mathematics

MDPI

Article

Parameter Optimization in a Leaky Integrator Echo State Network with an Improved Gravitational Search Algorithm

Shuxian Lun [1,*,†], Zhenqian Zhang [1,†], Ming Li [1] and Xiaodong Lu [2,*]

[1] School of Control Science and Engineering, Bohai University, Jinzhou 121013, China
[2] School of Information Engineering, Suqian University, Suqian 223800, China
* Correspondence: jzlunzi@163.com (S.L.); lxd2211@sina.com.cn (X.L.)
† These authors contributed equally to this work.

Abstract: In the prediction of a nonlinear time series based on a leaky integrator echo state network (leaky-ESN), building a reservoir related to the specific problem is a key step. For problems such as poor performance of randomly generated reservoirs, it is tough to determine the parameter values of the reservoirs. The work in this paper uses the gravitational search algorithm (GSA) to optimize the global parameters of a leaky-ESN, such as the leaking rate, the spectral radius, and the input scaling factor. The basic GSA has some problems, such as slow convergence and poor balance between exploration and exploitation, and it cannot solve some complex optimization problems well. To solve these problems, an improved gravitational search algorithm (IGSA) is proposed in this paper. First, the best agent and elite agents were archived and utilized to accelerate the exploration phase and improve the convergence rate in the exploitation phase. Second, to improve the effect of the poor fitness agents on the optimization result, a differential mutation strategy was proposed, which generated new individuals to replace original agents with worse fitness, increasing the diversity of the population and improving the global optimization ability of the algorithm. Finally, two simulation experiments showed that the leaky-ESN optimized by the IGSA had better prediction accuracy.

Keywords: leaky integrator echo state network (leaky-ESN); gravitational search algorithm (GSA); differential mutation; time series prediction

MSC: 93-10

Citation: Lun, S.; Zhang, Z.; Li, M.; Lu, X. Parameter Optimization in a Leaky Integrator Echo State Network with an Improved Gravitational Search Algorithm. *Mathematics* **2023**, *11*, 1514. https://doi.org/10.3390/math11061514

Academic Editors: Zaixing He, Mingqiang Wei, Qiong Wang and Meng Wu

Received: 15 February 2023
Revised: 12 March 2023
Accepted: 15 March 2023
Published: 21 March 2023

1. Introduction

A recurrent neural networks (RNN) is a potent machine learning model with a short-term memory and internal parameters that share characteristics. An RNN has some advantages in the nonlinear prediction of time series, so it is often used in natural language processing and various kinds of time series prediction. However, RNNs have the characteristics of extensive computation and a slow convergence rate. In 2001, Professor Jaeger proposed a new type of recurrent neural network, an echo state network (ESN) [1], which could solve the problems associated with a recurrent neural network and predict the Mackey–Glass chaotic time series. The prediction accuracy of an echo state network is 2400 times better than that of an RNN [2]. ESNs have been used in many fields, such as time series prediction [3,4], system modeling [5,6], dynamic pattern recognition [7], and so on. Compared with a traditional recurrent neural network, an ESN has the following characteristics:

1. It replaces the hidden layers of a recurrent neural network with pools of randomly sparse connected neurons;
2. It can map the input vector to the state vector;
3. It uses the linear regression method or least-squares algorithm to obtain the output weights, simplifying the network training process and obtaining the global optimal value.

In addition to the output connection weight matrix, the global parameters of an ESN include the spectral radius of the reservoir connection weight matrix, the input and output feedback scaling, the leaking rate, and so on. These parameters are usually selected based on historical experience, which makes the global approximation capability of an ESN suboptimal.

Professor Jaeger proposed an improved model of ESN, which is called the leaky integrator echo state network (leaky-ESN) [8]. A leaky-ESN can optimize the spectral radius, the input scaling factor, and the leaking rate of the internal connection weight matrix using the stochastic gradient descent method (SGD), which improves its prediction accuracy to a certain extent. However, the stochastic gradient descent method is sensitive to the initial value. If the initial value is not suitable, the stochastic gradient descent method easily falls into the local minimum, so many scholars use different methods to optimize the global parameter of an ESN. Zhang et al. [9] used an improved multiverse optimizer (MVO) to optimize the hyperparameters of a deep ESN, and the model obtained an engine fault recognition rate of 93%. In [10], the genetic algorithm (GA) was applied to a double-reservoir echo state network and adapted to the global parameters of an ESN. Wang et al. [11] optimized the echo state network with the artificial fish swarm algorithm (AFSA), satisfying the control requirements of the polyvinylchloride (PVC) polymerizing process. Zhang et al. [12] used a hybrid algorithm that was composed of the simulated annealing algorithm (SA) and the gray wolf optimization algorithm (GWO) to optimize an ESN, which was used for coverage optimization of wireless sensor networks, reducing the redundant distribution of sensor nodes and ensuring the smooth operation of sensor networks. Li et al. [13] used the particle swarm optimization (PSO) algorithm to optimize the output weight matrix and the number of neurons in the reservoir, and they proposed two boundary mutation strategies that verified the effectiveness of the model in electrical load prediction. Salah et al. [14] used a PSO-ESN to predict the remaining service life of an engine. According to reference [15], output layer neurons only need a partial connection to internal neurons in a reservoir, so Wang et al. proposed a binary particle swarm optimization algorithm (BPSO) to calculate the output connection weight matrix. Han et al. [16] used the quantum-behaved fruit fly optimization algorithm (QFFOA) to determine the size of the reservoir, the spectral radius, and the preprocessing of the original data. Many intelligent optimization algorithms have been used to optimize the parameters of an ESN; however, due to the reservoir containing more nodes and the search space of parameters being broad, the optimization effect needs to be improved.

The gravitational search algorithm is a heuristic optimization algorithm that simulates the phenomenon of gravity. In the algorithm, search agents on behalf of the objects are optimized, and their masses measure their performance. All agents interact with each other by the gravitational force, and tiny agents always move towards agents with heavier masses. With the continuous movement of the agents, the heaviest agent moves to the optimal position, i.e., it is the optimal solution to the problem. The GSA has high performance and good global search ability in solving nonlinear problems compared with other traditional optimization algorithms. However, there are still some problems, such as slow convergence rate and poor balance between exploration and exploitation.

The rest of the paper is organized as follows. Section 2 presents a brief introduction to the leaky-ESN and the GSA. Section 3 analyzes the problems that exist in the exploration phase and exploitation phase of the GSA, and an improved GSA is proposed. The simulation results and analysis of the leaky-ESN optimized by the IGSA are presented in Section 4. Finally, the conclusion is presented in Section 5.

