Knowledge-Based Systems 88 (2015) 195–209

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A novel single multiplicative neuron model trained by an improved glowworm swarm optimization algorithm for time series prediction ☆

Huimin Cui [a,b,*], Jianxin Feng [a], Jin Guo [a], Tingfeng Wang [a]

[a] State Key Laboratory of Laser Interaction with Matter, Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China
[b] University of Chinese Academy of Sciences, Beijing 100049, China

ARTICLE INFO

Article history:
Received 17 January 2015
Received in revised form 23 July 2015
Accepted 25 July 2015
Available online 30 July 2015

Keywords:
Glowworm swarm optimization (GSO)
Differential evolution (DE)
Linearly decreasing inertia weight
Single multiplicative recurrent neuron (SMRN)
Local convergence
Time series prediction

ABSTRACT

To better predict time series, in this paper the single multiplicative recurrent neuron (SMRN) is constructed by adding feedforward and feedback links at the nodes of the original single multiplicative neuron (SMN). Glowworm swarm optimization (GSO) algorithm as a method for training the parameters of various kinds of neural network models gets easily into locally optimal traps during optimization process and its movement stability is also poor because of no memory about search history. To overcome the aforementioned disadvantages, firstly the linearly declining inertia weight is incorporated into the location update formula of standard GSO (LWGSO). After that in order to further enhance the robustness capability, differential evolution (DE) algorithm is introduced into LWGSO forming LWGSODE. Standard unimodal and multi-modal static test functions in high dimensions have been used to test its properties. The statistically experimental results show that the proposed LWGSODE approach performs much better than basic GSO whatever in terms of solutions precision, robustness or convergence speed. Moreover, the function optimization results are also competitive when compared with other state-of-the-art methods in the literature. Finally, the LWGSODE algorithm is used to train SMRN for time series prediction, and approximation results have improved significantly. All the results obtained reveal the novel SMRN model combined with the proposed LWGSODE algorithm provides a promising means to approximate nonlinear series in the future.

1. Introduction

Time series prediction is actually a complex function approximation problem in the real world. Among many existing techniques, neural networks (NNs) simulating the human brain NNs' structures and characteristics by mathematical tools have strong input–output mapping capabilities, which makes them a powerful tool for modeling time series [1,2,4,12,49]. There are various types of NNs according to different nodes, layers, connection distribution and activation functions such as multi-layer feed-forward architecture, radial basis function network and recurrent neural network [1–11,39].

Artificial single neuron model has been used to solve many engineering problems [12,14,15,49], exhibiting its advantages of simpler structure and lower computing complexity when compared with other neural network models. In [12], a single multiplicative neuron (SMN) model in which the input signal of the neuron is estimated by the multiplication function was proposed by Yadav, and its input–output mapping ability, nonlinear generalization ability, and noise fault tolerance capability were also studied in this literature. This model has been applied in many occasions where multi-layer neural networks with multiple neurons are needed [16]. For example, in [49] single multiplicative neuron model has been used to forecast time series and yielded excellent forecasting performance.

Considering the dynamically temporal property in the inter of NNs has great impact on network prediction performance, in the paper, feed-forward and feedback links have been added to the SMN to make it change dynamically with the system state's changes. That means the current outputs of SMRN model are a function of the previous outputs and the current inputs. Thus, SMRN has the advantage over SMN in feedbacking the data generated by the network which is to be used in future iterations, in which way the feedback path enables the network to learn

☆ This work was supported by the National 973 Program of China (Grant Nos. 51334020202-2 and 51334020204-2).

* Corresponding author at: State Key Laboratory of Laser Interaction with Matter, Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China.
E-mail address: cui_huimin2012@126.com (H. Cui).

http://dx.doi.org/10.1016/j.knosys.2015.07.032
0950-7051/

temporal patterns or sequences dynamically, thus capturing dynamically nonlinear properties more effectively.

Currently, there have been many various algorithms used to train NNs. Among them the mostly used one is the back-propagation (BP) algorithm. However, it does not provide a suitable solution. One main reason is that the objective function's gradient information must be know to BP algorithm. Another important reason is that it is very easy for the BP algorithm to get trapped into local optimum, especially for complex function approximation problems [14,21,24,26,35]. During recent decades, inspired from natural phenomena, many heuristic search algorithms have been developed [13] such as Gravitation Search Algorithm (GSA) [46], Particle Swarm Optimization (PSO) [36,39,49,51], Ant Colony Optimization (ACO) [19], Differential Evolution (DE) [17,34,40], Artificial Bee Colony (ABC) [20], Glowworm Swarm Optimization (GSO) [21], Artificial Immune System (AIS) [23] and Estimation of Distribution Algorithm (EDA) [18]. The above-mentioned algorithms as well as their variants have been proven efficient in a wide range of problems, such as function optimization [20,22,29,31,38,41,42], image processing [26], data clustering [25] and economic dispatch [23,24,48]. However, suffering from premature convergence as to miss the global optimum is usually inevitable for many of them. But it should be noted that there is no specific algorithm which is capable of obtaining adequately high performance for all the optimization problems in comparison with other alternatives [27]. In other words, some algorithms give a better solution for some particular problems than others. Hence, it remains an open issue to search for new effective heuristic algorithms for optimization areas [28].

In 2005, inspired from the nature phenomenon that glowworms exchange information of searching food with their peers through luciferin dissemination in the night, Indian scholars proposed a new kind of stochastic, meta-heuristic optimization algorithm, artificial GSO algorithm [21]. In the algorithm, glowworm particles as initial solutions are randomly distributed in the search space, and then each glowworm moves towards the brighter neighbors generation by generation just as the movement of glowworms in nature. Ultimately particles gather around the brightest one which locates at or near the optimal solutions of problems so as to achieve the purpose of optimizing functions. Compared with a majority of other intelligent algorithms, GSO is characterized as a simple concept that is easy to implement with higher computational efficiency and smaller numbers of parameters to adjust.

Following its introduction, dynamic decision domain for GSO was improved by Krishnan and Ghose in 2006. In 2009, the algorithm was applied to multi-extremum function optimization and was successfully applied to clustering analysis and wireless sensor network layout. Up to now, artificial GSO has exhibited excellent performance in multi-modal function optimization, multi-source tracing, harmful gas leak location, multi-source localization, combinational optimization and other problems [29–31].

However, the GSO algorithm is not free from the drawback of premature convergence in optimization process which is common for many heuristic, stochastic optimization algorithms. When premature convergence occurs glowworms tend to carry the same luciferin. That means particles will not be able to explore new areas, conducing low optimizing accuracy and slow search efficiency. In addition, since the original GSO is a memory-less algorithm and ignores each individual's fitness history in the process of iterations, its movement stability is also poor. To overcome the disadvantages, strategies of making glowworms jump out of local optimum to explore new areas more stably should be proposed.

Differential evolution (DE) proposed by Storn and Price is an optimization algorithm based on groups evolution, which uses the basic framework of the genetic algorithm but designs a unique differential mutation operator drawing on the Nelder Mead simplex method. Differential evolution algorithm can dynamically track the current search situation to adjust its search strategy by virtue of its unique ability of remember. It has strong global convergence capability and robustness, less undetermined parameters and fast convergence speed. More importantly, it is difficult to fall into local optima. DE has also been widely applied in practice, and many comparative studies have shown that DE is an algorithm with excellent global optimization performance and numerical stability [17,37,38,40–42].

In order to achieve more rapid convergence speed, stronger robustness and better convergence accuracy, an improved variant of GSO algorithm is proposed in this paper. The work is done as follows. First, to overcome premature convergence, linearly declining inertia weight is introduced to help advance the search into the regions that otherwise might not be exploited. Furthermore, in allusion to GSO's poor robustness the evolutionary operations of differential evolution algorithm have been made full use by generating new solutions through combining with the ever found best solution. Overall, The twofold modifications try to balance between the explorative and exploitative tendencies of the swarm with the objective of achieving better search performance.

This paper is organized as follows. The next section depicts the models of the original SMN and the proposed SMRN in detail. In the third section, a brief description of nature-inspired heuristic GSO algorithm and differential evolution algorithm is given. And the variant of GSO algorithm is also presented in the same section. The fourth section presents the comparative study of LWGSODE with PSO, GSO and other representative heuristic stochastic algorithms against ten recognized test functions as well as the degree of glowworms assemble for GSO and LWGSODE algorithms at different iterations for the same function. To show the efficiency of the SMRN model trained by LWGSODE algorithm, the fourth section provides prediction results of the SMRN model combined with LWGSODE algorithm for Box–Jenkins (BJ), Mackey–Glass (MG) and Electroencephalogram (EEG) time series and the results comparison with other algorithms has also been provided. Finally, the last section concludes the paper and gives some suggestions for the relevant future work.

