Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 17(4): 3767–3783.
DOI: 10.3934/mbe.2020211
Received: 16 March 2020
Accepted: 17 May 2020
Published: 25 May 2020

Research article

An improved spotted hyena optimizer for PID parameters in an AVR system

Guo Zhou[1], **Jie Li**[2,3], **Zhonghua Tang**[2,3], **Qifang Luo**[2,3] **and Yongquan Zhou**[2,3,4,*]

[1] Department of Science and Technology Teaching, China University of Political Science and Law, Beijing 100088, China
[2] College of Artificial Intelligence, Guangxi University for Nationalities, Nanning 530006, China
[3] Key Laboratory of Guangxi High Schools Complex System and Computational Intelligence, Nanning 530006, China
[4] Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning 530006, China

* **Correspondence:** Email: yongquanzhou@126.com; Tel: +867713260264; Fax: +867713260264.

Abstract: In this paper, an improved spotted hyena optimizer (ISHO) with a nonlinear convergence factor is proposed for proportional integral derivative (PID) parameter optimization in an automatic voltage regulator (AVR). In the proposed ISHO, an opposition-based learning strategy is used to initialize the spotted hyena individual's position in the search space, which strengthens the diversity of individuals in the global searching process. A novel nonlinear update equation for the convergence factor is used to enhance the SHO's exploration and exploitation abilities. The experimental results show that the proposed ISHO algorithm performed better than other algorithms in terms of the solution precision and convergence rate.

Keywords: spotted hyena optimizer; opposition-based learning; nonlinear convergence factor; PID parameter optimization; metaheuristic

1. Introduction

A metaheuristic optimization algorithm is a novel population-based global search algorithm, and is more suitable for solving complex problems [1,2]. Rao et al. proposed a teaching-learning-based optimization (TLBO) algorithm to solve large-scale optimization problems. The simulation results of the standard test function showed that the TLBO algorithm effectively solves complex optimization problems [3]. To solve a complex constrained optimization problem, Sayed E et al. proposed a decomposition evolutionary algorithm [4].

Mohapatra et al. proposed a competitive swarm optimizer algorithm [5]. To overcome the shortcoming of particle swarm optimization (PSO) falling easily into a local optimum, an improved quantum PSO algorithm with the cultural gene algorithm and memory mechanism was proposed to solve continuous nonlinear problems [6,7]. Ali et al. presented a multi-population differential evolution global optimization algorithm [1]. Ant colony optimization, proposed by Ismkhan, has been applied to solving complex problems [8]. Using the symbiotic organism search algorithm for fractional fuzzy controllers [9] and so on.

In an industrial control system, the proportional integral derivative (PID) controller has been widely applied. It accounts for more than 90% of the actual control system [10]. The PID parameter tuning problem proposed by Ziegler and Nichols has caused extensive concern. However, the traditional PID parameter tuning method has the following problems: The control performance index is not ideal and, typically, the method has a large overshoot and long adjustment time. The control effect of an intelligent optimization algorithm in PID parameter tuning is better than that of the traditional tuning method, and it can avoid some shortcomings of traditional methods [11]. Jiang et al. proposed a PID tuning premature with a genetic algorithm (GA) to enhance the search and convergence speed, but there were problems of premature convergence and parameter dependence [12]. Yu et al. proposed seeker search algorithm optimization PID controller parameters; improved the control precision of the system; accelerated the response speed and robustness of the system; and optimized the optimal parameters for the control system PID, but the optimization formula complex, need to set more parameters [13]. P. B. de Moura Oliveira, et al. designed Posicast PID control systems using a gravitational search algorithm (GSA) [14]. Guo-qiang Zeng et al. designed multivariable PID controllers using real-coded population-based extremal optimization [15]. A. Belkadi et al. proposed a PSO-based approach on the robust PID adaptive controller for exoskeletons [16]. M. Gheisarnejad proposed an effective hybrid harmony search (HS) and cuckoo optimization algorithm-based fuzzy PID controller for load frequency control [17]. Amal Moharam, et al. designed an optimal PID controller using hybrid differential evolution and PSO with an aging leader and challengers [18].

The spotted hyena optimizer (SHO) [19] is a novel intelligence algorithm proposed by Dhiman and Kumar in 2017. It was inspired by the social and collaborative behavior of spotted hyenas, which exist in nature. Spotted hyenas typically perform four processes: Search, encirclement, hunt, and attack prey. The SHO has the characteristics of simple, easy to implement programming and adjust the parameters set less features. Since the SHO was proposed, there have been various improved versions of the SHO algorithm. For example, N. Panda, et al. used an improved SHO (ISHO) with space transformational search to train a pi-sigma higher-order neural network [20]. H. Moayedi et al. proposed using the SHO and ant lion optimization to predict the shear strength of soil [21]. Q. Luo, et al. proposed using the SHO with lateral inhibition for image matching [22]. G. Dhiman et al. proposed a multi-objective optimization algorithm for engineering problems [23] and used the SHO to solve the nonlinear economic load power dispatch problem [24]. Xu Y, et al. proposed an enhanced moth-flame optimizer with a mutation strategy for global optimization [25].

In function optimization and engineering optimization, it has been proved that the performance of the SHO is superior to that of the grey wolf optimizer (GWO), binary GWO [26], PSO, moth-flame optimization (MFO), multi-verse optimizer, sine cosine algorithm (SCA), GSA, GA, HS, Harris hawks optimization [27], bacterial foraging optimization [28], and flower pollination algorithm [29] in terms of precision and the convergence speed [22]. In this paper, an ISHO algorithm is applied to solve PID parameter problems in an automatic voltage regulator (AVR).