2. The Leaky-ESN and the GSA

2.1. The Leaky-ESN

The network structure of a standard ESN includes three layers: input layer, reservoir, and output layer, as shown in Figure 1. The input, $u = u(t)$, is connected to the reservoir through the input weight matrix, W^{in}; the output, $y = y(t)$, is fed back to the reservoir with

an output feedback weight matrix, W^{fb}; the state of the reservoir is $x = x(t)$; W denotes the weight matrix of the reservoir with state $x = x(t)$ through the output weight matrix, W^{out}, which is connected to the output layer. Assuming that the number of inputs, reservoirs, and outputs is M, N, and L, respectively, the dimensions of W^{in}, W, W^{fb}, and W^{out} are $N \times M$, $N \times N$, $N \times L$, and $L \times (M + N)$, respectively.

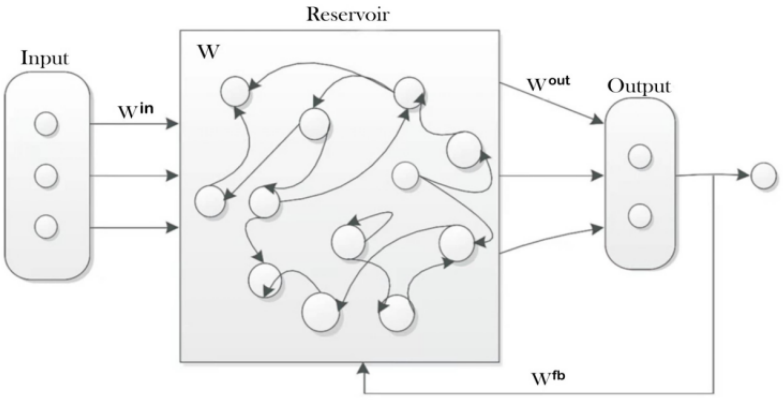


Figure 1. The structure of a standard ESN.

The state update equation of a standard ESN is as follows:

$$x(n+1) = f(W^{in}u(n+1) + Wx(n) + W^{fb}y(n)) \tag{1}$$

$$y(n) = g(W^{out}[x(n); u(n)]), \tag{2}$$

where f is an internal activation function (the tanh function is generally chosen), g is an output activation function (usually the identity function), and $[x; y]$ denotes the concatenation of the two vectors x and y. A leaky-ESN is an improved model of a standard ESN, and its reservoir is made up of leak-integrator neurons. A leaky-ESN has the same topology as the standard ESN model. The modified status update equation for a leaky-ESN is as follows:

$$x(n+1) = (1-a)x(n) + f(S^{in}W^{in}u(n+1) + \rho Wx(n) + S^{fb}W^{fb}y(n)), \tag{3}$$

where $a \in (0, 1]$ denotes the leaking rate, $\rho \in (0, 1]$ denotes the spectral radius of the weight matrix W of the reservoir, and S^{in} and S^{fb} are the input scaling factor and the output feedback scaling factor, respectively. To ensure that a leaky-ESN has the echo state property, Equation (4) must be satisfied.

$$a - \rho \geq 0. \tag{4}$$

In particular, only the output weight matrix, W^{out}, needs to be trained in a leaky-ESN and a standard ESN, while the rest of the connection weight matrix remains unchanged once it is determined. In a training ESN, the weight matrices, W^{in}, W, W^{fb}, are determined randomly and no longer change, except to train W^{out}. In the training stage, the state, $x(n)$ is collected in rows into X, and the output values, $y(n)$, corresponding to $x(n)$ are stored row by row in a vector, Y, so that the following formula can calculate W^{out}:

$$W^{out} = (X^T X + \epsilon I)^{-1} X^T Y, \tag{5}$$

where $\cdot^T$ denotes the transpose of a matrix or vector, ϵ is a regularization coefficient, and $\cdot^{-1}$ denotes the inversion of a square matrix.

A leaky-ESN can use the normalized root mean square error (E_{NRMSE}) to evaluate the training accuracy, and E_{NRMSE} is calculated as follows:

$$E_{NRMSE}(y, y_{teach}) = \sqrt{\frac{\langle || \, y(n) - y_{teach}(n) \, || \rangle^2}{\langle || \, y_{teach}(n) - \langle y_{teach}(n) \rangle \, || \rangle^2}}, \tag{6}$$

where $y(n)$ is the training output and y_{teach} is the teacher output, $\| \cdot \|$ denotes the Euclidean distance, and $\langle \cdot \rangle$ denotes the mean function.

2.2. The Gravitational Search Algorithm

The gravitational search algorithm (GSA) is a heuristic search algorithm based on the law of universal gravitation and the interaction between objects [17]. Fitness determines the mass, and the heavier agent represents a better solution in the GSA. Under the action of universal gravitation, an object will always move towards the object with the heaviest mass.

In the GSA, consider a model with N agents where the position of the ith agent is defined as follows:

$$X_i = (x_i^1, x_i^2 \dots x_i^d \dots x_i^D) \qquad for \;\; i = 1, 2 \dots D, \tag{7}$$

where x_i^d denotes the position of the ith agent in the dth dimension. The mass of each agent is calculated by Equation (8):

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{8}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)}, \tag{9}$$

where $M_i(t)$ represents the mass of the agent i at iteration t, $fit_i(t)$ represents the fitness of the agent i at iteration t, $best(t)$ and $worst(t)$ denote the optimal fitness and the worst fitness in the tth iteration, respectively.

The force acting on agent i is the sum of the forces acting on it by other agents:

$$F_i^d(t) = \sum_{j \in Kbest, j \neq i} rand_j^d G(t) \frac{M_j(t) M_i(t)}{R_{ij}(t) + \epsilon} (x_j^d(t) - x_i^d(t)) \tag{10}$$

where ϵ is a small constant and $R_{ij}(t)$ represents the Euclidean distance between agents i and j. Only gravitational agents can attract others, $Kbest$ is the number of attractive agents. As time goes by, $Kbest$ decreases linearly from N (total number of agents) to 1. The value $rand_j^d$ is a random number between 0 to 1, which gives a stochastic characteristic to the GSA. The gravitational constant, G, is a function of time, where the initial value of G, G_0, decreases with time:

$$G(t) = G_0 e^{-\beta \frac{t}{t_{max}}}, \tag{11}$$

where $\beta = 20$, t is the current number of iterations, and t_{max} is the total number of iterations.

