Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 3343505, 38 pages
https://doi.org/10.1155/2022/3343505

Research Article

Fractional-Order Boosted Jellyfish Search Optimizer with Gaussian Mutation for Income Forecast of Rural Resident

Yang Lei,[1] Lingyun Fan ,[2] Juntao Yang ,[3] and Wenhu Si[3]

[1]*School of Public Administration, Northwest University, Xi'an 710127, China*
[2]*School of Architecture and Urban Planning, Suzhou University of Science and Technology, Suzhou 215011, China*
[3]*Academy of Architecture, Chang'an University, Xi'an 710061, China*

Correspondence should be addressed to Lingyun Fan; yun1223@163.com and Juntao Yang; yangjt@chd.edu.cn

Received 18 March 2022; Accepted 28 April 2022; Published 13 June 2022

Academic Editor: Maciej Lawrynczuk

Copyright © 2022 Yang Lei 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 disposable income of residents can reflect the living standard of people in the area. For government departments, it is necessary to master the trend of rural resident income to formulate corresponding policies benefiting farmers. Thus, this paper proposes a grey model with an improved jellyfish search optimizer to predict the rural resident income in Shaanxi Province. Firstly, by applying fractional-order modified strategy and Gaussian mutation mechanism to the original algorithm, the proposed algorithm shows better performance in solving accuracy, stability, and convergence acceleration when compared with different classical methods on cec2017 and cec2019 test functions. Then, based on the fractional time-delayed grey model, a discrete fractional time-delayed grey model with triangular residual correction (TDFTDGM) is proposed by replacing the derivative with a first-order difference and introducing the triangular residual correction functions. Finally, the improved jellyfish search optimizer is used to explore the optimal order of the TDFTDGM model. The all-around performance of the forecast model is incomparable to additional grey models compared on four measure criteria, which means it is a practical approach for long-term prediction with small samples. Moreover, the forecast data of rural resident income in Shaanxi Province from 2021 to 2025 are given for reference.

1. Introduction

Agriculture, rural areas, and farmers are important issues for the long-term stability of the country in China [1]. In addition, the income of rural residents is a key index, which reflects the living standard of people in rural areas and the economic development of rural farmers [2]. Only by understanding the trend of rural residents' income, the government is able to formulate a series of policies to improve the income of rural residents [3]. However, there are only a few empirical studies on income prediction in the current literature. It is because that it is highly difficult and time-consuming to get exact information about the disposable income of a region in a long period [4]. Meanwhile, due to the income being affected by policies, climate, and other uncertainty factors, it is a challenging task to predict the income accurately [5].

Though the forecasting models for resident income are scarce, there are many forecasting approaches for other areas. For example, Maaouane et al. used the multiple linear regression method to predict the industry energy demand in Morocco [6]. Radial neural network is also a popular tool, which was used for energy consumption forecasting and wind speed forecasting in [7, 8], respectively. The authors in [9, 10] used the ARIMA model to the daily production prediction of wells in Sulige and to forecast the rural population in China from 1970 to 2015. Although these approaches can complete the task of data prediction according to different features, there are some defects. A tremendous amount of sample data is required, which means the above methods are not suitable for problems with a small sample [11]. As a choice, the grey forecasting algorithm solves the prediction problem of a small sample data set.

The grey model (GM) is an effective forecast approach with microscopic samples, which was proposed in 1982 [12]. It has the benefits of simple calculation, heightened precision, and wide application. As scholars have a deeper understanding of GM, some enhanced models were presented to enhance the accuracy. The classic grey model (GM) is mixed with the trigonometric residual modification strategy. Zhou et al. proposed a novel trigonometric grey prediction approach (TRGM) to forecast electricity needs and obtain effective results [13]. Then, in [14], an unknown discrete grey forecasting model called the DGM was designed. It showed outstanding performance in predicting the long-term developing tendency of an information series. Meanwhile, Wu et al. proposed a novel nonlinear grey Bernoulli model with fractional-order accumulation, shortened as the FANGBM model in 2019 [15]. This model was used to predict increase trend of the future China's renewable consumption. Though the curtain-raiser of fractional-order collection has created meaningful contributions to forecasting methods, some issues may also be mistaken as they do not consider the time-delayed effect. Thus, the authors in [16] introduce a new fractional grey model, called the fractional delay grey model (FTDGM). We design a novel grey model to obtain better-predicted results considering the significance of the discrete model and trigonometric residual modification technique.

Moreover, there is a parameter to be determined in the fractional grey model, the fractional order. Then, how to choose the most suitable parameters becomes another thorny problem. The authors in [16] provided a practical solution, which applied a metaheuristic algorithm to select parameters.

Metaheuristic methods have been grown rapidly in current years and show outstanding performance in solving continuous, discrete, or nonlinear optimizations problems [17, 18]. Generally, metaheuristic algorithms can be categorized into four varieties, swarm intelligence (SI) algorithms, evolutionary algorithms (EAs), physics-based algorithms (PhAs), and human-based algorithms [19]. The cooperative and hunting behavior of social animals in nature inspire SI algorithms. Particle swarm optimization (PSO) is the most classical one, which has been employed to solve different problems [20]. With the exploration of animal habits in recent years, lots of SI algorithms have emerged. In 2015, Wang et al. proposed the monarch butterfly optimization (MBO) algorithm by simplifying and idealizing the migration of monarch butterflies [21]. After being compared with other algorithms on thirty-eight benchmark functions, the results showed the capability of the MBO method significantly outperformed the other five algorithms [21]. In 2020, inspired by a unique mathematical model that slime mould forms the optimal path for connecting food through the positive and negative feedback of the propagation wave, Li et al. proposed the slime mould algorithm (SMA) [22]. In addition, in 2021, by simulating the behavior of African vultures and emperor penguin, respectively, the African vulture optimization algorithm (AVOA) [23] and Aptenodytes Forsteri Optimization (AFO) [24] were designed and provided excellent performance. Similar algorithms are available for moth search algorithm (MSA) [25], colony predation algorithm (CPA) [26], and so on. EA algorithms are inspired by the natural laws of population development and evolution. Among EA algorithms for solving various optimization tasks, the genetic algorithm (GA) [27] and differential evolution algorithm (DE) [28] are undoubtedly the most touted. PhA algorithms rely on physical regulation to suggest solutions to optimization difficulties. Such as multiverse optimizer (MVO) was inspired by the multiverse theory in physics [29]. In addition, the Archimedes optimization algorithm (AOA) is a novel PhA algorithm created with motivations from an exciting regulation of physics Archimedes regulation [30]. In 2021, based on the logic of slope variations computed by the Runge–Kutta method, the Runge–Kutta optimizer (RUN) was proposed and offered outstanding performance on 50 mathematical test functions and four real-world engineering problems [31]. The last set of nature-inspired methods simulates some natural human behaviors. Such as teaching-based learning algorithm (TBLA) [32], socioevolution learning optimization algorithm (SELOA) [33], preaching optimization algorithm (POA) [34], and hunger games search (HGS) [35].

Jellyfish search (JS) optimizer is a high-profile metaheuristic algorithm suggested in 2020, which was roused by the conduct of jellyfish in the ocean [36]. After being compared with ten prominent metaheuristic algorithms on the encyclopedic set of mathematical standard functions and used in a sequence of structural engineering concerns, JS is potentially a flawless algorithm for solving optimization problems. Unavoidably, the original JS algorithm also suffers defects in solving accuracy and premature convergence. Thus, this paper introduces the fractional-order modified strategy, and Gaussian mutation mechanism into the original JS algorithm, jellyfish search algorithm based on fractional-order modified, and Gaussian mutation mechanism (FOGJS). In addition, we apply the improved algorithm to the novel grey model to obtain the optimal order of the forecast model. The contribution of this paper can be outlined as follows:

An enhanced version of the jellyfish search algorithm with fractional-order modified and Gaussian mutation mechanism is proposed. And the validity of the improved algorithm is discussed on test functions of cec2017 and cec2019 by being compared with the original JS and other ten additional algorithms.

Based on the fractional time-delayed grey model, we alternate the derivative with a first-order difference. It introduces the trigonometric residual modification technique to design a novel forecast model—a discrete fractional time-delayed grey model with triangular residual correction (TDFTDGM).

Taking the rural income data of Shaanxi Province as an example, apply the FOGJS to TDFTDGM to search the most suitable fractional order of the forecast model. Then, compared with other optimization algorithms and grey models, the fitting and predicted errors of the FOGJS + TDFTDGM approach are discussed.

The rest of the paper is systematized as tracks. In Section 2, we give the theory of jellyfish search optimizer. The improved JS algorithm is proposed in Section 3. Section 4 examinations the performance of improved algorithms on miscellaneous test functions. Section 5 presents the novel grey model and predicts the income of rural residents by different models. Eventually, the work of this paper is abstracted in Section 6.

2. The Theory of Jellyfish Search Optimizer

The jellyfish search (JS) is a recent swarm intelligence method founded on jellyfish demeanor in the ocean. The jellyfish's search behavior and movement mode in the ocean encourages the algorithm [36]. As Figure 1 shows, the jellyfish will move with the ocean current or move in the population. Firstly, it is affected by the ocean current. Each jellyfish will follow the ocean current to form a jellyfish gathering. Secondly, once the surrounding food changes, jellyfish will move within the group. These motions are switched by using a time control mechanism.

The different initialization distributions of exact solutions in the search space will affect whether they will eventually fall into local solutions. After observing the typical random methods, it is finally found that the jellyfish search optimizer performs well under the logistic map. The mathematical description is as follows [36]:

$$Z_{i+1} = \alpha Z_i \left(1 - Z_i\right), 0 \leq Z_0 \leq 1, \tag{1}$$

where Z_i represents the i-th jellyfish logistic chaotic value, Z_0 is the first people of developed jellyfish, Z_0 is set between 0 and 1, but Z_0 cannot take some particular values, such as 0.0, 0.25, 0.5, 0.75, 1.0, and α is selected to 4.0.

The ocean currents contain a lot of nutrients and are easier to survive, attracting jellyfish. Therefore, the current ocean direction is usually specified by the average vectors from all jellyfish in the ocean group to the jellyfish presently in the optimal situation. The mathematical term of ocean current is as tails [36]:

$$Z_i(t+1) = Z_i(t) + r_1 \times \left(Z^* - \beta \times r_2 \times \mu\right), \tag{2}$$

where r_1 and r_2 are the accidental numerals between 0 and 1, Z^* is a jellyfish in the most suitable place at present, $\beta > 0$ is a distribution coefficient, and β is usually taken as 3.

The movement in the jellyfish group can be divided into active and passive movements. Initially, when the jellyfish group was just formed, most jellyfish showed passive movement. Passive motion is the movement of jellyfish near its place. At this time, the updated connected place of individual position is

$$Z_i(t+1) = Z_i(t) + \gamma \times r_3 \times (ub - lb), \tag{3}$$

where ub is the upper attached and lb is the lower bound of the tracking area, in addition, γ is a move coefficient and its value > 0, which is connected to the move length of the jellyfish near its place, and the original algorithm usually takes $\gamma = 0.1$, and r_3 is an arbitrary number in the range of 0 and 1.

The active movement relies on comparing the food quantity of two jellyfish to judge whether there is relative movement. When the food quantity near the other jellyfish is higher, it will move towards it. The expression of active motion is as tails [36]:

$$Z_i(t+1) = Z_i(t) + r_4 \times \overrightarrow{\text{Direction}}, \tag{4}$$

where r_4 is a random number between 0 and 1, and $\overrightarrow{\text{Direction}}$ is used to determine the movement direction of the current jellyfish in the next iteration. This movement always moves in the direction of better food position, and represents the following formula [36]:

$$\overrightarrow{\text{Direction}} = \begin{cases} Z_k(t) - Z_i(t), & \text{if } f(Z_i) \geq f(Z_i), \\ Z_i(t) - Z_k(t), & \text{if } f(Z_i) < f(Z_i), \end{cases} \tag{5}$$

where k is the index of a jellyfish determined haphazardly, and f is an objective function.

A time control tool is presented to control the tendency of people between observing the ocean current and driving within the people. Figure 2 is the allocation of activity within the people. The time control instrument contains a time control function $c(t)$ and a regular c_o, and the time control function is an arbitrary matter that fluctuates from 0 to 1. The term of the time control instrument is as follows [36]:

$$c(t) = \left(1 - \frac{t}{t_{\max}}\right) \times \left(2 \times r_5 - 1\right), \tag{6}$$

where t is the recent iterations, $t_{\max}$ is the total iterations, and r_5 is a random number between 0 and 1. When $c(t) \geq c_o$, swim inward with other passive or active actions; when the randomly generated random number is more significant than $(1 - c(t))$, the current people use the inactive motion. Otherwise, the active movement is involved.

3. An Improved Jellyfish Search Optimizer

An improved JS algorithm based on fractional-order modified and Gaussian mutation mechanism (FOGJS) is proposed to solve the problem that the jellyfish search algorithm quickly falls into optimal local solution and improves accuracy.

3.1. Fractional-Order Modified Strategy. Fractional calculus extends the order of calculus from integer to fraction and

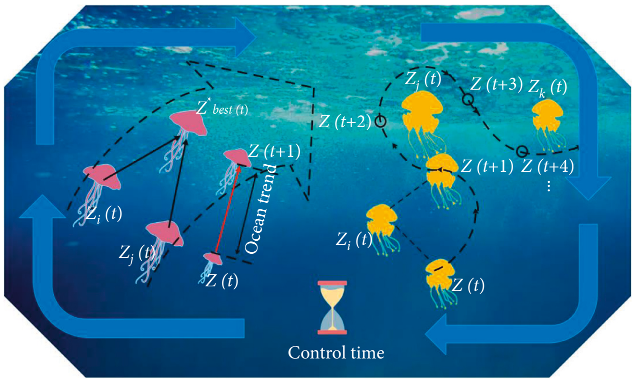


FIGURE 1: The simulation of ocean current and swarm.

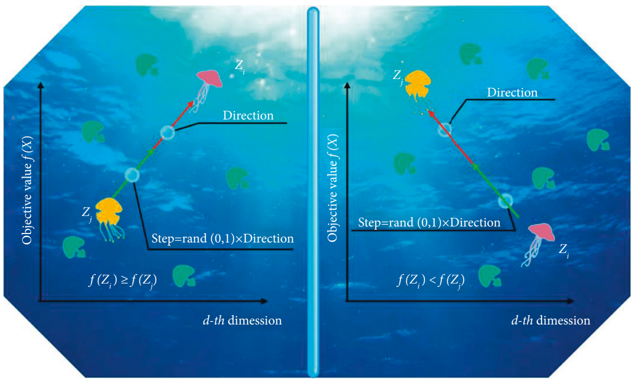


FIGURE 2: The direction of movement of jellyfish.

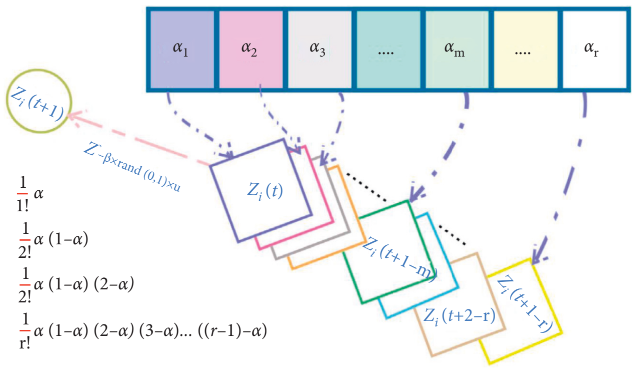


FIGURE 3: FOC memory property-based Z concept.

recursively deduces the solution limit through the difference approximation of integer calculus, that is, the differentiation and integration with the order of fraction. There are many definitions of derivatives, such as Grünwald–Letnikov, Caputo fractional derivatives [37], Riesz potential [38], and so on. The most commonly used definitions is Grünwald–Letnikov (G-L) [39] definition:

$$D^{\alpha}[z(t)] = \lim_{T\longrightarrow 0}\left[\frac{1}{T^{\alpha}}\sum_{k=0}^{+\infty}\frac{(-1)^k\Gamma(\alpha+1)z(t-kT)}{\Gamma(k+1)\Gamma(\alpha-k+1)}\right], \tag{7}$$

where α is the fractional coefficient of a public signal $z(t)$, Γ is a gamma function, and T is the truncation term.

The discrete expression of G-L can be expressed as

$$D^{\alpha}[z(t)] = \frac{1}{T^{\alpha}}\sum_{k=0}^{r}\frac{(-1)^k\Gamma(\alpha+1)z(t-kT)}{\Gamma(k+1)\Gamma(\alpha-k+1)}. \tag{8}$$

Taking advantage of the fractional learning and training algorithm is easy to leap out of the local extreme points. The jellyfish search algorithm is integrated with the fractional-order modified strategy to adjust the fractional-order by updating the jellyfish position in the ocean current and the jellyfish passive motion position. Let $\alpha = 4$ in equation (8) have the following:

$$\begin{aligned} D^{\alpha}[z(t+1)] \approx{}& z(t+1)-\alpha z(t)+\frac{1}{2}\alpha(\alpha-1)z(t-1)\\ &-\frac{1}{6}\alpha(\alpha-1)(\alpha-2)z(t-2)\\ &+\frac{1}{24}\alpha(\alpha-1)(\alpha-2)(\alpha-3)z(t-3). \end{aligned} \tag{9}$$

The fractional derivative results are related to the current term and previous state values, and the influence of past events decreases with time. The position update of jellyfish moving under the effect of ocean current and the position update of people passive movement when jellyfish group just formed are equations (2) and (3), respectively. Figure 3 shows how fractional-order correction affects the update. The left side of equations (2) and (3) is fractional-order G-L, which defines the discrete form when order α is 1, and T is 1; that is,

$$\begin{aligned} D^{\alpha}[z(t+1)] &= r_1\times\left(Z^{*}-\beta\times r_2\times\mu\right),\\ D^{\alpha}[z(t+1)] &= \gamma\times r_3\times\left(U_b-L_b\right). \end{aligned} \tag{10}$$

Therefore, the position update formula of jellyfish affected by ocean current after fractional-order modified is as follows:

$$\begin{aligned} Z_{i,j}^{t+1} ={}& \alpha Z_{i,j}^{t}-\frac{1}{2}\alpha(\alpha-1)Z_{i,j}^{t-1}-\frac{1}{6}\alpha(\alpha-1)(\alpha-2)Z_{i,j}^{t-2}\\ &+\frac{1}{24}\alpha(\alpha-1)(\alpha-2)(\alpha-3)Z_{i,j}^{t-3}+r_1\\ &\times\left(Z^{*}-\beta\times r_2\times\mu\right). \end{aligned} \tag{11}$$

The update formula of passive motion position of jellyfish after fractional-order modified is

$$\begin{aligned} Z_{i,j}^{t+1} ={}& \alpha Z_{i,j}^{t}-\frac{1}{2}\alpha(\alpha-1)Z_{i,j}^{t-1}-\frac{1}{6}\alpha(\alpha-1)(\alpha-2)Z_{i,j}^{t-2}\\ &+\frac{1}{24}(\alpha-1)(\alpha-2)(\alpha-3)Z_{i,j}^{t-3}+\gamma\times r_3\times\left(U_b-L_b\right). \end{aligned} \tag{12}$$

It is worth noting that when the terms of 1/120 or 1/720 or higher are multiplied by the remaining terms of equation (8), the values of these terms become negligible and hardly affect the update of position. Therefore, the higher-order terms are discarded.

3.2. Gaussian Mutation Mechanism

3.2. Gaussian Mutation Mechanism. The Gaussian distribution, also understood as the standard distribution, is a substantial probability distribution in mathematics, artificial intelligence, and other related fields [40]. The Gaussian mutation tool is employed to generate a further variant, which retains a better position by comparing it with the target value of the current optimal individual situation. This mechanism makes the algorithm's local results and global results well balanced [41]. The probability density function (PDF) formula of Gaussian distribution is as tails:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}}\exp\left(-\frac{(z-u)^2}{2\sigma^2}\right), \tag{13}$$

where μ is the anticipation of Gaussian distribution and σ is defined as the standard deviation of Gaussian distribution. We can describe the resulting new mutation location as

$$Z_{i,j}^{new}(t+1) = Z_{i,j}^{\text{best}}(t+1) \times (1 + \theta \times G(0,1)), \tag{14}$$

where θ is a declining random integer in the range of 0 and 1, $G(0,1)$ is the expected Gaussian distribution, and $Z_{i,j}^{best}$ is the best location for the current iteration.

3.3. Explicit Steps for the Improved Jellyfish Search Optimizer.

The fractional-order modified strategy and Gaussian mutation mechanism are presented in JS. The convergence precision of the JS algorithm is effectively enhanced, and the tracking performance of the original JS algorithm is improved. The exact steps of FOGJS are as follows:

Step 1. Give some parameters associated with FOGJS, such as people dimensions N, varying dimension Dim, upper set range ub, lower set range lb, total iterations M_{iter}, the distribution coefficient β, and the motion coefficient γ.

Step 2. Aimlessly initialize N people size according to chaotic logistic map (equation (1)), and set time $t = 1$.

Step 3. Compute the fitness value for each people, document the optimal fitness value and the related optimal place Z^*.

Step 4. While $t < M_{iter}$, compute the control time $c(t)$ according to equation (6), if $c(t) \geq 0.5$, jellyfish will follow ocean current, update the position of jellyfish by equation (11).

Step 5. If $c(t) < 0.5$, randomly generate an accidental number r, ranging from 0 to 1, if $r > (1 - c(t))$, the jellyfish will update its position by passive motion of equation (12). If $r \leq (1 - c(t))$, the position of the jellyfish is computed by active motion according to equation (4).

Step 6. Judge whether the updated new location crosses the boundary. If yes, the site is set near the border by default. At the same time, the fitness value of individual jellyfish is estimated. If it is more undersized than the optimal value, it is accepted as the new optimal value, and the related position is accepted as the recent optimal position Z^*.

Step 7. Through the Gaussian mutation mechanism of equation (14), the optimal position is mutated to produce a new solution, and judge its fitness with the optimal solution, to choose whether to update the mutated new solution to the optimal solution.

Step 8. Update the value of t $(t = t + 1)$, if $t < M_{iter}$, continue Step 4. Otherwise, output the optimal value and the optimal place.

Step 9. Otherwise, output the optimal value and the optimal place.

To express the enhanced algorithm more obviously, Algorithm 1 gives the pseudocode of the FOGJS. Among them, line 1 is to update the candidate solution position through the logistic chaotic map, lines 9 and lines 14 are the update operation process of applying the fractional modified update formula, and lines 22–26 are the update process after generating the mutation solution through the Gaussian mutation mechanism. Figure 4 shows the flowchart of the proposed FOGJS algorithm. It can be found that it first defines the parameters listed above and then executes the updated JS based on fractional-order modified. The optimal solution is taken as the initial point of mutation of the Gaussian mutation mechanism. The loop will continue, jump out of the loop when the termination conditions are met and then output the optimal solution of the search.

3.4. The Complexity Analysis of FOGJS.

The computational complexity of FOGJS mainly relies on three aspects: initialization stage, fitness function evaluation, and coordinated update. The complexity of the initialization stage is $O(N)$, where N is the number of candidate solutions. Different problems lead to additional complexity of the fitness function, so we will not be concerned about it here. Finally, the complexity update location is $O(N \times M)$, where M represents maximum iterations. Therefore, the computational complexity of FOGJS the algorithm is $O(N \times M)$. In the following sections, we will use diverse benchmark functions and actual optimization problems to verify the implementation of FOGJS in dealing with different optimization problems.