The remainder of the paper is organized as follows: The basic SHO algorithm is presented in Section 2. In Section 3, the ISHO is introduced. In Section 4, the ISHO is proposed to optimize PID parameters and compared with well-known algorithms. Finally, conclusions are provided in Section 5.

2. Spotted hyena optimizer

The social relationships and habits of animals are the source of inspiration for our work. This social behavior is also present in the spotted hyena, whose scientific name is Crocuta. According to llany et al. [30], spotted hyenas typically live in groups, with as many as 100 group members, and they have mutual trust and interdependence. The communication between spotted hyenas is typically posed, and given a special signal, they track prey using sight, hearing, and smell. There are four main steps for a spotted hyena to attack prey: Search for prey, encircle prey, hunt prey, and attack prey. To generate a mathematical model for encircling, the equations are as follows [31]:

$$\vec{D} = \left|\vec{B} \cdot \vec{X}_p(t) - \vec{X}(t)\right| \tag{2.1}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{E} \cdot \vec{D} \ , \tag{2.2}$$

where t is the current iteration, $\vec{B}$ and $\vec{E}$ are the coefficient vectors, $\vec{X}_p$ is the position vector of the prey, $\vec{X}$ is the position vector, $\vec{D}$ is the distance between the prey and spotted hyena, and $||\bullet||$ represents the absolute value.

The coefficient vectors $\vec{B}$ and $\vec{E}$ are calculated as follows:

$$\vec{B} = 2 \cdot \vec{rd}_1 \tag{2.3}$$

$$\vec{E} = 2\vec{h} \cdot \vec{rd}_2 - \vec{h} \tag{2.4}$$

$$\vec{h} = 5 - (t * (5 / T)) \ , \tag{2.5}$$

where $t = 1,2,3,\cdots,T$ to balance exploration and exploitation, $\vec{h}$ linearly decreases from 5 to 0, and $\vec{rd}_1$ and $\vec{rd}_2$ are random vectors in [0,1].

To define the spotted hyena behavior mathematically, the best search agent represents the location of the prey. The other search agents move toward the best search agent and save the best solutions obtained thus far to update their positions. The mathematical model can be formulated as follows:

$$\vec{D}_h = \left|\vec{B} \cdot \vec{X}_h - \vec{X}_k\right| \tag{2.6}$$

$$\vec{X}_k = \vec{X}_h - \vec{E} \cdot \vec{D}_h \tag{2.7}$$

$$\vec{C}_h = \vec{X}_k + \vec{X}_{k+1} + \cdots + \vec{X}_{k+N} \tag{2.8}$$

$$N = count_{nos}(\vec{X}_h, \vec{X}_{h+1}, \vec{X}_{h+2}, \cdots, (\vec{X}_h + \vec{M})), \tag{2.9}$$

where $\vec{X}_h$ is the position of the best spotted hyena, $\vec{X}_k$ is the position of other spotted hyena, N is the number of spotted hyenas, $\vec{M}$ is a random vector in [0.5,1], nos is the number of solutions, $count_{nos}$ is the count of all candidate solutions, and $\vec{C}_h$ is a cluster of N optimal solutions.

For spotted hyenas in the attack prey stage, to determine the optimal solution, it is necessary to continuously reduce the value of $\vec{h}$, where $\vec{h}$ is the step size that the spotted hyena takes to attack prey, and it is clear that, when looking for prey, the spotted hyena continues to increase the number of steps gain steps. The formulation for attacking prey is

$$\vec{X}(t+1) = \frac{\vec{C}_h}{N}, \tag{2.10}$$

where $\vec{X}(t+1)$ is the position of the current solution and t is the number of iterations. The SHO allows its agents to update their positions in the direction of the prey.

Algorithm 1 Pseudocode of the SHO

```
1.  Initialize the spotted hyena population $X_i$ $(i = 1,2,\cdots,n)$
2.  Initialize $h, B, E,$ and $N$
3.  Calculate the fitness of each search agent
4.  $\vec{X}_h$ = best search agent
5.  $\vec{C}_h$ = group or cluster of all far optimal solutions
6.  while ( $t$ < max number of iterations)
7.          for each search agent do
8.              Update the position of the current search agent by Eq (2.10)
9.          end for
10.           Update $h, B, E$ and $N$
11.           Check if any search agent goes beyond the search space and revamp it
12.           Calculate the fitness of each search agent
13.           Update $P_h$ if there is a better solution by Eqs (2.6) and (2.7)
14.       $t = t + 1$
15.  end while
16. Return $\vec{X}_h$
```

The parameters $\vec{B}$ and $\vec{E}$ oblige the SHO algorithm to explore and exploit the search space. As $\vec{B}$ decreases, half the iterations are dedicated to exploration (when $|\vec{E}|>1$) and the remainder are dedicated to exploitation (when $|\vec{E}|<1$) [32]. From the above, the $\vec{B}$ vector contains random values in [0,2]. This component provides random weights for prey to stochastically emphasize ($\vec{B}>1$) or deemphasize ($\vec{B}<1$) the effect of prey in defining the distance in Eq (2.3). This helps the random behavior of the SHO to increase during the course of optimization, and favors exploration and local optima avoidance. The pseudocode of the SHO algorithm is as above.

3. Improved spotted hyena optimizer

3.1. Population initialization based on opposite learning

Haupt et al. found that the initial population affects the algorithm's accuracy and convergence speed [32]. The better than initial population can lay the foundation for the global search of the SHO algorithm [21]. However, without any prior knowledge of the global optimal solution of the problem, the SHO algorithm typically adopts a random method when generating the initial search agent, which thus affects the search efficiency. The opposition learning strategy is a new technology that has emerged in the field of intelligent computing. So far, the opposite learning strategy has been successfully applied to swarm intelligent algorithms, such as PSO, HS, and DE algorithms [33,34]. In this paper, the opposite learning strategy is embedded into the SHO for initialization.