The acceleration of the agent i at time iteration t is calculated as follows:

$$acc_i^d(t) = \frac{F_i^d(t)}{M_i(t)}. \tag{12}$$

Furthermore, the next position and velocity can be calculated as follows:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + acc_i^d(t) \tag{13}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1), \tag{14}$$

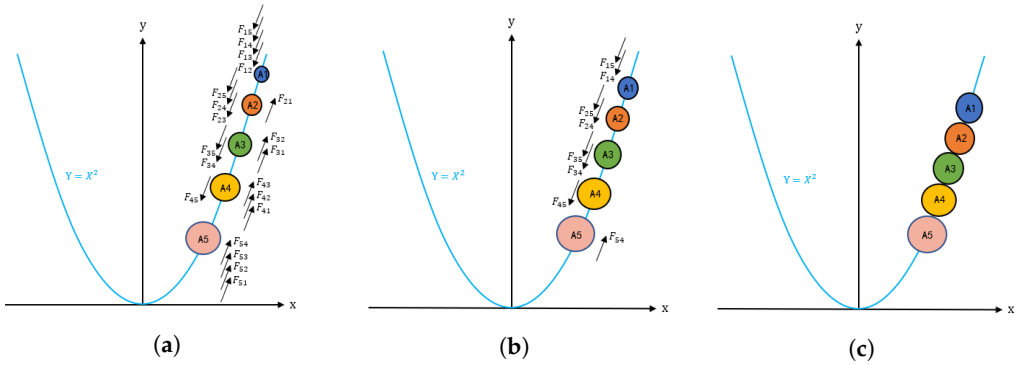
where d is the problem dimension and $rand_i$ is a random number in the interval [0,1].

3. The Improved Gravitational Search Algorithm (IGSA)

All optimization algorithms based on population behavior have two important characteristics: the ability to explore the wide search space, namely exploration, and the ability to

converge near the most promising candidate solution, namely exploitation. The exploration weakens and the exploitation strengthens gradually with each iteration. In the GSA, according to Equations (10) and (11), a high value of G can guarantee the exploration ability in the initial stage. As the program running G continues to decrease, the movement of the agents slows down, the exploration weakens, and the exploitation strengthens. Unfortunately, *Kbest* also decreases linearly over time and there are fewer and fewer gravitational agents, which can lead to the exploration weakening but the exploitation does not necessarily strengthen.

Figure 2 shows a case where the fitness function is $y = x^2$, where F_{mn} indicates that agent m is attracted by agent n. As we can see, A5 is the agent with the best fitness value, so A5 will always attract other agents to move toward it. However, A5 is also attracted to these agents and gradually moves away from the optimum, towards the center of all agents. After a period of time, the number of attractive agents continues to decrease and the strength of gravity decreases, which leads to a weakening in the exploration ability and a slow convergence rate in the exploitation, and the system fails to converge near the optimum. To solve these problems, we put forward the following solutions:

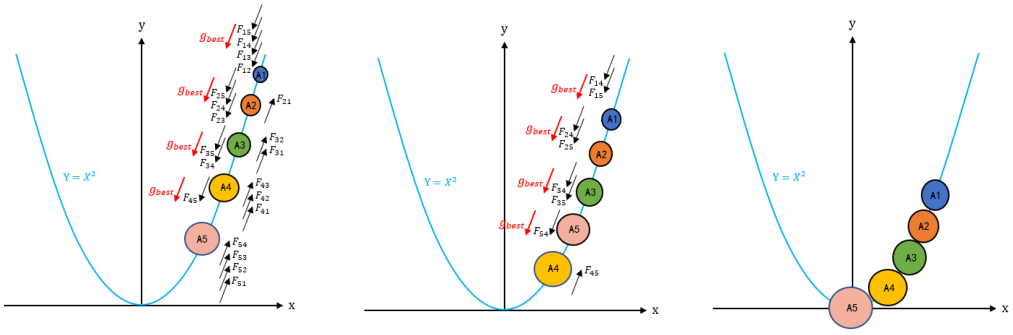


(**a**) (**b**) (**c**)

Figure 2. Movement behavior of agents in the GSA: (**a**) movement behavior of agents when $kbest = 5$; (**b**) movement behavior of agents when $kbest = 2$; (**c**) result of movement behavior.

3.1. The Adaptive Gbest and Elite-Agent-Guided GSA

This method mainly includes two scenarios: one is when the location of the agent with the best global fitness is saved and utilized to speed up the exploration phase; the second is when there are too few attractive agents, which results in slow convergence, and a fixed number of elite agents should be used to provide attraction and speed up the exploitation phase. Figure 3 shows the movement behavior of agents after joining the gbest and elite individuals. As shown in the figure, the gbest provides an additional attraction to the other agents, helping them move to the best global position, which can help a suboptimal agent to surpass the current optimal agent and become the new gbest in the next iteration. In general, 20% of agents with a better fitness value for each iteration are selected as elite agents, when the kbest value is less than the number of elite agents, elite agents will attract the others. As shown in Figure 3b, A4 and A5 are elite agents. When kbest < 2, elite agents attract other agents, ensuring that the lighter agents can quickly move to the heavier agents, improving the convergence rate, and, finally, the optimal solution can be found.



(**a**) (**b**) (**c**)

Figure 3. Movement behavior of agents after using the gbest and elite agents: (**a**) movement behavior of agents when $kbest = 5$; (**b**) movement behavior of agents when $kbest \leq 2$; (**c**) result of movement behavior.

When kbest is less than the number of elite agents, Equation (10) is changed to Equation (15):

$$F_i^d(t) = \sum_{j \in elite, j \neq i} rand_j^d G(t) \frac{M_j(t) M_i(t)}{R_{ij}(t) + \epsilon} (x_j^d(t) - x_i^d(t)), \tag{15}$$

where *elite* represents the elite agents.

The modified velocity formula and position formula using gbest are as follows:

$$V_i(t+1) = rand_i \times V_i(t) + c_1 \times acc_i(t) + c_2 \times (gbest - X_i(t)), \tag{16}$$

$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{17}$$

where $V_i(t)$ is the velocity of agent i at iteration t, $rand_i$ is a random number within 0 to 1, c_1 and c_2 are acceleration coefficients, $acc_i(t)$ is the acceleration of agent i at iteration t, and $gbest$ is the location of the global optimal solution.

3.2. The Differential Mutation Strategy

As can be seen from the above, the better agents may become the new global optimal solution through the guidance of the gbest, while the agents with poor fitness can only gradually approach the current optimal position through the guidance of the gbest and elite agents, and it is difficult to be the best agent after one iteration. To increase the probability of the inferior agents becoming the best agent in the next iteration, and improve the contribution to the optimization results, the work in this paper proposes a differential mutation strategy.

The agents are sorted according to fitness, where the half of agents that have better fitness are selected as the high-quality solutions, and the other agents are the inferior solutions. After the mutation operation of Equation (18), we obtain the new agents:

$$U_i = X_{r1} + F \cdot (X_{r2} - X_{r3}), \tag{18}$$

where U_i is the candidate agent after mutation, X_{r1} is an agent randomly selected from the inferior solutions, X_{r2} and X_{r3} are different agents randomly selected from the high-quality solutions, F controls the scale of the different agents, and, usually, $F = 0.5$.