2. The models of single multiplicative neuron and single multiplicative recurrent neuron

Generally, the fitness function to be calculated during the training of neural network models is the mean square error (MSE). It is calculated from the difference between output values and target values for all training samples as below:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(d_i - y_i)^2 \tag{1}$$

where d_i and y_i represent the target value and the output of network corresponding to the ith learning sample, respectively.

The output function for SMN model or the newly proposed SMRN model is logistic function whose output range is between 0 and 1, so it is very necessary to normalize time series data to the interval $[0.1, 0.9]$. Normalization formula is as follows [49]:

$$x_{new} = 0.1 + \frac{0.8(x_{old} - x_{min})}{x_{max} - x_{min}} \tag{2}$$

where x_{old} and x_{new} represent data sets before normalization and after normalization, respectively. x_{min} and x_{max} represent the minimum and the maximum among learning data sets before normalization.

2.1. Single multiplicative neuron (SMN)

The structure of single multiplicative neuron model for n inputs is given in Fig. 1.

The model has a single neuron and multiplication is performed to the signal coming into the neuron, which is different from other neural network models. Function $\Omega(\theta, x)$ is the product of the weighted inputs. As shown in Fig. 1 the multiplicative neural model with n inputs has $2*n$ parameters, of which n are the weights corresponding to the inputs and the others are the biases. Suppose that activation function is taken as logistic, which is given as follows [12,14,15,49]:

$$f(x) = \frac{1}{1+e^{-x}} \tag{3}$$

In this case, the net value of the neuron is obtained as follows:

$$net = \Omega(x,\theta) = \Pi_{i=1}^{n}(\omega_i x_i + b_i) \tag{4}$$

Thus, as the net value passes through activation function, output of the network is obtained as $y = f(net)$.

2.2. Single multiplicative recurrent neuron (SMRN)

With the deepening of neurophysiological studies, it has been recognized that biological nervous systems has the dynamic nature showing in the delay of synaptic connections, the pulse transmission and stimulated membrane excitability, etc. In order to simulate dynamic functions such as learning, adaptation, memory and recall, and thus better reflect the dynamic properties of biological neurons, it is necessary to introduce the feedback link into the mathematical modeling of biological neurons. In this paper we propose a single multiplicative recurrent neuron (SMRN) model, by adding feed-forward and feedback links in the feed-forward channel of the original SMN model as shown in Fig. 2. It is obvious from the figure that the dynamic SMRN model with feed-forward, feedback and delay links has different structures from traditional static SMN model. The following experimental results will prove that the introduction of feed-forward and feedback loops make SMRN have a stronger dynamic tracking capability than the original SMN.

$$s(k) = \prod_{i=1}^{n}(w_i x_i(k) + b_i) \tag{5}$$

$$u(k) = -b_{11}u(k-1) - b_{12}u(k-2) + a_0 s(k) + a_1 s(k-1) + a_2 s(k-2) \tag{6}$$

$$y(k) = f(u(k)) \tag{7}$$

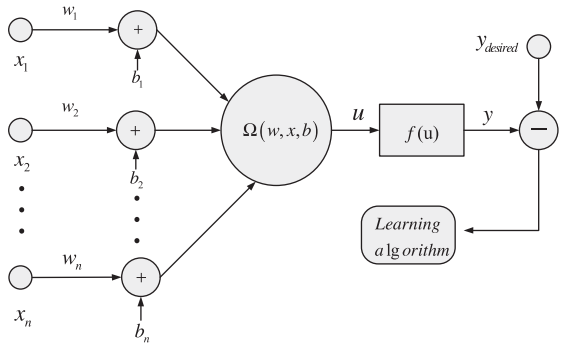

Fig. 1. The structure of single multiplicative neuron (SMN).

The model of SMRN is described in Fig. 2, where $x_i(k)$ $(i = 1, 2, \ldots, n)$ are the inputs to the whole system, w_i $(i = 1, 2, \ldots, n)$ are weights, b_i $(i = 1, 2, \ldots, n)$ are biases, $a_{ff} = [a_0, a_1, a_2]$ are feed-forward weights, $b_{ff} = [b_{11}, b_{12}]$ are feedback weights, $s(k)$ is the input to a linear dynamic system and is also the output to a linear dynamic system, $u(k)$ is the input variable to nonlinear activation function of sigmoid, $y(k)$ is the output to the whole system and k is the moment.

3. The glowworm swarm optimization algorithm and its variants

3.1. Glowworm swarm optimization (GSO) algorithm

In the artificial GSO algorithm each agent i is thought of as a glowworm that carries a luminescence quantity called luciferin $(l_i(t))$ along with them and emits a light intensity of which is proportional to the associated luciferin. Every agent i has a variable decision range r_d^i which is bounded by a circular sensor range r_s $(0 < r_d^i < r_s)$ and all other glowworms located within its current decision domain are identified as its neighbors. Once there are brighter neighboring glowworms, the current agent will be attracted and moves towards one of them based on a probabilistic mechanism.

Initially, all the agents contain an equal quantity of luciferin. During each iteration a luciferin-update phase is followed by a movement-phase which is based on a transition rule and a local-decision range update phase.

3.1.1. Luciferin update phase

The luciferin update depends on the function value at the glowworm's current position. Although all glowworms own the same luciferin value at the initial iteration, these values change according to the function values of their positions. Meanwhile, to simulate the decay in luciferin with time a fraction of the luciferin value is subtracted. The luciferin update rule is given by:

$$l_i(t) = (1-\rho)l_i(t-1) + \gamma \times f(x_i(t)) \tag{8}$$

where ρ is the luciferin decay constant $(0 < \rho < 1)$ and γ is the luciferin enhancement constant and $f(x_i(t))$ represents the value of the objective function at agent i's location at iteration t.

3.1.2. Movement phase

During the movement phase, each glowworm decides to move towards a brighter neighbor according to a probabilistic mechanism. For each glowworm i, the probability of moving towards a neighbor j is given by:

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)} \tag{9}$$

where $N_i(t)$ behaves agent i's neighbor muster during iteration t.

$N_i(t)$ is shown as follows:

$$N_i(t) = \{j : \|x_j(t) - x_i(t)\| < r_d^i(t);\ l_i(t) < l_j(t)\} \tag{10}$$

where $x_j(t)$ is the location of agent j at iteration t, $\|x_j(t) - x_i(t)\|$ represents the euclidian distance between glowworms i and j at iteration t, $l_j(t)$ represents the luciferin level associated with glowworm j at iteration t, $r_d^i(t)$ is the variable local-decision range associated with glowworm i at iteration t. Let the glowworm i select a glowworm j $(j \in N_i(t))$ with $p_{ij}(t)$ given by Eq. (9). Then the discrete-time model of the glowworm movements can be stated as:

$$x_i(t+1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}\right) \tag{11}$$

where s is the step-size.

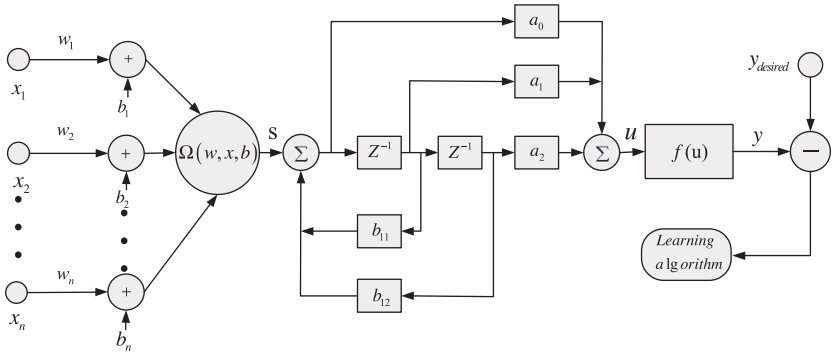

Fig. 2. The structure of single multiplicative recurrent neuron (SMRN).

3.1.3. Local-decision range update phase

When the glowworms determine their movements only depending on their local information, it is expected that the number of peaks captured would be a strong function of the radial sensor range. For instance, if the sensor range of each agent covers the entire workspace, all the agents move to the global optimum point thus ignoring the local optima. Sometimes, we need to detect multiple peaks for a special problem, in this case the sensor range must be made a varying parameter. That means, each agent i is associated with a local-decision domain whose radial range r_d^i is dynamic in nature. The local-decision domain update rule is given below:

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\}\} \tag{12}$$

where $r_d^i(t)$ is the agent i's decision radius at iteration t and satisfies $0 < r_d^i < r_s$, r_s is the sensor radial range, β is change rate of neighbor-domain and n_t is threshold for numbers of glowworms within decision range.