Algorithm 2 Initialization method based on opposite learning

```
Set the population size to N
    for i=1 to N do
        for j=1 to d do
            $X_i^j = l_i^j + rand(0,1) \cdot (u_i^j - l_i^j)$
        end for
    end for
        for i=1 to N do
            for j=1 to d do
                $\breve{X}_i^j = l_i^j + u_i^j - X_i^j$
            end for
        end for
Output $\{X(N) \cup \breve{X}(N)\}$, where N denotes the individuals with the best fitness selected as the initial population.
```

Definition: Opposition-based [35]. Suppose X exists in $\left[l,u\right]$. The opposite point is $\breve{X}=l+u-X$. Let $X=(X_1,X_2,\cdots,X_d)$ be a point in d-dimensional space, where $X_1,X_2,\cdots,X_d\in R$ and $X_i\in\left[l_i,u_i\right]$ $\forall i\in\left\{1,2,\cdots,d\right\}$. The opposition-based $\breve{X}=\left(\breve{X}_1,\breve{X}_2,\cdots,\breve{X}_d\right)$ is completely defined by its components:

$$\breve{X}_i=l_i+u_i-X_i. \tag{3.1}$$

According to the above definition, the specific steps for using the opposite learning strategy to generate the initial population are as above.

3.2. Nonlinear variation convergence factor

Similar to other group intelligent optimization algorithms based on population iteration, it is crucial for the SHO to coordinate its exploration and exploitation capabilities. During exploration, groups need to detect a wider search area and avoid the SHO algorithm becoming stuck in a local optimum. The exploitation capacity mainly uses the group's existing information to search some local solution's areas of the solution. The convergence rate of the SHO algorithm has a decisive influence. Clearly, robustness and fast convergence are achieved only when the SHO algorithm improves the coordination of the exploration and exploitation capabilities.

According to [21], the SHO algorithm's exploration and exploitation abilities depend on the change of the convergence factor $\vec{h}$. The larger the convergence factor $\vec{h}$, the better the global search ability and the more likely the SHO algorithm avoids falling into a local optimum. The smaller the convergence factor $\vec{h}$, the stronger the local search ability, which speeds up the convergence of the SHO algorithm. However, in the basic SHO algorithm, the convergence factor $\vec{h}$ decreases linearly from 5 to 0 as the number of iterations increases. The linear decreasing strategy of the convergence factor $\vec{h}$ has a good global search ability in the early stages of the algorithm, but the convergence speed is slow. In the latter part of the algorithm to speed up the convergence rate, but easy to fall into a local optimum, particularly in multimode functions problems. Therefore, in the evolutionary search process, the convergence factor $\vec{h}$ with the number of iterations linearly decreasing strategy cannot fully reflect the actual optimization of the search process in the SHO algorithm [36]. In fact, the SHO is expected to have a strong global search ability in the pre-search period while maintaining a fast convergence rate. Additionally, Enns, et al. and Zeng, et al. found that performance improved if the control parameter was chosen as a nonlinearly decreasing quantity rather than using a linearly decreasing strategy [30,32]. Thus, the control parameter $\vec{h}$ is modified as follows:

$$\vec{h} = \vec{h}_{initial} - (\vec{h}_{initial} - \vec{h}_{final}) \times (\frac{Max\_iteration - t}{Max\_iteration})^{u}, \tag{3.2}$$

where t is the current iteration, $Max\_iteration$ is the maximum number of iterations, u is the nonlinear modulation index, and $\vec{h}_{initial}$ and h_{final} are the initial value and final value of control parameter $\vec{h}$, respectively. According to [32], when $\vec{h}_{initial}$ is set to 5, h_{final} becomes 0.

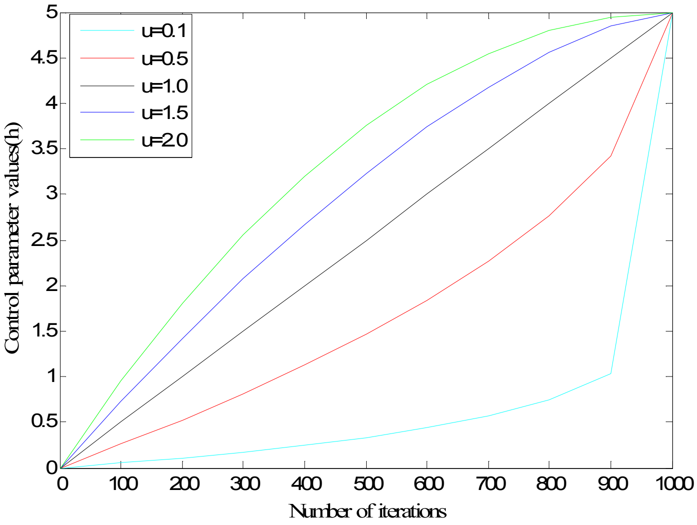

Figure 1. Control parameter $(\vec{h})$ with iterations for different values of the nonlinear modulation index (u).

Figure 1 shows typical control parameter $\vec{h}$ variations with iterations for different values of u. We conducted several SHO experiments with a nonlinear modulation index u in the interval (0, 2.0). On average, the results are better than those of existing algorithms: The larger the value of u ($u > 2.0$), the greater the failed convergence rate.