By Equation (19), $U_i = (u_i^1, u_i^2 \dots u_i^d \dots u_i^D)$ is changed to $N_i = (n_i^1, n_i^2 \dots n_i^d \dots n_i^D)$ after crossover:

$$n_i^d = \begin{cases} u_i^d & rand(0,1) < CR \quad or \quad d = d_{rand} \\ x_i^d & otherwise, \end{cases} \tag{19}$$

where X_i is agent i from the inferior solutions, CR is the crossover probability, and d_{rand} is a random integer within the range of the dimension, D, which ensures that at least one dimension value comes from the agent generated by the mutation. The N_i values are new

agents that have been updated by the differential mutation strategy, and they form the new population together with the high-quality solutions.

In the process of generating new agents, the mutation and crossover make full use of the combination of better agents and inferior agents, increasing the probability of appearing as the better agents and the diversity of population, which both improve the global optimization ability of the algorithm. In the early stages of the algorithm, the agents are far apart, and the differential mutation strategy gives the algorithm a strong global search ability and the ability to escape the local optimum. In the later periods of the algorithm, the agents are close to each other, so the differential mutation strategy can improve the local search ability and speed up the convergence rate.

3.3. IGSA Evaluation

To evaluate the performance and the effectiveness of the IGSA, we applied it to 13 benchmark functions [18]. Table 1 shows the unimodal test functions, mainly showing the convergence rate rather than the final results. Table 2 shows the multimodal test functions, which have many local minima, so it is important for them to obtain a global optimal value. The minimum value of F_8 was $-418.9829 \times n$, the minimum values of the other functions were zero. The value n was the dimension of the functions in Tables 1 and 2.

Table 1. Unimodal test functions.

Test Function	Dimension
$F_1(X) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
$F_2(X) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$
$F_3(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^n$
$F_4(X) = \max\{\lvert x_i \rvert, 1 \le i \le n\}$	$[-100, 100]^n$
$F_5(X) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	$[-30, 30]^n$
$F_6(X) = \sum_{i=1}^{n} \left([x_i + 0.5] \right)^2$	$[-100, 100]^n$
$F_7(X) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	$[-1.28, 1.28]^n$

Table 2. Multimodal test functions.

Test Function	Dimension
$F_8(X) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert} \right)$	$[-500, 500]^n$
$F_9(X) = \sum_{i=1}^{n} \left[x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$	$[-5.12, 5.12]^n$
$F_{10}(X) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$	$[-32, 32]^n$
$F_{11}(X) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	$[-600, 600]^n$
$F_{12}(X) = \frac{\pi}{n} \{ 10 sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 sin^2(\pi y_{i+1})] + (y_n - 1)^2 \} + \sum_{i=1}^{n} u(X_i, 10, 100, 4)$	
$y_i = 1 + \frac{X_i + 1}{4}$	
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]^n$
$F_{13}(X) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_i + 1) \right] + (x_n - 1)^2 \left[1 + \sin^2(2\pi x_n) \right] \right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^n$

We applied the IGSA to these benchmark functions and compared the results with the GSA and PSO. In all algorithms, the population size was $N = 50$, the dimension was $n = 30$, and the total number of iterations was 1000. In PSO, positive constants $c1$ and $c2$ were 0.5 and the inertia factor was $w = 0.8$. In the GSA and the IGSA, G_0 was set to 100 and $\beta = 20$. For each algorithm, every function ran 30 times and the results are shown in Tables 3 and 4.

Table 3. Experimental results of unimodal functions.

		PSO	GSA	IGSA
F_1	Average best-so-far.	2.04×10^{-17}	4.18×10^{-22}	9.21×10^{-61}
	Average mean fitness.	1.11×10^{-17}	5.31×10^{-21}	2.21×10^{-59}
F_2	Average best-so-far.	5.81×10^{-10}	2.75×10^{-8}	3.75×10^{-19}
	Average mean fitness.	3.44×10^{-10}	2.11×10^{-8}	3.56×10^{-19}
F_3	Average best-so-far.	267.44	216.36	50.25
	Average mean fitness.	443.03	227.01	103.45
F_4	Average best-so-far.	2.72×10^{-5}	3.42×10^{-9}	3.67×10^{-10}
	Average mean fitness.	2.21×10^{-5}	3.35×10^{-9}	1.95×10^{-10}
F_5	Average best-so-far.	24.88	26.01	2.12
	Average mean fitness.	25.12	62.43	10.21
F_6	Average best-so-far.	0.75387	8.83×10^{-30}	0
	Average mean fitness.	1.0012	1.88×10^{-30}	0
F_7	Average best-so-far.	0.018	0.0078	0.00071
	Average mean fitness.	0.049	0.0204	0.0014

Table 4. Experimental results of multimodal functions.

		PSO	GSA	IGSA
F_8	Average best-so-far.	−3474.6477	−5731.6815	−6491.628
	Average mean fitness.	−3237.771	−3237.771	−5994.9608
F_9	Average best-so-far.	17.9093	11.9395	6.9496
	Average mean fitness.	24.874	15.9193	9.5961
F_{10}	Average best-so-far.	4.66×10^{-9}	5.08×10^{-11}	4.44×10^{-15}
	Average mean fitness.	3.18×10^{-9}	1.46×10^{-11}	4.21×10^{-15}
F_{11}	Average best-so-far.	3.92	0.15	0.0098
	Average mean fitness.	4.14	0.18	0.0265
F_{12}	Average best-so-far.	1.84×10^{-19}	4.12×10^{-24}	8.11×10^{-32}
	Average mean fitness.	1.38×10^{-19}	1.41×10^{-23}	1.18×10^{-31}
F_{13}	Average best-so-far.	0.0233	0.1957	1.46×10^{-7}
	Average mean fitness.	0.0963	0.2366	3.43×10^{-6}

As shown in Table 3, the IGSA provided better results than the GSA and PSO for functions in Table 1. This was especially visible in F_1 and F_2. Figures 4–6 show the convergence rate of all algorithms. According to these figures, the IGSA was the fastest to find the global optimum. This was due to the extra attractive power of the gbest and elite agents.