4. Numerical Examples and Analysis

This part evaluates the comprehensive performance of the improved FOGJS algorithm in solving challenging test functions. Two series of popular functions are used, including 29 cec2017 benchmarks and 10 cec2019 benchmarks [42]. The optimization results of FOGJS are analyzed and compared with other famous optimization algorithms. Meanwhile, to ensure the consistency and reliability of the improved FOGJS, 20 independent tests were conducted on FOGJS and other algorithms. The mean (mean), standard deviation (STD), the worst solution to date, and the best solution to date are reported. To achieve a reasonable comparison, other algorithms considered achieving the same number of iterations 1000 and population 30 as the FOGJS.

4.1. Evaluation Index.

The following section will analyze the performance differences between the improved FOGJS algorithm and other comparison algorithms through the following evaluation indicators.

4.1.1. The Best Results

$$\text{best} = \min(\text{best}_1, \text{best}_2, \ldots, \text{best}_{\text{runs}}), \tag{15}$$

where $best_i$ is the best result of each run. *Runs* is the number of runs of each algorithm in each benchmark function, which is taken as 20 in this paper.

4.1.2. The Worst Results

$$\text{worst} = \min(\text{worst}_1, \text{worst}_2, \ldots, \text{worst}_{\text{runs}}). \tag{16}$$

```
Input: The parameters of JS such as the distribution coefficient $\beta$, and the motion coefficient $\gamma$. People size $N$, dimension $Dim$, and maximum iterations $M_{iter}$.
Output: Optimal fitness value.
    Construct the initial value for the population through logistic chaotic map ($Z_i$).
  While ( $t < M_{iter}$) do
      Calculate the fitness function for an individual population.
      Choose the best location $Z^*$.
      Compute the time control $c(t)$ using equation (6)
      For $i = 1$ to $N$ do
        If $c(t) \geq 0.5$ then
            Jellyfish follows ocean current
            The jellyfish position is updated by equation (11) with fractional-order modified.
        Else
          Jellyfish move inside a swarm
          If $rand(0, 1) \geq (1 - c(t))$ then
            Jellyfish exhibits type passive motion.
            The jellyfish position is updated by equation (12) with fractional-order modified.
          Else
            Jellyfish exhibits type active motion.
            The jellyfish position is updated by equation (4).
          End if
        End if
        Check the boundary of the jellyfish location and calculate the new area.
        Update the location of the jellyfish and the most location $Z^*$
        Gaussian mutation mechanism
          $Z_{i,j}^{\text{new}}(t) = Z^*(t) \times (1 + \theta \times G(0,1))$.
          If $f(Z^{new}(t)) < f(Z^*(t+1))$ then
            update the optimal location to $Z^{new}(t)$.
          End if
      End for
      $t = t + 1$
End while
```

ALGORITHM 1: The proposed FOGJS.

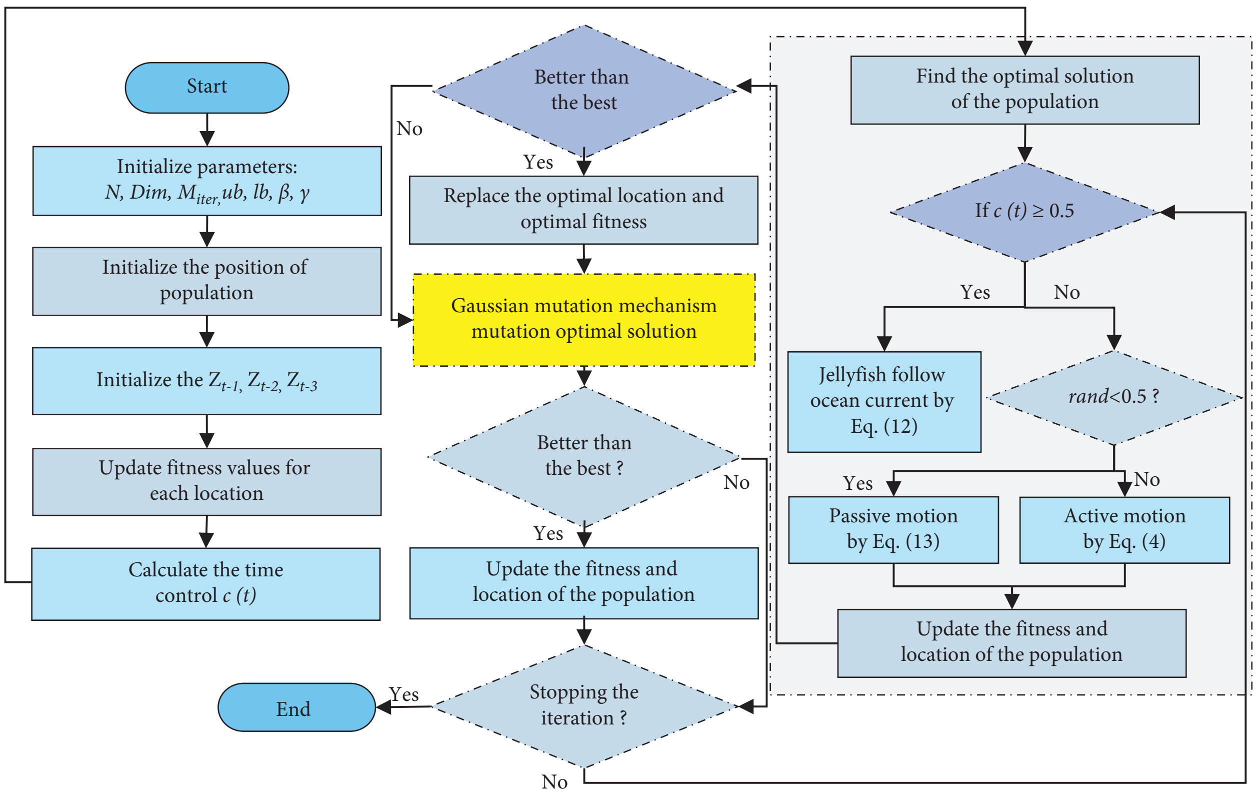

Start
Initialize parameters: N, Dim, M_{iter}, ub, lb, β, γ
Initialize the position of population
Initialize the Z_{t-1}, Z_{t-2}, Z_{t-3}
Update fitness values for each location
Calculate the time control $c(t)$
Better than the best
No
Yes
Replace the optimal location and optimal fitness
Gaussian mutation mechanism mutation optimal solution
Better than the best ?
No
Yes
Update the fitness and location of the population
Stopping the iteration ?
Yes
No
End
Find the optimal solution of the population
If $c(t) \geq 0.5$
Yes
No
Jellyfish follow ocean current by Eq. (12)
$rand<0.5$?
Yes
No
Passive motion by Eq. (13)
Active motion by Eq. (4)
Update the fitness and location of the population

FIGURE 4: Flowchart for the proposed FOGJS optimization algorithm.

Table 1: Parameters setting.

Algorithm	Parameters	Values
AOA [43]	Control parameter	$\mu = 0.499$
	Sensitive parameter	$a = 5$
GSA [44]	ElitistCheck, Rpower, Rnorm, alpha, $G0$	1, 1, 2, 20, 100
PSO [45]	c_1, c_2, v_{Max}, w	2, 2, 6, 1
WOA [46]	A	Decreased from 2 to 0
	b	2
SSA [47]	Leader position update probability	$c3 = 0.5$
GBO [48]	$\beta_{\min}$, $\beta_{\min}$	0.2, 1.2
	pr	0.5
SOA [48]	Control parameter (A)	[2, 0]
	fc	2
JS [36]	β	3
	γ	0.1
LSA [49]	Channel time	10
DE [28]	Scalingfactor, crossoverprobability	0.5, 0.5
WHO [50]	Crossover percentage	PC = 0.13
	Stallions percentage (number of groups)	PS = 0.2
	Crossover	Mean
MVO [29]	WEP_Max, WEP_Min	1, 0.2

4.1.3. *The Average Value (AVE)*

$$\text{Mean} = \frac{1}{\text{Runs}} \sum_{i=1}^{\text{Runs}} \text{best}_i. \tag{17}$$

4.1.4. *The Standard Deviation (STD)*

$$\text{Std} = \sqrt{\frac{1}{\text{runs} - 1} \sum_{i=1}^{\text{runs}} \left(\text{best}_i - \text{Mean}\right)^2}. \tag{18}$$

4.1.5. Wilcoxon's Rank-Sum Test. Wilcoxon's rank-sum test is a nonparametric examination used to test the statistical difference between two sample data sets. Wilcoxon's rank-sum is an effective tool to check whether the FOGJS is significantly better than other comparison algorithms in the general distribution of the examination results.

4.2. Parameter Setting. Algorithms with various characteristics are selected to be compared with the FOGJS in this paper. These algorithms include the original JS and other intelligent optimization algorithms, which are differential evolution (DE) [28], multiverse optimizer (MVO) [29], games search (HGS) [35], arithmetic optimization algorithm (AOA) [43], gravity search algorithm (GSA) [44], particle swarm optimization (PSO) [45], whale optimization algorithm (WOA) [46], sparrow search algorithm (SSA) [47], gradient-based optimizer (GBO) [48], lightning search algorithm (LSA) [49], wild horse optimizer (WHOA) [50], crisscross optimization algorithm (CSO) [51], Harris hawks optimization (HHO) [52], atom search optimization (ASO) [53], Aquila optimizer (AO) [19], heap-based optimizer (HBO) [54], seagull optimization algorithm (SOA) [55], grasshopper optimization algorithm (GOA) [56], and lion swarm optimization (LSO) [57]. Considering that the parameters of algorithms will have an impact on performance, these parameters are set the same as those in the corresponding literature, which are shown in Table 1.

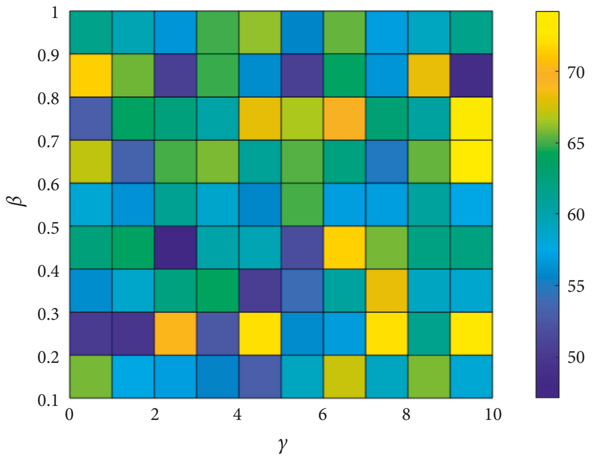
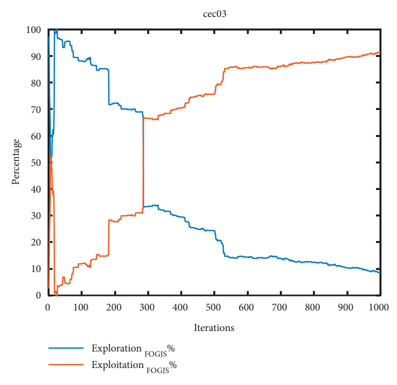
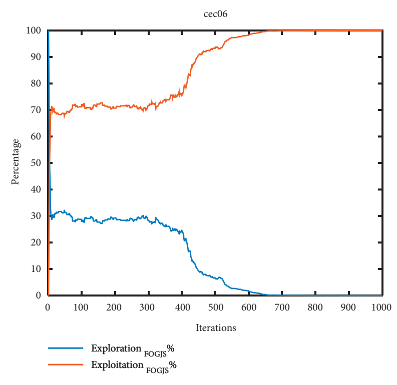


Figure 5: The average rank of FOGJS algorithms with different parameters on cec2017.

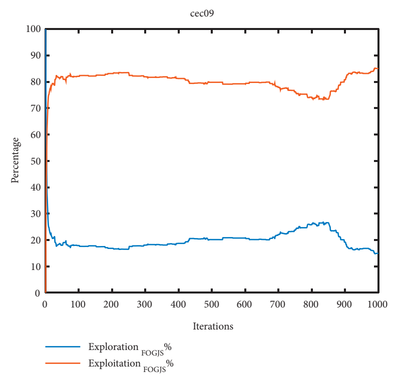
4.3. Sensitivity Analysis of Parameters. Several JS algorithms introduce parameters that affect the performance when working with improvements, such as the distribution coefficient β parameter related to the trend length and the motion coefficient γ parameter associated with the update of the position motion length. A sensitivity analysis of the response to parameter changes is now performed to properly understand the effectiveness of these parameters on improving the FOGJS situation.



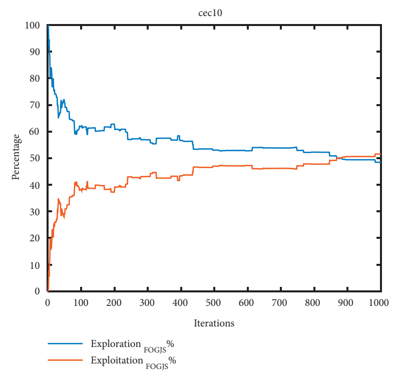
(a)



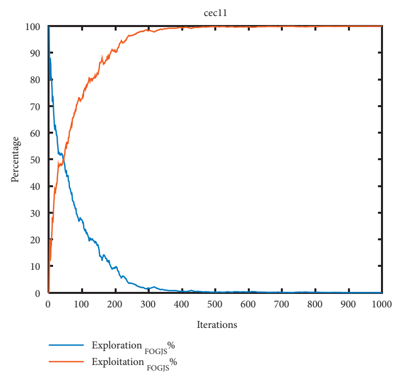
(b)



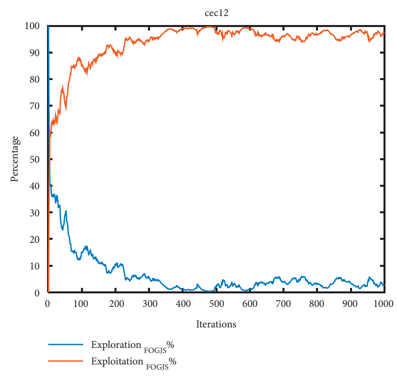
(c)



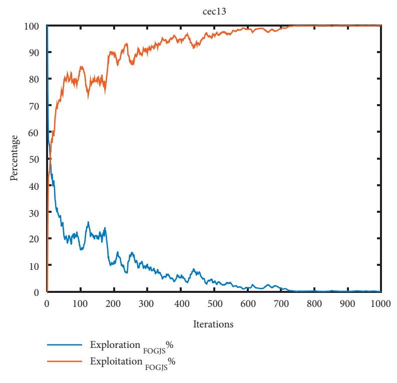
(d)



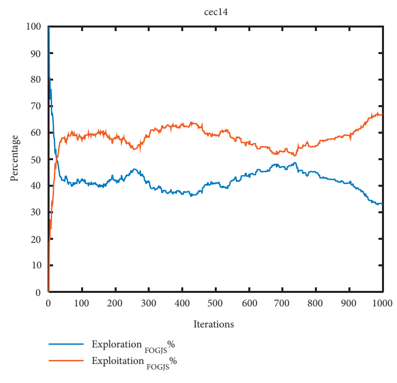
(e)



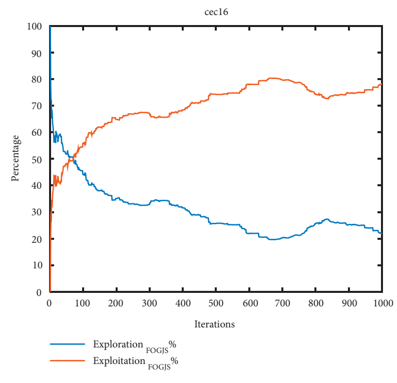
(f)



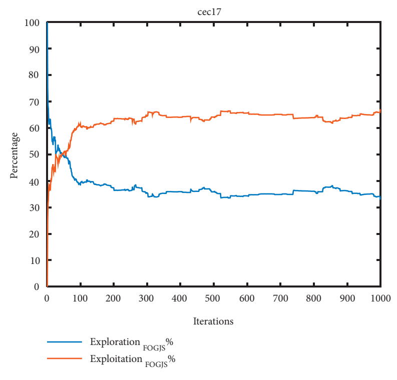
(g)



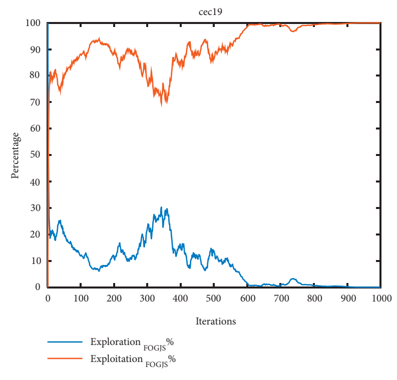
(h)



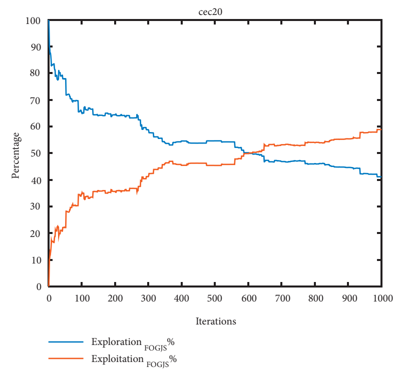
(i)



(j)

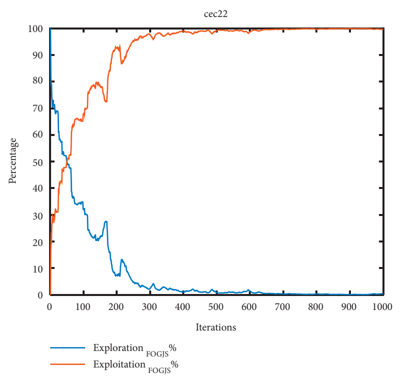


(k)

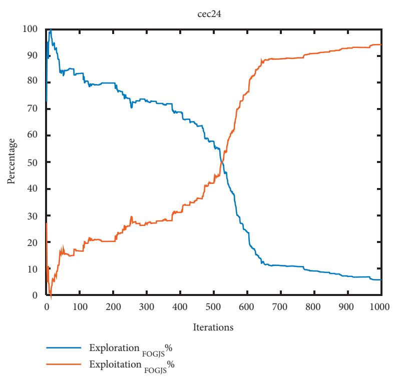


(l)

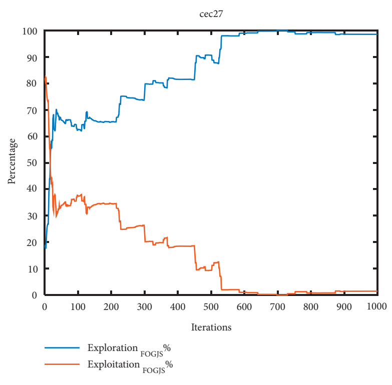
Figure 6: Continued.



(m)



(n)



(o)

Figure 6: The exploration and exploitation diagrams of FOGJS. (a). cec03. (b) cec06. (c) cec09. (d) cec10. (e) cec11. (f) cec12. (g) cec13. (h) cec14. (i) cec16. (j) cec17. (k) cec19. (l) cec20. (m) cec22. (n) cec24. (o) cec27.

The coefficient β was changed from 0.1 to 1 in steps of 0.1. The coefficient γ was changed from 1 to 10 in Step 1. The algorithm was run on 29 cec2017 test functions, where the dimension was set to 30, and the population was set to 30. Figure 5 shows the average rank obtained for each analysis for the two parameters (β, γ). The optimal values of the coefficients include for β equal to 0.4 and γ equal to 2. It is important to note that we set the parameters only when the two strategies of fractional order and Gaussian variation are more effective in improving the JS algorithm. There will be differences with the parameters provided by the JS algorithm. Experiments prove that the FOGJS algorithm is feasible and suitable at $\beta = 0.4$ and $\gamma = 2$.

4.4. Exploration and Exploitation. The distance between individuals in different dimensions and the overall trend can determine whether the whole tends to be divergence or aggregation. When there is a trend of separation, the difference between each individual in multiple dimensions will become more prominent, which means that each individual explores the space in a differentiated way. This trend will make the algorithm do a more comprehensive exploration of the solution space through the temporary characteristics of the population. In addition, when there is a tendency of aggregation, the population is based on a widely recognized partial exploration space, reducing the differentiation of each individual and doing more detailed exploitation of the region. At the same time, maintaining a good balance between these two modes of exploration and exploitation is a necessary guarantee to find the optimal solution. Usually, algorithms with poorer methods fail to produce satisfactory results.

Studying the trend of convergence plots and the statistical mean, best, worst, and standard deviation over multiple runs do not help us understand an algorithm's exploration process, leading to the failure to solve the speed of accuracy problem faced by an algorithm. Therefore, we resorted to the dimensional diversity measure proposed by Hussain et al. in [58], which calculates the variability between individuals and populations in terms of dimensionality by

$$\begin{aligned}\mathrm{Div}_k &= \frac{1}{n}\sum_{k=1}^{n}\mathrm{Median}\left(z^j\right) - z_i^j,\\ \mathrm{Div} &= \frac{1}{\mathrm{Dim}}\sum_{k=1}^{\mathrm{Dim}}\mathrm{Div}_k,\end{aligned} \tag{19}$$

where $Median(z^j)$ is the median of jth dimension in the whole group, $z^j{}_i$ is the jth dimension of the people i, and n is the number of the population. After calculating the average Div_j of each individual, divide the result by Dim to estimate the diversity index of the average dimension.

By obtaining the average diversity of the estimated population after each iteration, we can calculate the exploration and utilization percentage of all iterations according to the following formula:

$$\begin{aligned}\mathrm{Exploration\%} &= \frac{\mathrm{Div}}{\mathrm{Div}_{\max}} \times 100,\\ \mathrm{Exploitation\%} &= \frac{\left|\mathrm{Div} - \mathrm{Div}_{\max}\right|}{\mathrm{Div}_{\max}} \times 100,\end{aligned} \tag{20}$$

where Div is the average diversity of the population in each iteration and Div_{max} is the maximum of the average variety in all iterations. *Exploration*% and *Exploitation*% are the exploration and development percentages of iterations, respectively.

Figure 6 shows the exploration and exploitation diagrams of FOGJS for some of the cec2017 test functions (cec03, cec06, cec09, cec10, cec11, cec12, cec13, cec14, cec16, cec17, cec19, cec20, cec22, cec24, and cec27). From cec03, cec11, cec20, and cec24, we can find that FOGJS has a high exploration rate at the beginning of the iteration and a rapid transition to a high exploitation rate in the middle and finally finishes the iteration with a high exploitation profile. This process indicates that the higher exploration rate in the early stage of FOGJS ensures the global search and prevents getting into the optimal local solution, while the higher exploitation rate in the later stage ensures the higher accuracy of the optimal solution. And in cec10, cec14, cec17, and cec20 test functions, the exploration and exploitation maintain almost the same ratio throughout the iterations, indicating that FOGJS can ensure the balance between the two when handling these test functions. Whereas in cec06, cec09, cec12, cec13, and cec19, the exploitation rate has been dominant throughout the iterative process, and in cec27, the opposite is true.

In summary, FOGJS has different strategies in dealing with varying test functions and therefore has a solid dynamic and adaptable nature. Figure 7 shows the histogram of the percentage of both exploration and exploitation for different test functions. As can be seen, there are 12 test functions with more than 50% exploitation, while at cec10, cec20, and cec27 while more focused on exploration.

4.5. Comparison Results Using cec2017 Test Functions. To further prove the performance of the improved JS algorithm, a challenging set of test functions named cec2017 is selected to be solved. It contains 29 functions, at least half of which are challenging mixed and combined functions. To evaluate the stability of the improved FOGJS performance, these functions are used to test FOGJS. Meanwhile, the FOGJS has achieved 1000 iterations and 30 population sizes for 20 independent operations. The results are compared with other excellent algorithms, including JS [36], GSA [44], MVO [29], WOA [46], GOA [56], LSO [57], HHO [52], ASO [53], AOA [43], and AO [19]. The AVE (average value) and STD (standard deviation) values of the benchmark test bench considered are calculated through the executed algorithm. In addition, to make the data more convincing, the last few rows of the table contain the statistical results of the Wilcoxon rank-sum test, where "+" indicates that the results of other algorithms are better than FOGJS, "−" means the opposite, and "=" suggests that there is no significant contrast between FOGJS and different comparison methods. The best average obtained by 12 algorithms in tables is also drawn in bold.