3.2. The differential evolution algorithm

In [41,42], DE is introduced to solve constrained optimization problems. It creates new candidate solutions by perturbing the parent individual with the weighted difference of several other randomly chosen individuals of the same population. A parent is replaced by the candidate only when it is better than its parent. Thereafter, DE guides the population towards the vicinity of the global optimum through repeated cycles of mutation, crossover and selection operation. The main procedure of DE is explained in detail as follows [50]:

3.2.1. Mutation

For each individual in n dimensions at generation $t, X_i^t = \{x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,n}^t\}\ \{i \in 1,2,\ldots,N\}$ where N is the number of agents in a swarm. An associated mutant individual $Y_i^t = \{y_{i,1}^t, y_{i,2}^t, \ldots, y_{i,n}^t\}\ \{i \in 1,2,\ldots,N\}$ can be created by using one of the mutation strategies. The most commonly used strategies are:

Rand/1:

$$y_{i,j}^t = x_{r[a],j}^t + F(x_{r[b],j}^t - x_{r[c],j}^t) \tag{13}$$

Best/1:

$$y_{i,j}^t = x_{best,j}^t + F(x_{r[a],j}^t - x_{r[b],j}^t) \tag{14}$$

where $r[k]\ (k \in 1,2,\ldots,N)$ is a uniformly distributed random number in the range $[1,N], x_{best,j}^t$ is the best individual of the population at generation t, and $F(F \in [0,3])$ is a amplification factor.

3.2.2. Crossover

DE applies a crossover operator on X_i^t and Y_i^t to generate the offspring individual $Z_i^t = \{z_{i,1}^t, z_{i,2}^t, \ldots, z_{i,n}^t\}\ \{i \in 1,2,\ldots,N\}$. The genes of Z_i^t are inherent from X_i^t or Y_i^t, determined by a parameter called crossover probability $CR \in [0,1]$, as follows:

$$z_{i,j}^t = \begin{cases} y_{i,j}^t, & \text{if } rand <= CR \text{ or } j = j_{rand} \\ x_{i,j}^t, & \text{otherwise} \end{cases} \tag{15}$$

where $rand$ is a uniformly distributed random number in the range $[0,1]$, and j_{rand} is a uniformly distributed random number in the range $[1,n]$.

3.2.3. Selection

The offspring individual Z_i^t competes against the parent individual X_i^t using the greedy criterion and the survivor enters the $t+1$ generation:

$$X_i^{t+1} = \begin{cases} Z_i^t, & \text{if } f(Z_i^t) < f(X_i^t) \\ X_i^t, & \text{otherwise} \end{cases} \tag{16}$$

where finding the minimum of function f is optimization objective.

3.3. The proposed algorithm

3.3.1. The LWGSO algorithm

In the speed update formula of standard particle swarm optimization algorithm, the inertia weight ω is usually taken to be a constant. Thus, it cannot make full use of feedback information in the particle search process. Consequently, it cannot make adjustments to the parameters of the update equation in time, causing unsatisfying experimental results. Through the study it is found that a greater inertia weight ω favors the global search and a smaller inertia weight ω is conducive to the local search. Generally, at early in the search process, using a larger inertia weight can make a stronger global search capability. In the latter evolution process it can enhance the local search ability to use a smaller inertia weight, thus reducing the number of iterations to find the optimal solution. Therefore, this paper chooses a linearly decreasing inertia weight strategy for GSO algorithm. The inertia weight is calculated as follows [32,33]:

$$\omega(t) = \frac{t_{max} - t}{t_{max}}(\omega_{max} - \omega_{min}) + \omega_{min} \tag{17}$$

where t_{max} behaves the maximum iteration number, t is the current iteration number, ω_{max} and ω_{min} behave the maximum and minimum inertia weight. Then, during the movement phase, Eq. (11) of the glowworm movements will be changed to be:

$$x_i(t+1) = \omega(t) \times x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}\right) \quad (18)$$

This is so called linearly decreasing inertia weight glowworm swarm optimization (LWGSO).

3.3.2. The proposed LWGSODE algorithm

In order to further improve the search performance of LWGSO, for each agent when there is no better solution in the perception of its area, a new solution will be generated randomly by the differential mutation operation. The difference between selected individuals is used to perturb a third individual (called target vector), thus obtaining a new trial solution. This process, dubbed mutation operator, can be performed by choosing one among many alternatives. If location of the newly produced solution is better it will replace the original solution and continue to evolve as the process of LWGSO.

The pseudocode of LWGSODE is described in detail as follows:

```
Initialize fluorescein volatile factor, fitness extraction ratio, rates of changes in the field, field threshold, fluorescein concentration, step, perception radius, decisions radius, population size, function space dimension, maximum number of iterations; set Cr and F; set the maximum inertia weight and minimum inertia weight; set upper and lower bounds of the variables.
while (t < the maximum iteration number)
  for i = 1 : n
    Update fluorescein according to Eq. (8)
  end
  Find location of the current optimal particle
  for i = 1 : n
    Find better particles in the decision-radius
      if (There is no better one)
        Operate mutation according to Eq. (13) or Eq. (14)
        Restrict the newly generated solution in the setting range
        Operate crossover according to Eq. (15)
        Operate selection according to Eq. (16)
        if (The new solution is better)
          Replace the original solution with the new one
        end
      else
        Select the better neighboring glowworm according to Eq. (10)
        Accumulate and normalize the selection probability according to Eq. (9)
        Decide to make forward to one glowworm finally
        Determine value of the inertia weight ω according to Eq. (17)
        Update the current location of the glowworm according to Eq. (18)
        Update the decision radius according to Eq. (12)
        if (The updated decision radius < 0)
          Decision radius = 0
        end
        if (The updated decision radius > the sensor radius)
          Decision radius = sensor radius
        end
      end
    t = t + 1
  end
  Select the optimal particle
end
```

4. Results and discussion

4.1. Unconstrained multidimensional functions optimization

It is usually difficult to derive the best solutions of some multi-dimensions functions because of their many peaks in search space, so they are commonly used to evaluate the performance of evolution algorithms. In the paper, in order to validate the good performance of the proposed LWGSODE, ten well-known benchmark functions with different complexity listed in Table 1 have been adopted. As we know, unimodal functions have only one local minimum while multimodal functions have more than one local minimum. Among the used functions, functions f_1–f_5 are unimodal while the remaining five functions are multimodal. Table 1 shows formulations, search domains, and minimum values for corresponding functions.

It should be noted that during experiments the initialization domain is set the same as its corresponding function's search range. Additionally, experiments for the same functions but in different dimensions have also been carried out and the relative convergence results are similar. Consequently, we only set optimization of functions in 30 dimensions as an example to clarify the proposed method's effectiveness. All the experiments are conducted in a Windows XP Professional system using Intel Core i5, 2.67 GHz, 2G RAM and the codes are performed in Matlab 2011b.

The numerical function optimization's task is to obtain the minimal value of each function. The results derived have been compared with the standard GSO, the commonly used PSO, LWGSO and other algorithms available in the literature.

For the comparisons to be fair, all algorithms are forced to use the value of each function as their fitness and have the same number of function evaluations. In addition, the same setting of parameters is assigned for GSO, PSO, LWGSO and LWGSODE. That is, in all the experiments carried out in the paper, the population size is set to be 50, the dimension of all the benchmark functions is determined as 30, and the maximum generation of glowworms' search is assumed as 1000. For PSO, $\omega = 0.5$, $c_1 = c_2 = 2$ while for LWGSO and LWGSODE ω is declining linearly with generation increasing as indicated in Eq. (17). In addition, for GSO as well as its variants, the fluorescein volatile factor $\rho = 0.4$, fitness extraction ratio $\gamma = 0.6$, rates of changes in the field $\beta = 0.08$, field threshold $n_t = 5$, fluorescein concentration $iot0 = 5$, step $s = 0.03$, perception radius $r_s = 500$ and decisions radius $r_0 = 500$. For LWGSODE algorithm, the mutation operator $F = 1.2$ and the crossover operator $C_r = 0.9$. For each algorithm and each function, 50 dependent runs are performed. The best solutions, the worst solutions, the average standard deviations and the total computation time of CPU for four methods with 50 runs over all functions are presented in Table 2.