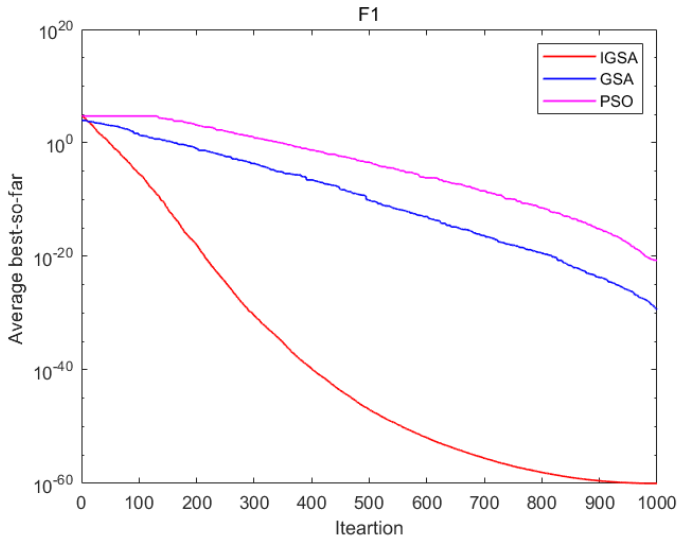


Figure 4. The convergence behavior of F_1.

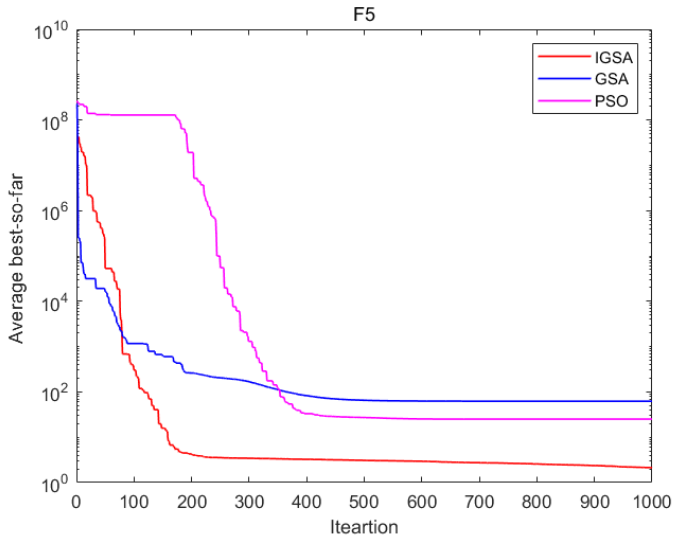


Figure 5. The convergence behavior of F_5.

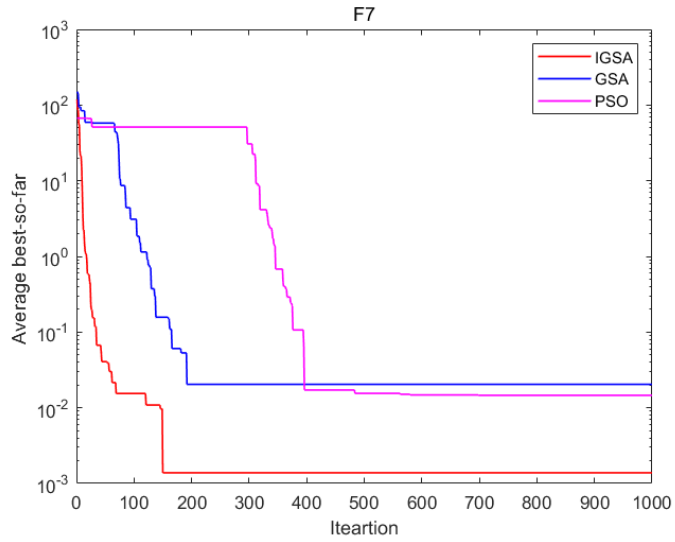


Figure 6. The convergence behavior of F_7.

For multimodal functions, the final result was the most important as it reflected an algorithm's ability to escape from local optima. As shown in Table 4, due to the differential mutation strategy, the IGSA had more chance to escape from local optima, so it had better results. The convergence behavior of some functions are shown in Figures 7 and 8.

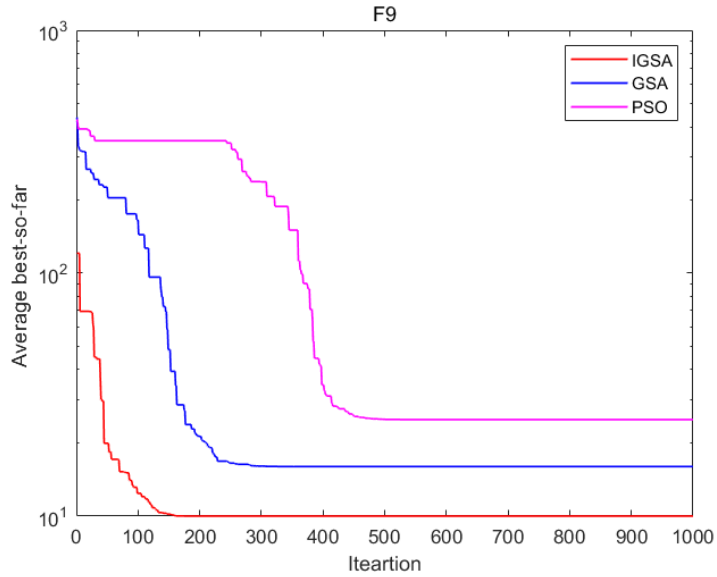


Figure 7. The convergence behavior of F_9.

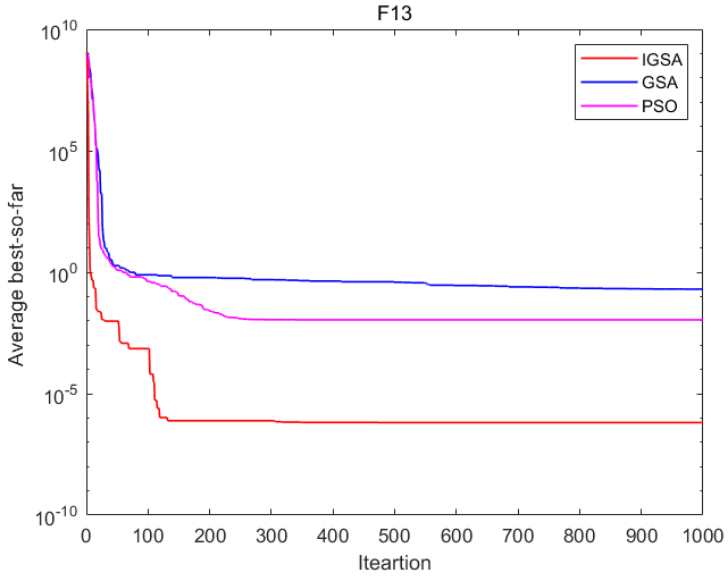


Figure 8. The convergence behavior of F_{13}.