Table 2 provides the evaluation results of the best solution to date. The average rank of the FOGJS is 2.28, ranking first. Among them, the FOGJS provides the best results of all algorithms in 10 functions and also shows strong competitiveness in other functions. For other algorithms, MVO ranks second, second only to FOGJS, and has successfully implemented 10 functions, but it has poor competitiveness in other functions. JS and CSO successfully implemented 6 and 3 functions, respectively, while GSA, WOA, GOA, LSO, HHO, ASO, AOA, and AO did not show the best performance. In conclusion, the FOGJS is superior to other optimization algorithms in accuracy and occupies the first place. In addition, the ranking of the original JS is 2.69, which is lower than that of the FOGJS. It can be seen that the fractional-order modified strategy and Gaussian mutation mechanism improve the effective exploration ability of the JS in finding the optimal solution. The FOGJS algorithm can search for better solutions with faster convergence rates established on the actual algorithm. The results of the Wilcoxon rank-sum test are shown in the final row of Table 2. The Wilcoxon rank-sum test results of JS, GSA, CSO, MVO, WOA, and GOA are 3/26/0, 1/4/24, 2/12/15, 9/10/10,

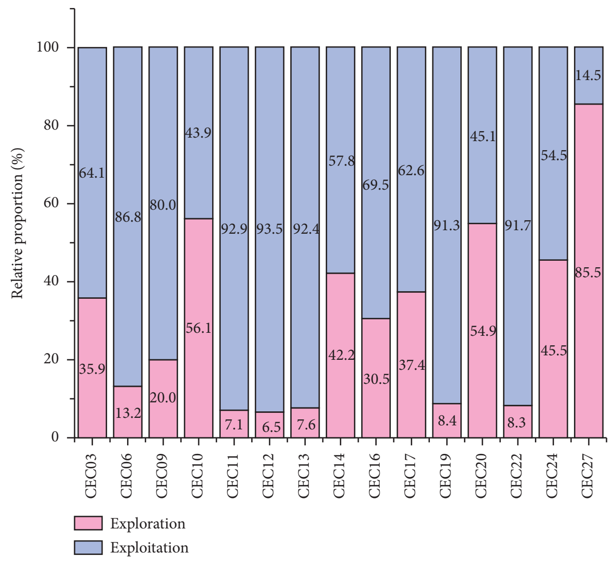


Figure 7: The histogram of the percentage of both exploration and exploitation.

0/2/27, 1/5/23, LSO, HHO, ASO, AOA, and AO are 0/0/29, 1/2/26, 0/2/27, 0/1/28, and 1/4/24, respectively. Figure 8 displays the average convergence process of 29 test functions in 30-dimensions. By analyzing the curve, it can be found that the iterative curve of FOGJS can escape the local solution and connect to the approximate optimal solution in the early phase of iteration, and the optimization will be found near the optimal solution in the later development stage. Specifically, in the 30-dimensional convergence curve, the FOGJS shows that the convergence effects of cec01, cec05, cec16, cec21, and cec29 are more obvious. This observation shows that the FOGJS can be regarded as one of the most reliable algorithms. Figure 9 shows the box plots of different algorithms. Obviously, in most test functions, the box of the FOGJS algorithm is more concentrated, and its target distribution is smaller than other improved algorithms, which shows that the enhanced algorithm has good stability. The results of the Wilcoxon rank-sum test are shown in Table 3. Table 4 shows the number of function evaluations and execution times for each comparison algorithm. It can be found that FOGJS has a longer execution time compared to the original algorithm, which is due to the added improvement strategy. FOGJS has a shorter execution time than the GOA and ASO algorithm but still requires a long running time. NFFEs are the number of equation evaluations of an algorithm, which is another expression of the execution efficiency of an algorithm. JS, GSA, MVO, WOA, ASO, and AOA all have smaller NFFEs, while FOGJS has 60030 NFFEs in one run.

4.6. Comparison Results Using cec2019 Test Functions. This section describes in detail the analysis of FOGJS results when tested with the ten functions of the latest CEC benchmark (cec2019). All results are obtained after 20 independently runs by set the population size as 30 and the iterations as 1000 times. The results were compared with JS

Table 2: Results of FOGJS and different comparison methods on cec2017 test functions.

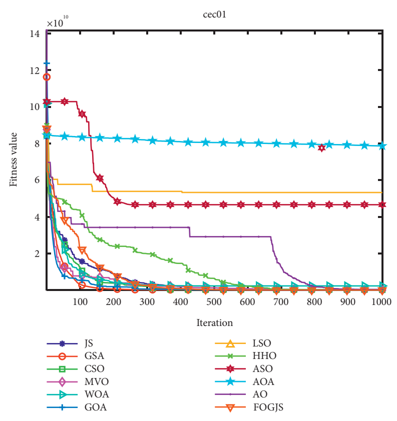
Function	Index	Algorithms											
		JS	GSA	CSO	MVO	WOA	GOA	LSO	HHO	ASO	AOA	AO	FOGJS
	AVE	1.68E+06	3.17E+07	2.13E+08	5.40E+05	1.61E+09	4.82E+08	5.55E+10	2.99E+07	4.93E+10	7.16E+10	6.19E+08	5.80E+05
cec01	STD	2.65E+06	6.36E+07	7.85E+07	2.29E+05	7.47E+08	6.91E+08	9.22E+09	6.42E+06	2.29E+10	7.95E+09	2.82E+08	1.30E+06
	Rank	3	5	6	1	9	7	11	4	10	12	8	2
	AVE	4.74E+04	9.74E+04	8.87E+04	1.68E+03	2.73E+05	6.38E+04	8.47E+04	4.07E+04	3.66E+05	9.19E+04	5.46E+04	3.79E+04
cec03	STD	8.78E+03	1.20E+04	2.72E+04	8.95E+02	6.69E+04	3.38E+04	8.51E+03	6.32E+03	1.17E+05	3.13E+03	8.44E+03	5.46E+03
	Rank	4	10	8	1	11	6	7	3	12	9	5	2
	AVE	5.39E+02	6.62E+02	6.01E+02	4.97E+02	8.48E+02	5.60E+02	1.61E+04	5.60E+02	1.23E+04	2.54E+04	6.57E+02	5.39E+02
cec04	STD	2.60E+01	1.02E+02	8.15E+01	1.37E+01	1.56E+02	9.31E+01	4.27E+03	4.31E+01	7.78E+03	4.93E+03	6.68E+01	2.81E+01
	Rank	2	8	6	1	9	5	11	4	10	12	7	3
	AVE	6.58E+02	7.42E+02	6.78E+02	6.18E+02	8.22E+02	8.28E+02	9.02E+02	7.51E+02	8.14E+02	9.21E+02	7.03E+02	6.71E+02
cec05	STD	2.93E+01	2.76E+01	3.30E+01	4.56E+01	4.56E+01	7.64E+01	3.48E+01	4.05E+01	1.03E+02	2.24E+01	3.33E+01	6.16E+01
	Rank	2	6	4	1	9	10	11	7	8	12	5	3
	AVE	6.23E+02	6.58E+02	6.01E+02	6.32E+02	6.76E+02	6.77E+02	6.82E+02	6.66E+02	6.62E+02	6.67E+02	6.55E+02	6.27E+02
cec06	STD	7.21E+00	3.61E+00	1.09E+00	1.48E+01	1.13E+01	1.17E+01	7.66E+00	5.25E+00	2.57E+01	5.16E+00	7.82E+00	9.51E+00
	Rank	2	6	1	4	10	11	12	8	7	9	5	3
	AVE	9.65E+02	9.61E+02	9.82E+02	8.70E+02	1.28E+03	1.01E+03	1.41E+03	1.28E+03	1.46E+03	1.34E+03	1.10E+03	9.53E+02
cec07	STD	5.18E+01	4.31E+01	5.08E+01	2.71E+01	9.97E+01	1.38E+02	4.69E+01	7.66E+01	4.41E+02	3.45E+01	4.92E+01	6.10E+01
	Rank	4	3	5	1	9	6	11	8	12	10	7	2
	AVE	9.09E+02	9.63E+02	9.36E+02	9.16E+02	1.03E+03	1.08E+03	1.12E+03	9.80E+02	1.10E+03	1.12E+03	9.72E+02	9.37E+02
cec08	STD	1.70E+01	2.51E+01	2.52E+01	2.72E+01	4.21E+01	5.13E+01	1.89E+01	1.97E+01	1.05E+02	3.33E+01	3.00E+01	2.19E+01
	Rank	1	5	3	2	8	9	12	7	10	11	6	4
	AVE	3.65E+03	4.30E+03	3.52E+03	4.01E+03	1.18E+04	1.01E+04	9.98E+03	8.20E+03	1.29E+04	5.92E+03	6.66E+03	4.42E+03
cec09	STD	1.42E+03	6.54E+02	1.15E+03	2.79E+03	3.19E+03	2.01E+03	1.40E+03	1.06E+03	5.39E+03	4.74E+02	1.57E+03	2.31E+03
	Rank	2	4	1	3	11	10	9	8	12	6	7	5
	AVE	7.96E+03	5.45E+03	4.40E+03	4.55E+03	7.00E+03	5.52E+03	8.74E+03	6.12E+03	7.14E+03	7.28E+03	5.64E+03	6.51E+03
cec10	STD	7.11E+02	4.87E+02	4.93E+02	6.70E+02	7.41E+02	7.04E+02	4.59E+02	6.35E+02	7.43E+02	4.25E+02	6.44E+02	1.22E+03
	Rank	11	3	1	2	8	4	12	6	9	10	5	7
	AVE	1.21E+03	4.12E+03	2.40E+03	1.35E+03	7.35E+03	1.89E+03	1.14E+04	1.31E+03	2.47E+04	1.15E+04	2.50E+03	1.21E+03
cec11	STD	4.19E+01	9.40E+02	2.14E+03	4.61E+01	3.28E+03	7.22E+02	3.55E+03	4.51E+01	1.76E+04	3.38E+03	8.16E+02	3.52E+01
	Rank	2	8	6	4	9	5	10	3	12	11	7	1
	AVE	1.89E+06	9.50E+07	1.66E+06	1.36E+07	2.72E+08	1.92E+08	1.24E+10	3.00E+07	8.19E+09	2.19E+10	6.26E+07	1.58E+06
cec12	STD	1.58E+06	1.13E+08	1.73E+06	1.16E+07	1.44E+08	7.09E+08	4.23E+09	1.82E+07	5.57E+09	4.08E+09	5.53E+07	1.42E+06
	Rank	3	7	2	4	9	8	11	5	10	12	6	1
	AVE	8.20E+03	3.39E+04	3.37E+05	1.26E+05	2.14E+06	1.15E+05	5.58E+09	5.61E+05	8.50E+09	2.32E+10	1.46E+06	5.90E+03
cec13	STD	7.39E+03	1.03E+04	8.82E+05	5.71E+04	1.69E+06	6.57E+04	4.20E+09	2.09E+05	5.02E+09	8.00E+09	2.51E+06	4.64E+03
	Rank	2	3	6	5	9	4	10	7	11	12	8	1
	AVE	1.94E+04	1.07E+06	2.00E+06	2.10E+04	2.10E+06	7.61E+04	5.94E+06	8.01E+05	1.32E+07	3.72E+06	8.63E+05	1.74E+04
cec14	STD	2.38E+04	4.81E+05	2.17E+06	2.08E+04	3.04E+06	9.07E+04	5.38E+06	9.98E+05	1.10E+07	3.27E+06	9.52E+05	1.92E+04
	Rank	2	7	8	3	9	4	11	5	12	10	6	1

Table 2: Continued.

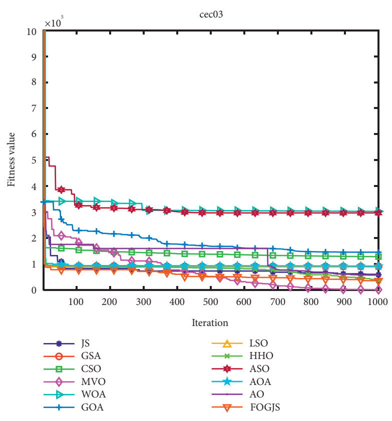
Function	Index	Algorithms											
		JS	GSA	CSO	MVO	WOA	GOA	LSO	HHO	ASO	AOA	AO	FOGJS
cec15	AVE	3.99E+03	1.65E+04	8.85E+03	4.94E+04	2.69E+06	8.04E+04	4.81E+08	9.60E+04	1.67E+09	4.65E+08	1.10E+05	4.57E+03
	STD	2.76E+03	5.06E+03	9.96E+03	3.84E+04	4.51E+06	8.43E+04	3.44E+08	8.92E+04	1.26E+09	4.38E+08	5.92E+04	2.27E+03
	Rank	1	4	3	5	9	6	11	7	12	10	8	2
cec16	AVE	2.55E+03	3.54E+03	2.99E+03	2.65E+03	4.06E+03	3.26E+03	5.05E+03	3.53E+03	4.41E+03	7.12E+03	3.41E+03	2.77E+03
	STD	3.09E+02	4.10E+02	3.21E+02	2.95E+02	5.67E+02	5.22E+02	8.08E+02	3.42E+02	5.33E+02	2.57E+03	4.33E+02	2.56E+02
	Rank	1	8	4	2	9	5	11	7	10	12	6	3
cec17	AVE	2.05E+03	2.77E+03	2.58E+03	2.08E+03	2.72E+03	2.60E+03	3.89E+03	2.69E+03	4.04E+03	9.59E+03	2.48E+03	2.02E+03
	STD	1.31E+02	2.66E+02	2.63E+02	1.24E+02	3.10E+02	2.61E+02	1.08E+03	2.65E+02	8.24E+02	1.40E+04	3.02E+02	1.67E+02
	Rank	2	9	5	3	8	6	10	7	11	12	4	1
cec18	AVE	2.38E+05	4.55E+05	2.51E+06	4.32E+05	1.13E+07	1.25E+06	4.73E+07	2.63E+06	3.35E+07	7.00E+07	4.19E+06	3.78E+05
	STD	1.97E+05	3.96E+05	2.28E+06	2.82E+05	1.16E+07	1.43E+06	4.72E+07	2.74E+06	2.79E+07	6.74E+07	4.75E+06	3.21E+05
	Rank	1	4	6	3	9	5	11	7	10	12	8	2
cec19	AVE	5.65E+03	1.63E+05	9.55E+03	1.13E+06	1.10E+07	3.09E+06	8.91E+08	1.22E+06	1.48E+09	6.38E+08	2.13E+06	4.90E+03
	STD	3.26E+03	1.93E+05	5.79E+03	7.32E+05	9.76E+06	2.12E+06	6.30E+08	1.22E+06	1.63E+09	4.82E+08	1.97E+06	2.09E+03
	Rank	2	4	3	5	9	8	11	6	12	10	7	1
cec20	AVE	2.32E+03	3.06E+03	2.75E+03	2.54E+03	2.85E+03	2.93E+03	2.83E+03	2.82E+03	3.07E+03	2.92E+03	2.59E+03	2.41E+03
	STD	5.94E+01	2.54E+02	1.82E+02	1.81E+02	2.28E+02	2.39E+02	1.63E+02	2.08E+02	1.59E+02	1.76E+02	1.62E+02	1.14E+02
	Rank	1	11	5	3	8	10	7	6	12	9	4	2
cec21	AVE	2.41E+03	2.62E+03	2.47E+03	2.41E+03	2.65E+03	2.65E+03	2.72E+03	2.57E+03	2.62E+03	2.72E+03	2.47E+03	2.40E+03
	STD	2.20E+01	3.23E+01	2.73E+01	2.91E+01	5.90E+01	8.14E+01	7.78E+01	5.72E+01	1.08E+02	4.20E+01	4.77E+01	1.86E+01
	Rank	3	7	4	2	10	9	11	6	8	12	5	1
cec22	AVE	2.33E+03	7.21E+03	4.19E+03	4.91E+03	7.86E+03	6.81E+03	8.77E+03	6.78E+03	8.69E+03	9.23E+03	2.52E+03	2.31E+03
	STD	2.32E+01	3.84E+02	2.14E+03	1.60E+03	1.71E+03	8.03E+02	1.10E+03	1.79E+03	7.16E+02	5.79E+02	7.75E+01	1.80E+01
	Rank	2	8	4	5	9	7	11	6	10	12	3	1
cec23	AVE	2.81E+03	3.80E+03	2.83E+03	2.76E+03	3.11E+03	3.24E+03	3.48E+03	3.22E+03	2.99E+03	3.52E+03	2.96E+03	2.78E+03
	STD	5.06E+01	1.55E+02	4.98E+01	2.99E+01	7.35E+01	9.98E+01	1.59E+02	1.19E+02	6.72E+01	1.63E+02	5.99E+01	3.59E+01
	Rank	3	12	4	1	7	9	10	8	6	11	5	2
cec24	AVE	2.97E+03	3.55E+03	3.13E+03	2.93E+03	3.23E+03	3.25E+03	3.63E+03	3.41E+03	3.08E+03	4.00E+03	3.09E+03	3.01E+03
	STD	2.89E+01	1.58E+02	1.12E+02	3.78E+01	1.11E+02	1.25E+02	1.82E+02	1.09E+02	4.67E+01	2.25E+02	5.16E+01	1.15E+02
	Rank	2	10	6	1	7	8	11	9	4	12	5	3
cec25	AVE	2.93E+03	2.99E+03	2.99E+03	2.90E+03	3.09E+03	2.93E+03	5.64E+03	2.94E+03	5.14E+03	7.89E+03	3.00E+03	2.91E+03
	STD	2.58E+01	2.68E+01	3.46E+01	1.80E+01	5.19E+01	4.53E+01	6.17E+02	2.49E+01	2.27E+03	6.65E+02	2.42E+01	1.55E+01
	Rank	4	7	6	1	9	3	11	5	10	12	8	2
cec26	AVE	5.52E+03	8.08E+03	5.00E+03	4.78E+03	7.77E+03	7.33E+03	1.13E+04	7.66E+03	7.93E+03	1.25E+04	4.97E+03	5.67E+03
	STD	1.35E+03	6.47E+02	1.74E+03	2.90E+02	1.20E+03	1.73E+03	7.97E+02	7.55E+02	4.99E+02	1.44E+03	1.28E+03	1.63E+03
	Rank	4	10	3	1	8	6	11	7	9	12	2	5
cec27	AVE	3.30E+03	5.13E+03	3.27E+03	3.23E+03	3.46E+03	3.36E+03	4.05E+03	3.50E+03	3.35E+03	4.79E+03	3.39E+03	3.20E+03
	STD	2.79E+01	3.90E+02	2.52E+01	1.44E+01	1.16E+02	1.66E+02	3.78E+02	1.15E+02	3.92E+01	6.27E+02	7.24E+01	1.15E-04
	Rank	4	12	3	2	8	6	10	9	5	11	7	1
cec28	AVE	3.35E+03	3.58E+03	3.32E+03	3.25E+03	3.60E+03	3.36E+03	6.94E+03	3.33E+03	6.63E+03	8.71E+03	3.47E+03	3.31E+03
	STD	4.36E+01	2.04E+02	5.87E+01	4.26E+01	9.05E+01	3.54E+02	8.10E+02	3.02E+01	2.61E+02	8.66E+02	8.74E+01	2.49E+01
	Rank	5	8	3	1	9	6	11	4	10	12	7	2

Table 2: Continued.

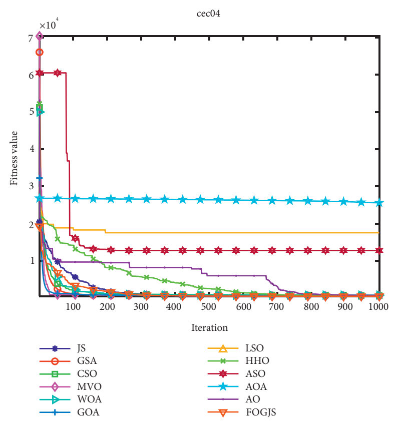
Function	Index	Algorithms											
		JS	GSA	CSO	MVO	WOA	GOA	LSO	HHO	ASO	AOA	AO	FOGJS
cec29	AVE	$3.83E+03$	$5.20E+03$	$4.01E+03$	$3.96E+03$	$5.21E+03$	$4.61E+03$	$7.89E+03$	$4.61E+03$	$5.14E+03$	$1.15E+04$	$4.53E+03$	$3.91E+03$
	STD	$2.40E+02$	$3.57E+02$	$1.53E+02$	$2.80E+02$	$4.99E+02$	$3.26E+02$	$1.69E+03$	$2.53E+02$	$4.81E+02$	$8.77E+03$	$3.38E+02$	$2.70E+02$
	Rank	1	9	4	3	10	6	11	7	8	12	5	2
cec30	AVE	$1.38E+04$	$2.60E+06$	$1.01E+06$	$4.24E+06$	$4.55E+07$	$6.65E+06$	$1.06E+09$	$4.85E+06$	$1.32E+09$	$3.02E+09$	$1.14E+07$	$1.19E+04$
	STD	$6.07E+03$	$1.37E+06$	$2.78E+06$	$3.53E+06$	$4.24E+07$	$5.50E+06$	$1.26E+09$	$2.36E+06$	$1.15E+09$	$1.54E+09$	$6.51E+06$	$4.76E+03$
	Rank	2	4	3	5	9	7	10	6	11	12	8	1
Mean rank		2.69	6.97	4.24	2.59	8.90	6.76	10.55	6.28	9.76	11.00	6.00	2.28
Final ranking		3	8	4	2	9	7	11	6	10	12	5	1



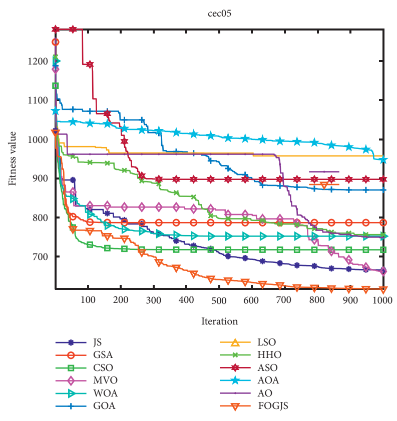
(a)



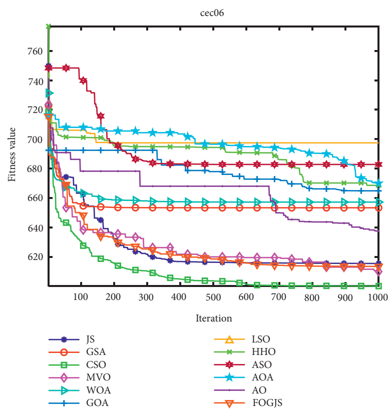
(b)



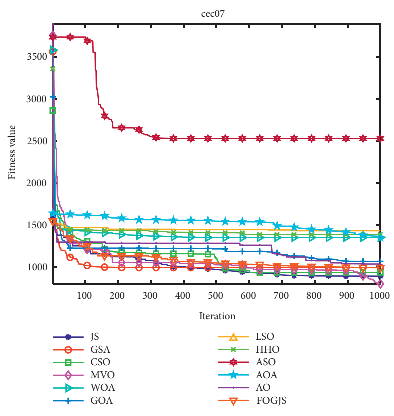
(c)



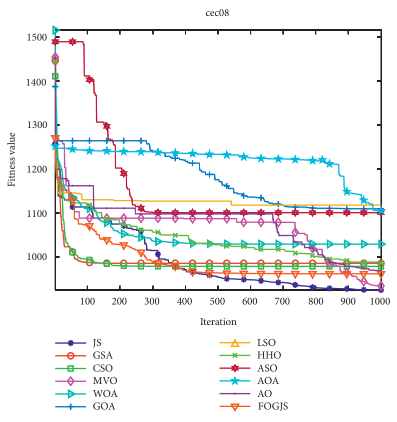
(d)



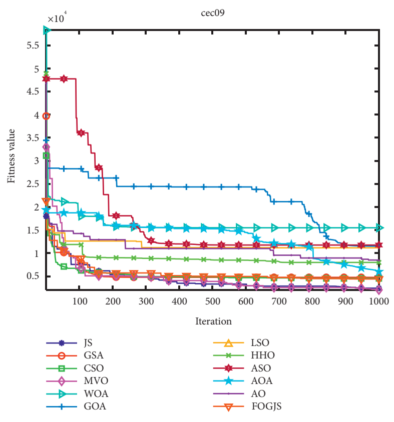
(e)



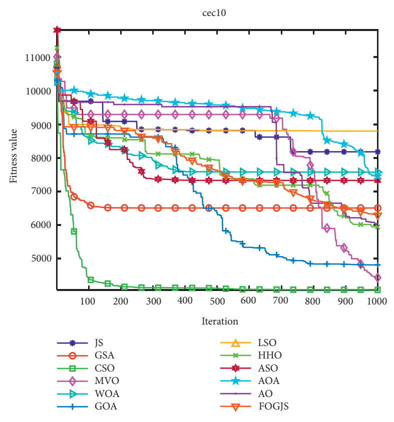
(f)



(g)

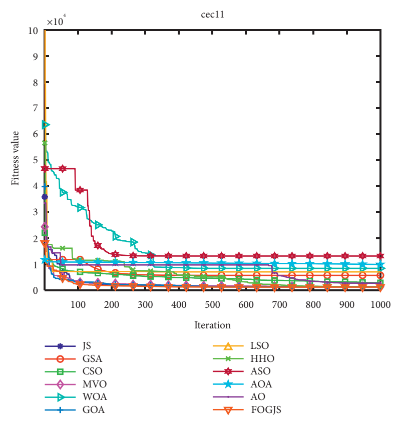


(h)

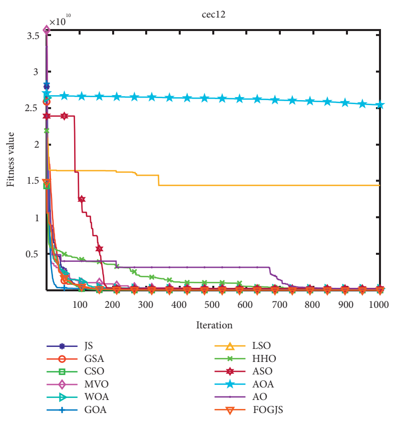


(i)

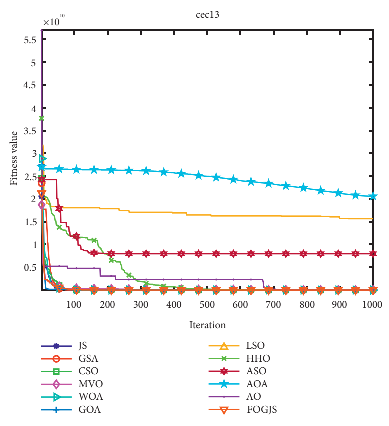
Figure 8: Continued.