It can be observed from the numerical optimization results in Table 2 that LWGSO and LWGSODE greatly outperform GSO and PSO with better best, worst, mean and *standard deviation* items for all the test functions. The derived best, worst values and the mean best results illustrate a better exploring ability for promising area and a better exploiting ability for locating the optima. We can conduct robustness comparisons of the algorithms by checking the standard deviations. With respect to significantly low standard deviations of LWGSO and LWGSODE in Table 2, we conclude both of them own more stable agents' movements compared with GSO and PSO. For the comparison of LWGSO and LWGSODE, the mean convergence precise and the average standard deviation in LWGSODE method is a little higher than LWGSO or at least comparative with it for all the ten functions except for f_6. That illustrates the introduction of DE not only further improve LWGSO's ability of searching global optimum but also enhance its robustness.

Table 1
High-dimensional benchmark functions.

Benchmark functions	Domain	f_{min}
$f_1 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10,10]^n$	0
$f_2 = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-10,10]^n$	0
$f_3 = \sum_{i=1}^{n} x_i^2$	$[-100,100]^n$	0
$f_4 = \sum_{i=1}^{n} i x_i^4$	$[-10,10]^n$	0
$f_5 = \sum_{i=1}^{n} (0.2x_i^2 + 0.1x_i^2 \sin 2x_i)$	$[-10,10]^n$	0
$f_6 = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i + 10)$	$[-5.12,5.12]^n$	0
$f_7 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-32,32]^n$	0
$f_8 = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32,32]^n$	0
$f_9 = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-30,30]^n$	0
$f_{10} = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$	$[-5,10]^n$	0

Table 2
The performance comparison among PSO, GSO, LWGSO and LWGSODE with ten unconstrained high-dimensional functions. All results are the statistical results over 50 independent runs, where "*Best*", "*Worst*", "*Mean*", "*Stdard deviation*" are the best, the worst, the average, the standard deviation of the final results.

Function	Algorithm	Best	Worst	Mean	Standard deviation
f_1	PSO	7.90E+1	2.05E+2	1.43E+2	2.9E+1
	GSO	3.77E+1	5.65E+1	4.70E+1	4.25
	LWGSO	1.20E−4	5.00E−3	1.80E−3	1.2E−3
	LWGSODE	2.78E−4	2.00E−3	1.10E−3	4.09E−4
f_2	PSO	2.59E+4	5.35E+5	1.84E+5	1.15E+5
	GSO	4.25E+5	8.29E+6	2.82E+6	1.90E+6
	LWGSO	4.12E−8	4.47E−6	9.42E−7	9.27E−7
	LWGSODE	5.96E−9	2.46E−6	5.77E−7	5.43E−7
f_3	PSO	1.54E+3	5.35E+5	1.93E+4	1.10E+4
	GSO	4.63E+4	7.17E+4	6.12E+4	5.82E+3
	LWGSO	2.35E−9	6.42E−7	1.68E−7	1.63E−7
	LWGSODE	1.13E−8	6.94E−7	1.68E−7	1.63E−7
f_4	PSO	3.25E+3	3.53E+5	1.13E+5	7.68E+4
	GSO	3.73E+3	1.50E+4	9.16E+3	3.01E+3
	LWGSO	8.74E−19	1.46E−11	1.55E−12	4.13E−12
	LWGSODE	1.34E−16	1.46E−11	8.32E−13	2.24E−12
f_5	PSO	1.65E+1	1.35E+2	2.19E+1	2.19E+1
	GSO	1.04E+1	2.14E+1	1.67E+1	2.12
	LWGSO	3.44E−10	3.49E−7	4.41E−8	5.67E−8
	LWGSODE	1.81E−10	2.00E−7	3.60E−8	3.46E−8
f_6	PSO	1.79E+2	4.57E+2	3.22E+2	6.55E+1
	GSO	1.17E+2	2.28E+2	1.86E+2	1.91E+1
	LWGSO	1.65E−6	1.21E−4	3.25E−5	2.90E−5
	LWGSODE	8.77E−7	1.95E−4	3.85E−5	3.75E−5
f_7	PSO	1.09	2.07	1.51	2.40E−1
	GSO	2.05	2.40	2.21	8.00E−2
	LWGSO	2.84E−11	2.96E−8	7.91E−9	7.18E−9
	LWGSODE	1.44E−10	4.27E−8	7.24E−9	7.08E−9
f_8	PSO	7.13	2.03E+1	1.86E+1	2.33
	GSO	1.90E+1	2.00E+1	1.90E+1	1.30E−1
	LWGSO	6.28E−5	1.43E−3	3.06E−4	2.27E−4
	LWGSODE	6.82E−5	8.75E−4	2.99E−4	1.86E−4
f_9	PSO	1.87E+5	2.40E+8	3.91E+7	5.14E+7
	GSO	8.09E+7	1.34E+8	1.10E+8	1.21E+7
	LWGSO	2.66E+1	2.87E+1	2.60E+1	2.70E−2
	LWGSODE	2.66E+1	2.87E+1	2.60E+1	2.70E−2
f_{10}	PSO	7.08E+2	1.46E+3	1.08E+3	1.71E+2
	GSO	3.20E+2	5.39E+9	1.05E+9	1.47E+9
	LWGSO	3.75E−9	9.28E−7	2.43E−7	2.21E−7
	LWGSODE	1.08E−8	1.05E−6	2.43E−7	2.02E−7

In order to compare convergence rates of the methods, the mobility of average best fitness evaluation in logarithmic scale for the ten functions in 30 dimensions has been shown in Figs. 3 and 4. As Table 2 illustrates, it is obvious that there is a statistically significant difference between the four different algorithms in average convergence speed and precision. While GSO and PSO get stuck in local minima, LWGSO and LWGSODE escape from traps and acquire better results.

To further illustrate LWGSODE method's superiority, function optimization results including mean best solutions and standard deviations have been compared with GSA [46], MGSA [47], GSAWM [47], HS [44], APO [45] and pPSA [43] in Table 3.

From examined results provided in this table, the LWGSODE method finds best mean solutions for f_2 and f_7 and it is always located first three rank for all the test functions. More importantly, the mean rank of LWGSODE is first among all the listed competing algorithms, which means the proposed method is generally successful no matter for unimodal functions or multimodal functions.

In order to get a more thorough analysis of LWGSODE's performance, the searching process of GSO and LWGSODE is illustrated with more details. The contours for function f_6 in two dimensions have been plotted and the global optimum of it is $(0,0)$ which is just located in the center of each figure. In the LWGSODE algorithm, the agents' positions for different iterations are shown in Fig. 5. For comparison the agents' evolution process for the same function in GSO method has also been shown in Fig. 5. It is necessary to be mentioned that both GSO and LWGSODE start the search process with the same initial population. The optimization results at different iteration numbers have been depicted in Table 4.

Both Fig. 5 and Table 4 reveal that the GSO algorithm loses rapidly the exploration ability while the LWGSODE preserves the swarm diversity not only in the beginning iterations but also in the final iterations. In other words, the strong exploration ability of the proposed LWGSODE could be concluded, while the GSO misses the global optimum because of loss of diversity in the beginning iterations.

4.2. Time series prediction problems

4.2.1. Box–Jenkins (BJ) gas furnace time series prediction problem

Box–Jenkins gas furnace data is widely used to verify the performance of a new identification model [12,49]. Data is recorded from the methane–air mixed gas combustion process. The data sets ranging from time $t = 1$ to $t = 296$ include 296 pairs of data $y(t)$ and $u(t)$. $y(t)$ is the concentration of output CO_2, $u(t)$ is the input flow of gas. It is found that there is a good performance when using $y(t-1)$ and $u(t-4)$ to predict $y(t)$. This chapter uses 150 samples to train the proposed model and the testing is performed on 140 samples.