3.3. Diversity mutation operation

Similar to other population-based intelligent optimization algorithms, in the late iteration of the SHO, all the spotted hyenas move closer to the optimal individual region, which results in a reduction of population diversity. In this case, if the current optimal individual is the local optimal, then the SHO algorithm falls into a local optimum. This is also an inherent characteristic of other group intelligent optimization algorithms. To reduce the probability of premature convergence for the SHO algorithm, in this paper, a diversity mutation operation is performed on the current optimal spotted hyena individuals. The steps are as follows:

Assume that an individual $X_i = (x_{i1}, x_{i2}, \cdots, x_{id})$ of the spotted hyena species selects one element $x_k (k = 1,2,\cdots,d)$ randomly from the individual X_i with a probability of $1/d$ and randomly generates a real number in the range $[l_i, u_i]$ instead of the element x_{ik} from the individual X_i, thus producing a new individual $X_i^{'} = (x_{i1}^{'}, x_{i2}^{'}, \cdots, x_{id}^{'})$. The variation mutation operation is

$$X_i^{'} = \begin{cases} l_i + \lambda \cdot (u_i - l_i) & i = k \\ X_i & otherwise \end{cases}, \tag{3.3}$$

where l_i and u are the upper and lower bounds of the variable *x*, respectively, and $\lambda \in [0,1]$ is a random. The ISHO optimizer steps are presented in Algorithm 3.

Algorithm 3 Pseudocode of the **ISHO**

```
Set the population size $N$ using the opposite learning strategy described in Algorithm 2 to generate an initialized spotted hyena population $X_i (i = 1,2,\cdots,n)$
Initialize the parameters $h$, $B$, $E$, and $N$
Calculate the fitness of each agent
$P_h$ = best search agent
$C_h$ = group or cluster of all far optimal solutions
while $(t < t_{\max})$ do
    for $i$ =1 to $N$ do
        Using Eq (3.2), calculate the value of the convergence factor $h$
Update the other parameters $B$ and $E$ using Eqs (2.3) and (2.4), respectively
        if $(|E| \geq 1)$ do
            According to Eq (2.7), update the spotted hyena individual's position
        end if
        if $(|E| < 1)$ do
            According to Eq (2.10), update the spotted hyena individual's position
        end if
    end for
    Perform diversity mutation on the current spotted hyena individual using Eq (3.3)
    Calculate the fitness of each agent
    Update $X_i$ if there is a better solution
    $t = t+1$
end while
end
```

4. Improved spotted hyena optimizer for PID parameters in an AVR system

4.1. Description and modeling of an AVR system

Providing constancy and stability at rated voltage levels in electricity, the network is also one of the main problems in power system control. If the rated voltage level deviates from this value, then the performance degrades and the life expectancy reduces. Another important reason for this control is true line loss, which depends on the real and reactive power flow. The reactive power flow is largely dependent on the terminal voltage of the power system. However, it is necessary to reduce the loss caused by the solid line by controlling the rated voltage level. To solve these control problems, an AVR system is applied to power generation units [36]. The role of the AVR is to maintain the terminal voltage of the synchronous alternator at the rated voltage value.

4.1.1. PID controller

Using the PID controller to improve the dynamic response while reducing or eliminating the steady-state error, the derivative controller adds a finite zero to the open-loop plant, which enables the improvement of the transient response. The PID controller transfer function is

$$C(s) = K_p + \frac{K_i}{S} + K_d S \; . \tag{4.1}$$

A simple AVR system has four parts: Amplifier, exciter, generator, and sensor. The mathematical transfer function of the above four components is considered as linear and time constant. To analyze the dynamic performance of an AVR, the transfer functions of these components are in [37,38].

4.1.2. Amplifier model

The amplifier model is represented by a gain K_A and time constant τ_A. The transfer function is given by

$$\frac{V_r(s)}{V_e(s)} = \frac{K_A}{1+\tau_A s} \, , \tag{4.2}$$

where the range of K_A is [10,400] and the amplifier time constant ranges from 0.02–0.1 s.

4.1.3. Exciter model

The transfer function of an exciter is modeled by a gain K_E and time constant τ_E, and given by

$$\frac{V_f(s)}{V_r(s)} = \frac{K_E}{1+\tau_E s} \, , \tag{4.3}$$

where K_E is typically in the range [10,400] and the time constant τ_E is in the range 0.5–1.0 s.

4.1.4. Generator model

The generator model is represented by a gain K_G and time constant τ_G. The transfer function is given by

$$\frac{V_t(s)}{V_f(s)} = \frac{K_\mathrm{G}}{1+\tau_G s}, \tag{4.4}$$

where K_R is in the range [0.7,1.0] and τ_R is in the range 1.0–2.0 s. The generator gain K_R and time constant τ_R are load dependent.

4.1.5. Sensor model

The sensor is modeled by a gain K_R and time constant τ_R. The transfer function is given by

$$\frac{V_s(s)}{V_t(s)} = \frac{K_R}{1+\tau_R s}, \tag{4.5}$$

where K_R is in the range [10, 400] and τ_R is in the range 0.001–0.06 s.

The complete transfer function model of the AVR system is given in Figure 2. In the work of Gozde and Taplanmacioglu [37], the parameters of the AVR system were $K_A = 10.0$, $\tau_A = 0.1$, $K_E = 1.0$, $\tau_E = 0.4$, $K_G = 1.0$, $\tau_G = 1.0$, $K_R = 1.0$, and $\tau_R = 0.01$.

The transformer function of the AVR system with the above parameters is

$$\frac{\Delta V_t(s)}{\Delta V_{ref}(s)} = \frac{0.1s+10}{0.0004s^4 + 0.045s^3 + 0.555s^2 + 1.51s^2 + 1.51s + 11}. \tag{4.6}$$

To improve the dynamic response of the AVR system and maintain the terminal voltage at 1.0 pu, a PID controller is included, as shown in Figure 2.