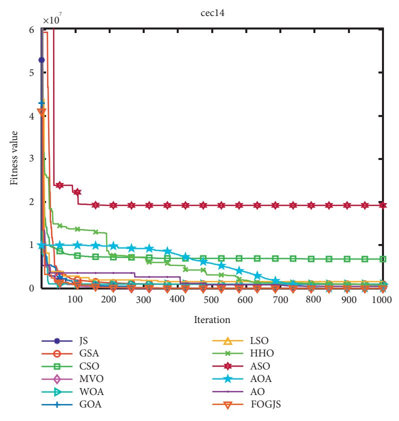
(j)



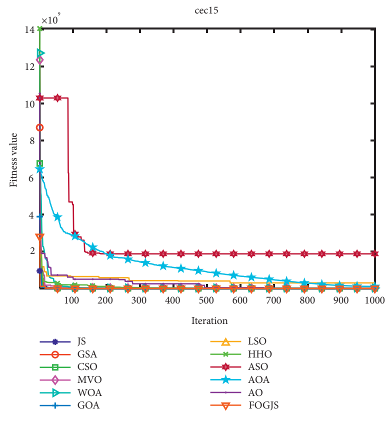
(k)



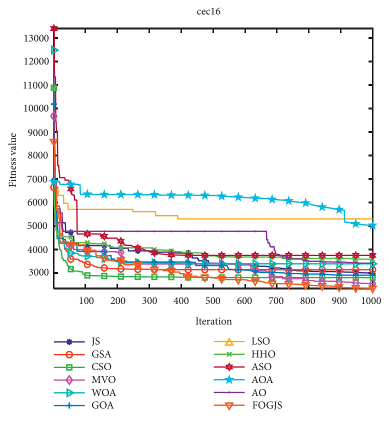
(l)



(m)

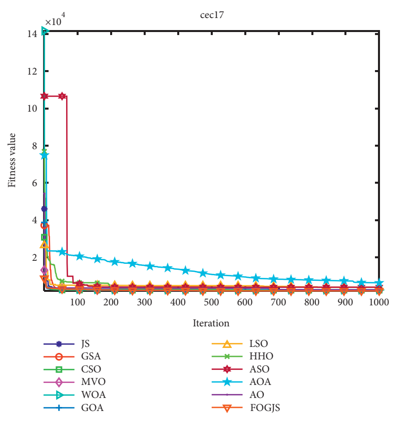


(n)

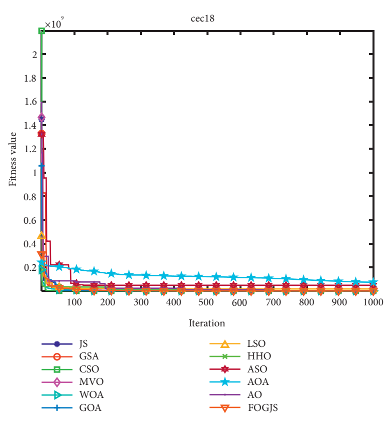


(o)

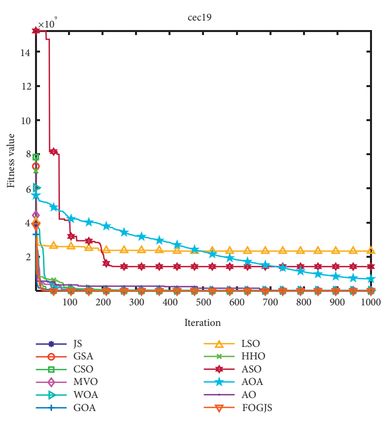
Figure 8: Continued.



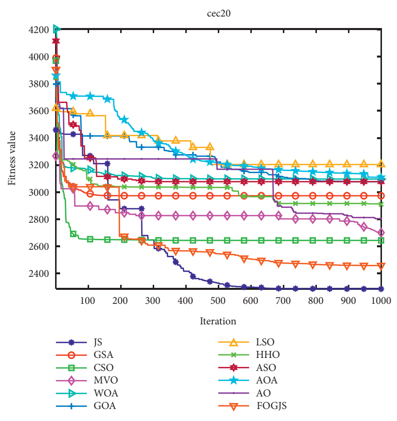
(p)



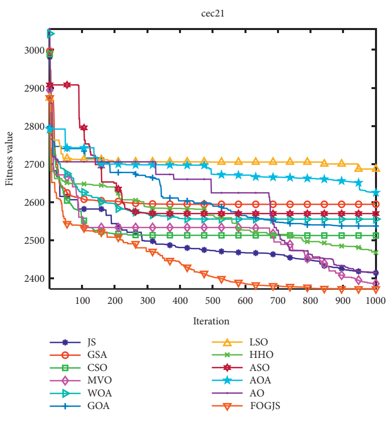
(q)



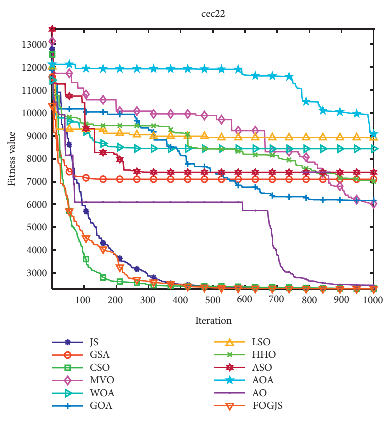
(r)



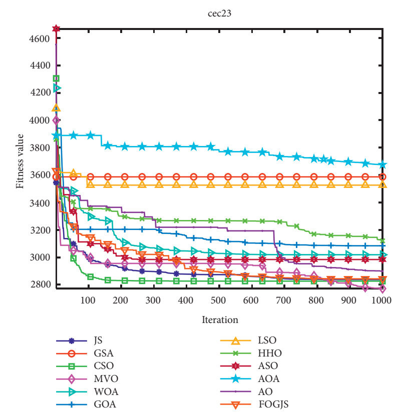
(s)



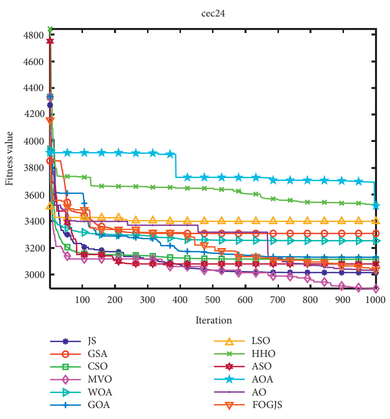
(t)



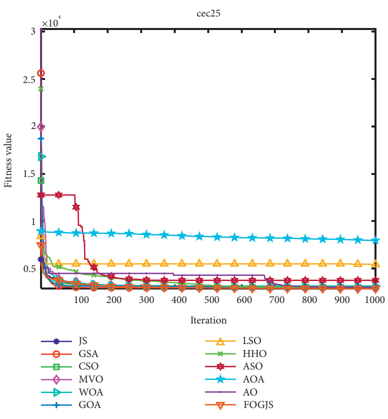
(u)



(v)

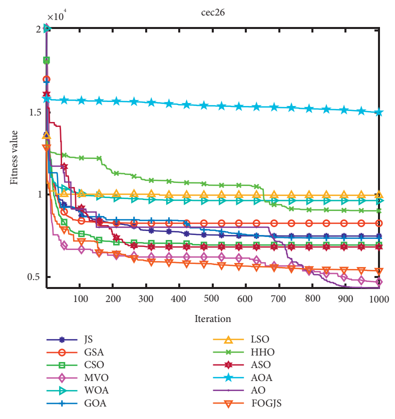


(w)

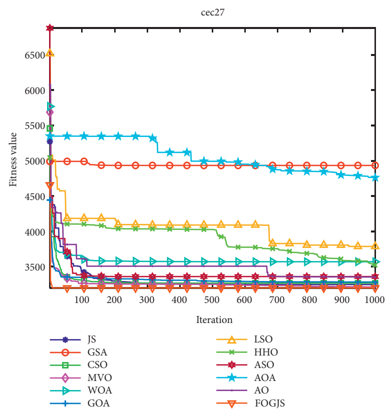


(x)

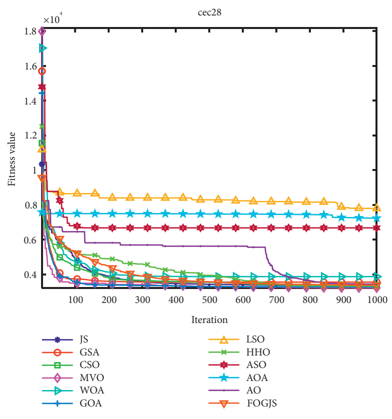
Figure 8: Continued.



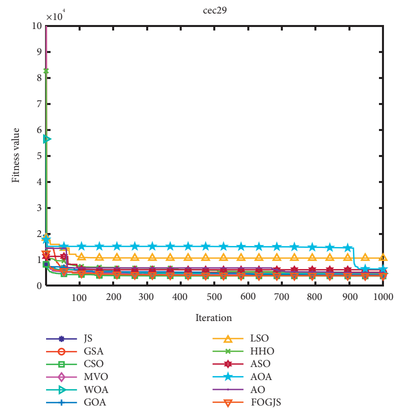
(y)



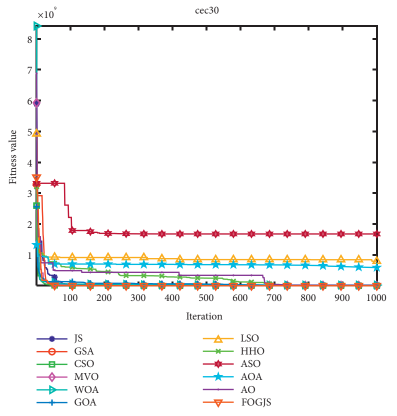
(z)



(z.1)



(z.2)



(z.3)

Figure 8: The convergence curves of the FOGJS and other comparison algorithms on cec2017 functions. (a) cec01. (b) cec03. (c) cec04. (d) cec05. (e) cec06. (f) cec06. (g) cec08. (h) cec09. (i) cec10. (j) cec11. (k) cec12. (l) cec13. (m) cec14. (n) cec15. (o) cec16. (p) cec17. (q) cec18. (r) cec19. (s) cec20. (t) cec21. (u) cec22. (v) cec23. (w) cec24. (x) cec25. (y) cec26. (z) cec27. (z.1) cec28. (z.2) cec29. (z.3) cec30.

[36], PSO [45], DE [28], LSA [49], GBO [48], SOA [55], HGS [35], SSA [47], HBO [54], WHOA [50], and AOA [43]. As shown in Table 5, in the whole process of the function, the worst value, average value, the best value, and the STD value are compared through the considered algorithm. In addition, the results of the Wilcoxon rank-sum test are shown in Table 6. According to the data results in Table 5, the average rank of the FOGJS is 2.8, ranking first. FOGJS provides the best results in cec01, cec02, cec08, and cec10. Compared with other algorithms, FOGJS is also very competitive in other test functions. WHOA ranks second, ranking worse than FOGJS. At the same time, it successfully implements one function and shows good competitiveness in other functions. For other comparison functions, HBO successfully implemented four functions, while SSA and HGS successfully implemented one function. The results show that the improved FOGJS has better convergence accuracy and accuracy than other algorithms. The last row of Table 5 gives the average and rank ranking. The Wilcoxon rank-sum test results of JS, PSO, DE, GBO, LSA, and SOA algorithms are 1/5/4, 0/2/8, 0/0/10, 2/3/5, 1/4/5, and 0/0/10, respectively. The Wilcoxon rank-sum test results of SSA, HGS, HBO, WHOA, and AOA are 0/5/5, 0/6/4, 4/2/4, 1/5/4, and 0/0/10, respectively. Figure 10 shows the convergence curve of 10 test functions. FOGJS can get closer to the optimal solution faster than other algorithms by analyzing the curve. At the same time, it will find the optimization near the optimal solution in the middle development stage of iteration and replace the original solution with the newly found better solution. Specifically, FOGJS shows that the convergence effects of cec01, cec02, cec04, cec06, and cec08 are apparent in the convergence curve. Figure 11 shows the box plot of 12 algorithms in 10 test functions. We can find that the length of the FOGJS box is small, and there are few outliers when combining the results of 20 runs, thus showing the stability and balance of FOGJS. Table 7 shows the number of function evaluations and execution times for each comparison algorithm.

5. Income Forecast Model of Rural Resident Based on Improved Jellyfish Search Optimizer

Agriculture, rural areas, and farmers are important issues for the long-term stability of the country. In order to adopt a series of corresponding policies to benefit and support

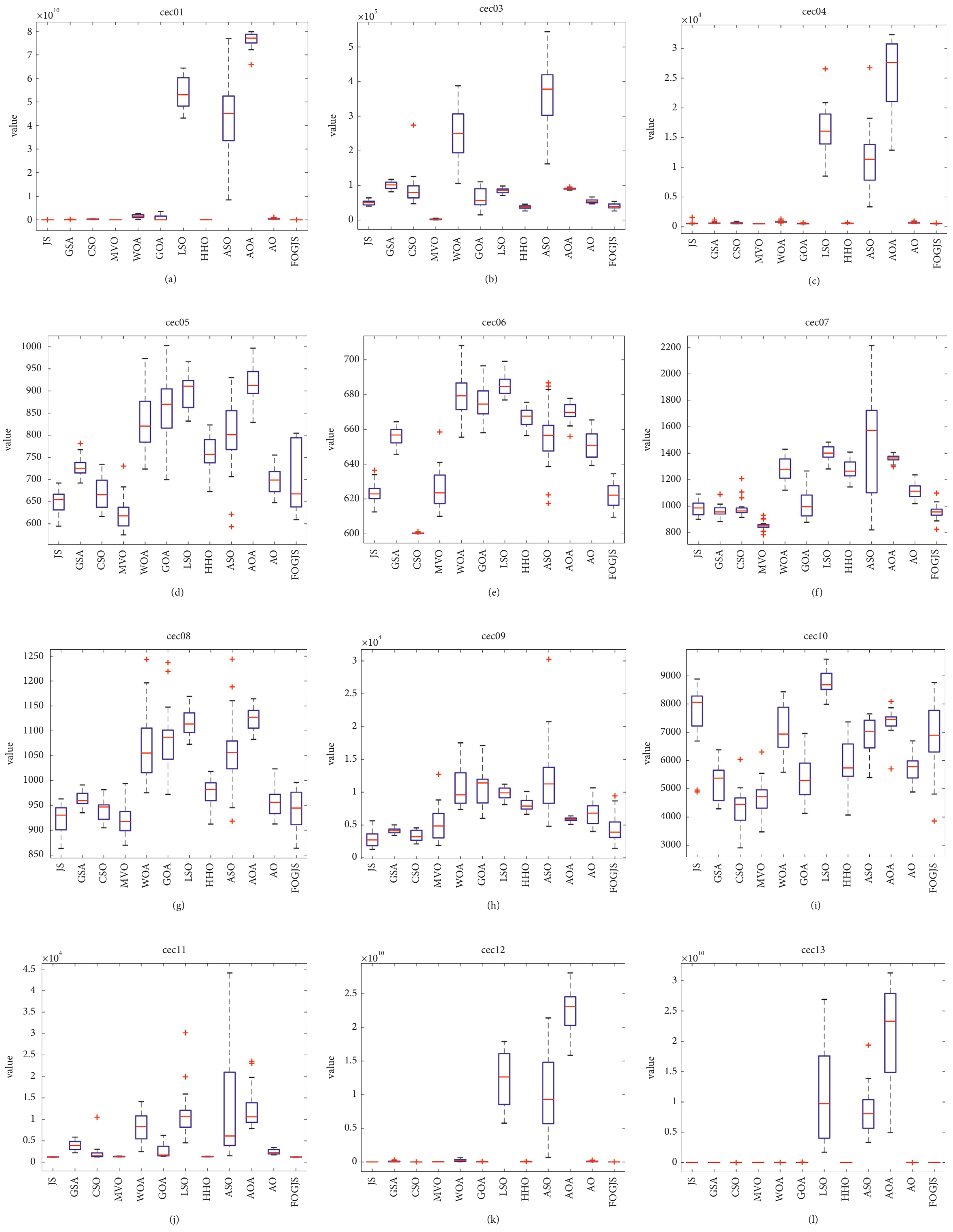


Figure 9: Continued.

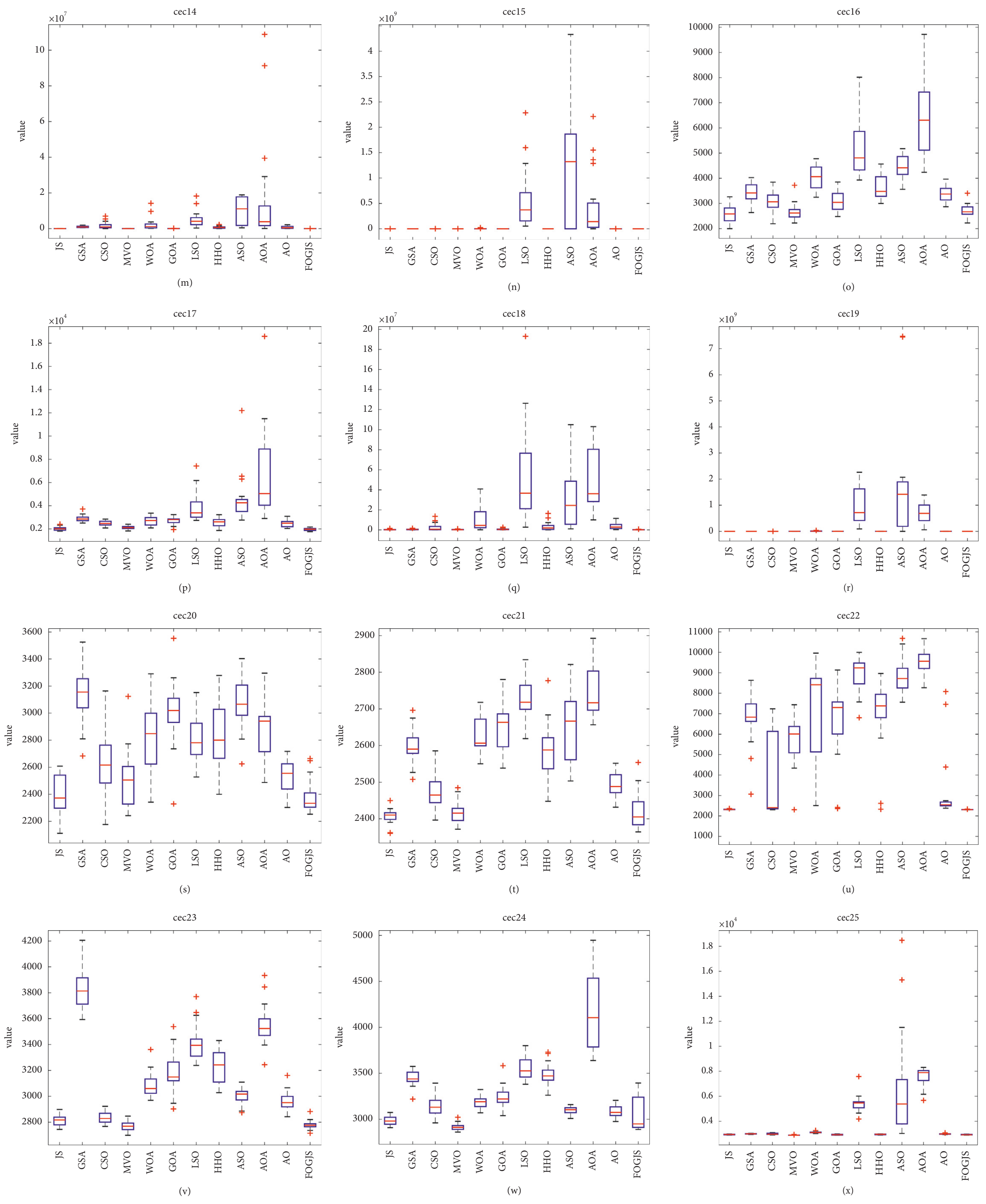


Figure 9: Continued.