In simulation, the SMRN model has 9 parameters to be identified. The length of agents is set to be 30. In addition, $r_s = 100$, $r_0 = 70$, the maximum iteration number itermax $= 1000$, $F = 2$ and $C_r = 0.9$. The error precision goal of

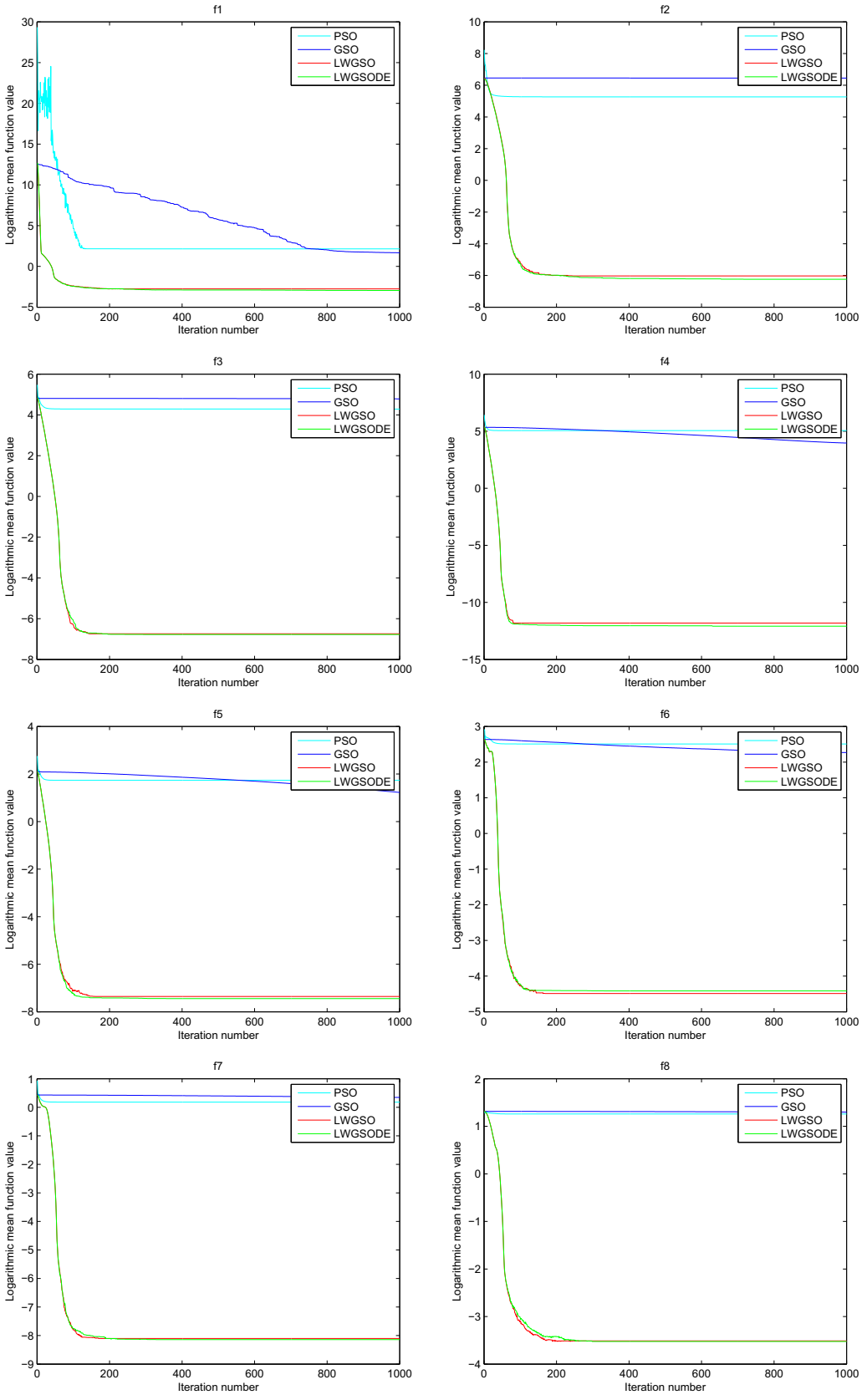

Fig. 3. Average best evolution performance comparison among PSO, GSO, LWGSO and LWGSODE for benchmark functions f_1–f_8.

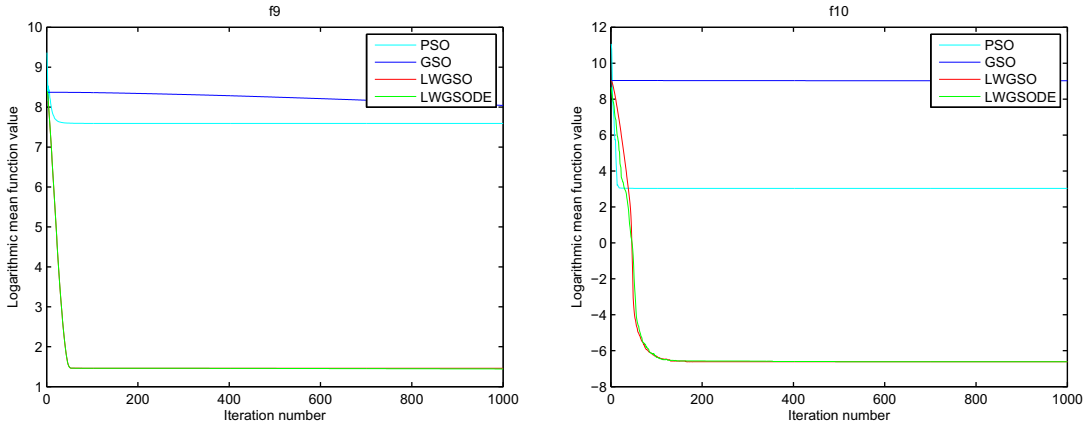

Fig. 4. Average best evolution performance comparison among PSO, GSO, LWGSO and LWGSODE for benchmark functions f_9–f_{10}.

Table 3
The performance comparison among GSA, MGSA, GSAWM, HS, APO and pPSA for function f_2,f_3,f_6,f_7,f_8,f_9. All results are the statistical results over 50 independent runs, where the digital in bracket is average result and the digital out of bracket is the standard deviation of the final results.

	LWGSODE	GSA	MGSA	GSAWM	HS	APO	pPSA
f_2	5.77E−7 (5.43E−7) 1	6.87E+1 (2.69E+1) 5	8.06E+1 (2.76E+1) 6	3.00E−1 (5.84E−1) 2	2.06E+1 (1.64E+1) 4	3.82E+2 (3.44E+1) 7	1.27E+1 (3.98E+1) 3
f_3	1.68E−7 (1.63E−7) 2	2.31E−16 (2.90E−17) 1	5.42E−4 (7.25E−4) 5	8.42E−4 (1.74E−3) 6	2.05E−7 (9.13E−5) 3	3.56E+2 (2.99E+1) 7	7.88E−6 (1.01E−6) 4
f_6	3.85E−5 (3.75E−5) 2	9.15 (1.93) 4	1.24E+1 (2.83) 5	1.11E−3 (1.48E−3) 3	2.75E−7 (8.74E−6) 1	3.17E+2 (1.19E+1) 7	8.11E+1 (2.04E+1) 6
f_7	7.24E−9 (7.08E−9) 1	5.66E−3 (8.38E−3) 4	5.79E−3 (8.59E−3) 5	4.42E−3 (9.71E−3) 3	7.31E−6 (1.07E−5) 2	3.18E+1 (4.02) 7	1.21E−2 (1.23E−2) 6
f_8	2.99E−4 (1.86E−4) 2	1.10E−8 (7.86E−10) 1	7.01E−2 (4.31E−3) 5	1.19E−2 (9.91E−3) 4	4.21E−3 (4.54E−3) 3	1.94E+1 (2.90) 7	1.19 (2.35) 6
f_9	2.6E+1 (2.70E−2) 3	2.63E+1 (7.70E−2) 4	2.77E+1 (3.27E−1) 5	3.34E−2 (3.98E−3) 1	2.56 (5.47) 2	7.30E+5 (1.06E+5) 7	6.37E+1 (8.96E+1) 6
Total	11	19	31	19	15	42	31
Rank	1	3	4	3	2	5	4

training and testing is set to be 0.0040. For each algorithm, 50 dependent experiments have been implemented.

The mean, the best, the worst solutions, the standard deviation and the success rate for training data sets have been presented in Table 5, in which the "baseline" methods (i.e., XPSO, GSO, BP and GA) [49] are with the "baseline" SMN model when they are used for BJ time series prediction. The results of LWGSODE method are superior to the original GSO. More importantly, when SMRN is combined with LWGSODE, the approximation results get significantly improved. The best mean MSE 0.0015, the smallest MSE 8.76E−4, the biggest MSE 0.0016 and the best standard deviation 2.69E−4 among all these listed algorithms have been derived and the success rate also adds up to 100%.

The testing results for BJ series have been presented in Table 6. It can be concluded from this table that the SMRN model trained by LWGSODE algorithm owns the best values considering all four items for 50 independent experiments. Thus, SMRN model based on LWGSODE training algorithm can be used to predict BJ time series better.

