, q_α is equal to 3.102 at the significance level $\alpha = 0.05$, and thus, CD takes the value of 2.1934 ($k = 9$ and $N = 30$). Figure 5 shows the CD diagrams for each of the four evaluation criteria about the experimental results of the 10-dimensional space. If any compared SIOA whose average rank was within one CD to that of FCSO, it would be connected to FCSO with a red line, as described in Figure 5. The algorithms that were unconnected to FCSO were considered to have a significantly different performance between them. In MAX (Figure 5(a)), for example, the average rank for FCSO was 1.9333, and the critical value would be 4.1267 by adding CD. Since PSO, GA, FPSO, CSO, FGA, AFS, and FAFS obtained 6.6667, 6.5, 6.2, 5.6, 5.0667, 4.8, and 4.1333 for their respect average rankings, they were significantly worse compared with FCSO. From Table 5, we can see that FCSO obtained the best average ranks on all four criteria and has obvious better performance than the other SIOAs.

Figure 6 shows the CD diagrams for each of the four evaluation criteria about the experimental results of the 100-dimensional space. FCSO obtained the best average rank on the MAX and STD criteria and ranked the second and third best average ranks on the MEAN and MIN criteria, respectively. Although FAFS obtained the best average ranks on the MIN and MEAN criteria, its results are not stable

TABLE 5: The time complexities of 9 compared SIO algorithms.

Algorithms	Time complexity
FCSO	$O(D \cdot N + (2D \cdot N + N) \cdot T + N \cdot T / G + T_{FZ} \cdot T / G) \approx O(D \cdot N \cdot T)$
FGA	$O(D \cdot N + (3N \cdot D + 3 \cdot N) \cdot T + T_{FZ} \cdot T) \approx O(D \cdot N \cdot T)$
FAFS	$O(D \cdot N + (2N(N-1) + 4N \cdot \text{Try}_{\text{number}} \cdot D) \cdot T + T_{FZ} \cdot N \cdot T) \approx O(D \cdot N \cdot T \cdot \text{Try}_{\text{number}})$
FPSO	$O(D \cdot N + (3D \cdot N + N) \cdot T + T_{FZ} \cdot T) \approx O(D \cdot N \cdot T)$
CSO	$O(D \cdot N + (2D \cdot N + N) \cdot T + N \cdot T / G) \approx O(D \cdot N \cdot T)$
ICSO	$O(D \cdot N + (2D \cdot N + N) \cdot T + N \cdot T / G) \approx O(D \cdot N \cdot T)$
GA	$O(D \cdot N + (3N \cdot D + 3 \cdot N) \cdot T) \approx O(D \cdot N \cdot T)$
AFS	$O(D \cdot N + (2N(N-1) + 4N \cdot \text{Try}_{\text{number}} \cdot D) \cdot T) \approx O(D \cdot N \cdot T \cdot \text{Try}_{\text{number}})$
PSO	$O(D \cdot N + (3D \cdot N + N) \cdot T) \approx O(D \cdot N \cdot T)$

TABLE 6: Summary of the Friedman statistics F_F ($k = 9$ and $N = 30$) and the critical value in terms of each evaluation criterion.

Dimensions	Criteria	F_F	Critical value ($\alpha = 0.05$)
10-dimensional space	MAX	12.3695	1.9785
	MIN	9.86242	
	MEAN	10.6978	
	STD	12.2192	
	MAX	14.0527	
100-dimensional space	MIN	18.2798	
	MEAN	16.4334	
	STD	16.9399	

k : #comparing algorithms; N : #datasets.

Critical distance = 2.1934 Average rank

(a)

Critical distance = 2.1934 Average rank

(b)

Critical distance = 2.1934 Average rank

(c)

Critical distance = 2.1934 Average rank

(d)

FIGURE 5: Comparison of FCSO (control algorithm) against other compared algorithms using the Nemenyi test for the experimental results in the 10-dimensional space. (a) MAX. (b) MIN. (c) MEAN. (d) STD.

(only ranked the fifth on the STD criterion), but FCSO is the most stable algorithm in both 10- and 100-dimensional spaces.