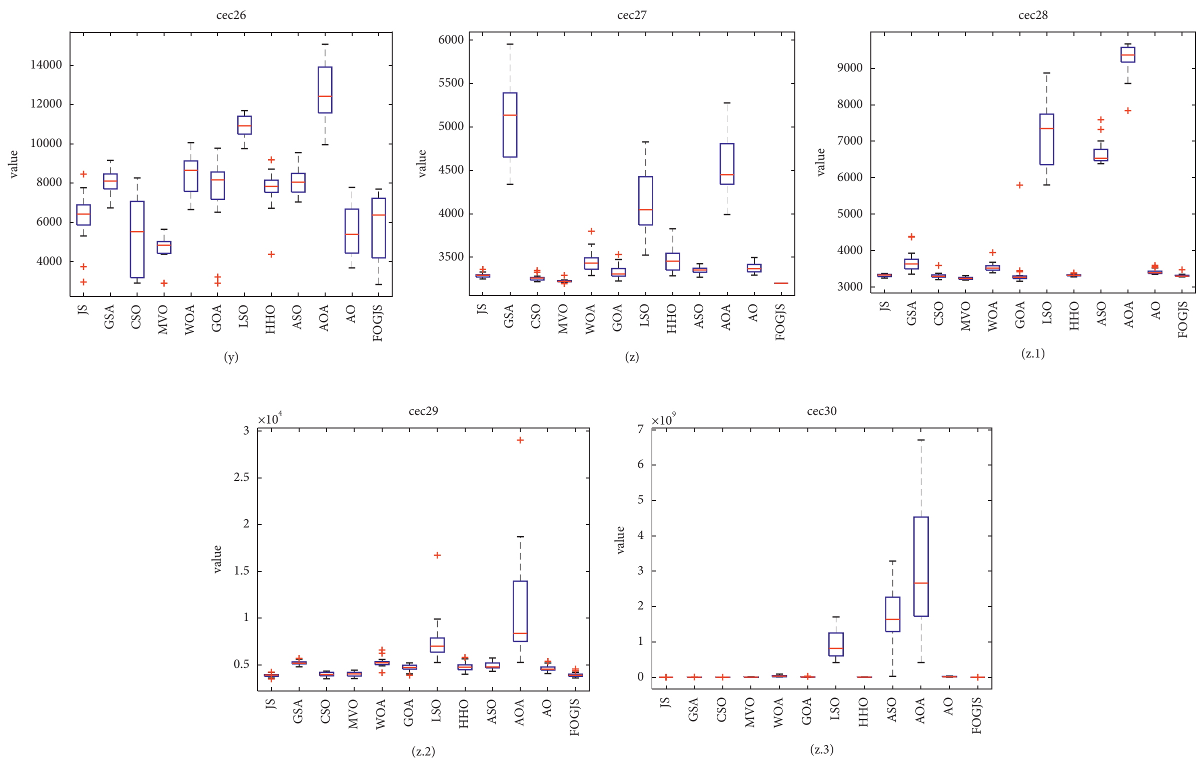


Figure 9: Box plot of the FOGJS algorithm and other algorithms on cec2017 functions (red minus $\longrightarrow$ median; red plus $\longrightarrow$ outlier). (a) Boxplot of cec01. (b) Boxplot of cec03. (c) Boxplot of cec04. (d) Boxplot of cec05. (e) Boxplot of cec06. (f) Boxplot of cec07. (g) Boxplot of cec08. (h) Boxplot of cec09. (i) Boxplot of cec10. (j) Boxplot of cec11. (k) Boxplot of cec12. (l) Boxplot of cec13. (m) Boxplot of cec14. (n) Boxplot of cec15. (o) Boxplot of cec16. (p) Boxplot of cec17. (q) Boxplot of cec18. (r) Boxplot of cec19. (s) Boxplot of cec20. (t) Boxplot of cec21. (u) Boxplot of cec22. (v) Boxplot of cec23. (w) Boxplot of cec24. (x) Boxplot of cec25. (y) Boxplot of cec26. (z) Boxplot of cec27. (z.1) Boxplot of cec28. (z.2) Boxplot of cec29. (z.3) Boxplot of ce30.

agriculture, the forecast of farmers' income trend becomes the task need to be solve. Disposable income represents the sum of the ultimate consumer spending and savings obtained by residents, which can play an important role in estimating per capita consumption power and understanding the productivity of an area. Thus, this paper discusses the per capita disposable income of rural residents in Shaanxi Province. For ease to be understood, per capita disposable income of rural resident in this paper is simply referred to as income of rural resident. As Figure 12 shows, income of rural resident in Shaanxi Province is increasing from 1989 to 2020. In this section, a novel discrete fractional time-delayed grey model is established to solve the problem of income forecast.

5.1. Income Forecast Model Based on Improved Jellyfish Search Optimizer

5.1.1. Establishment of the TDFTDGM Model. Grey model (GM) is a popular predicted approach by establishing a grey differential prediction model through a small amount of incomplete information and the development trend of the internal system is described, which has been widely applied to population forecast, economy forecast, and climate prediction. In [16], a fractional time-delayed grey model was proposed. However, the conversion from the discrete equation to the continuous equation will bring conversion error, which will decrease the accuracy of prediction. Thus, this paper establishes a novel discrete fractional time-delayed grey model with triangular residual correction (TDFTDGM).

Firstly, given a raw nonnegative data set $Z^{(0)} = (z^{(0)}(1), z^{(0)}(2), \cdots, z^{(0)}(n))$. Then, calculate the m-order accumulated sequence $Z^{(m)} = (z^{(m)}(1), z^{(m)}(2), \cdots, z^{(m)}(n))$ by

$$z^{(m)}(k) = \sum_{i=1}^{k} \binom{k-i+m-1}{k-i} z^{(0)}(i), k = 1, 2, \cdots, n, \quad (21)$$

where $\binom{k-i+m-1}{k-i} = (k-i+m-1)(k-i+m-2)\cdots(m+1)m/(k-i)!$.

Differential equation (23) is the basic fractional grey model (FGM) with order m:

Table 3: The p value of the Wilcoxon rank-sum test on cec2017 test functions.

Functions	JS	GSA	CSO	MVO	WOA	GOA	LSO	HHO	ASO	AOA	AO
CEC01	1.18E − 02/−	1.00E + 00/=	1.91E − 06/−	4.14E − 02/+	1.91E − 06/−	1.15E − 01/=	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC03	4.02E − 04/−	1.91E − 06/−	4.01E − 05/−	1.91E − 06/+	1.91E − 06/−	1.18E − 02	1.91E − 06/−	8.24E − 01/=	1.91E − 06/−	1.91E − 06/−	4.01E − 05/−
CEC04	8.24E − 01/=	2.58E − 03/−	4.14E − 02/−	2.58E − 03/+	1.91E − 06/−	8.24E − 01/=	1.91E − 06/−	1.15E − 01/=	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC05	5.03E − 01/=	5.03E − 01/=	1.15E − 01/=	1.18E − 02/+	1.18E − 02/−	1.91E − 06/−	1.91E − 06/−	4.14E − 02/−	1.18E − 02/−	1.91E − 06/−	2.63E − 01/=
CEC06	1.00E + 00/=	1.91E − 06/−	1.91E − 06/+	1.00E + 00/=	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC07	5.03E − 01/=	1.00E + 00/=	5.03E − 01/=	4.02E − 04/+	1.91E − 06/−	5.03E − 01/=	1.91E − 06/−	1.91E − 06/−	4.02E − 04/−	1.91E − 06/−	1.91E − 06/−
CEC08	1.15E − 01/=	4.14E − 02/−	5.03E − 01/=	4.14E − 02/+	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	4.14E − 02/−	4.01E − 05/−	1.91E − 06/−	5.03E − 01/=
CEC09	1.15E − 01/=	5.03E − 01/=	2.63E − 01/=	8.24E − 01/=	4.01E − 05/−	4.02E − 04/−	4.01E − 05/−	4.02E − 04/−	4.01E − 05/−	1.18E − 02/−	4.14E − 02/−
CEC10	1.15E − 01/=	4.02E − 04/+	4.01E − 05/+	1.91E − 06/+	8.24E − 01/=	4.02E − 04/+	4.01E − 05	1.18E − 02/+	1.00E + 00/=	2.63E − 01/=	2.58E − 03/+
CEC11	2.63E − 01/=	1.91E − 06/−	1.91E − 06/−	4.01E − 05/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	4.01E − 05/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC12	5.03E − 01/=	1.91E − 06/−	8.24E − 01/=	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC13	8.24E − 01/=	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC14	8.24E − 01/=	1.91E − 06/−	1.91E − 06/−	1.15E − 01/=	1.91E − 06/−	4.14E − 02/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC15	2.63E − 01/=	1.91E − 06/−	8.24E − 01/=	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC16	2.63E − 01/=	4.02E − 04/−	4.14E − 02/−	5.03E − 01/=	1.91E − 06/−	1.18E − 02/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	4.01E − 05/−
CEC17	2.63E − 01/=	1.91E − 06/−	1.91E − 06/−	4.02E − 04/−	1.91E − 06/−	4.01E − 05/−	1.91E − 06/−	4.01E − 05/−	1.91E − 06/−	1.91E − 06/−	4.01E − 05/−
CEC18	8.24E − 01/=	4.14E − 02/−	4.02E − 04/−	1.18E − 02/−	4.01E − 05/−	4.02E − 04/−	1.91E − 06/−	2.58E − 03/−	1.91E − 06/−	1.91E − 06/−	4.01E − 05/−
CEC19	8.24E − 01/=	1.91E − 06/−	2.63E − 01/=	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC20	5.03E − 01/=	1.91E − 06/−	1.18E − 02/−	5.03E − 01/=	1.91E − 06/−	4.01E − 05/−	1.91E − 06/−	4.01E − 05/−	1.91E − 06/−	1.91E − 06/−	2.58E − 03/−
CEC21	1.00E + 00/=	1.91E − 06/−	1.18E − 02/−	5.03E − 01/=	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	2.58E − 03/−
CEC22	1.15E − 01/=	1.91E − 06/−	4.01E − 05/−	4.01E − 05/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC23	1.15E − 01/=	1.91E − 06/−	4.01E − 05/−	5.03E − 01/=	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC24	1.00E + 00/=	4.01E − 05/−	2.63E − 01/=	8.24E − 01/=	2.63E − 01/=	1.15E − 01/=	1.91E − 06/−	1.91E − 06/−	2.63E − 01/=	1.91E − 06/−	5.03E − 01/=
CEC25	8.24E − 01/=	1.91E − 06/−	4.02E − 04/−	2.58E − 03/+	1.91E − 06/−	1.00E + 00/=	1.91E − 06/−	4.14E − 02/−	1.91E − 06/−	1.91E − 06/−	4.01E − 05/−
CEC26	8.24E − 01/=	1.91E − 06/−	8.24E − 01/=	1.15E − 01/=	1.91E − 06/−	2.58E − 03/−	1.91E − 06/−	4.02E − 04/−	4.02E − 04/−	1.91E − 06/−	8.24E − 01/=
CEC27	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	4.01E − 05/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC28	2.63E − 01/=	1.91E − 06/−	1.00E + 00/=	4.01E − 05/+	1.91E − 06/−	4.14E − 02/−	1.91E − 06/−	4.14E − 02/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
CEC29	8.24E − 01/=	1.91E − 06/−	5.03E − 01/=	1.15E − 01/=	1.91E − 06/−	4.01E − 05/−	1.91E − 06/−	4.01E − 05/−	1.91E − 06/−	1.91E − 06/−	4.01E − 05/−
CEC30	1.00E + 00/=	1.91E − 06/−	1.15E − 01/=	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−	1.91E − 06/−
+/−/=	0/26/3	1/4/24	2/12/15	9/10/10	0/2/27	1/5/23	0/0/29	1/2/26	0/2/27	0/1/28	1/4/24

Table 4: The number of function evaluations and execution times on cec2017 test functions.

Function	JS	GSA	CSO	MVO	WOA	GOA	LSO	HHO	ASO	AOA	AO	FOGJS
CEC01	2.7065	12.3922	6.0170	6.0348	2.2613	35.1929	1.9855	11.3714	20.9821	4.5290	11.6457	15.1768
CEC03	3.5131	12.2369	7.8752	5.3787	2.7771	35.5317	2.3484	10.1844	23.2933	4.9004	24.9698	16.0245
CEC04	2.6789	11.6298	5.9341	5.5321	2.0978	34.7803	1.6983	9.6181	22.0126	4.0862	10.4534	15.4520
CEC05	2.7505	11.7143	6.2998	5.7581	2.2609	35.0219	1.8799	11.9700	20.3889	4.8477	12.0491	14.5897
CEC06	5.0495	13.8682	12.9905	7.9945	4.5030	37.2236	4.1648	17.9729	26.0110	6.8288	17.8733	19.3789
CEC07	2.9198	11.9703	6.8966	5.9520	2.4580	35.2540	2.0748	12.3911	19.7669	4.9447	14.2293	14.9892
CEC08	2.8103	11.7521	6.5594	5.7971	2.3564	35.1671	1.9380	12.2690	20.5061	4.8929	12.2444	14.7867
CEC09	4.3631	13.1465	10.7201	7.3054	3.8079	38.6865	3.3674	14.1672	20.5218	4.8905	12.3131	18.0388
CEC10	5.0373	12.7053	9.6647	6.9376	3.4602	36.3130	3.2438	15.5772	29.0926	5.2186	13.1325	18.3623
CEC11	2.9487	11.8532	6.7883	5.1665	2.4030	35.3573	2.7793	10.4996	26.0504	4.4148	11.2691	15.3322
CEC12	3.6138	12.1908	7.9926	6.2011	2.7580	35.4513	2.6014	11.2374	26.3553	5.1620	11.8367	16.3858
CEC13	3.1011	11.8712	6.9422	5.8020	2.4297	35.2583	2.1285	10.7652	25.2373	4.4944	10.9107	15.3066
CEC14	4.2574	12.2911	8.1456	5.9011	2.8667	35.6623	2.5319	12.8246	26.5752	5.3377	12.6115	17.4529
CEC15	2.9989	11.7677	6.6305	5.6208	2.3407	35.2033	2.0122	10.3497	25.4023	4.3027	10.7015	15.3175
CEC16	3.6415	12.4632	7.8490	6.2351	2.7876	35.6124	2.3901	11.1907	26.7440	4.6606	11.5330	18.2657
CEC17	5.6991	14.3658	14.4855	8.4687	5.0384	37.9059	4.8134	16.2210	30.2044	6.7846	15.0185	22.2148
CEC18	3.3428	11.9307	7.0916	5.6584	2.4941	35.2894	2.0984	11.2186	25.6128	4.6969	11.3155	16.2910
CEC19	9.0490	17.8390	24.8333	11.8436	8.4414	41.2716	8.2289	39.8321	34.6625	15.8344	34.1780	28.8875
CEC20	6.7150	14.7896	15.6685	8.9579	5.4461	38.0852	5.2075	16.8818	30.0451	6.8492	15.5655	23.0113
CEC21	6.2876	15.0089	16.4831	9.1370	5.6521	38.3453	5.3954	19.7756	33.9069	8.1039	17.9104	22.4235
CEC22	7.4123	16.0146	19.7128	10.4190	6.8338	39.5789	6.6221	22.1463	31.5021	8.9067	20.0652	24.8958
CEC23	8.3140	16.7610	22.0370	11.2247	7.6950	40.2366	10.7281	23.7808	34.3246	9.9222	22.1497	27.9339
CEC24	8.0441	16.6313	21.4595	10.8171	7.3408	40.0270	8.5336	33.3528	35.4490	10.5150	23.8269	29.6309
CEC25	7.8619	16.7512	21.3756	10.8456	7.3697	39.9249	11.0523	29.1524	33.6303	9.3849	22.1971	25.8589
CEC26	9.8584	18.4609	26.9625	12.9988	9.4126	41.8311	12.4126	35.0726	35.9213	11.4311	26.3548	30.0543
CEC27	11.3057	19.5524	30.5458	14.0669	11.3285	42.9386	11.8015	38.1281	38.6731	12.6044	29.0552	46.5453
CEC28	9.8053	18.3297	26.6680	12.6377	10.6993	41.8145	11.3436	33.5285	35.7967	11.1153	25.8306	32.0283
CEC29	9.2615	17.8411	24.7765	11.9704	8.8397	41.0065	10.5643	29.9246	35.2396	9.2298	21.0481	29.3321
CEC30	12.7780	21.1413	35.0944	15.4022	12.1838	44.4547	14.7088	58.1867	40.5053	18.4674	39.2704	36.0754
NFFEs	30030	30000	90030	30000	30000	60000	31030	98461	30000	30030	60000	60030

$$\frac{\mathrm{d}z^{(m)}(t)}{\mathrm{d}t}+az^{(m)}(t)=bt^{(m)}+c. \tag{22}$$

For the derivative of the left of equation (22), the first backward difference can be approximately expressed by equation (23) when $t=k$:

$$\left(\frac{\mathrm{d}z^{(m)}(t)}{\mathrm{d}t}\right)_{t=k}\approx\lim_{\Delta t\longrightarrow 1}\frac{z^{(m)}(k)-z^{(m)}(k-\Delta t)}{\Delta t}=z^{(m)}(k)-z^{(m)}(k-1). \tag{23}$$

Thus, when $t=k$, the left of (22) can be approximated by follows:

$$\left(\frac{\mathrm{d}z^{(m)}(t)}{\mathrm{d}t}+az^{(m)}(t)\right)_{t=k}\approx z^{(m)}(k)-z^{(m)}(k-1)+az^{(m)}(k)=(1+az)^{(m)}(k)-z^{(m)}(k-1). \tag{24}$$

Substituting equations (24) into (22), equation (25) can be obtained:

$$(1+a)z^{(m)}(k)-z^{(m)}(k-1)=bk^{(m)}+c, \tag{25}$$

where $k^{(m)}=\sum_{i=1}^{k}\begin{pmatrix}k-i+m-1\\ k-i\end{pmatrix}\cdot i$ is the time delay term with m-order.

Then, discrete formula (25) can be obtained by letting $\beta_1=1/1+a,\beta_2=b/1+a,\beta_3=c/1+a$:

$$z^{(m)}(k)=\beta_1 z^{(m)}(k-1)+\beta_2 k^{(m)}+\beta_3. \tag{26}$$

Equation (26) is the discrete fractional time-delayed grey model (FTDGM), which can be expressed by matrix:

$$\mathbf{B\beta}=\mathbf{Y}, \tag{27}$$

where $\mathbf{B}=\begin{bmatrix}z^{(m)}(1) & 2^{(m)} & 1\\ z^{(m)}(2) & 3^{(m)} & 1\\ \vdots & \vdots & \vdots\\ z^{(m)}(n-1) & n^{(m)} & 1\end{bmatrix},\mathbf{Y}=\begin{bmatrix}z^{(m)}(2)\\ z^{(m)}(3)\\ \vdots\\ z^{(m)}(n)\end{bmatrix},\beta=\begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\end{bmatrix}.$

If the fractional order m was determined, parameters $\beta 1$, $\beta 2$, and $\beta 3$ can be estimated by the least squares solution as equation (28):

$$\widehat{\beta}=\left[\widehat{\beta}_1,\widehat{\beta}_2,\widehat{\beta}_3\right]^{\mathrm{T}}=\left(\mathbf{B}^{\mathrm{T}}\mathbf{B}\right)^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{Y}. \tag{28}$$

Then, the value of $\widehat{z}^{(r)}(k)$ can be obtained by equation (29) after determining the parameters $\beta 1$, $\beta 2$, and $\beta 3$:

Table 5: Results of FOGJS and other comparison algorithms on cec2019 test functions.

Function	Index	Algorithms											
		JS	PSO	DE	GBO	LSA	SOA	SSA	HGS	HBO	WHOA	AOA	FOGJS
Cec01	Best	1.00E+00	8.24E+03	4.08E+04	1.00E+00	3.08E+03	1.00E+00	1.00E+00	1.00E+00	2.72E+05	1.00E+00	3.06E+07	1.00E+00
	Worst	2.70E+04	6.96E+07	9.04E+07	1.00E+00	5.30E+05	1.75E+07	1.00E+00	1.00E+00	3.06E+06	5.74E+04	6.56E+08	1.00E+00
	Mean	1.86E+03	5.30E+06	1.96E+07	1.00E+00	1.05E+05	1.83E+06	1.00E+00	1.00E+00	1.50E+06	1.04E+04	1.88E+08	1.00E+00
	Std	6.05E+03	1.60E+07	2.88E+07	8.16E−13	1.27E+05	4.13E+06	5.95E−15	0.00E+00	8.95E+05	1.83E+04	1.46E+08	0.00E+00
	Rank	5	10	11	4	7	9	1	1	8	6	12	1
Cec02	Best	4.12E+00	1.99E+02	3.22E+02	4.25E+00	1.55E+02	5.00E+00	4.22E+00	4.22E+00	2.02E+03	4.33E+00	7.55E+03	4.21E+00
	Worst	3.10E+01	7.17E+03	1.35E+04	5.00E+00	5.12E+02	4.45E+03	5.00E+00	2.48E+01	5.73E+03	5.73E+00	2.15E+04	4.28E+00
	Mean	8.09E+00	1.74E+03	3.39E+03	4.38E+00	3.05E+02	5.23E+02	4.33E+00	5.37E+00	3.42E+03	4.78E+00	1.43E+04	4.26E+00
	Std	6.52E+00	1.82E+03	3.69E+03	1.69E−01	1.06E+02	1.06E+03	2.28E−01	4.59E+00	9.18E+02	4.36E−01	3.97E+03	2.05E−02
	Rank	6	9	10	3	7	8	2	5	11	4	12	1
Cec03	Best	1.09E+00	1.82E+00	4.61E+00	1.41E+00	1.00E+00	4.00E+00	1.45E+00	1.41E+00	4.48E+00	1.00E+00	4.49E+00	1.41E+00
	Worst	4.69E+00	9.71E+00	1.11E+01	6.71E+00	7.70E+00	8.71E+00	9.71E+00	7.71E+00	8.41E+00	7.56E+00	9.10E+00	4.17E+00
	Mean	2.90E+00	5.25E+00	8.79E+00	1.67E+00	2.40E+00	7.25E+00	5.63E+00	3.83E+00	7.13E+00	2.63E+00	7.32E+00	2.68E+00
	Std	1.01E+00	2.37E+00	1.63E+00	1.19E+00	2.18E+00	1.94E+00	2.66E+00	2.44E+00	1.08E+00	1.82E+00	1.19E+00	8.85E−01
	Rank	5	7	12	1	2	10	8	6	9	3	11	4
Cec04	Best	8.96E+00	7.99E+00	1.49E+01	6.97E+00	1.69E+01	6.18E+01	5.97E+00	1.09E+01	5.05E+00	5.97E+00	3.78E+01	4.98E+00
	Worst	4.68E+01	7.06E+01	6.32E+01	4.78E+01	8.26E+01	1.24E+02	9.75E+01	3.78E+01	1.89E+01	3.03E+01	1.09E+02	5.17E+01
	Mean	1.92E+01	3.14E+01	3.71E+01	2.33E+01	4.32E+01	8.62E+01	3.53E+01	2.39E+01	1.27E+01	1.42E+01	6.52E+01	2.06E+01
	Std	7.40E+00	1.61E+01	1.56E+01	1.03E+01	1.75E+01	2.00E+01	2.02E+01	7.82E+00	4.84E+00	6.06E+00	2.10E+01	1.10E+01
	Rank	3	7	9	5	10	12	8	6	1	2	11	4
Cec05	Best	1.02E+00	1.16E+00	1.02E+00	1.02E+00	1.02E+00	2.78E+01	1.05E+00	1.02E+00	1.00E+00	1.02E+00	5.75E+01	1.02E+00
	Worst	1.16E+00	1.96E+01	1.68E+01	1.38E+00	1.30E+00	1.12E+02	1.37E+00	1.90E+00	1.09E+00	1.02E+01	1.81E+02	1.26E+00
	Mean	1.08E+00	5.00E+00	2.53E+00	1.15E+00	1.13E+00	6.56E+01	1.14E+00	1.19E+00	1.02E+00	1.53E+00	1.28E+02	1.08E+00
	Std	3.35E−02	5.42E+00	3.67E+00	7.61E−02	6.11E−02	2.18E+01	7.29E−02	1.96E−01	2.48E−02	2.05E+00	3.32E+01	5.97E−02
	Rank	2	10	9	6	4	11	5	7	1	8	12	3

Table 5: Continued.