From Table 7 it can be seen that SMRN combined with LWGSODE has obtained the best MSE, MAD and *MAPE* values. In addition, the coefficient of determination R^2 indicating the goodness of fit of the proposed method is good.

The mobility of average best fitness for 50 runs is provided in Fig. 6. Fig. 7 depicts the real model, the training and testing approximation results. The approximation error for training and testing data sets has also been presented in Fig. 8.

In order to show how sensitive the proposed approach is to the initial settings of these parameters, how far the improvements shown in the experimental evaluation depend on the "lucky" selection of these parameters and how far they are "consistent"/"systematic" over various initial settings, different parameters settings have been made, i.e. the parameters at the beginning are $F = 2$, $C_r = 0.9$, $n = 30$, $r_s = 100$, $n_t = 5$, $\gamma = 0.6$ and experiments are preformed when single variant is changed. For each algorithm, 10 dependent experiments have been implemented. The training and testing results have been shown in Tables 8 and 9. From the two tables we know that when initial parameters change within small ranges, the training and testing results for all items are similar to the initial results. That means the proposed method is robust. However, each parameter has a threshold value. From the changes of parameter F we know when the value of F changes to

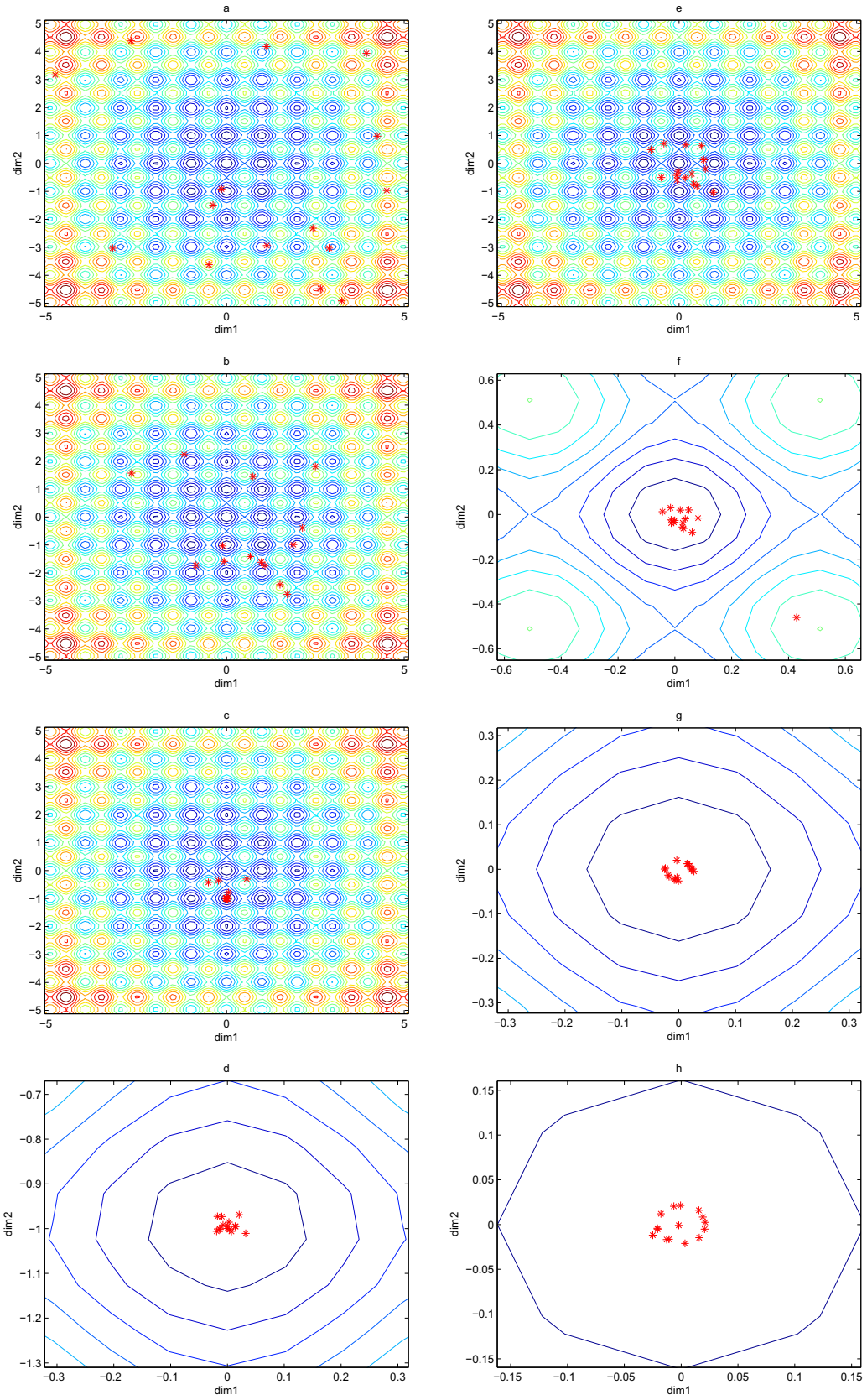

Fig. 5. The particles' positions for two dimensional function f_6. a, b, c and d exhibit the swarm positions of GSO in iterations 5, 100, 200 and 400, respectively. e, f, g and h exhibit the swarm positions of LWGSODE at iterations 5, 10, 20 and 100, respectively. In these figures a star sign represents the position of one particle of the swarm and the position of the optimum is located at (0,0).

Table 4
The function optimization results comparison of LWGSODE with GSO for f_6 at different iteration numbers. In the table x_1 behaves one dimension of function f_6, x_2 represents the other dimension, and y is the finally optimized function value within fixed iteration numbers.

Iteration numbers		5	10	20	100	200	400
GSO	x_1	−1.43E−1	–	–	−1.12E−1	3.37E−3	1.26E−4
	x_2	−9.19E−1	–	–	−1.03	−9.95E−1	−9.95E−1
	y	5.94	–	–	3.67	9.97E−1	9.95E−1
LWGSODE	x_1	9.64E−1	−5.80E−4	−3.02E−3	−2.06E−3	–	–
	x_2	−1.03	−2.06E−2	−1.84E−2	−8.85E−4	–	–
	y	2.49	1.40E−1	6.92E−2	9.97E−4	–	–

Table 5
The training performance comparison for predicting the BJ time series. All results are the statistical results over 50 independent runs, where "*Mean*", "*Best*", "*Worst*", "*Std.dev.*" and "*S.R.*" are the mean, the best, the worst, the standard deviation and the success rate of the results.

	XPSO	GSO	BP	PSO	GA	LWGSODE	LWGSODE + SMRN
Mean	0.0017	0.0018	0.0030	0.0029	0.0018	0.0018	0.0015
Best	0.0016	0.0016	0.0016	0.0016	0.0016	0.0016	8.76E−4
Worst	0.0039	0.0050	0.0080	0.0078	0.0042	0.0036	0.0016
Std.dev.	6.55E−4	8.33E−4	0.0019	0.0019	7.78E−4	7.74E−4	2.69E−4
S.R. (%)	100	74	66	67	96	100	100

Table 6
The testing performance comparison for predicting the BJ time series. "*Mean*", "*Best*", "*Worst*" and "*Std.dev.*" are the mean, the best, the worst and the standard deviation of the results.

	XPSO	BP	PSO	GA	LWGSODE	LWGSODE + SMRN
Mean	0.0021	0.0056	0.0054	0.0023	0.0026	0.0020
Best	0.0018	0.0019	0.0019	0.0018	0.0018	0.0018
Worst	0.0045	0.0095	0.0096	0.0046	0.0047	0.0040
Std.dev.	0.0015	0.0038	0.0040	0.0021	0.0014	4.17E−4

Table 7
The testing performance comparison for predicting the BJ time series. "*MSE*", "*MAD*", "*MAPE*" and "R^2" are the mean square error, the mean absolute deviation, the mean absolute percentage error and the coefficient of determination of the results.