6. Optimization on Engineering Problems

The CSO algorithm has been widely applied in various scientific and engineering domains. In the wireless sensor network (WSN) field, Fouad et al. [25] used it to optimize the topological structure of the WSN, and Al Shayokh and Shin [26] and Yu et al. [27] applied the CSO algorithm to handle the localization problem of the WSN. In the cloud computing domain, CSO has been used to solve deadlock-free

migration of the virtual machine (VM) [28] and dynamic task scheduling problem [29]. In the military field, CSO is applied to optimize the warhead's reentry trajectory [30] and ascent trajectory of hypersonic vehicles [31]. In the data mining domain, it is used to optimize the K -means clustering algorithm [32], adaptive neurofuzzy inference system [33], and feature selection problem [34]. Still, more fields have CSO been applied to, including architecture [35], transport [36], mechanical engineering [37], environmental protection [38], power [39], and robot [40].

Many real-world engineering optimization problems (EOPs) are very complex in nature and have many constrained conditions. It is difficult for the traditional

optimization methods (TOMs) to deal with the objective functions and the constraints with multiple or sharp peaks due to their instability [1]. In contrast, SIOAs outperform TOMs in solving the EOPs. In this section, we apply the proposed FCSO algorithm to four types of constrained EOPs to further evaluate the performance of FCSO. These engineering optimization problems are often used to test the performance of SIOAs [11, 15]. In the next, we first introduce these four engineering designs and their constraints and then present the experiment results of applying the CSO and FCSO to these problems and make a brief comparison of their performance.

6.1. Description of Engineering Optimization Problems. In this section, the four types of classic engineering design problems are discussed, including tension/compression spring [1], pressure vessel design [2], three-bar truss design [3], and cantilevered beam design [4]. The description of the four engineering optimization problems is shown in Figure 7.

Problem 1. (Tension/compression spring design). The design, as shown in Figure 7(a), aims to minimize the weight of a tension/compression spring. In this design, the constraints, required in the design, include the minimum deflection, shear stress, surge frequency, and the limit of outer diameter. There are three designed variables: the average coil diameter x_1 , the wire diameter x_2 , and the number of active coils x_3 , which together define the following constraints (named as F30):

$$\begin{aligned} \min F_{16}(X) &= (x_3 + 2)x_2x_1^2, \\ \text{s.t. } g_1(X) &= 1 - \frac{x_2^3x_3}{71785x_1^3} \leq 0, \\ g_2(X) &= \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0, \\ g_3(X) &= 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0, \\ g_4(X) &= \frac{x_1 + x_2}{1.5} - 1 \leq 0, \end{aligned} \quad (38)$$

where $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.3$, $2 \leq x_3 \leq 15$.

Problem 2. (Pressure vessel design). The aim of the design of a cylindrical vessel which is capped at both ends by hemispherical heads, as shown in Figure 7(b), is to solve a hybrid constrained optimization problem, that is, to minimize the costs of welding, materials, and forming. The four variables to describe this problem are the thickness of the cylindrical skin (T_s), the thickness of the spherical head (T_h), the inner radius (R), and the length of the cylindrical segment of the vessel (L). It is required that the thickness of the cylinder can only be taken as an integral multiple of 0.0625. This problem

(named F31) is defined as formula (39), and the above four variables are represented by x_1 , x_2 , x_3 , and x_4 , respectively.

$$\begin{aligned} \min F_{17}(X) &= 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 \\ &\quad + 19.84x_1^2x_3, \\ \text{s.t. } g_1(X) &= -x_1 + 0.0193x_3 \leq 0, \\ g_2(X) &= -x_2 + 0.00954x_3 \leq 0, \\ g_3(X) &= -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0, \\ g_4(X) &= x_4 - 240 \leq 0, \end{aligned} \quad (39)$$

where $0.0625 \leq x_1, x_2 \leq 99 \times 0.0625$, $x_3 \geq 10$, $x_4 \leq 200$.

Problem 3. (Three-bar truss design). The objective of this design is to optimize the volume of a statistically loaded three-bar truss. The three-bar truss is subjected to vertical and horizontal forces, and the volume of the three-bar truss is subjected to stress constraints. As shown in Figure 7(c), the lengths of the bars A1 (or A3) and A2 are denoted as x_1 and x_2 in the following formula (40), which is used to define the problem (named as F32):

$$\begin{aligned} \min F_{18}(X) &= (2\sqrt{2}x_1 + x_2) \times l, \\ \text{s.t. } g_1(X) &= \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0, \\ g_2(X) &= \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0, \\ g_3(X) &= \frac{1}{x_1 + \sqrt{2}x_2} P - \sigma \leq 0, \end{aligned} \quad (40)$$

where $0 \leq x_1 \leq 1$, $0 \leq x_2 \leq 1$, $l = 100$ cm,
 $P = 2$ kN/cm², and $\sigma = 2$ kN/cm².