Function	Index	Algorithms											
		JS	PSO	DE	GBO	LSA	SOA	SSA	HGS	HBO	WHOA	AOA	FOGJS
Cec06	Best	1.17E+00	3.06E+00	1.12E+00	1.54E+00	1.03E+00	7.71E+00	2.70E+00	2.97E+00	1.00E+00	1.33E+00	9.48E+00	1.17E+00
	Worst	4.61E+00	9.39E+00	9.49E+00	7.52E+00	7.60E+00	1.36E+01	7.76E+00	8.46E+00	1.64E+00	8.41E+00	1.43E+01	4.16E+00
	Mean	2.75E+00	6.38E+00	5.80E+00	4.07E+00	5.30E+00	1.10E+01	5.60E+00	5.61E+00	1.25E+00	3.88E+00	1.20E+01	2.37E+00
	Std	1.17E+00	1.56E+00	2.33E+00	1.64E+00	1.49E+00	1.37E+00	1.71E+00	1.71E+00	1.81E−01	1.92E+00	1.24E+00	8.73E−01
	Rank	3	10	9	5	6	11	7	8	1	4	12	2
Cec07	Best	3.60E+02	6.08E+02	3.82E+02	5.35E+02	1.35E+02	1.31E+03	4.35E+02	3.39E+02	4.71E+02	3.23E+02	7.59E+02	4.19E+02
	Worst	1.40E+03	2.43E+03	2.22E+03	1.52E+03	1.59E+03	2.77E+03	1.46E+03	1.13E+03	1.06E+03	1.11E+03	1.69E+03	1.36E+03
	Mean	8.82E+02	1.19E+03	1.37E+03	9.56E+02	9.63E+02	2.24E+03	8.42E+02	7.05E+02	7.69E+02	6.51E+02	1.36E+03	8.59E+02
	Std	2.66E+02	4.41E+02	5.07E+02	2.49E+02	3.77E+02	4.19E+02	2.85E+02	2.29E+02	1.72E+02	2.00E+02	2.69E+02	2.47E+02
	Rank	6	9	11	7	8	12	4	2	3	1	10	5
Cec08	Best	3.27E+00	3.53E+00	4.28E+00	3.37E+00	3.09E+00	4.38E+00	2.87E+00	3.20E+00	3.72E+00	2.96E+00	4.02E+00	3.21E+00
	Worst	4.27E+00	4.85E+00	5.43E+00	4.58E+00	4.53E+00	5.19E+00	4.96E+00	4.80E+00	4.44E+00	4.47E+00	5.50E+00	4.14E+00
	Mean	3.76E+00	4.29E+00	4.93E+00	3.95E+00	3.89E+00	4.88E+00	4.20E+00	4.08E+00	4.08E+00	3.74E+00	5.04E+00	3.70E+00
	Std	2.99E−01	3.77E−01	2.94E−01	3.38E−01	4.12E−01	1.76E−01	5.29E−01	3.95E−01	1.83E−01	3.82E−01	4.15E−01	3.03E−01
	Rank	3	9	11	5	4	10	8	7	6	2	12	1
Cec09	Best	1.12E+00	1.06E+00	1.12E+00	1.06E+00	1.12E+00	1.30E+00	1.15E+00	1.13E+00	1.12E+00	1.07E+00	2.91E+00	1.14E+00
	Worst	1.37E+00	1.44E+00	1.84E+00	1.45E+00	1.50E+00	5.23E+00	1.72E+00	1.63E+00	1.27E+00	1.42E+00	5.00E+00	1.35E+00
	Mean	1.23E+00	1.24E+00	1.35E+00	1.21E+00	1.29E+00	3.25E+00	1.36E+00	1.31E+00	1.21E+00	1.22E+00	3.78E+00	1.28E+00
	Std	6.64E−02	1.11E−01	1.71E−01	1.01E−01	1.07E−01	1.04E+00	1.92E−01	1.21E−01	3.25E−02	1.06E−01	4.95E−01	5.14E−02
	Rank	4	5	9	2	7	11	10	8	1	3	12	6
Cec10	Best	5.66E+00	2.10E+01	2.13E+01	2.16E+00	2.10E+01	2.16E+01	2.10E+01	2.10E+01	2.11E+01	2.10E+01	2.10E+01	3.01E+00
	Worst	2.15E+01	2.14E+01	2.20E+01	2.14E+01	2.15E+01	2.19E+01	2.13E+01	2.11E+01	2.13E+01	2.11E+01	2.10E+01	2.15E+01
	Mean	2.05E+01	2.12E+01	2.16E+01	2.03E+01	2.12E+01	2.17E+01	2.10E+01	2.10E+01	2.12E+01	2.10E+01	2.10E+01	1.89E+01
	Std	3.52E+00	1.41E−01	1.97E−01	4.26E+00	1.63E−01	7.66E−02	8.48E−02	2.59E−02	4.91E−02	1.63E−02	3.66E−03	6.03E+00
	Rank	3	10	11	2	8	12	7	6	9	5	4	1
Mean rank		4	8.6	10.2	4	6.3	10.6	6	5.6	5	3.8	10.8	2.8
Final ranking		3	9	10	3	8	11	7	6	5	2	12	1

Table 6: The p value of the Wilcoxon rank-sum test on cec2019 test functions.

Functions	JS	PSO	DE	GBO	LSA	SOA	SSA	HGS	HBO	WHOA	AOA
CEC01	1.91E − 06/−	8.01E − 09/−	8.01E − 09/−	NaN/=	1.91E − 06/−	7.63E − 06/−	1.00E + 00/=	NaN/=	8.01E − 09/−	7.63E − 06/−	1.91E − 06/−
CEC02	1.91E − 06/−	6.80E − 08/−	6.80E − 08/−	1.41E − 05/−	1.91E − 06/−	1.91E − 06/−	2.63E − 01/=	3.23E − 01/=	6.80E − 08/−	1.91E − 06/−	1.91E − 06/−
CEC03	1.15E − 01/−	1.66E − 07/−	6.80E − 08/−	4.36E − 07/+	1.18E − 02/+	1.91E − 06/−	4.14E − 02/−	5.07E − 01/=	6.80E − 08/−	5.03E − 01/=	1.91E − 06/−
CEC04	5.03E − 01/=	1.44E − 02/−	3.38E − 04/−	2.98E − 01/=	4.02E − 04/−	1.91E − 06/−	1.18E − 02/−	6.55E − 01/=	1.33E − 02/+	2.63E − 01/=	1.91E − 06/−
CEC05	4.14E − 02/+	2.22E − 07/−	3.38E − 04/−	6.22E − 04/−	4.14E − 02/−	1.91E − 06/−	4.14E − 02/−	3.75E − 04/−	2.79E − 03/+	8.24E − 01/=	1.91E − 06/−
CEC06	2.63E − 01/=	6.92E − 07/−	1.58E − 06/−	6.22E − 04/−	4.01E − 05/−	1.91E − 06/−	4.02E − 04/−	1.80E − 06/−	7.58E − 06/+	1.18E − 02/−	1.91E − 06/−
CEC07	8.24E − 01/=	1.23E − 02/−	2.04E − 05/−	2.39E − 01/=	2.63E − 01/=	1.91E − 06/−	2.63E − 01/=	6.79E − 02/=	8.39E − 01/=	1.18E − 02/+	4.02E − 04/−
CEC08	2.63E − 01/=	3.75E − 04/−	6.80E − 08/−	4.39E − 02/−	5.03E − 01/=	1.91E − 06/−	2.58E − 03/−	1.33E − 02/−	2.47E − 04/−	2.63E − 01/=	1.91E − 06/−
CEC09	5.03E − 01/=	5.25E − 01/=	6.55E − 01/−	1.14E − 02/+	1.00E + 00/=	1.91E − 06/−	8.24E − 01/=	4.09E − 01/=	4.70E − 03/+	1.15E − 01/=	1.91E − 06/−
CEC10	4.14E − 02/−	5.98E − 01/=	9.28E − 05/−	4.25E − 01/=	8.24E − 01/=	1.91E − 06/−	8.24E − 01/=	7.11E − 03/−	7.64E − 02/=	4.14E − 02/−	4.14E − 02/−
+/=/−	1/5/4	0/2/8	0/0/10	2/3/5	1/4/5	0/0/10	0/5/5	0/6/4	4/2/4	1/5/4	0/0/10

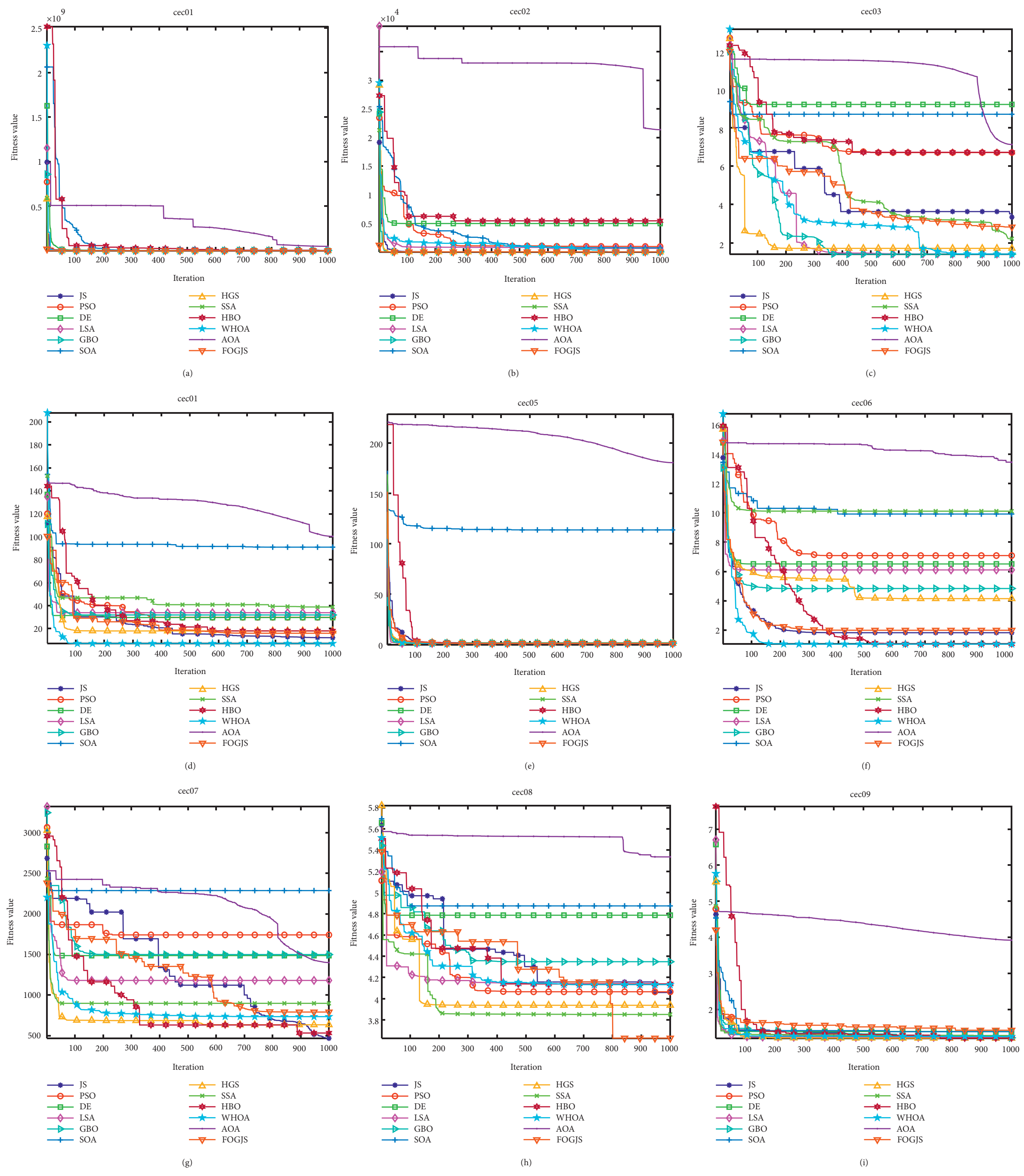


Figure 10: Continued.

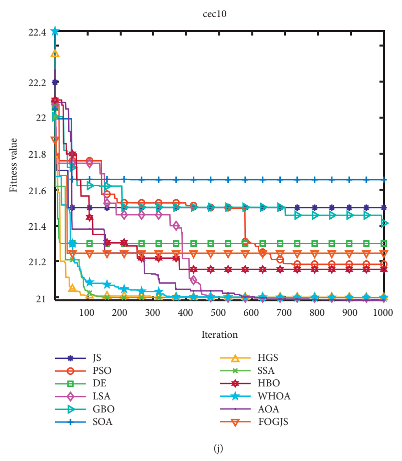


Figure 10: The convergence curves of the FOGJS and other intelligent algorithms on cec2019 test functions. (a) cec01. (b) cec02. (c) cec03. (d) cec04. (e) cec05. (f) cec06. (g) cec07. (h) cec08. (i) cec09. (j) cec10.

$$\begin{aligned}\hat{z}^{(m)}(k) &= \hat{\beta}_1 z^{(m)}(k-1) + \hat{\beta}_2 k^{(m)} + \hat{\beta}_3 \\ &= z^{(0)}(1)\cdot\left(\ddot{\beta}_1\right)^{(k-1)} + \sum_{i=2}^{k}\left(\hat{\beta}_2 i^{(m)} + \hat{\beta}_3\right)\cdot\left(\hat{\beta}_1\right)^{(k-i)}.\end{aligned} \tag{29}$$

Finally, the predicted value $\hat{Z}^{(0)} = (\hat{z}^{(0)}(1), \hat{z}^{(0)}(2), \cdots, \hat{z}^{(0)}(n))$ is obtained by m-order inverse fractional-order accumulation:

$$\hat{z}^{(0)}(k) = \left(\hat{z}^{(m)}(k)\right)^{(-m)} = \sum_{i=1}^{k}\binom{k-i-m-1}{k-i}\hat{z}^{(m)}(i). \tag{30}$$

Furthermore, the prediction precision of the model can be improved by analyzing the error between the raw data and predicted data. Thus, this paper introduces the triangular residual correction function into the discrete fractional time-delayed grey model to obtain a new prediction model with higher precision. According to the predicted value in equation (30), the residual error sequence can be calculated as follows:

$$r(k) = z^{(0)}(k) - \hat{z}^{(0)}(k), k = 2, 3, \cdots, n. \tag{31}$$

In addition, the definition of the triangle model is shown as

$$\begin{aligned} r(k+1) &= b_0 + b_1 k + b_3 \sin\frac{2\pi k}{T} + b_4 \cos\frac{2\pi k}{T} \\ &\quad + \varepsilon_{k+1}, k = 1, 3, \cdots, n-1,\end{aligned} \tag{32}$$

where ε is the random error and T is the parameter of time period. b_1, b_2, b_3, and b_4 are obtained by the least squares solution as

$$\mathbf{b} = [b_0, b_1, b_2, b_3]^{\mathrm{T}} = \left(\mathbf{C}^{\mathrm{T}}\mathbf{C}\right)^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{R}, \tag{33}$$

where $\mathbf{C} = \begin{bmatrix} 1 & 1 & \sin 2\pi/T & \cos 2\pi/T \\ 1 & 2 & \sin 4\pi/T & \cos 4\pi/T \\ \vdots & \vdots & \vdots & \vdots \\ 1 & n-1 & \sin 2(n-1)\pi/T & \cos 2(n-1)\pi/T \end{bmatrix}$, $\mathbf{R} = [r(2), r(3) \cdots, r(n)]$.

Submitting b_1, b_2, b_3, and b_4 into (32), the finial predicted values $Z^* = (z^*(1), z^*(2), \cdots, z^*(n))$ can be obtained as follows:

$$\begin{cases} z^*(1) = z^{(0)}(1), \\ z^*(k) = \hat{z}^{(0)}(k) + \hat{r}(k), k = 2, 3, \ldots, \end{cases} \tag{34}$$

when $k \le n$, $\hat{z}^{(0)}(k)$ are the fitting values of the raw data. And when $k > n$, $\hat{z}^{(0)}(k)$ are the predicted values.

5.1.2. Optimization Model Based on the FOGJS Algorithm. Obviously, once given the order m, the predicted value can be calculated by the above TDFTDGM model. Regarding the order m as a variable, the improved jellyfish search optimizer just can be employed to search the most suitable m to obtain predicted value with higher precision. Thus, an optimization model can be established by minimizing the mean absolute percentage error between the calculated values and the actual value. The final established optimization model is shown as equation (35).

The data set used in this experiment is the income of rural residents from 1989 to 2020 in Shaanxi Province. The data from 1989 to 2013 are used as training data, and the rest is regarded as test data. Then, the process for FOGJS to solve the rural resident's income forecasting model is displayed in Figure 13:

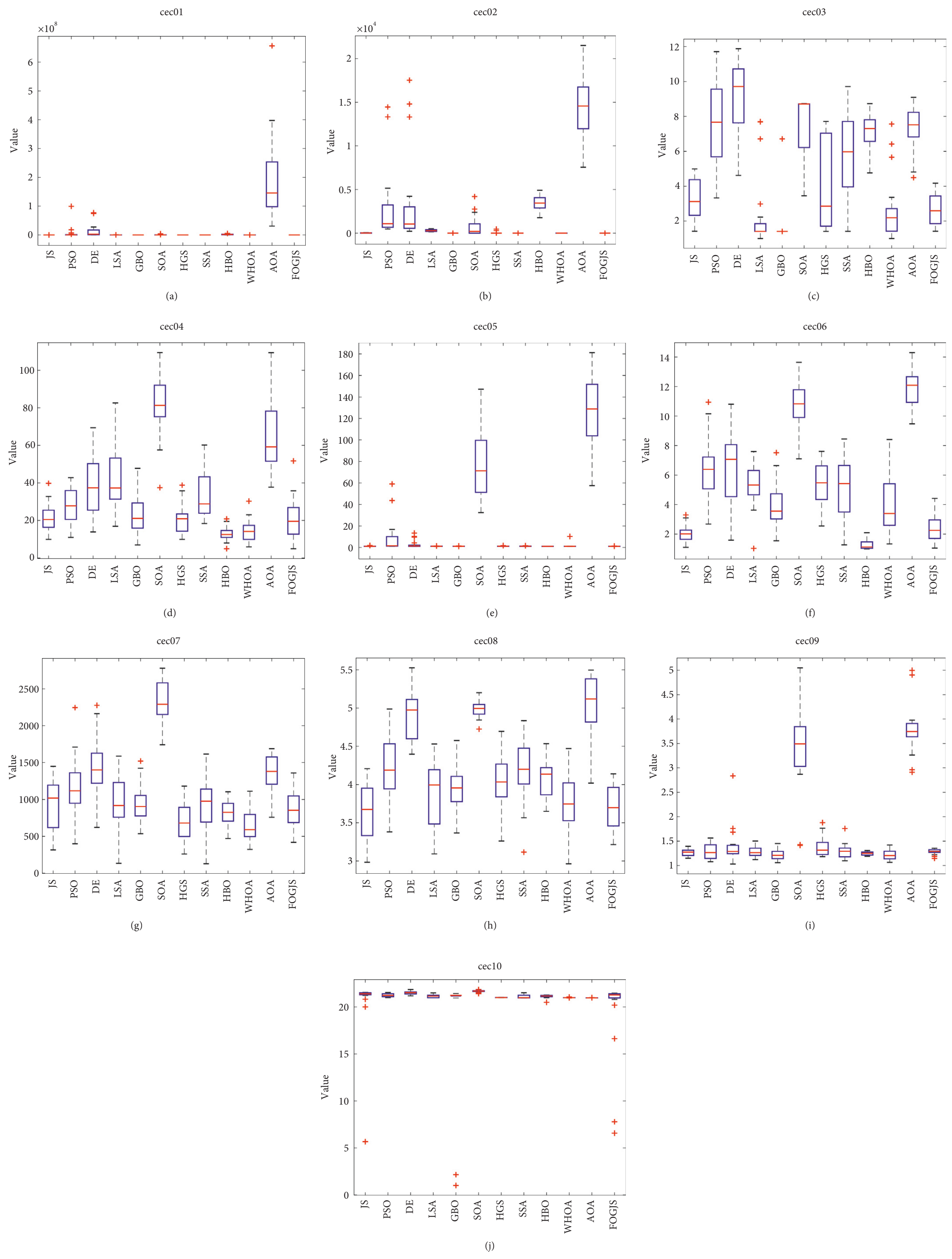


Figure 11: Box plot of the FOGJS and other algorithms for cec2019 test functions (red minus ⟶ median; red plus⟶ outlier). (a) Boxplot of cec01. (b) Boxplot of cec02. (c) Boxplot of cec03. (d) Boxplot of cec04. (e) Boxplot of cec05. (f) Boxplot of cec06. (g) Boxplot of cec07. (h) Boxplot of cec08. (i) Boxplot of cec09. (j) Boxplot of cec10.

Table 7: The number of function evaluations and execution times on cec2019 test functions.