	SVR	ANFIS	MLR	LWGSODE + SMRN
MSE	0.0033	0.1615	0.0157	0.0018
MAD	0.0479	0.3005	0.1251	0.0256
MAPE	0.1371	0.6263	0.2871	0.0629
R^2	0.8492	7.0846	3.6435	0.6248

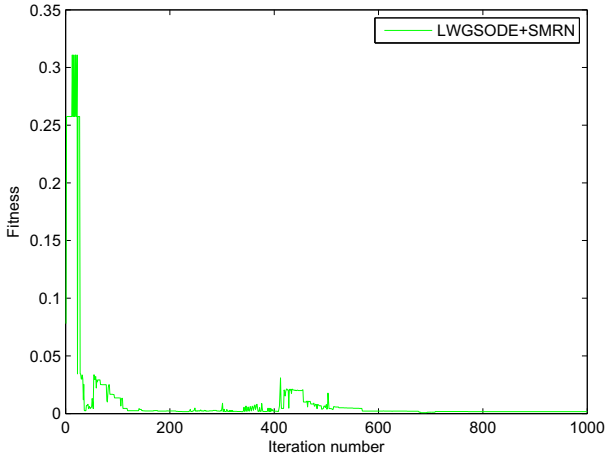

Fig. 6. The fitness mobility of LWGSODE for BJ time series prediction.

be 2.2 or 1.6, the training and testing improvements decrease obviously.

4.2.2. Mackey–Glass (MG) time series prediction problem

Mackey–Glass (MG) time series represents a model for white blood cell production in leukemia patients and has nonlinear oscillation which is often used to test the performance of neural network models as a benchmark problem [12,49]. It is a chaotic time sequence which is produced by the following formula [49]:

$$\frac{dy(t)}{dt} = \frac{ay(t-\tau)}{1+y^{10}(t-\tau)} - by(t) \tag{19}$$

where $\tau = 17$, $a = 0.2$, $b = 0.1$.

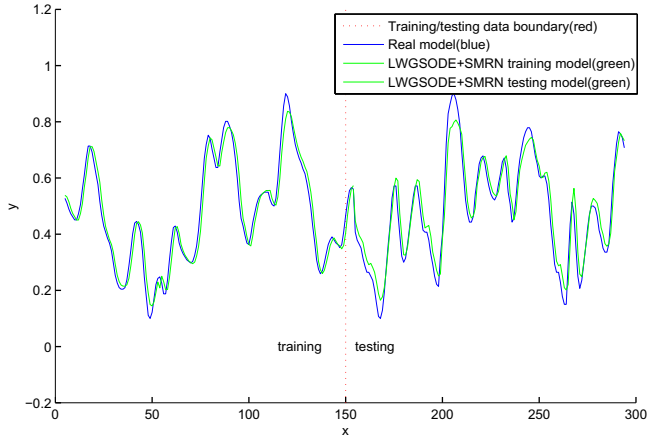

Fig. 7. The training and testing results of LWGSODE method for BJ time series prediction.

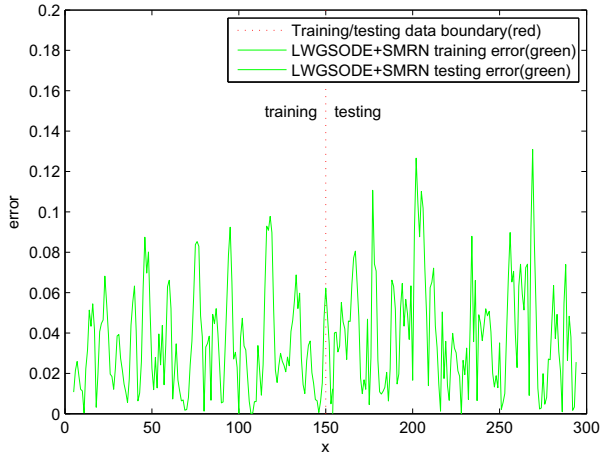

Fig. 8. The training and testing approximation error of SMRN model trained by LWGSODE algorithm for BJ time series prediction.

The destination of this research is to use four previous measurements, i.e. $y(t)$, $y(t-6)$, $y(t-12)$ and $y(t-18)$, to predict $y(t+1)$. In this case, 450 training samples are taken and 500 samples are used to test the model's generation property. The error goal of training and testing is set to be 0.001. In the SMRN model 13 parameters should be identified. The size of particles' population is set to be 100. In addition, $r_s = 30$, $r_0 = 27$, the maximum iteration number itermax $= 1000$, $F = 0.9$ and $C_r = 1.2$. The experiment has been carried out for 50 times independently.

The detailed training and testing results for different algorithms have been given in Tables 10 and 11, respectively. Similarly, in the two tables the "baseline" methods (i.e., XPSO, GSO, BP and GA) [49] are with the "baseline" SMN model when they are used for MG time series prediction. From the training table we know the results for the LWGSODE method has improved greatly in comparison with GSO for whatever items. Furthermore, the SMRN model combined with LWGSODE training algorithm has reached the best results for the Mean, Best, Worst, and *S.R.* items among all the

Table 8
The training performance comparison for predicting the BJ time series with different initial parameters. "*Mean*", "*Best*", "*Worst*", "*Std.dev.*" and "*S.R.*" are the mean, the best, the worst, the standard deviation and the success rate of the results.

	$F = 2.2$	$F = 2$	$F = 1.6$	$C_r = 0.8$	$n = 35$	$r_s = 110$	$n_t = 7$	$\gamma = 0.8$
Mean	0.0016	0.0016	0.0016	0.0016	0.0016	0.0016	0.0016	0.0016
Best	8.47E−4	8.75E−4	0.0016	8.76E−4	8.65E−4	8.77E−4	8.96E−4	9.01E−4
Worst	0.0025	0.0020	0.0028	0.0021	0.0021	0.0022	0.0022	0.0021
Std.dev.	8.73E−4	5.75E−4	3.11E−4	5.88E−4	5.78E−4	6.87E−4	6.90E−4	5.84E−4
S.R. (%)	98	100	81	98	98	98	97	96

Table 9
The testing performance comparison for predicting the BJ time series with different initial parameters. "*Mean*", "*Best*", "*Worst*" and "*Std.dev.*" are the mean, the best, the worst and the standard deviation of the results.

	$F = 2.2$	$F = 2$	$F = 1.6$	$C_r = 0.8$	$n = 35$	$r_s = 110$	$n_t = 7$	$\gamma = 0.8$
Mean	0.0030	0.0020	0.0179	0.0021	0.0020	0.0020	0.0021	0.0020
Best	0.0022	0.0018	0.0028	0.0018	0.0020	0.0019	0.0020	0.0019
Worst	0.0090	0.0042	0.0180	0.0045	0.0043	0.0051	0.0049	0.0045
Std.dev.	9.90E−4	4.20E−4	0.0141	4.52E−4	4.65E−4	4.50E−4	4.51E−4	4.83E−4

Table 10
The training performance comparison for predicting the MG time series. All results are the statistical results over 50 independent runs, where "*Mean*", "*Best*", "*Worst*", "*Std.dev.*" and "*S.R.*" are the mean, the best, the worst, the standard deviation and the success rate of the results.

	XPSO	GSO	BP	PSO	GA	LWGSODE	LWGSODE + SMRN
Mean	5.24E−4	0.0057	0.0038	0.0017	5.95E−4	0.0022	3.40E−4
Best	5.23E−4	0.0021	5.35E−4	5.25E−4	5.38E−4	4.05E−4	3.22E−4
Worst	0.0012	0.0096	0.0078	0.0052	0.0012	0.0031	9.62E−4
Std.dev.	2.19E−6	0.0038	0.0037	0.0025	7.17E−5	9.03E−4	2.58E−5
S.R. (%)	98	0	56	78	98	58	100

Table 11
The testing performance comparison for predicting the MG time series. "*Mean*", "*Best*", "*Worst*", "*Std.dev.*" are the mean, the best, the worst and the standard deviation of the results.

	XPSO	BP	PSO	GA	LWGSODE	LWGSODE + SMRN
Mean	5.49E−4	0.0046	0.0018	6.22E−4	0.0019	5.02E−4
Best	5.46E−4	5.65E−4	5.31E−4	5.18E−4	5.10E−4	4.52E−4
Worst	0.00139	0.0057	0.0046	0.0017	0.0030	9.87E−4
Std.dev.	3.05E−6	0.0040	0.0027	7.25E−4	0.0012	3.16E−4

Table 12
The testing performance comparison for predicting the MG time series. "*MSE*", "*MAD*", "*MAPE*" and "R^2" are the mean square error, the mean absolute deviation, the mean absolute percentage error and the coefficient of determination of the results.