Problem 4. (Cantilevered beam design). The design problem of the cantilevered beam is described in Figure 5(d), and its goal is to determine the best combination of five different cross-section areas to minimize the volume of the cantilever beam. Each cross section has a width and a height, and in total, there are ten variables, denoted as h_i and b_i ($i = 1, \dots, 5$), respectively, see Figure 7(d). The constraints on the design are briefly presented as follows. The maximum allowable stress at the left end of each part is $\sigma_{\max} = 14000$ N/cm², the material modulus of elasticity is $E = 200$ GPa, the length of each section l_i ($i = 1, \dots, 5$) is 100 cm, the maximum allowable deflection is $y_{\max} = 2.715$ cm, and the height-to-width ratio of each cross section is restricted to less than 20. This optimization problem (named as F33) can be defined as follows:

FIGURE 6: Comparison of FCSO (control algorithm) against other compared algorithms using the Nemenyi test for the experimental results in the 100-dimensional space.

$$\begin{aligned}
 X &= [b_1, h_1, b_2, h_2, b_3, h_3, b_4, h_4, b_5, h_5]^T = [x_1, x_2, x_3, x_4, x_5]^T, \\
 \min \quad & F_{19}(X) = 100(x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8 + x_9x_{10}), \\
 \text{s.t.} \quad & g_1(X) = 10.7143 - \frac{x_1x_2^2}{10^3} \leq 0, \\
 & g_2(X) = 8.5714 - \frac{x_3x_4^2}{10^3} \leq 0, \\
 & g_3(X) = 6.4286 - \frac{x_5x_6^2}{10^3} \leq 0, \\
 & g_4(X) = 4.2957 - \frac{x_7x_8^2}{10^3} \leq 0, \\
 & g_5(X) = 2.1429 - \frac{x_9x_{10}^2}{10^3} \leq 0, \\
 & g_6(X) = 10^4 \left(\frac{244}{x_1x_2^3} + \frac{148}{x_3x_4^3} + \frac{76}{x_5x_6^3} + \frac{28}{x_7x_8^3} + \frac{4}{x_9x_{10}^3} \right) - 10.86 \leq 0, \\
 & g_7(X) = x_2 - 20x_7 \leq 0, \\
 & g_8(X) = x_4 - 20x_3 \leq 0, \\
 & g_9(X) = x_6 - 20x_5 \leq 0, \\
 & g_{10}(X) = x_8 - 20x_7 \leq 0, \\
 & g_{11}(X) = x_{10} - 20x_9 \leq 0,
 \end{aligned} \tag{41}$$

where $1 \leq x_i \leq 5$ ($i = 1, 3, 5, 7, 9$), $30 \leq x_i \leq 65$ ($i = 2, 4, 6, 8, 10$).

FIGURE 7: Four engineering optimization problems. (a) The design of the tension/compression spring. (b) The pressure vessel design problem. (c) Three-bar truss design. (d) The cantilevered beam.

TABLE 7: Experimental results of four EOPs.

Function	Algorithms	MAX	MIN	MEAN	STD
F30	CSO	0.014145	0.013208	0.013363	0.000284
	FCSO	0.013194	0.012776	0.013007	0.000135
F31	CSO	18187.57	2367.618	7502.164	5331.018
	FCSO	12272.28	1864.725	4803.109	2945.724
F32	CSO	267.883	264.7858	266.0553	1.07483
	FCSO	267.2195	264.3374	265.2885	1.016607
F33	CSO	87581.08	76219.26	83708.51	4209.323
	FCSO	76775.52	73063.19	75500.1	1342.61

6.2. Experimental Results' Analysis of Engineering Optimization Problems. Previously, we presented the constraint settings for the problems of the four types of engineering designs, which can be seen as to seek for an optimal solution to the problems of F30, F31, F32, and F33. Because FCSO and CSO are proposed to handle unconstrained optimization problems, for the above four constrained engineering problems, after each evolution operation, the individuals in FCSO and CSO will judge whether the solutions satisfy the constraints of engineering problems. If not, they need to be re-evolved until the constraints are satisfied. The settings for the experiment environment are the same as being described in Section 4.1. In this section, we only discuss the comparison results of the performance of the algorithms of CSO and FCSO. Four criteria, including MAX, MIN, MEAN, and STD, are selected for performance comparison, and their

results are derived from 10 independent experiments, which can comprehensively reflect the optimization accuracy and stability. As shown in Table 7, the FCSO algorithm has obvious advantage over the CSO algorithm for all the four engineering problems on all the four criteria, so FCSO performs well in four engineering problems than the CSO algorithm.