With the PID controller, the transfer function of the AVR system of Figure 2 becomes

$$\frac{\Delta V_t(s)}{\Delta V_{ref}(s)} = \frac{0.1K_d s^2 + (0.1K_p + 10K_d)s^2 + (0.1K_i + 10K_p)s + 10K_i}{0.0004s^5 + 0.045s^4 + 0.555s^3 + (1.51 + 10K_d)s^2 + (1+10K_p)s + 10K_i}. \tag{4.7}$$

4.2. PID controller based on the ISHO

An AVR system with a PID controller tuned by the ISHO algorithm is shown in Figure 2. The gains of the PID controller are regulated by the ISHO algorithm. If the proportional gain is too high, then the system becomes unstable and the proportional gain becomes too low, which results in a larger error and lower sensitivity. For an AVR system, the ranges commonly used in the literature are [0.0,1.5] and [0.2,2.0] in [38,39]. To increase the search space for better optimization gains, the resulting lower and upper bounds are chosen to be 0.01 and 2, respectively.

To improve the performance control, the optimal PID parameters use the ISHO. The maximum overshoot, rise time, steady state error as a typical time domain analysis method index, and integral time multiplied by the absolute error (ITAE) are considered as control performance indicators in the design. The ITAE is the objective function with time value:

$$ITAE = \int_0^t t\left|V_t - V_{ref}\right| dt . \tag{4.8}$$

An ISHO-PID controller is presented for searching the optimal or near optimal controller parameters K_p, K_i, and K_d using the ISHO algorithm. Each individual *K* contains three members: K_p, K_i, and K_d. The searching procedures of the proposed ISHO-PID controller are presented in Algorithm 4. The three controller parameters set in the algorithm are shown in Table 1.

Algorithm 4 ISHO solution for the PID control system algorithm

```
1. Determine the parameters of the PID, proportional gain $K_p$, integral gain $K_i$, and differential gain $K_d$.
2. Randomly initialize population $X$ of $N$ individuals (solutions), $iter = 0$, and set the parameters of the ISHO, $h$, $B$, and $E$, and maximum number of iterations $iter_{\max}$.
3. Set the lower and upper bounds of the three controller parameters for each individual, apply the PID controller with gains specified by that individual to the PID controller, run all the system steps, and calculate the fitness value of each individual using Eq (4.8).
4. Determine the optimal spotted hyena individual.
        while stop if the termination criterion is not satisfied do
5. for each individual $x \in X$ do
5.1. Propagate each spotted hyena individual $x$ to a new individual $x'$ using Eq (2.7).
5.2. if $f(x') > f(x)$ then
6. Update the position of all individuals using Eq (2.10).
7. Calculate each individual fitness function using Eq (4.8).
8. $iter = iter + 1$.
9. if $iter < iter_{\max}$, then go to step 5.
10. Output the best solution and the optimal controller parameters.
```

4.3. Experimental results and analysis

The ISHO algorithm was applied to optimize the PID controller for an AVR system and determine a set of optimal gains that minimize the value of the objective function. To prove the superiority of the ISHO algorithm, we compared the ISHO with other algorithms that contain the SHO [19], GWO [40], PSOGSA [41], FPA [42], and SCA [43]. The results showed that the ISHO performed better than the other algorithms. All the parameters set in the algorithms are given in Table 2.

Setting the six algorithms' parameters, we obtained the best parameter values for each algorithm to optimize the PID parameters using the population size of 50, 20 iterations, and 20 runs independently. "Best" is the optimal fitness value, "Worst" is the worst fitness value, "Mean" is the mean fitness value, and "Std." is the standard deviation. Table 3 shows that the best fitness value of ISHO was significantly better than those of the other algorithms, and the standard error of the ISHO was the smallest. It is thus proved that the ISHO is better than the standard SHO and other algorithms (SCA, FPA, PSOGSA, and GWO) in terms of obtaining the optimal PID parameters.

Table 1. Three controller parameters set in the algorithm.

Controller parameters	Min. value	Max. value
K_p	0	1.5
K_i	0	1.0
K_d	0	1.0

Table 2. Parameters set in the six algorithms.

Algorithms	Parameter values
SCA	$r_2 \in [0,2\pi]$, $a = 2$, $r_4 \in [0,1]$. The population size is 50.
FPA	The proximity probability p = 0.8, the population size is 50.
PSOGSA	$c_1 = c_2 = 2$, $\omega_{\max} = 0.3, \omega_{\min} = 0.1$, $G_0 = 1$, $\alpha = 5$ the population size is 50.
GWO	Components $\vec{\alpha} \in [0,2]$ over the course of iterations. The population size is 50.
SHO	The parameter is $\vec{h} \in [0,5]$ over the course of iterations, the population size is 50.
ISHO	$\vec{h}_{initial} = 5$, $h_{final} = 0$, $u \in [0,2.0]$. The population size is 50.

Table 3. Optimum parameters of the PID controller.

Algorithms	SCA	FPA	PSOGSA	GWO	SHO	ISHO
K_p	1.4155	1.4230	1.4012	1.3168	1.3079	**1.0263**
K_i	0.9721	0.9974	0.9602	0.9051	0.9234	**0.7115**
K_d	0.4546	0.4302	0.4601	0.4219	0.3985	**0.3154**
Best	0.0328	0.0329	0.0328	0.0329	0.0334	**0.0327**
Worst	0.0382	0.0380	0.0333	0.0369	0.0370	**0.0331**
Ave	0.0344	0.0341	0.0330	0.0344	0.0347	**0.0328**
Std	0.0015	0.0012	1.5427×10^{-4}	0.0012	9.8111×10^{-4}	$\mathbf{8.4078 \times 10^{-5}}$

In Table 2, we label the optimal PID parameters with the optimal fitness values, and the minimum standard error is indicated by the black bold line. The table shows that, although the PSOGSA algorithm is also a hybrid of the PSO and GSA algorithms, the effect of optimally searching for PID parameters was still not as good as that of the ISHO algorithm.