Function	JS	PSO	DE	GBO	LSA	SOA	SSA	HGS	HBO	WHOA	AOA	FOGJS
CEC01	3.6200	2.8114	6.0888	15.7707	30.2908	4.3671	7.5269	4.3323	4.6441	13.9102	3.4599	15.6979
CEC02	2.1869	1.3686	6.0189	14.4759	47.7168	4.2256	5.3505	2.7221	3.7760	12.2332	2.2003	13.3965
CEC03	2.8174	1.3118	6.3777	15.5053	52.3774	4.5143	5.4665	2.7589	3.7545	12.3701	2.2264	13.4125
CEC04	2.5652	1.5433	4.9453	14.2501	36.4910	3.2141	5.7330	2.6113	3.2024	12.3221	2.1680	13.8916
CEC05	2.5422	1.5809	4.9875	14.3659	34.9856	3.2794	5.7593	2.6728	3.2467	12.3997	2.2298	13.9198
CEC06	20.3736	18.8581	22.2178	33.9569	74.9322	20.5988	34.6231	19.9502	20.2233	30.5913	19.6429	49.0128
CEC07	3.2563	1.7084	5.1264	14.5384	36.4959	3.3727	6.0191	2.7732	3.4223	12.5532	2.3344	15.2189
CEC08	2.7577	1.5725	4.9911	14.4529	30.2979	3.2464	5.7523	2.6618	3.2323	12.4473	2.1979	14.4652
CEC09	2.1947	1.3554	4.7838	14.1492	37.5057	3.0529	5.2918	2.4728	3.0199	11.8400	2.0393	13.3624
CEC10	3.2142	1.5874	4.9943	14.4306	41.9134	3.2798	5.8652	2.6894	3.2357	12.4172	2.2075	15.1973
NFFEs	30030	30030	30030	30030	107584	30030	90030	30000	30030	150030	30030	60030

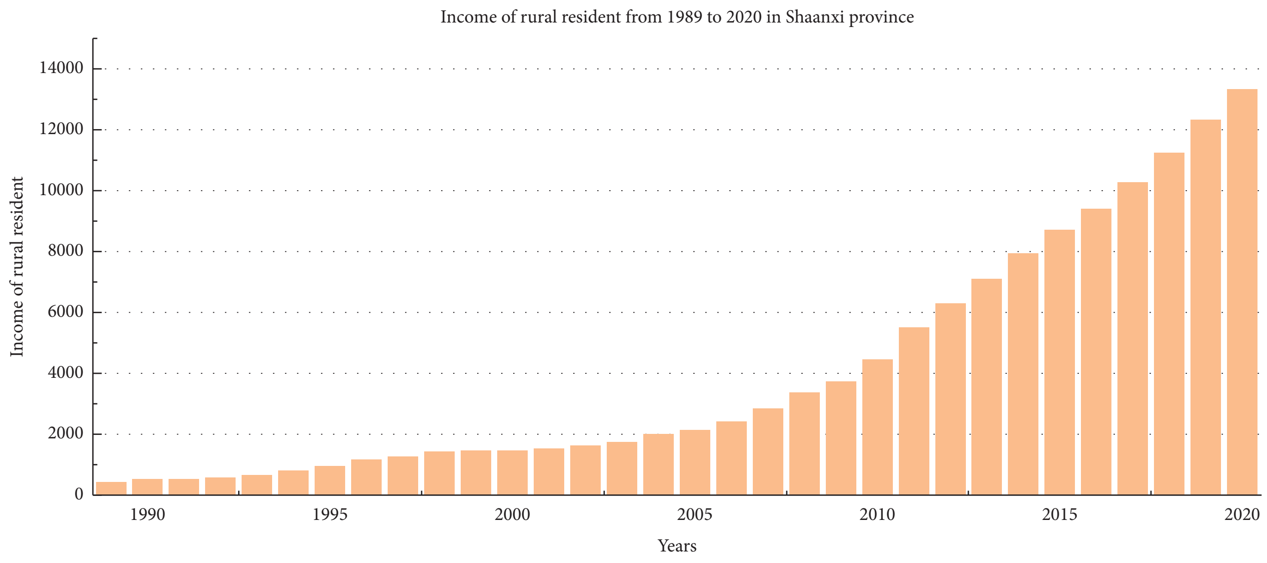


Figure 12: Income of rural resident from 1989 to 2020 in Shaanxi Province.

$$\min \arg(f(m)) = \frac{1}{n-1}\sum_{k=2}^{n}\left|\frac{z^*(k) - z^{(0)}(k)}{z^{(0)}(k)}\right| \times 100\%,$$

$$s.t.\begin{cases} \left[\hat\beta_1, \hat\beta_2, \hat\beta_3\right]^{\mathrm{T}} = \left(\mathbf{B}^{\mathrm{T}}\mathbf{B}\right)^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{Y} \\ \hat z^{(m)} = (k) z^{(m)}(1)\cdot\left(\hat\beta_1\right)^{(k-1)} + \sum_{i=2}^{k}\left(\hat\beta_2 i^{(r)} t + n\hat\beta_3\right)\cdot\left(\hat\beta_1\right)^{(k-i)}, \\ \left[b_0, b_1, b_2, b_3\right]^{\mathrm{T}} = \left(\mathbf{C}^{\mathrm{T}}\mathrm{C}\right)^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{R} \\ \hat r(k+1) = b_0 + b_1 k + b_3 \sin\frac{2\pi k}{T} + b_4 \cos\frac{2\pi k}{T} + \varepsilon_{k+1}, k = 1, 3, \ldots, n-1, \\ z^*(1) = z^{(0)}(1), \\ z^*(k) = \hat z^{(0)}(k) + \hat r(k), k = 2, 3, \ldots, \end{cases} \tag{35}$$

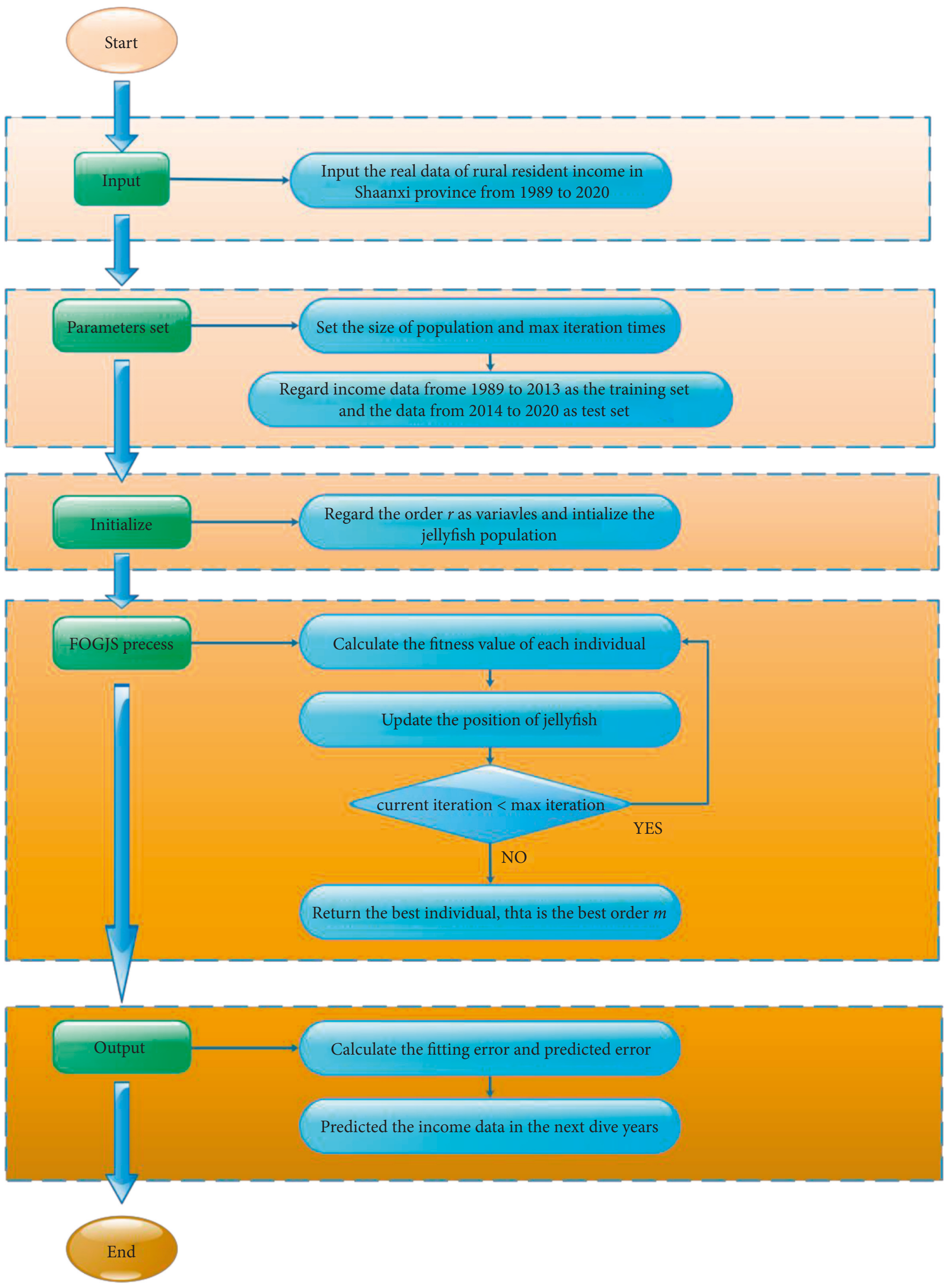

Start
Input
Input the real data of rural resident income in Shaanxi province from 1989 to 2020
Parameters set
Set the size of population and max iteration times
Regard income data frome 1989 to 2013 as the training set and the data from 2014 to 2020 as test set
Initialize
Regard the order r as variavles and intialize the jellyfish population
FOGJS precess
Calculate the fitness value of each individual
Update the position of jellyfish
current iteration < max iteration
YES
NO
Return the best individual, thta is the best order m
Output
Calculate the fitting error and predicted error
Predicted the income data in the next dive years
End

Figure 13: The process of the TDFTDGM prediction model based on the FOGJS algorithm.

5.2. The Comparison Results Based on Different Optimization Algorithms. In Section 5.1, an optimization model for resident income forecasting is proposed by considering the order m as a decision variable. However, lots of optimization algorithms can be used to solve the optimization model. Thus, to verify the validity of the prediction model based on the FOGJS algorithm, the original JS algorithm and other algorithms are employed to solve this optimization model. The selected comparison algorithms are the original JS algorithm [36], Aquila optimizer (AO) [19], sine cosine

Table 8: Result of the predicted model based on different algorithms.

	FOGJS	JS	AO	SCA	GWO	RSO	SOA	DE	PSO
Best	**10.3858**	10.3859	10.3893	10.3895	10.3859	10.3860	10.3888	10.3859	**10.3858**
Mean	**10.3860**	10.3873	10.4338	10.4149	10.3974	10.4267	10.4174	10.3877	10.3970
Worst	**10.3865**	10.3909	10.6427	10.4926	10.4329	10.5582	10.4819	10.3940	10.4117
Std	**2.1697*E* − 04**	1.6785*E* − 03	7.4951*E* − 02	3.0414*E* − 02	1.4576*E* − 02	5.6484*E* − 02	3.1718*E* − 02	2.4713*E* − 03	8.7028*E* − 03

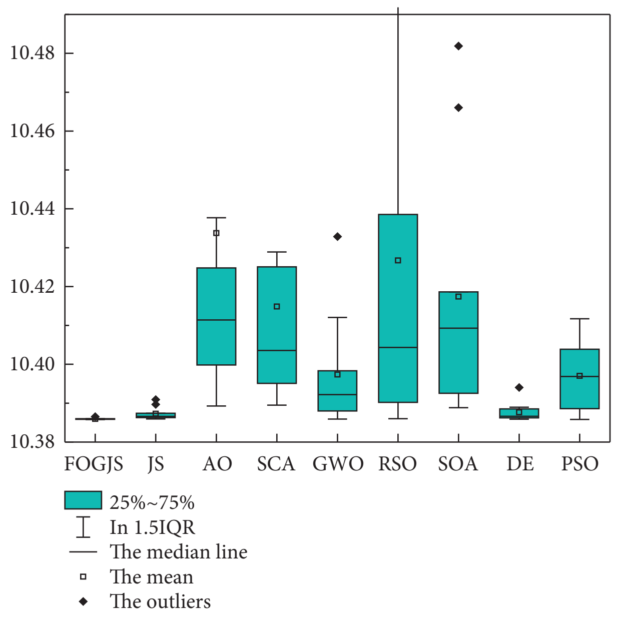


Figure 14: The boxplots of different algorithms.

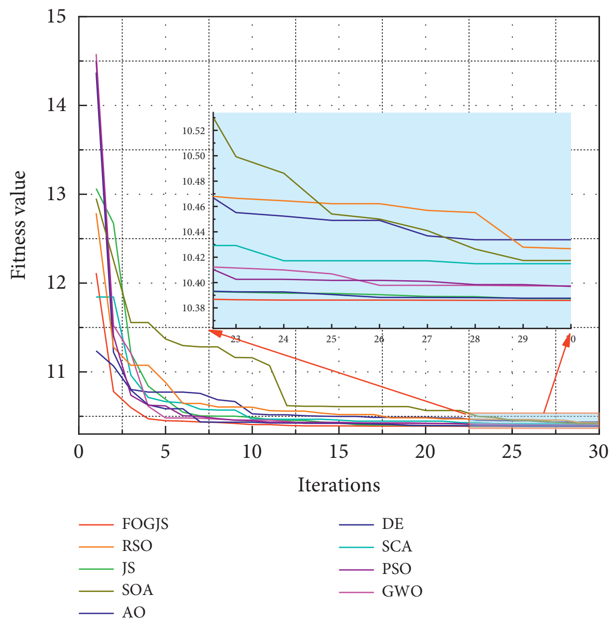


Figure 15: The convergence curves of different algorithms.

algorithm (SCA) [59], grey wolf optimizer (GWO) [60], rat swarm optimizer (RSO) [61], seagull optimization algorithm (SOA) [55], DE [28], and PSO algorithm [45]. Meanwhile, to reflect the efficiency of the algorithm, the number of iterations is set as 30 times to highlight the ability of the algorithm to solve problems in a short time. Here, the parameters of the FOGJS algorithm are the same as the results obtained by the sensitivity analysis in Section 4.3. That is, $\beta = 0.4$ and $\gamma = 2$. All algorithms run 10 times independently setting the size of the population as 50. Then, Table 8 provides the results after 10 times runs, including the best value (Best), the average value (Mean), the worst value (Worst), and the standard deviation (Std). According to the error between predicted data and the real data in Table 8, the FOGJS algorithm is the most suitable one to solve the predicted model than others, because it has the best performance on all measure indexes. Though JS, DE GWO, and PSO also perform well on the best values, they are much worse than the FOGJS algorithm in terms of average values. It illustrates other algorithms are highly susceptible to local optimums, thus causing instability of the solutions.

Figure 14 also supports the above conclusion through boxplots. The lowest position and the smallest height of the box plotted by the obtained results indicate that the FOGJS algorithm has strong robustness in solving the problem. Both JS and DE algorithms have some outliers, showing that the quality of the solutions is easily affected by other factors, which needs to be avoided in practice applications. Moreover, Figure 15 shows the convergence curves of different algorithms in solving the predicted model of resident income. The FOGJS algorithm has the fastest convergence speed in the early period of iteration, especially being compared with the original JS algorithm. That is, the fractional-order modified mechanism improves the quality of the whole population, speeding up the convergence rate to the optimal solution. In the later stages of the search, Gaussian mutation also plays a role in avoiding local optimums, which can be observed in the enlarged subplot.

Table 9: Evaluation indicators.

Measure indicators	Formula
Mean absolute error (MAE)	$(1/n)\sum_{k=1}^{n}\lvert\hat{z}^{(0)}(k) - z^{(0)}(k)\rvert$
Mean absolute percentage error (MAPE)	$(1/n)\sum_{k=1}^{n}\lvert\hat{z}^{(0)}(k) - z^{(0)}(k)\rvert \times 100\%$
Mean square error (MSE)	$(1/n)\sum_{k=1}^{n}(\hat{z}^{(0)}(k) - z^{(0)}(k))^2$
Root mean square percentage error (RMSPE)	$\sqrt{(1/n)\sum_{k=1}^{n}(\hat{z}^{(0)}(k) - z^{(0)}(k)/z^{(0)}(k))^2} \times 100\%$

Table 10: Fitting results of different forecast models.

Years	Actual income	GM	DGM	TRGM	FANGBM	FTDGM	DFTDGM	TDFTDGM
1989	433.67	434.00	434.00	434.00	434.00	434.00	434.00	434.00
1990	530.00	306.97	311.02	569.77	483.14	537.18	483.23	542.11
1991	533.96	347.14	351.70	635.64	537.88	640.47	538.04	534.00
1992	558.79	392.57	397.71	709.11	598.85	743.90	599.08	648.42
1993	652.99	443.95	449.73	791.08	666.77	847.48	667.03	657.57
1994	804.84	502.04	508.56	882.53	742.44	951.26	742.70	786.08
1995	962.89	567.75	575.08	984.54	826.76	1055.29	826.94	816.24
1996	1165.10	642.04	650.30	1098.35	920.71	1159.60	920.75	963.04
1997	1273.30	726.07	735.36	1225.31	1025.41	1264.27	1025.19	1018.92
1998	1415.08	821.08	831.55	1366.95	1142.11	1369.37	1141.48	1189.29
1999	1474.96	928.53	940.32	1524.96	1272.19	1475.00	1270.96	1294.69
2000	1471.67	1050.05	1063.32	1701.24	1417.21	1581.27	1415.13	1477.34
2001	1529.11	1187.46	1202.40	1897.89	1578.89	1688.31	1575.66	1585.97
2002	1648.04	1342.86	1359.68	2117.28	1759.19	1796.31	1754.39	1842.90
2003	1741.11	1518.60	1537.53	2362.02	1960.25	1905.46	1953.40	2035.64
2004	1952.57	1717.33	1738.65	2635.06	2184.52	2016.01	2174.98	2305.63
2005	2161.68	1942.07	1966.07	2939.66	2434.71	2128.27	2421.69	2522.53
2006	2396.33	2196.22	2223.24	3279.46	2713.85	2242.60	2696.39	2890.23
2007	2824.05	2483.63	2514.05	3658.55	3025.34	2359.47	3002.26	3217.49
2008	3373.46	2808.65	2842.90	4081.46	3373.00	2479.41	3342.81	3627.64
2009	3722.07	3176.21	3214.76	4553.25	3761.09	2603.10	3722.00	4011.96
2010	4477.21	3591.86	3635.26	5079.58	4194.40	2731.35	4144.20	4520.87
2011	5483.79	4061.92	4110.77	5666.76	4678.29	2865.13	4614.29	5092.62
2012	6285.00	4593.48	4648.48	6321.80	5218.78	3005.67	5137.71	5725.99
2013	7092.20	5194.61	5256.51	7052.56	5822.62	3154.42	5720.50	6370.73

Table 11: Fitting error of different forecast models.

	GM	DGM	TRGM	FANGBM	FTDGM	DFTDGM	TDFTDGM
MAE	519	499	320	248	627	256	**221**
MAPE (%)	25.606	24.725	16.212	**9.696**	17.258	9.763	9.873
MSE	488381	454256	200193	163109	1546543	185217	**85490**
RMSPE (%)	28.488	27.717	20.539	**11.653**	24.240	11.796	12.034

5.3. The Comparison Results of Different Predicted Models. According to the analysis of Section 5.2, the predicted model based on FOGJS algorithm is effective in solving the predicted problem of resident income. Then, the TDFTDGM + FOGJS approach also needs to be compared with other classical predicted models. In this section, the other six predicted approaches are selected, including GM [12], DGM [14], TRGM [13], FANGBM [15], FTDGM [16], and DFTDGM. Table 9 displays the measure indicators in the prediction of resident income to illustrate the difference of all models. To compare different models more fairly, two types of errors will be calculated. The fitting error is the error between data obtained by models and training data. And the predicted error represents the error between data obtained by models and test data.

Table 10 displays the fitting results obtained by different forecast models. In addition, Table 11 shows the fitting error compared with the actual data. Obviously, the TDFTDGM predicted model shows better performance than others on two evaluation indicators, which is marked in bold. Compared with GM, TRGM has a smaller fitting error. Meanwhile, under similar results in MAPE and RMSPE, TDFTDGM performs significantly better than DFTDGM on MAE and MSE, which illustrates that applying the triangular residual correction method into the original model is an effective method to improve prediction accuracy.

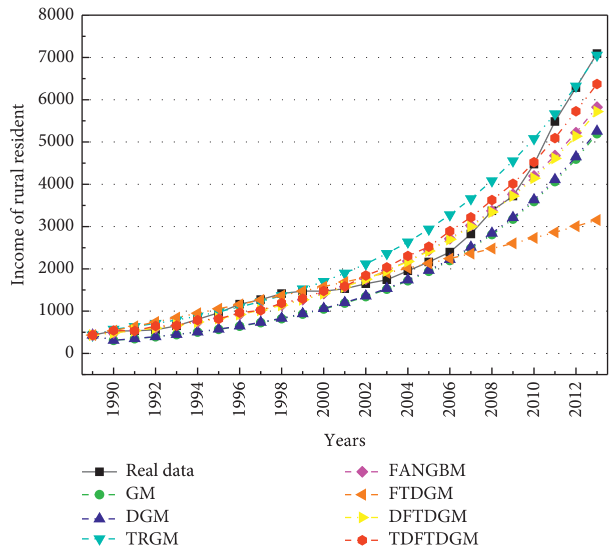


Figure 16: Fitting curves of income by different forecast models.

Table 12: Predicted results of different forecast models.

Years	Actual income	GM	DGM	TRGM	FANGBM	FTDGM	DFTDGM	TDFTDGM
2014	7932.21	5874.41	5944.09	7867.80	6497.38	3313.16	6369.39	7232.41
2015	8689.00	6643.17	6721.60	8777.27	7251.58	3484.06	7091.90	8057.86
2016	9396.00	7512.53	7600.81	9791.88	8094.78	3669.75	7896.36	9045.05
2017	10264.51	8495.67	8595.03	10923.76	9037.73	3873.45	8792.07	10096.47
2018	11213.00	9607.46	9719.29	12186.49	10092.53	4099.05	9789.38	11331.47
2019	12326.00	10864.75	10990.61	13595.18	11272.79	4351.32	10899.83	12737.23
2020	13316.00	12286.57	12428.23	15166.70	12593.82	4636.02	12136.24	14283.60

Table 13: Predicted error of different forecast models.

	GM	DGM	TRGM	FANGBM	FTDGM	DFTDGM	TDFTDGM
MAE	1693	1591	757	1185	6530	1452	**478**
MAPE (%)	17.239	16.271	6.477	12.056	62.095	14.502	**4.731**
MSE	2981279	2662247	940980	1458475	44515496	2123232	**308408**
RMSPE (%)	18.250	17.361	7.885	12.749	62.140	14.929	**5.507**

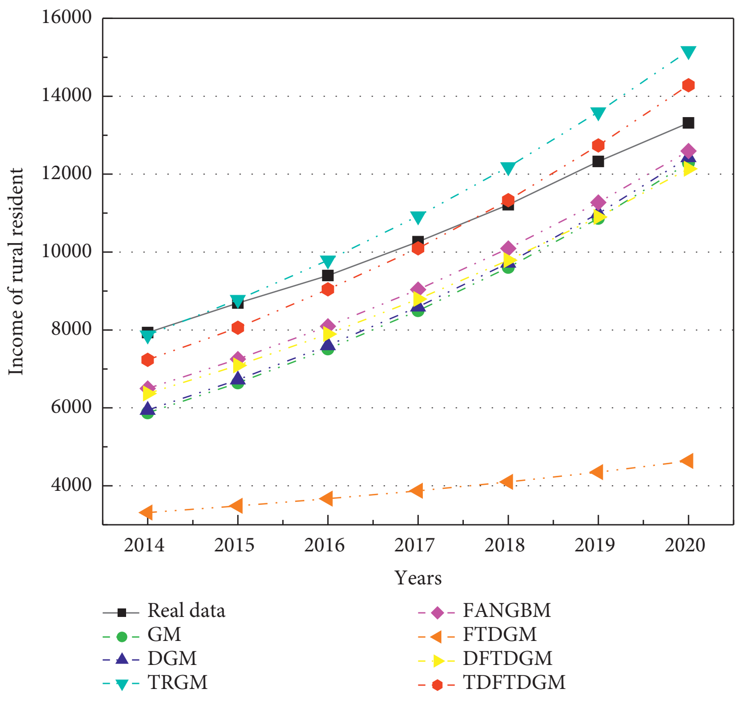


Figure 17: Predicted curves of income by different forecast models.

Table 14: The predicted results from 2021 to 2025 based on different forecast models.

Years	GM	DGM	TRGM	FANGBM	FTDGM	DFTDGM	TDFTDGM
2021	13894.46	14053.89	16919.89	14072.94	4960.21	13512.89	14782.68
2022	15712.77	15892.20	18875.73	15729.55	5332.43	15045.71	16450.74
2023	17769.04	17970.96	21057.66	17585.65	5763.11	16752.40	18307.02
2024	20094.40	20321.64	23491.80	19666.03	6264.93	18652.68	20372.76
2025	22724.07	22979.79	26207.32	21998.72	6853.30	20768.53	22671.59

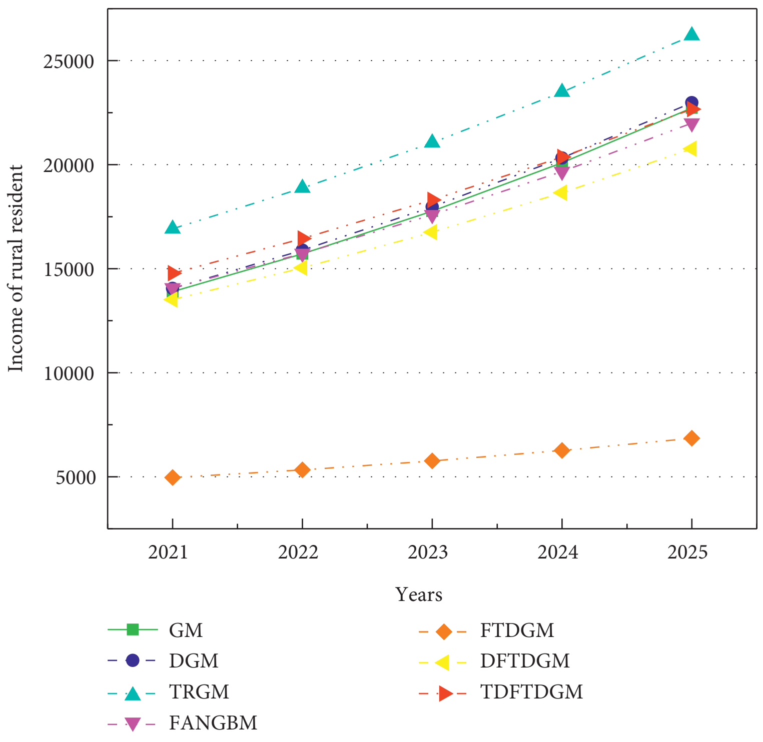


FIGURE 18: The curves of resident income from 2021 to 2025 through various forecast models.