7. Conclusions and Future Work

As having been already demonstrated, SIOAs possesses a great advantage in solving NP-hard problems [28]. As a newly proposed SIO algorithm, CSO has been widely used in different applications owing to its good characteristics in recent years, for example, high optimization accuracy [9]. However, generally speaking, the CSO algorithms inherently suffer from slow convergence speed causing a long execution time and easily falling into the local optimum, which make it hard to produce a general optimal solution. In this paper, we propose a fuzzy chicken swarm optimization method, FCSO, to improve the normal CSO algorithms by applying the fuzzy system method to the adjustment of the chicken number and random factors adaptively.

The major contributions of this paper to the field of swarm intelligence include the following three aspects. Firstly, the fuzzy system has been introduced to the CSO algorithm for the first time to adaptively adjust random factors and the number of chickens to address the three

monitoring indicative issues, optimization speed, chicken aggregation, and iteration times. The introduced fuzzy system makes an adaptive balance between the exploration and exploitation capabilities in the CSO algorithm to overcome the two major drawbacks. Secondly, the cosine function has been integrated into the solution to the position update of roosters that can mitigate the randomness of the original strategy and make the local exploration more accurate with the increase of iteration times. Thirdly, we design a collection of comprehensive experiments in both low- and high-dimensional spaces with the nine algorithms on 30 BBOB functions to show that the proposed FCSO has advantages in accuracy and convergence speed. Owing to the obvious advantages of FCSO, it can be applied to handle the constrained (as described in Section 6) and unconstrained optimization problems, especially for the problems requiring high stability and rapid convergence speed. Although FCSO achieved better results than other SIOAs in both 10- and 100-dimensional spaces, it performed not very well on some composition functions (for example, F26, F28, and F29) in the 100-dimensional space, which indicates that the fuzzy system needs to be further improved to adapt high-dimensional complex optimization problems.

In the future, there are efforts required to improve and extend the proposed method and algorithm. First, in this work, we only integrate one kind of cosine function into rooster position update, cosine functions with different periods and other trigonometric functions should be studied in depth in order to more clearly determine the timing and extent of adjusting step length for different optimization problems, and proper step length can enhance optimization accuracy and convergence. Second, the membership functions are constructed by experience in this work, and the influence of different membership functions on the results of FCSO needs further discussion. For different optimization problems, it is necessary to study how to choose the most suitable membership functions, which can improve the performance of the fuzzy system. Third, this work mainly focuses on adaptively adjusting parameters of CSO algorithm, the learning strategies of FCSO can be further improved by hybridizing other SIOAs, such as differential evolution (DE) algorithm [41, 42], and the proposed FCSO can be applied to more practical optimization problems [43, 44].

Data Availability

All the test data used and the generated results are included within the manuscript.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Science Foundation of China (Grant nos. 71972177 and 61363075), National High Technology Research and Development Program ("863" Program) of China (Grant no.

2012AA12A308), Yue Qi Young Scholars Program of China University of Mining and Technology, Beijing (Grant no. 800015Z1117), the Department of Science and Technology of Jiangxi Province of China (Grant nos. 20161BBG70078 and KJLD13031), and the Department of Education of Jiangxi Province of China (Grant no. GJJ180270).

Supplementary Materials

Supplementary Materials A and B give the detailed experimental results of 9 algorithms' accuracy and convergence speed. Specifically, Tables A1 and A2 in Supplementary Material A give the accuracy comparison on 10- and 100-dimensional spaces, respectively; the bold font represents the optimal values of 9 algorithms. In Supplementary Material B, Figures S1~F30 and F31~F60 show the comparison results of convergence speed on low- and high-dimensional spaces, respectively. (*Supplementary Materials*)