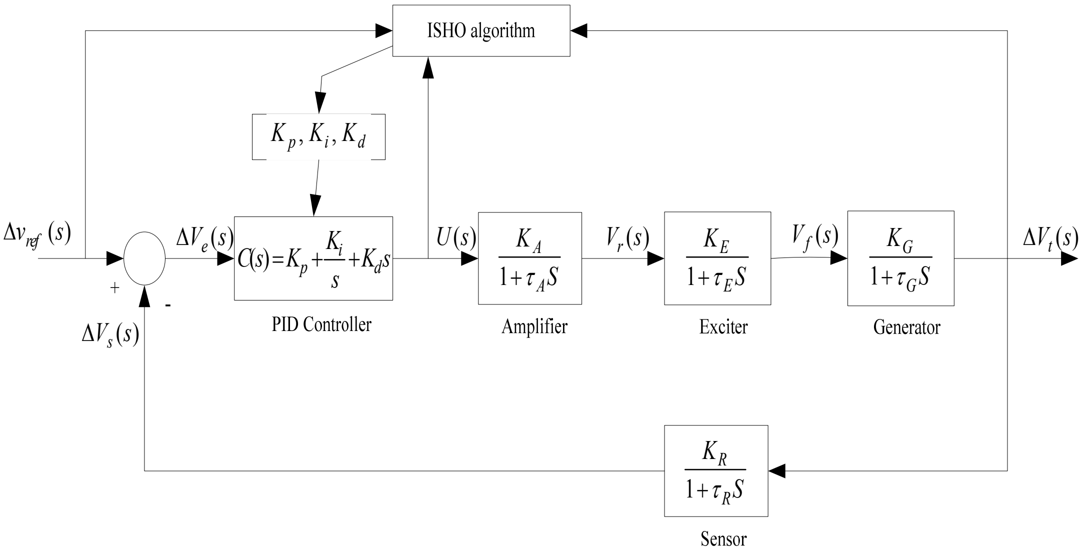

Figure 2. Block diagram of an AVR system with an ISHO-PID controller.

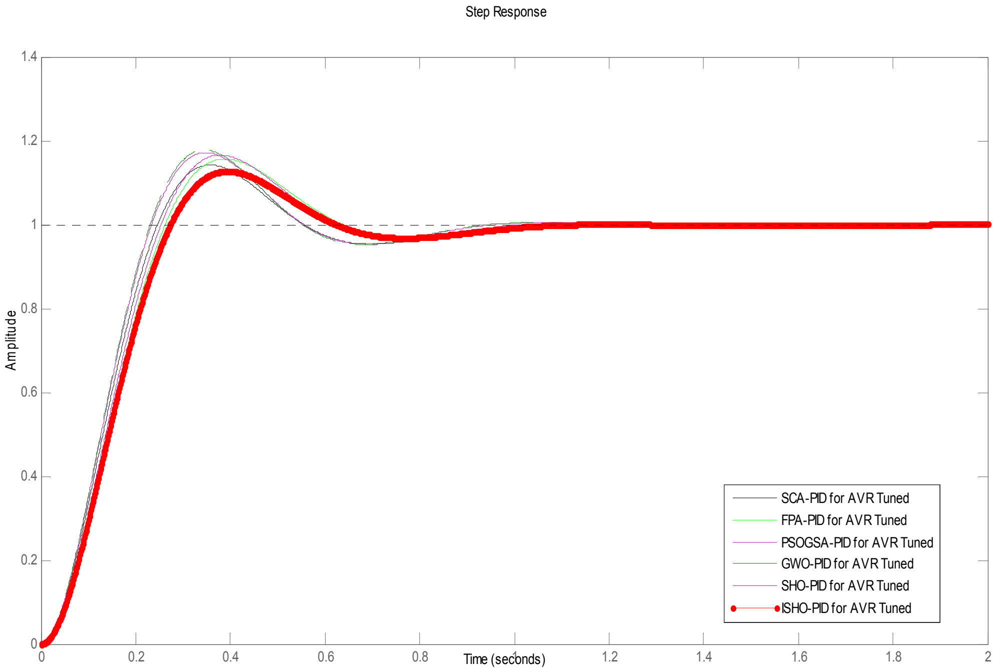

Figure 3. AVR system terminal voltage curves for different algorithms.

Table 4. Results of the transient response for different algorithms.

Algorithms	SCA	FPA	PSOGSA	GWO	SHO	ISHO
Maximum overshoots	1.1751	1.1601	1.1383	1.1767	1.2017	1.120
Peak time (s)	0.3821	0.3923	0.3883	0.3924	0.3951	0.4160
Settling time (s)	0.6930	1.003	0.9723	0.9494	0.9288	0.8481
Rise time (s)	0.2382	0.2207	0.2883	0.2274	0.2010	0.3021

The transient and steady-state behavior of the system can be analyzed from the transient analysis of the ISHO optimized PID controller in the AVR system, as shown in Figure 4. For comparison, responses for the SCA, FPA, PSOGSA, GWO, and SHO algorithms are shown in Table 8. The figure shows that the maximum overshoot for the ISHO algorithm is 5% less than that of the SCA algorithm, 3.5% less than that of the FPA algorithm, 4.82% less than that of the GWO algorithm, and 6.8% less than that of the SHO algorithm. The peak time for the ISHO algorithm is more than that of the SHO, GWO, PSOGSA, FPA, and SCA algorithms. The maximum overshoot and settling time for the ISHO algorithm are better than those of the SHO, GWO, PSOGSA, FPA, and SCA algorithms, and are major factors for comparing the stability analysis of systems.