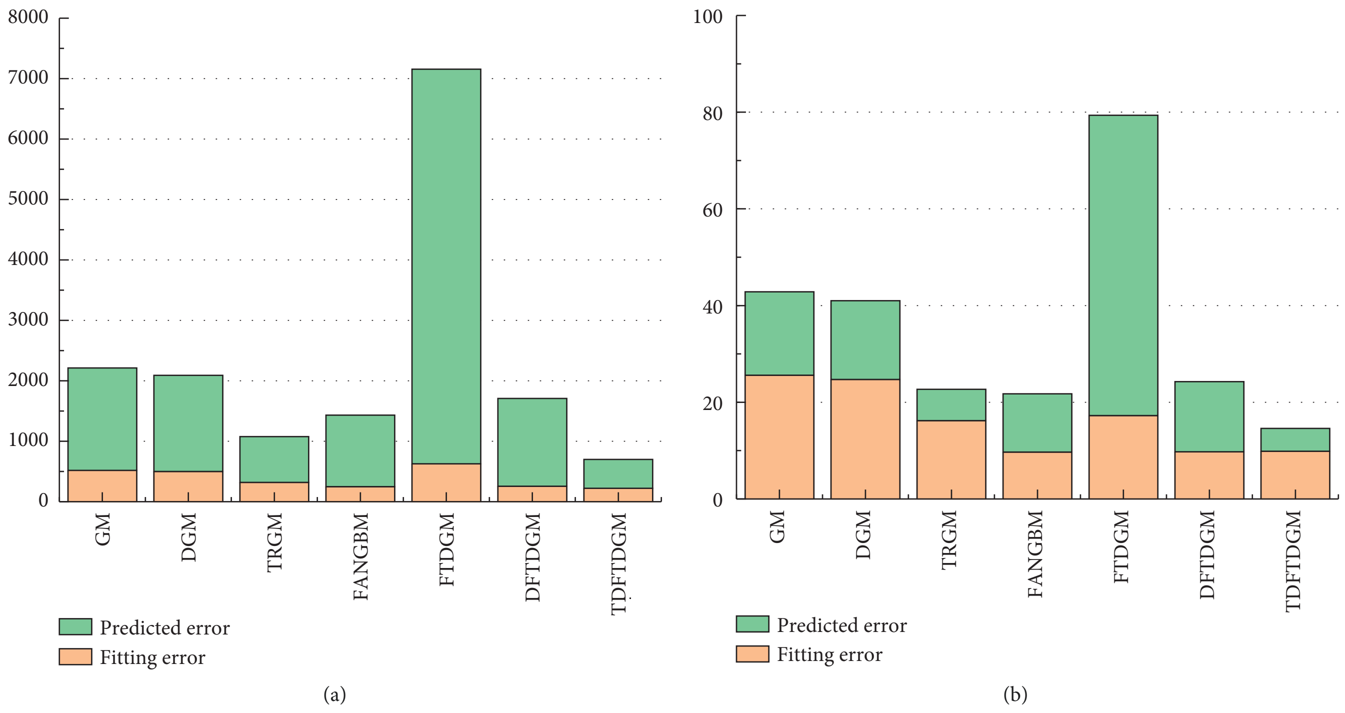
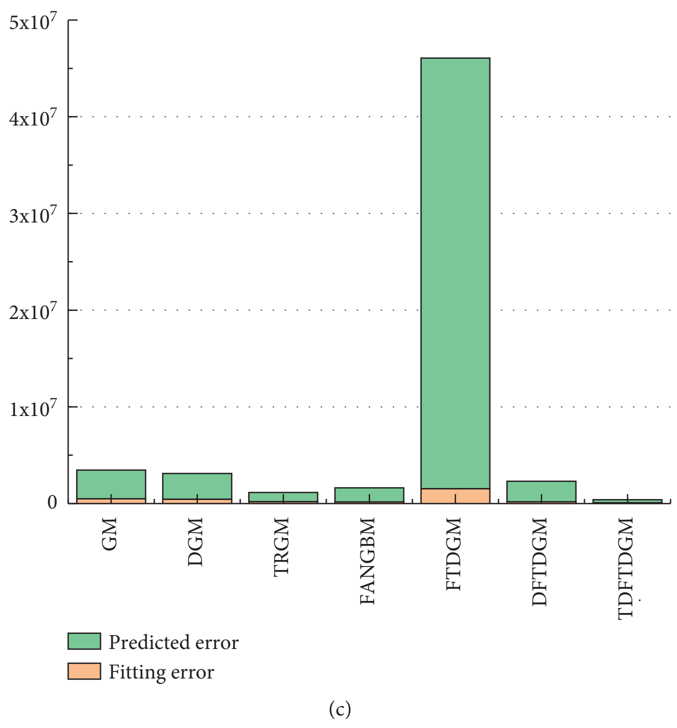


FIGURE 19: Continued.



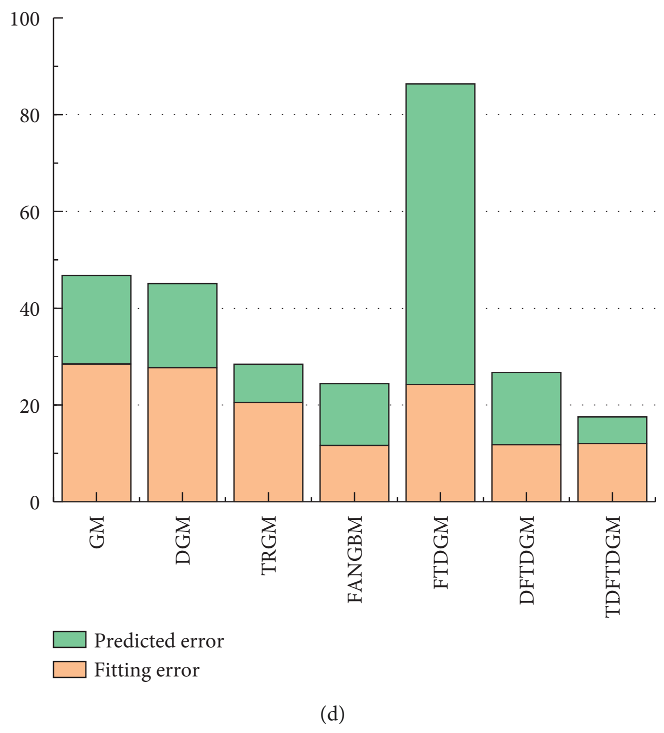


Figure 19: Comprehensive comparison of different forecast models. (a) MAE. (b) MAPE. (c) MSE. (d) RMSPE.

Figure 16 shows the fitting curve of different models. The FTDGM and FANGBM models perform well before about 2004. However, after that, the fitting data of FTDGM and FANGBM differ greatly from the real income data. That is, these two models have great defects when facing a set of data with large fluctuation. After discretizing the FTDGM model, DFTDGM overcomes this disadvantage as shown in the yellow curve in Figure 16. However, after 2010, the gap between the income data obtained using the DFTDGM model and the real income become larger. The red curve of the TDFTDGM model is closer to the real data for all fitting data with time growth, which means the trigonometric correction function further improves the accuracy of the prediction model to let it more suitable for long-term forecasting. The analysis of the combined results shows that the fitted data obtained by TDFTDGM model are more in line with the actual income changes in the process.

Table 12 shows the predicted results obtained by all models, and the predicted error is summarized in Table 13. For the predicted error, the advantages of TDFTDGM are even more obvious on all four evaluation indexes. In addition, the predicted error of the FANGBM model is large, though it provides great performance in fitting results. The mean absolute percentage error (MAPE) and root mean square percentage error (RMSPE) of FANGBM are all over 10, which means its poor prediction ability.

Meanwhile, it can be observed in Figure 17 that the predicted curve of TRGM is moving away from the real curve with the increase of years. Thus, though the TRGM model also has smaller MAPE and RMSPE, the predicted precision will decrease if the predicted data after 2020 is still needed. However, the predicted data obtained by the TDFTDGM model is able to maintain fluctuations in the vicinity of the real data. Hence, the proposed model with triangular residual correction is superior to others and can provide more reliable and informative predictions.

The income of rural residents in the next five years (2021–2025) is also predicted and shown in Table 14. In addition, Figure 18 is plotted by the predicted data. The curves of TRGM and FTDGM are growing too fast and slow, respectively, which are not conform to the trend of income growth. The red curve's growth rate of the TDFTDGM model is more stable, which is more suitable for the long-term forecast of rural residents' income. Moreover, due to the different performance on fitting and predicted error, Figure 19 draws the bar graphs to discuss the comprehensive performance of different models. On the four-measure indexes, the proposed TDFTDGM model has outstanding advantages over other models. That is, by introducing the trigonometric correction function and FOGJS into the discrete fractional time-delayed grey model, it becomes a competitive forecasting method in practical application.

6. Conclusion and Future Work

This paper predicts the rural resident income by combing the TDFTDGM model and FOGJS algorithm. Firstly, the fractional-order modified and Gaussian mutation mechanisms are introduced into the original JS algorithm. After analyzing the effect of different parameters, more suitable parameters are selected in the FOGJS algorithm to improve its capacity. Then, from the exploration and exploitation diagrams, the FOGJS algorithm keeps a balance between the two capacities. Meanwhile, by being compared with different kinds of algorithms on classical test functions, the FOGJS algorithm offers outstanding performance. For the solution precision, the improved algorithm ranks first in terms of mean rank on both cec2017 and cec2019. From the convergence curves and box plots, it can be observed that the FOGJS algorithm has advantages of convergence speed and stability. Moreover, the results of the Wilcoxon rank-sum test further support the conclusion that the introduction of improvement strategies forms the special search mechanism to provide superior performance.

Secondly, the discrete fractional time-delayed grey model with triangular residual correction is established. In addition, the FOGJS algorithm is used to optimize the order of the model. The experiment of income forecasting is divided into two parts. On the one hand, the original JS algorithm and the other seven popular algorithms are selected to solve the TDFTDGM model to be compared with the FOGJS algorithm. Results show that FOGJS algorithm has outstanding performance on precision and convergence speed, which indicates that the FOGJS + TDFTDGM approach is an effective tool for prediction of resident income. On the other hand, the FOGJS + TDFTDGM approach is compared with other six prediction models. Experiments show that TDFTDGM model obtains the predicted data closer to the real income data. Thus, a conclusion can be deduced that TDFTDGM model is more suitable for the long-term prediction of volatile data. The combination between TDFTDGM and FOGJS algorithm also provides an idea to determine the parameters in the forecast model.

In future work, the fractional-order modified and Gaussian mutation mechanism may also be suitable choices for some metaheuristic algorithms (e.g., MRFO algorithm [62] and hybrid arithmetic optimization algorithm [63]) to improve their performance. Moreover, the TDFTDGM model can deal with forecast problems in other fields, such as population forecast, climate forecast, and resource forecast.

Data Availability

All data generated or analyzed during this study are included in this published article (and its supplementary information files).

Conflicts of Interest

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

Acknowledgments

This work was supported by National Key R & D Program of China (Grant No. 2019YFD1100700), National Natural Science Foundation of China (Grant No. 51978432; 51678380), and Jiangsu Province “333” project.

Supplementary Materials

The Matlab Code of the proposed FOGJS. (*Supplementary Materials*)

References

[1] D. Sena and N. K. Nagwani, “Application of Time Series Based Prediction Model to Forecast Per Capita Disposable Income,” in *Proceedings of the 2015 IEEE International Advance Computing Conference (IACC)*, vol. 2015, pp. 454–457, Banglore, India, July 2015.

[2] P. Wang, T. Zhao, Y. Fan, and Q. Hao, “Study of Chinese Farmers Income Forecast Model Based on BP Neural Network,” in *Proceedings of the 2014 IEEE Workshop on Electronics, Computer and Applications*, vol. 2014, Ottawa, ON, 2014.

[3] O. Isengildina-Massa, B. Karali, and T. H. Kuethe, “Joint evaluation of the system of USDA’s farm income forecasts,” *Applied Economic Perspectives and Policy*, vol. 43, pp. 1140–1160, 2021.

[4] A. Kibekbaev and E. Duman, “Benchmarking regression algorithms for income prediction modeling,” *Information Systems*, vol. 61, pp. 40–52, 2016.

[5] S.-y. Zhai, G.-x. Song, Y.-c. Qin, and X.-y. Ye, “Climate change and Chinese farmers: perceptions and determinants of adaptive strategies,” *Journal of Integrative Agriculture*, vol. 17, no. 4, pp. 949–963, 2018.

[6] Z. Maaouane, S. Zouggar, and G. Krajačić, “Modelling industry energy demand using multiple linear regression analysis based on consumed quantity of goods,” *Energy*, vol. 225, Article ID 120270, 2021.

[7] Y.-R. Zeng, Y. Zeng, and B. Choi, “Multifactor-influenced energy consumption forecasting using enhanced backpropagation neural network,” *Energy*, vol. 127, pp. 381–396, 2017.

[8] A. Altan, S. Karasu, and E. Zio, “A new hybrid model for wind speed forecasting combining long short-term memory neural network, decomposition methods and grey wolf optimizer,” *Applied Soft Computing*, vol. 100, Article ID 106996, 2021.

[9] Q. Zhang, Z. Chen, and Y. Zeng, “Data-driven approaches for time series prediction of daily production in the Sulige tight gas field, China,” *Artificial Intelligence in Geosciences*, vol. 2, pp. 165–170, 2021.

[10] Y. Guan, X. Y. Li, and J. M. Zhu, “Prediction and Analysis of Rural Population in China based on ARIMA Model,” *The Journal of Shandong Agriculture and Engineering University*, vol. 36, pp. 15–20, 2019.

[11] B. Zeng, X. Ma, and J. J. Shi, “A new-structure grey Verhulst model for China’s tight gas production forecasting,” *Applied Soft Computing*, vol. 96, 2020.

[12] N. M. Xie and R. Z. Wang, “A historic review of grey forecasting models,” *J. Grey Syst-Uk.*, vol. 29, pp. 1–29, 2017.

[13] P. Zhou, B. Ang, and K. Poh, “A trigonometric grey prediction approach to forecasting electricity demand,” *Energy*, vol. 31, no. 14, pp. 2839–2847, 2006.

[14] N.-m. Xie and S.-f. Liu, “Discrete grey forecasting model and its optimization,” *Applied Mathematical Modelling*, vol. 33, no. 2, pp. 1173–1186, 2009.

[15] W. Wu, X. Ma, and B. Zeng, “Forecasting short-term renewable energy consumption of China using a novel fractional nonlinear grey Bernoulli model,” *Renewable Energy*, vol. 140, pp. 70–87, 2019.

[16] X. Ma, X. Mei, and W. Wu, “A novel fractional time delayed grey model with Grey Wolf Optimizer and its applications in forecasting the natural gas and coal consumption in Chongqing China,” *Energy*, vol. 178, pp. 487–507, 2019.

[17] G. Hu, B. Du, and X. F. Wang, “An enhanced black widow optimization algorithm for feature selection,” *Knowl.-Based Syst.*, vol. 235, 2021.

[18] G. Hu, X. N. Zhu, and G. Wei, “An improved marine predators algorithm for shape optimization of developable Ball surfaces,” *Engineering Applications of Artificial Intelligence*, vol. 105, 2021.

[19] L. Abualigah, D. Yousri, and M. A. Elaziz, “Aquila Optimizer: a novel meta-heuristic optimization algorithm,” *Computers & Industrial Engineering*, vol. 157, 2021.

[20] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of the ICNN’95-International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[21] G.-G. Wang, S. Deb, and Z. Cui, “Monarch butterfly optimization,” *Neural Computing & Applications*, vol. 31, no. 7, pp. 1995–2014, 2019.

[22] S. Li, H. Chen, and M. Wang, “Slime mould algorithm: a new method for stochastic optimization,” *Future Generation Computer Systems*, vol. 111, pp. 300–323, 2020.

[23] B. Abdollahzadeh, F. S. Gharehchopogh, and S. Mirjalili, “African vultures optimization algorithm: a new nature-inspired metaheuristic algorithm for global optimization problems,” *Computers & Industrial Engineering*, vol. 158, Article ID 107408, 2021.

[24] Z. Yang, L. Deng, and Y. Wang, “Aptenodytes Forsteri optimization: algorithm and applications,” *Knowl.-Based Syst.-* vol. 232, Article ID 107483, 2021.

[25] G.-G. Wang, “Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems,” *Memetic Computing*, vol. 10, no. 2, pp. 151–164, 2018.

[26] J. Tu, H. Chen, and M. Wang, “The Colony predation algorithm,” *Journal of Bionics Engineering*, vol. 18, no. 3, pp. 674–710, 2021.

[27] J. H. Holland, “Genetic algorithms,” *Scientific American*, vol. 267, no. 1, pp. 66–72, 1992.

[28] R. Storn and K. Price, “Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[29] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, “Multi-Verse Optimizer: a nature-inspired algorithm for global optimization,” *Neural Computing & Applications*, vol. 27, no. 2, pp. 495–513, 2016.

[30] F. A. Hashim, K. Hussain, and E. H. Houssein, “Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems,” *Applied Intelligence*, vol. 51, no. 3, pp. 1531–1551, 2021.

[31] I. Ahmadianfar, A. A. Heidari, and A. H. Gandomi, “RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method,” *Expert Systems with Applications*, vol. 181, 2021.

[32] R. V. Rao, V. J. Savsani, and D. P. Vakharia, “Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems,” *Computer-Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.

[33] M. Kumar, A. J. Kulkarni, and S. C. Satapathy, “Socio evolution & learning optimization algorithm: a socio-inspired optimization methodology,” *Future Generation Computer Systems*, vol. 81, pp. 252–272, 2018.

[34] D. Wei, Z. Wang, and L. Si, “Preaching-inspired swarm intelligence algorithm and its applications,” *Knowl.-Based Syst.* vol. 211, Article ID 106552, 2021.

[35] Y. Yang, H. Chen, and A. A. Heidari, “Hunger games search: visions, conception, implementation, deep analysis, perspectives, and towards performance shifts,” *Expert Systems with Applications*, vol. 177, Article ID 114864, 2021.

[36] J. S. Chou and D. N. Truong, “A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean,” *Applied Mathematics and Computation*, vol. 389, Article ID 125535, 2021.

[37] K. Luo, Y. Jiao, and J. Zhu, “Perturbation observer based fractional-order control for SMES systems based on jellyfish search algorithm,” *Frontiers in Energy Research*, vol. 9, Article ID 781774, 2021.

[38] M. D. Ortigueira and J. A. Tenreiro Machado, “What is a fractional derivative?” *Journal of Computational Physics*, vol. 293, pp. 4–13, 2015.

[39] E. J. Solteiro Pires, J. A. Tenreiro Machado, and P. B. de Moura Oliveira, “Particle swarm optimization with fractional-order velocity,” *Nonlinear Dynamics*, vol. 61, no. 1-2, pp. 295–301, 2010.

[40] H. M. Sultan, A. S. Menesy, and S. Kamel, “Parameter identification of proton exchange membrane fuel cells using an improved salp swarm algorithm,” *Energy Conversion and Management*, vol. 224, Article ID 113341, 2020.

[41] Y.-P. Xu, J.-W. Tan, and D.-J. Zhu, “Model identification of the proton exchange membrane fuel cells by extreme learning machine and a developed version of arithmetic optimization algorithm,” *Energy Reports*, vol. 7, pp. 2332–2342, 2021.

[42] L. Abualigah, M. Abd Elaziz, and P. Sumari, “Reptile Search Algorithm (RSA): a nature-inspired meta-heuristic optimizer,” *Expert Systems with Applications*, vol. 191, Article ID 116158, 2022.

[43] L. Abualigah, A. Diabat, and S. Mirjalili, “The arithmetic optimization algorithm,” *Computer Methods in Applied Mechanics and Engineering*, vol. 376, Article ID 113609, 2021.

[44] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, “GSA: a gravitational search algorithm,” *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[45] L. D. S. Coelho, “Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems,” *Expert Systems with Applications*, vol. 37, no. 2, pp. 1676–1683, 2010.

[46] S. Mirjalili and A. Lewis, “The Whale optimization algorithm,” *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[47] J. Xue and B. Shen, “A novel swarm intelligence optimization approach: sparrow search algorithm,” *Systems Science & Control Engineering*, vol. 8, no. 1, pp. 22–34, 2020.

[48] I. Ahmadianfar, O. Bozorg-Haddad, and X. Chu, “Gradient-based optimizer: a new metaheuristic optimization algorithm,” *Information Sciences*, vol. 540, pp. 131–159, 2020.

[49] H. Shareef, A. A. Ibrahim, and A. H. Mutlag, “Lightning search algorithm,” *Applied Soft Computing*, vol. 36, pp. 315–333, 2015.

[50] I. Naruei and F. Keynia, “Wild Horse Optimizer: A New Meta-Heuristic Algorithm for Solving Engineering Optimization Problems,” *Eng.with Computer*, vol. 17, 2021.

[51] A.-b. Meng, Y.-c. Chen, and H. Yin, “Crisscross optimization algorithm and its application,” *Knowledge-Based Systems*, vol. 67, pp. 218–229, 2014.

[52] A. A. Heidari, S. Mirjalili, and H. Faris, “Harris hawks optimization: algorithm and applications,” *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[53] W. Zhao, L. Wang, and Z. Zhang, "Atom search optimization and its application to solve a hydrogeologic parameter estimation problem," *Knowledge-Based Systems*, vol. 163, pp. 283–304, 2019.

[54] Q. Askari, M. Saeed, and I. Younas, "Heap-based optimizer inspired by corporate rank hierarchy for global optimization," *Expert Systems with Applications*, vol. 161, Article ID 113702, 2020.

[55] G. Dhiman and V. Kumar, "Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems," *Knowledge-Based Systems*, vol. 165, pp. 169–196, 2019.

[56] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: theory and application," *Advances in Engineering Software*, vol. 105, pp. 30–47, 2017.

[57] J. Liu, D. Li, and Y. Wu, "Lion swarm optimization algorithm for comparative study with application to optimal dispatch of cascade hydropower stations," *Applied Soft Computing*, vol. 87, Article ID 105974, 2020.

[58] K. Hussain, M. N. M. Salleh, and S. Cheng, "On the exploration and exploitation in popular swarm-based metaheuristic algorithms," *Neural Computing & Applications*, vol. 31, no. 11, pp. 7665–7683, 2019.

[59] S. Mirjalili, "SCA: a Sine Cosine Algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[60] Y.-t. Zhao, W.-g. Li, and A. Liu, "Improved grey wolf optimization based on the two-stage search of hybrid CMA-ES," *Soft Computing*, vol. 24, no. 2, pp. 1097–1115, 2020.

[61] G. Dhiman, M. Garg, and A. Nagar, "A novel algorithm for global optimization: Rat Swarm Optimizer," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 8, pp. 8457–8482, 2021.

[62] G. Hu, M. Li, and X. F. Wang, "An Enhanced Manta ray Foraging Optimization Algorithm for Shape Optimization of Complex CCG-Ball Curves," *Knowl.-Based Syst*, vol. 204, Article ID 108071, 2022.

[63] G. Hu, J. Zhong, B. Du, and G. Wei, "An enhanced hybrid arithmetic optimization algorithm for engineering applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 394, Article ID 114901, 2022.