References

- [1] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi, "Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems," *Applied Soft Computing*, vol. 13, no. 5, pp. 2592–2612, 2013.
- [2] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Mixed variable structural optimization using firefly algorithm," *Computers & Structures*, vol. 89, no. 23, pp. 2325–2336, 2011.
- [3] W. Yi, Y. Zhou, L. Gao, X. Li, and C. Zhang, "Engineering design optimization using an improved local search based epsilon differential evolution algorithm," *Journal of Intelligent Manufacturing*, vol. 29, no. 7, pp. 1559–1580, 2018.
- [4] H.-C. Kuo and C.-H. Lin, "A directed genetic algorithm for global optimization," *Applied Mathematics and Computation*, vol. 219, no. 14, pp. 7348–7364, 2013.
- [5] S. Boyd and L. Vandenberghe, *Convex Optimization*, pp. 466–496, Cambridge University Press, Cambridge, UK, 2004.
- [6] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, USA, 1975.
- [7] K. James and E. Russell, "Particle swarm optimization," *Neural Networks*, vol. 4, pp. 1942–1948, 1995.
- [8] X. L. Li, Z. J. Shao, and J. X. Qian, "An optimization method based on autonomous animats: fish-swarm algorithm," *Systems Engineering-Theory & Practice*, vol. 22, no. 11, pp. 32–38, 2002.
- [9] X. B. Meng, Y. Liu, X. Z. Gao, and H. Z. Zhang, "A new bio-inspired algorithm: chicken swarm optimization," in *Proceedings of the 5th International Conference on Swarm Intelligence*, pp. 86–94, Hefei, China, 2014.
- [10] D. Wu, S. Xu, and F. Kong, "Convergence analysis and improvement of the chicken swarm optimization algorithm," *IEEE Access*, vol. 4, pp. 9400–9412, 2016.
- [11] C. W. Qu, S. A. Zhao, Y. M. Fu, and W. He, "Chicken swarm optimization based on elite opposition-based learning," *Mathematical Problems in Engineering*, vol. 2017, 19 pages, 2017.
- [12] B. Li, G. J. Shen, and G. Sun, "Improved chicken swarm optimization algorithm," *Journal of Jilin University*, vol. 49, no. 4, pp. 1339–1344, 2019.
- [13] S. Torabi and F. Safi-Esfahani, "A hybrid algorithm based on chicken swarm and improved raven Roosting algorithm," *Soft Computing*, vol. 23, pp. 100129–100171, 2018.