3.4. The Process of Reservoir Parameter Selection Based on the IGSA

Due to the uncertainty of reservoir parameters in the leaky-ESN, the work in this paper used the improved gravitational search algorithm to optimize reservoir parameters and proposed a reservoir parameter selection model based on the IGSA, which gave the leaky-ESN better generalization ability and prediction performance.

The process of reservoir parameter selection was as follows: first, the initial population was generated using reservoir parameters and the fitness of all agents was calculated, the population was then updated with the gbest and elite agents and inferior agents using the differential mutation strategy were updated, eventually obtaining a new population. When the termination condition was met, the algorithm stopped and output the corresponding parameters. Otherwise, the optimization continued in the search space. The process of reservoir parameter selection is shown in Figure 9.

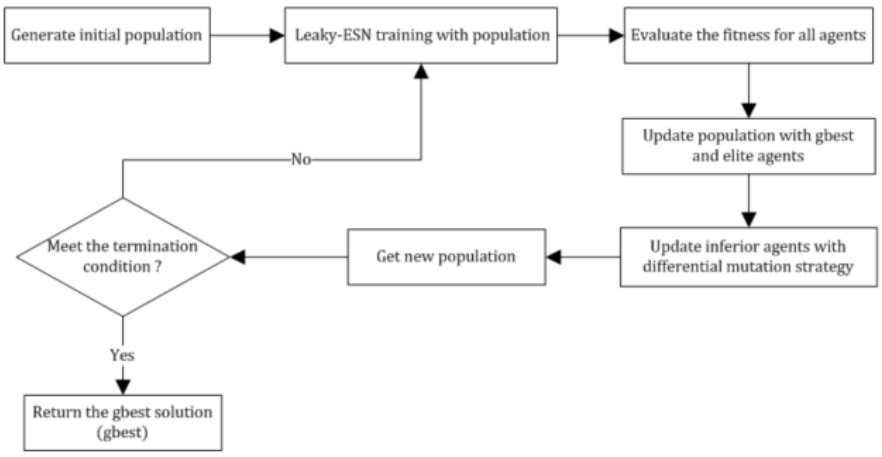


Figure 9. The process of reservoir parameter selection based on the IGSA.

Firstly, the input time series was preprocessed, which included noise removal and phase space reconstruction. The phase space reconstruction extended the time series into high-dimensional space and fully revealed the hidden information. Secondly, reservoir parameters were initialized, which included the input connection weight matrix, W^{in}, the reservoir connection weight matrix, W, and the leaky-ESN training model. Thirdly, the reservoir parameter selection model was used to obtain the optimal parameters, which included the leaking rate, a, the spectral radius, ρ, the input scaling factor, S^{in}, and the leaky-ESN prediction model with the optimal parameters. Finally, the leaky-ESN prediction model was used to predict the time series. The overall framework for time series prediction in the leaky-ESN is shown in Figure 10.

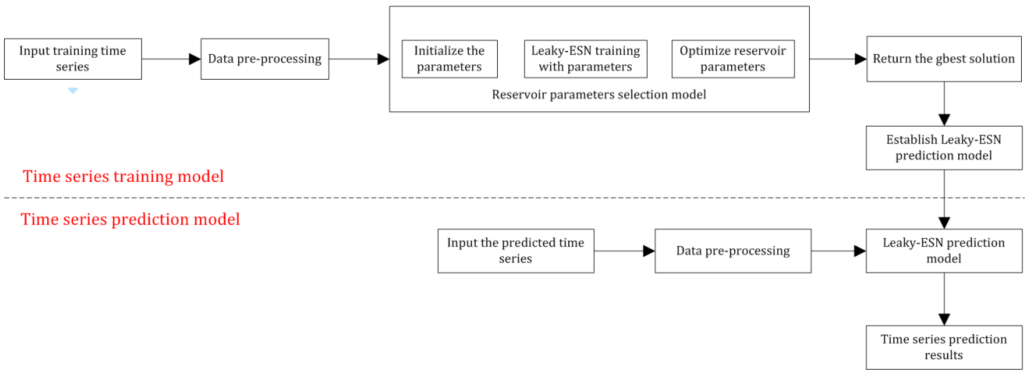


Figure 10. The overall framework for time series prediction in the leaky-ESN.

4. Simulation and Analysis

In this paper, we documented how we tested the performance of the leaky-ESN that was optimized by the IGSA using a time series prediction problem. The leaky-ESN that was optimized by the IGSA was compared with the leaky-ESN that was optimized by a basic GSA, the stochastic gradient descent (SGD) method, and the particle swarm optimization (PSO) algorithm. All of these algorithms optimized the leaking rate, a, the spectral radius, ρ, and the input scaling factor, S^{in}, of the reservoir parameters. In SGD, the learning rate, LR, was 0.0002. In PSO, positive constants $c1$ and $c2$ were 0.5, and the inertia factor $w = 0.8$. In the GSA and the IGSA, G_0 was set to 100 and $\beta = 20$. For the following time series prediction problems, the lengths of the training data and test data were both 10,000.

4.1. The Sine Time Series

The time series were created by the following equation:

$$u(n) = \sin(n) + \sin(0.51n) + \sin(0.22n) + \sin(0.1002n) + \sin(0.05343n). \quad (20)$$

This is a sum of essentially incommensurate sines with periods ranging from approximately 6 to approximately 120 discrete time steps. The desired output was $d(n) = u(n-5)$, indicating that this was a delay-5, short-term recall task. For SGD, the three groups of initial values, $X = [a, \rho, S^{in}]$, $X_1 = [0.33, 0.2, 0.3]$, $X_2 = [0.8, 0.2, 0.3]$, *and* $X_3 = [0.8, 0.66, 0.3]$ that are given in reference [8], were used. For the IGSA, the GSA, and PSO, 50 initial individuals were generated, respectively, under the condition of satisfying the echo state property. In each model, the input connection weight matrix, W^{in}, and the reservoir connection weight matrix, W, remain unchanged after random generation, and the output feedback weight matrix $W^{fb} = 0$. Meanwhile, the three models had the same number of reservoir units.

Figure 11 shows the training NRMSE of four different leaky-ESNs optimized by the IGSA, PSO, the SGD, and the GSA. It can be seen that all four algorithms had a good optimization effect. The SGD had the fastest convergence rate, but its NRMSE was the largest. The IGSA had a faster convergence rate than PSO and the GSA, and its NRMSE was the smallest among the three algorithms. The predicted errors of the four algorithms for the sine time series are shown in Figure 12. The prediction accuracy of the leaky-ESN that was optimized by using the IGSA is shown in Figure 13. We found that the IGSA's predicted error was the smallest and the IGSA had a good prediction accuracy.