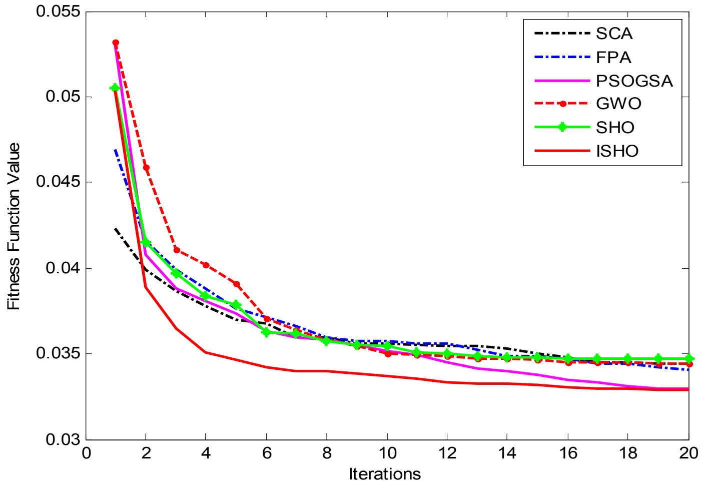

Figure 4. Evolution curves of the fitness values.

The convergence characteristics are shown in Figure 4. The figure shows that the ISHO algorithm's fitness value decreases the fastest compared with those of the other algorithms. This shows that the ISHO algorithm has a strong global search capability and higher precision. The ISHO algorithm is considered as an optimization of the PID controller parameters in the AVR system, which has promising potential applications.

5. Conclusions

The SHO was inspired by the social behavior of a spotted hyena swarm. Its mathematical model is relatively simple, but the control parameter directly affects the balance between the global search ability and local search ability in the SHO algorithm. Based on the analysis of the above characteristics of the SHO, in this paper, a nonlinear adjustment strategy was adopted for the control parameters and the mutation strategy was used to deal with the update of the intelligent individual position. The performance of the improved algorithm was verified by a simulation. The ISHO quickly approached the theoretical value and significantly improved the convergence speed and optimization efficiency. The ISHO algorithm was used to determine the parameters of the PID controller for an AVR system. It is clear from the results that the proposed ISHO algorithm avoided the shortcoming of the premature convergence of the SHO, GWO, PSOGSA, FPA, and SCA algorithms and obtained global solutions with better computation efficiency.

Acknowledgments

This work was supported by the Project of China University of Political Science and Law Research Innovation under Grant No. 10818441 and the Young Scholar Fund of China University of Political Science and Law under Grant No. 10819144. We thank Maxine Garcia, PhD, from Liwen Bianji, Edanz Group China (www.liwenbianji.cn/ac) for editing the English text of a draft of this manuscript.

Conflict of Interests

The authors declare no conflict of interest.

References

1. G. Huang, T. Li, Q. Lu, Artificial memory-based optimization, *Syst. Eng. Theory Pract.*, **11** (2014), 2900–2912.
2. S. Rahnamayan, G. Wang, Solving large scale optimization problems by opposition-based differential evolution (ODE), *WSEAS Trans. Comput.*, **7** (2008),1792–1804.
3. R. Rao, V. Savsani, D. Vakharia, Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems, *Comput. Aided Des.,* **43** (2011), 303–315.
4. E. Sayed, D. Essam, R. Sarker, S. Elsayed, Decomposition-based evolutionary algorithm for large scale constrained problems, *Inf. Sci.,* **316** (2015), 457–486.
5. P. Mohapatra, K. Das, S. Roy, A modified competitive swarm optimizer for large scale optimization problems, *Appl. Soft Comput.*, **59** (2017), 340–362.
6. D. Tang, Y. Cai, J. Zhao, Y. Xue, A Quantum-behaved particle swarm optimization with memetic algorithm and memory for continuous non-linear large scale problems, *Inf. Sci.,* **289** (2014), 162–189.
7. H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, M. Ventresca, Enhancing particle swarm optimization using generalized opposition-based learning, *Inf. Sci.,* **181** (2011), 4699–4714.
8. H. Ismkhan, Effective Heuristics for ant colony optimization to handle large-scale problems, *Swarm Evol. Comput.,* **32** (2017), 140–149.
9. Y Zhou, F. Miao, Q Luo, Symbiotic organisms search algorithm for optimal evolutionary controller tuning of fractional fuzzy controllers, *Appl. Soft Comput.* **77** (2019), 497–508