- [14] W. Fu, B. Wang, X. Li, L. Liu, and Y. Wang, "Ascent trajectory optimization for hypersonic vehicle based on improved chicken swarm optimization," *IEEE Access*, vol. 7, pp. 151836–151850, 2019.
- [15] J. Wang, Z. Cheng, O. K. Ersoy, M. Zhang, K. Sun, and Y. Bi, "Improvement and application of chicken swarm optimization for constrained optimization," *IEEE Access*, vol. 7, pp. 58053–58072, 2019.
- [16] M. Han and S. Y. Liu, "An improved binary chicken swarm optimization algorithm for solving 0-1 knapsack problem," in *Proceedings of 13th International Conference on Computational Intelligence and Security (CIS)*, pp. 207–2100, Hong Kong, China, 2017.
- [17] D. Zouache, Y. Ould Arby, F. Nouioua, and F. Ben Abdelaziz, "Multi-objective chicken swarm optimization: a novel algorithm for solving multi-objective optimization problems," *Computers & Industrial Engineering*, vol. 129, pp. 377–391, 2019.
- [18] L. X. Wang, *A Course in Fuzzy Systems and Control*, pp. 1–448, Prentice-Hall, Upper Saddle River, NJ, USA, 1996.
- [19] A. Jain, K. Nandakumar, and A. Ross, "Score normalization in multimodal biometric systems," *Pattern Recognition*, vol. 38, no. 12, pp. 2270–2285, 2005.
- [20] L. K. Lan, J. W. Zhu, W. L. Liu et al., "New method of fuzzy-based genetic algorithms," *Computer Engineering and Design*, vol. 29, no. 14, pp. 3714–3718, 2008.
- [21] D. Yazdani, A. N. Toosi, and M. R. Meybodi, "Fuzzy adaptive artificial fish swarm algorithm," in *Proceedings of 23rd Australasian Joint Conference*, pp. 334–343, Adelaide, Australia, 2010.
- [22] Y. H. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proceedings of Congress on Evolutionary Computation*, pp. 101–106, Seoul, South Korea, 2001.
- [23] N. H. Awad, M. Z. Ali, P. N. Suganthan, J. J. Liang, and B. Y. Qu, *Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*, Nanyang Technological University, Singapore.
- [24] J. Derrac, S. Garcia, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18.
- [25] M. M. Fouad, A. I. Hafez, and A. E. Hassanien, "Optimizing topologies in wireless sensor networks: a comparative analysis between the grey wolves and the chicken swarm optimization algorithms," *Computer Network*, vol. 163, pp. 1–9, 2019.
- [26] M. Al Shayokh and S. Y. Shin, "Bio inspired distributed WSN localization based on chicken swarm optimization," *Wireless Personal Communications*, vol. 97, no. 4, pp. 5691–5706, 2017.
- [27] X. Yu, L. Zhou, and X. Li, "A novel hybrid localization scheme for deep mine based on wheel graph and chicken swarm optimization," *Computer Networks*, vol. 154, pp. 73–78, 2019.
- [28] F. Tian, R. Zhang, J. Lewandowski, K.-M. Chao, L. Li, and B. Dong, "Deadlock-free migration for virtual machine consolidation using chicken swarm optimization algorithm," *Journal of Intelligent & Fuzzy Systems*, vol. 32, no. 2, pp. 1389–1400, 2017.
- [29] S. Torabi and F. Safi-Esfahani, "A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing," *Journal of Supercomputing*, vol. 74, pp. 2581–2626, 2016.
- [30] Y. Wu, B. Yan, and X. Qu, "Improved chicken swarm optimization method for reentry trajectory optimization," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–13, 2018.
- [31] Y. T. Li, Y. Wu, and X. J. Qu, "Chicken swarm-based method for ascent trajectory optimization of hypersonic vehicles," *Journal of Aerospace Engineering*, vol. 30, no. 5, pp. 1–12, 2017.
- [32] N. Irsalinda, I. T. R. Yanto, H. Chiroma, and T. Herawan, "A framework of clustering based on chicken swarm optimization," in *Proceedings of the 2nd International Conference on Soft Computing and Data Mining (SCDM)*, pp. 336–343, Bandung, Indonesia, 2016.
- [33] Roslina, M. Zarlis, and I. T. R. Yanto, "A framework of training anfis using chicken swarm optimization for solving classification problem," in *Proceedings of 1st International Conference on Informatics and Computing (ICIC)*, pp. 437–441, Mataram, Indonesia, 2016.
- [34] A. I. Hafez, H. M. Zawbaa, E. Emary, H. A. Mahmoud, and A. E. Hassanien, "An innovative approach for feature selection based on chicken swarm optimization," in *Proceedings of 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, pp. 19–24, Fukuoka, Japan, 2015.
- [35] Y. Li, S. Wang, and M. Han, "Truss structure optimization based on improved chicken swarm optimization algorithm," *Advances in Civil Engineering*, vol. 2019, pp. 1–16, 2019, Article ID 6902428.
- [36] D. He, G. Lu, and Y. Yang, "Research on optimization of train energy-saving based on improved chicken swarm optimization," *IEEE Access*, vol. 7, pp. 121675–121684, 2019.
- [37] G. Q. Fu, H. W. Gong, H. L. Gao, T. D. Gu, and Z. Q. Cao, "Integrated thermal error modeling of machine tool spindle using a chicken swarm optimization algorithm-based radial basic function neural network," *International Journal of Advanced Manufacturing Technology*, vol. 100, no. 5, pp. 2039–2055, 2019.
- [38] D. Liu, C. Liu, Q. Fu et al., "Projection pursuit evaluation model of regional surface water environment based on improved chicken swarm optimization algorithm," *Water Resources Management*, vol. 32, no. 4, pp. 1325–1342, 2018.
- [39] Z. Q. Wu, D. L. Yu, and X. H. Kang, "Application of improved chicken swarm optimization for MPPT in photo voltaic system," *Optimal Control Applications & Methods*, vol. 39, pp. 10029–10042, 2018.
- [40] Y. Mu, L. J. Zhang, X. R. Chen, and X. S. Gao, "Optimal trajectory planning for robotic manipulators using chicken swarm optimization," in *Proceedings of 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, pp. 369–373, Hangzhou, China, 2016.
- [41] W. Deng, J. J. Xu, X. Z. Gao, and H. M. Zhao, "An enhanced MSIQDE algorithm with novel multiple strategies for global optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics: System*, pp. 1–10, 2020.
- [42] W. Deng, J. J. Xu, Y. J. Song, and H. M. Zhao, "Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem," *Applied Soft Computing Journal*, vol. 100, pp. 1–16, 2020.
- [43] W. Deng, J. Xu, H. Zhao, and Y. Song, "A novel gate resource allocation method using improved PSO-based QEA," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–9, 2020.
- [44] W. Deng, H. L. Liu, J. J. Xu, H. M. Zhao, and Y. J. Song, "An improved quantum-inspired differential evolution algorithm for deep belief network," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 10, pp. 7319–7327, 2020.