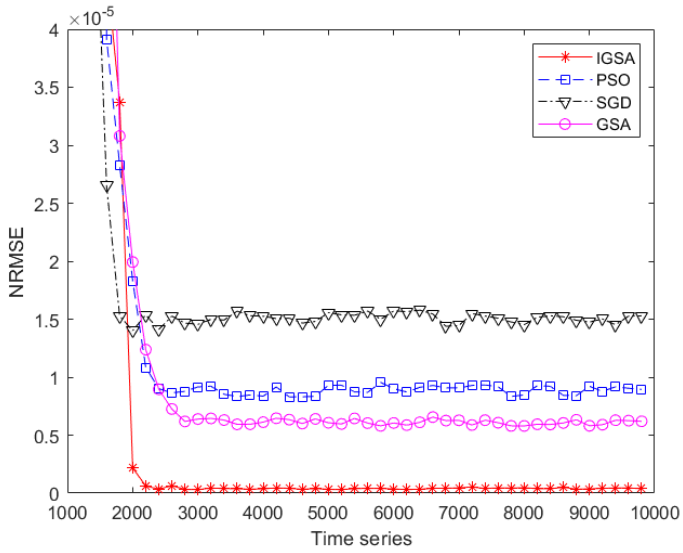


Figure 11. Training NRMSE for the sine time series.

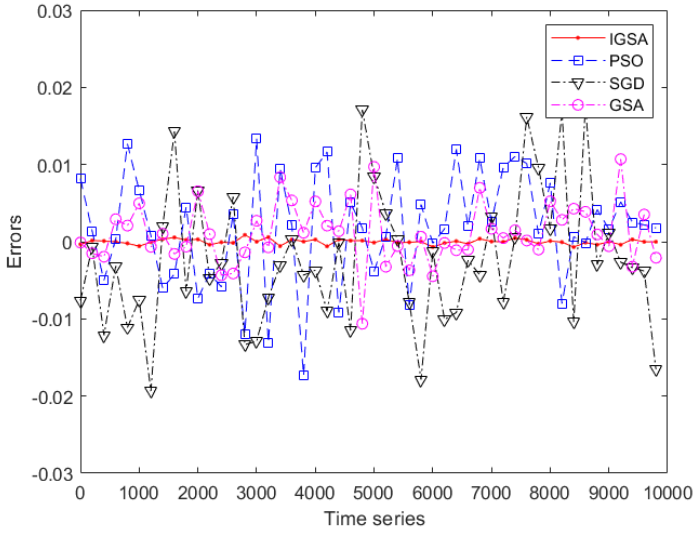


Figure 12. Predicted errors for the sine time series.

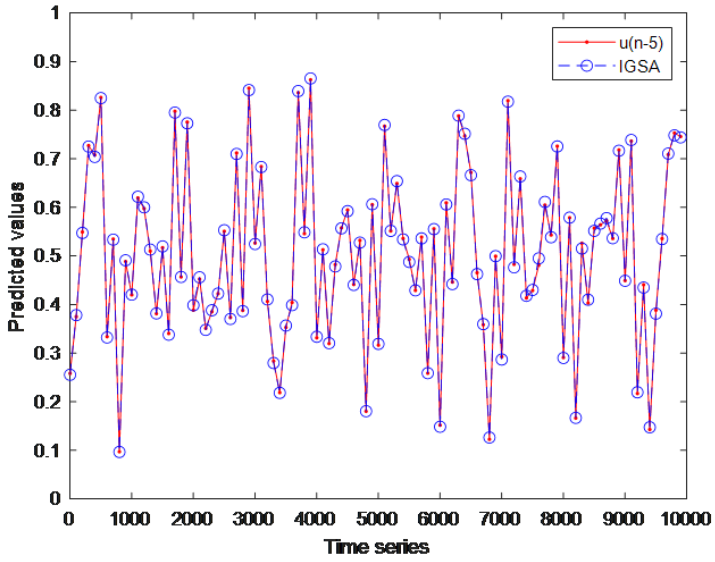


Figure 13. Predicted values for the sine time series.

4.2. The Mackey–Glass Chaotic Time Series

The Mackey–Glass time series is the standard data set in the research field of time series, which has obvious nonlinear and nonstationary characteristics. Due to its chaotic characteristics, it is difficult to predict accurately. The discrete-time form of the Mackey–Glass chaotic time series is as follows:

$$x(n+1) = x(n) + \triangle T \left(\frac{\alpha x(t - \frac{\tau}{\triangle T})}{1 + x(t - \frac{\tau}{\triangle T})^{10}} + \gamma x(n) \right). \tag{21}$$

In which, $\alpha = 0.2$, $\tau = 17$, $\gamma = -0.1$, and $\triangle T = 1/10$. The value τ denotes a delay factor and when $\tau > 16.8$ the time series (21) has a chaotic property. The initial values of the SGD and the leaky-ESN parameters were the same as the sine time series. Simulation results showed that the leaky-ESN that was optimized by the IGSA still showed good performance. Figure 14 shows that the IGSA had a minor training NRMSE. From Figures 15 and 16, we

found that the IGSA-optimized model had a smaller predicted error and could approximate the real value well.

The above two simulation experiments showed that compared with the leaky-ESN that was optimized by the SGD, PSO, and the GSA, the leaky-ESN that was optimized by the IGSA had better prediction performance.

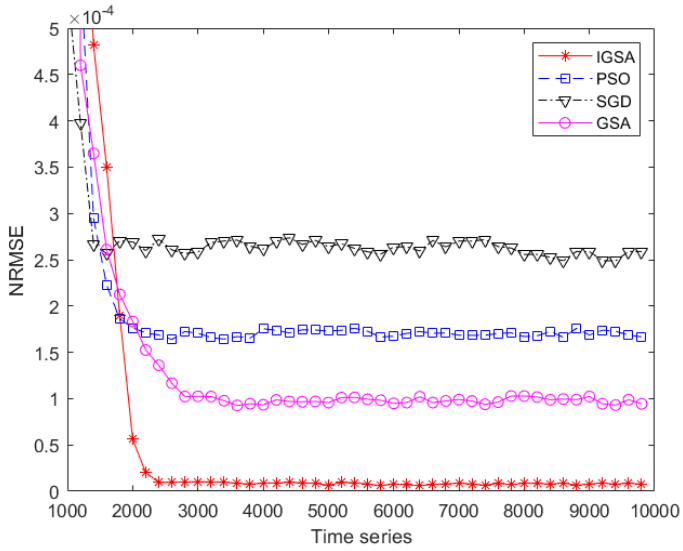


Figure 14. Training NRMSE for the MG time series.