10. Z. Yang, Z. Chen, Z. Fan, X. Li, A Tuning of PID controller based on improved particle-swarm optimization, *Control Theory Appl.*, **27** (2010), 1345–1352
11. L. Echevarría, O. Santiago, J. Fajardo, A. Silva Neto, D. Sánchez, A variant of the particle swarm optimization for the improvement of fault diagnosis in industrial systems via faults estimation, *Eng. Appl. Artif. Intell.,* **28** (2014), 36–51.
12. J. Jiang, Y. Xue, Q. Yang, Combined algorithm for PID tuning based on genetic algorithm and direct search, *Comput. Simul.,* **12** (2005), 139–142.
13. Y. Zhou, J. Zhang, X. Yang, Y. Ling, Optimization of PID controller based on water wave optimization for an automatic voltage regulator system, *Inf. Technol. Control,* **48** (2019), 160–171.
14. P. B. de Moura Oliveira, E. J. S. Pires, P. Novais, Design of Posicast PID control systems using a gravitational search algorithm, *Neurocomputing,* **167** (2015),18–23
15. G. Q. Zeng, J. Chen, M. R. Chen, Y. X. Dai, L. M. Li, K. D. Lu, et al., Design of multivariable PID controllers using real-coded population-based extremal optimization, *Neurocomputing*, **151** (2015), 1343–1353
16. A. Belkadi, H. Oulhadj, Y. Touati, S. A. Khan, B. Daachi, On the robust PID adaptive controller for exoskeletons: A particle swarm optimization based approach, *Appl. Soft Comput.,* **60** (2017), 87–100
17. M. Gheisarnejad. An effective hybrid harmony search and cuckoo optimization algorithm based fuzzy PID controller for load frequency control, *Appl. Soft Comput.*, **65** (2018), 121–138.
18. A. Moharam, M. A. El-Hosseini, H. A. Ali, Design of optimal PID controller using hybrid differential evolution and particle swarm optimization with an aging leader and challengers, *Appl. Soft Comput.* , **38** (2016), 727–737
19. G. Dhiman, V. Kumar, Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications, *Adv. Eng. Software*, **114** (2017), 48–70.
20. N. Panda, S. K. Majhi, Improved spotted hyena optimizer with space transformational search for training pi-sigma higher order neural network, *Comput. Intell.*, **36** (2020), 320–350.
21. H. Moayedi, D. T. Bui, D. Anastasios, B. Kalantar, Spotted Hyena Optimizer and Ant Lion Optimization in Predicting the Shear Strength of Soil, *Appl. Sci. Basel*, **9** (2019), 2.
22. Q. Luo, J. Li, Y. Zhou. Spotted hyena optimizer with lateral inhibition for image matching, *Multimedia Tools Appl.*, **78** (2019), 34277–34296.
23. G. Dhiman, V. Kumar. Multi-objective spotted hyena optimizer: A Multi-objective optimization algorithm for engineering problems, *Knowl. Based Syst.,* **150** (2018), 175–197.
24. G. Dhiman, S. Guo, S. Kaur, ED-SHO: A framework for solving nonlinear economic load power dispatch problem using spotted hyena optimizer, *Mod. Phys. Lett. A*, **33** (2018), 1850239.
25. Y. Xu, H. Chen, J. Luo, Q. Zhang, S. Jiao, X. Zhang, Enhanced Moth-flame optimizer with mutation strategy for global optimization, *Inf. Sci.*, **492** (2019), 181–203.
26. P. Hu, J. Pan, S. Chu, Improved Binary Grey Wolf Optimizer and Its application for feature selection, *Knowl. Based Syst.,* **195** (2020), 105746.
27. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872.
28. H. Chen, Q. Zhang, J. Luo, Y. Xu, X. Zhang, An enhanced Bacterial Foraging Optimization and its application for training kernel extreme learning machine, *Appl. Soft Comput.,* **86** (2020), 105884.

29. T. T. Nguyen, J. S. Pan, T. Dao, An improved flower pollination algorithm for optimizing layouts of nodes in wireless sensor network, *IEEE Access*, **7** (2019), 75985–75998.
30. A. Ilany, A. Booms, K. Holekamp, Topological effects of network structure on long-term social network dynamics in a wild mammal, *Ecol. Lett.*, **18** (2015), 687–695.
31. R. Haupt, S. Haupt, Practical genetic algorithms, second edition, New York, John Wiley & Sons, Inc. 2004.
32. X. Gao, X. Wang, S. J. Ovaska, K. Zenger, A hybrid optimization method of harmony search and opposition-based learning, *Eng. Optim.*, **44** (2012), 895–914.
33. H. Tizhoosh, *Opposition-based learning: A new scheme for machine intelligence*, International Conference on Intelligent Agents, *IEEE*, 2005, 695–701. Available from: https://ieeexplore.ieee.org/abstract/document/1631345.
34. M. Omran, S. Al-Sharhan, *Using Opposition-based learning to improve the performance of particle swarm optimization*, 2008 IEEE Swarm Intelligence Symposium, 2008, 1–6. Available from: https://ieeexplore.ieee.org/abstract/document/4668288.
35. M. A. Ahandani, H. Alavi-Rad, Opposition-based learning in the shuffled differential evolution algorithm, *Appl. Math. Comput.*, 1**6** (2012), 1303–1337.
36. M. Enns, Electric Energy Systems Theory, *IEEE Trans. Autom. Control*, **17** (1972), 749–750.
37. H. Gozde, M. C. Taplamacioglu, Comparative performance analysis of artificial bee colony algorithm for automatic voltage regulator (AVR) system, *J. Franklin Inst.*, **348** (2011), 1927–1946.
38. L. Coelho, Tuning of PID Controller for an automatic regulator voltage system using chaotic optimization approach, *Chaos Solitons Fractals*, **39** (2009), 1504–1514.
39. D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, *Appl. Math. Comput.*, **214** (2009), 108–132.
40. S. Mirjalili, S. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Software*, **69** (2014), 46–61.
41. S. Mirjalili, S. Hashim, *A new hybrid PSO + GSA algorithm for function optimization International Conference on Computer and Information Application,* 2010 International Conference on Computer and Information Application, 2012, 374–377. Available from: https://ieeexplore.ieee.org/abstract/document/6141614.
42. X. Yang, *Flower pollination algorithm for global optimization*, International Conference on Unconventional Computing and Natural Computation, 2012, 242–243. Available from: https://link.springer.com/chapter/10.1007/978-3-642-32894-7_27.
43. S. Mirjalili, SCA: A sine cosine algorithm for solving optimization problems, *Knowl. Based Syst.*, **96** (2016), 120–133.