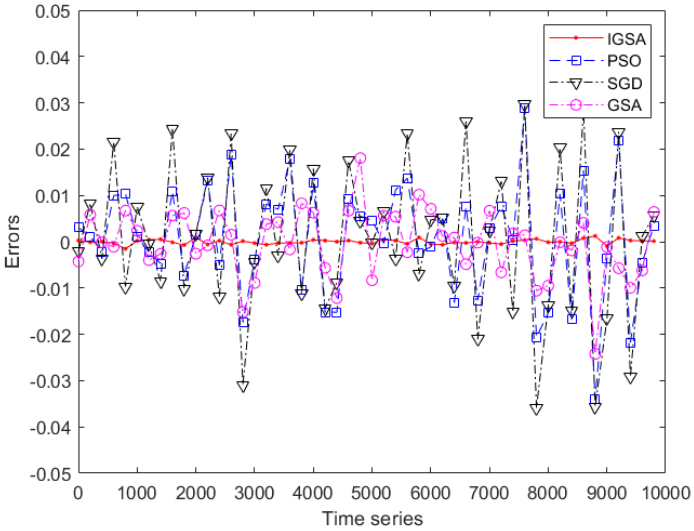


Figure 15. Predicted errors for the MG time series.

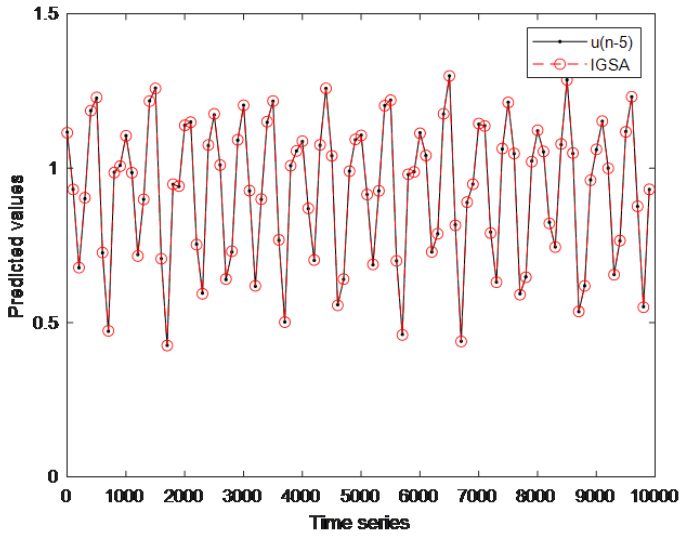


Figure 16. Predicted values for the MG time series.

5. Conclusions

The work in this paper applied the IGSA to a leaky-ESN to optimize the leaking rate, the spectral radius, and the input scaling factor and improve the leaky-ESN's prediction accuracy. Modifying the velocity formula, using the optimal global agent and elite agents to guide agents to move, gave the algorithm better global exploration and a faster convergence rate. Meanwhile, the work proposed a differential mutation strategy, which made full use of the combination of better agents and inferior agents, increasing the probability of the better agents appearing, and enhancing the ability to escape from the local optimum. The benchmark functions experiment showed that the IGSA could provide better final results and a faster convergence rate. Finally, two numerical experiments showed that the leaky-ESN that was optimized by the IGSA had better prediction accuracy.

Author Contributions: Methodology, Z.Z.; Software, Z.Z.; Validation, Z.Z.; Data curation, Z.Z.; Writing—original draft, Z.Z.; Writing—review & editing, S.L., M.L. and X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China (61773074) and Research Foundation of Education Bureau of Liaoning Province (LJKZZ20220118).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn Ger. Ger. Natl. Res. Cent. Inf. Technol. Gmd Tech. Rep.* **2001**, *34*, 13.
2. Jaeger, H.; Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* **2004**, *304*, 78–80. [CrossRef] [PubMed]
3. Zheng, K.; Qian, B. Long-short term echo state network for time series prediction. *IEEE Access* **2020**, *8*, 91961–91974. [CrossRef]
4. Kim, T.; King, B.R. Time series prediction using deep echo state networks. *Neural Comput. Appl.* **2020**, *32*, 17769–17787. [CrossRef]
5. Chen, Q.; Shi, H. Echo state network-based backstepping adaptive iterative learning control for strict-feedback systems: An error-tracking approach. *IEEE Trans. Cybern.* **2019**, *50*, 3009–3022. [CrossRef] [PubMed]
6. Liu, X.I.; Liu, Y.; Chen, Y. Trajectory design and power control for multi-UAV assisted wireless networks: A machine learning approach. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7957–7969. [CrossRef]
7. Wootton, A.J.; Day, C.R.; Taylor, S.L. Optimizing echo state networks for static pattern recognition. *Cogn. Comput.* **2017**, *9*, 391–399. [CrossRef]
8. Jaeger, H.; Lukoševičius, M. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Netw.* **2007**, *20*, 335–352. [CrossRef] [PubMed]
9. Li, X.; Bi, F.; Zhang, L. An Engine Fault Detection Method Based on the Deep Echo State Network and Improved Multi-Verse Optimizer. *Energies* **2022**, *15*, 1205. [CrossRef]

10. Zhong, S.; Xie, X. Genetic algorithm optimized double-reservoir echo state network for multi-regime time series prediction. *Neurocomputing* **2017**, *238*, 191–204. [CrossRef]
11. Wang, J.S.; Han, S. Echo state networks based predictive model of vinyl chloride monomer convention velocity optimized by artificial fish swarm algorithm. *Soft Comput.* **2014**, *18*, 457–468. [CrossRef]
12. Zhang, Y.; Cao, L.; Yue, Y. A novel coverage optimization strategy based on grey wolf algorithm optimized by simulated annealing for wireless sensor networks. *Comput. Intell. Neurosci.* **2021**, *2021*, 6688408. [CrossRef]
13. Wei, L.; Haitian, L. Electrical load forecasting using echo state network and optimizing by PSO algorithm. In Proceedings of the 2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA), Changsha, China, 9–10 October 2017.
14. Salah, S.B.; Fliss, I. Echo state network and particle swarm optimization for prognostics of a complex system. In Proceedings of the 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 30 October–3 November 2017.
15. Wang, H.; Yan, X. Optimizing the echo state network with a binary particle swarm optimization algorithm. *Knowl.-Based Syst.* **2015**, *86*, 182–193. [CrossRef]
16. Han, Y.; Jing, Y. Multi-step prediction for the network traffic based on echo state network optimized by quantum-behaved fruit fly optimization algorithm. In Proceedings of the 2017 29th Chinese Control and Decision Conference (CCDC), Chongqing, China, 28–30 May 2017.
17. Rashedi, E.; Nezamabadi-Pour, H. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
18. Wang, J.S.; Han, S. Evolutionary programming made faster. *IEEE Trans. Evol. Comput.* **1999**, *3*, 82–102.