	SVR	ANFIS	MLR	LWGSODE + SMRN
MSE	5.4947E−4	0.0591	9.1590E−4	5.7381E−4
MAD	0.0020	0.1501	0.0088	0.0015
MAPE	0.0032	0.2409	0.0142	0.0099
R^2	1.0018	3.1225	1.0286	1.0543

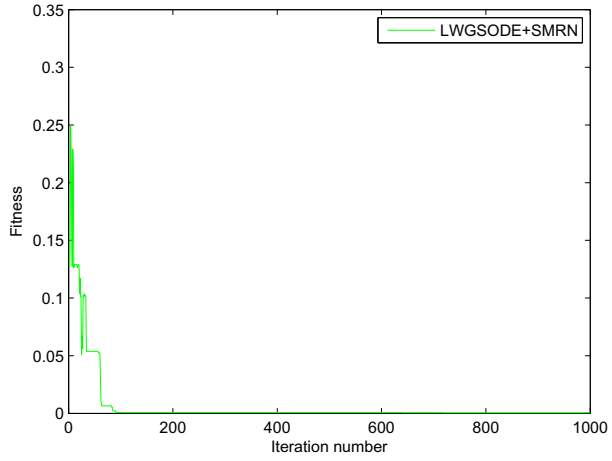

Fig. 9. The fitness mobility of LWGSODE for MG time series prediction.

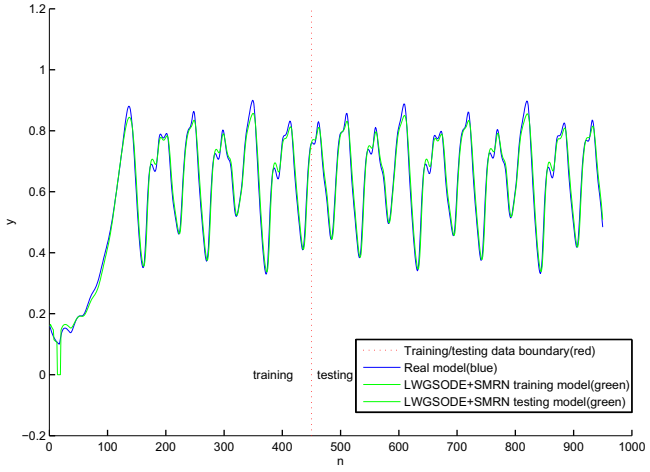

Fig. 10. The training and testing results of LWGSODE method for MG time series prediction.

listed algorithms. From the testing results table it is obvious that SMRN with LWGSODE performs much better than all the other methods for the Mean, Best, Worst items. In addition, the table also illustrates the SMRN combined with LWGSODE method has better standard deviation values compared with other methods. Overall, the novel SMRN model trained by the proposed LWGSODE training method is an effective alternative to predict MG time series.

From Table 12, it can be seen that SMRN combined with LWGSODE has obtained the best *MAD* values for MG prediction series. In addition, the proposed method has similar values of MSE, MAPE and R^2 with SVR whose coefficient of determination has reached 1.0018.

The mobility of average best fitness for 50 runs is presented in Fig. 9. Fig. 10 depicts the real model, the training and testing approximation for Mackey–Glass time series. The approximation error for training and testing data sets has also been presented in Fig. 11.

4.2.3. Electroencephalogram (EEG) time series prediction problem

EEG signal is the overall electrophysiological reflection of brain cells in the cerebral cortex. It includes very rich and useful information. Because EEG has characteristics of reflecting function changes of brain lesion, it has great significance in respects of physiological research and clinical applications such as epilepsy, encephalitis, brain tumors and other brain disease diagnoses. With the development of computers and signal processing techniques, EEG is playing an increasingly important role in the clinical diagnosis of brain diseases. One of the most obvious examples is the diagnosis of epilepsy.

Using four previous measurements, i.e. $y(t-1), y(t-2), y(t-4)$ and $y(t-8)$, to predict $y(t)$ is our research destination. In this case, 150 training samples are taken and 150 samples are used to test the model's generation property. The error goal of training and testing is set to be 0.009. In the SMRN model 13 parameters should be identified. The size of particles' population is set to be 50. In addition, $r_s = 500, r_0 = 498, \text{itermax} = 1000, F = 0.9$ and $C_r = 1.5$. The experiment has been carried out for 50 times independently.

The detailed training and testing results for different algorithms among which the "baseline" methods (i.e., XPSO, GSO, BP and GA) [49] are with the "baseline" SMN model when they are used for EEG time series prediction have been given in Tables 13 and 14, respectively. From the training table we know the results for the LWGSODE method have improved greatly in comparison with GSO. Furthermore, the SMRN model combined with LWGSODE training algorithm has reached the best results for the Mean, Best, Worst, Std.dev. and *S.R.* items among all the listed algorithms.

From the testing results table it is obvious that SMRN with LWGSODE performs better than all the other methods for the Mean, Best and *Worst* items. In addition, the table also illustrates the SMRN combined with LWGSODE method has better standard

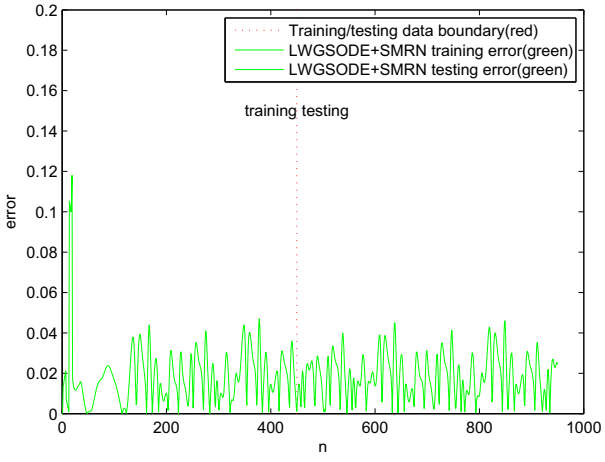

Fig. 11. The training and testing approximation error of SMRN model trained by LWGSODE algorithm for MG time series prediction.

Table 13
The training performance comparison for predicting the EEG time series. All results are the statistical results over 50 independent runs, where "*Mean*", "*Best*", "*Worst*", "*Std.dev.*" and "*S.R.*" are the mean, the best, the worst, the standard deviation and the success rate of the results.

	XPSO	GSO	BP	PSO	GA	LWGSODE	LWGSODE + SMRN
Mean	0.0082	0.0171	0.0142	0.0081	0.0081	0.0094	0.0080
Best	0.0080	0.0080	0.0082	0.0080	0.0080	0.0082	0.0070
Worst	0.0087	0.0301	0.0287	0.0081	0.0087	0.0098	0.0081
Std.dev.	4.56E−4	0.0030	0.0045	7.60E−9	5.77E−4	7.23E−4	2.23E−4
S.R. (%)	100	15	25	100	100	80	100

Table 14
The testing performance comparison for predicting the EEG time series. "*Mean*", "*Best*", "*Worst*", "*Std.dev.*" are the mean, the best, the worst and the standard deviation of the results.

	XPSO	BP	PSO	GA	LWGSODE	LWGSODE + SMRN
Mean	0.0067	0.0107	0.0066	0.0067	0.0069	0.0064
Best	0.0064	0.0066	0.0066	0.0064	0.0066	0.0063
Worst	0.0070	0.0145	0.0071	0.0070	0.0070	0.0067
Std.dev.	3.00E−4	0.0040	3.22E−7	3.00E−4	2.12E−4	2.00E−4

Table 15
The testing performance comparison for predicting the EEG time series. "*MSE*", "*MAD*", "*MAPE*" and "R^2" are the mean square error, the mean absolute deviation, the mean absolute percentage error and the coefficient of determination of the results.

	SVR	ANFIS	MLR	LWGSODE + SMRN
MSE	0.0090	0.0086	0.0087	0.0064
MAD	0.0687	0.0645	0.0708	0.0607
MAPE	0.1355	0.1212	0.1391	0.1144
R^2	0.0055	0.7972	0.1378	0.6337

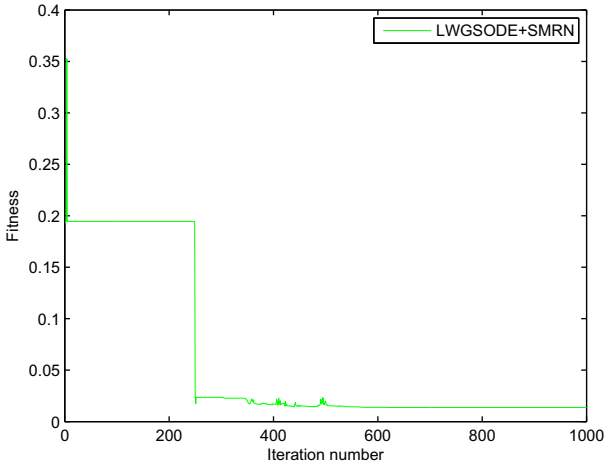

Fig. 12. The fitness mobility of LWGSODE for EEG time series prediction.

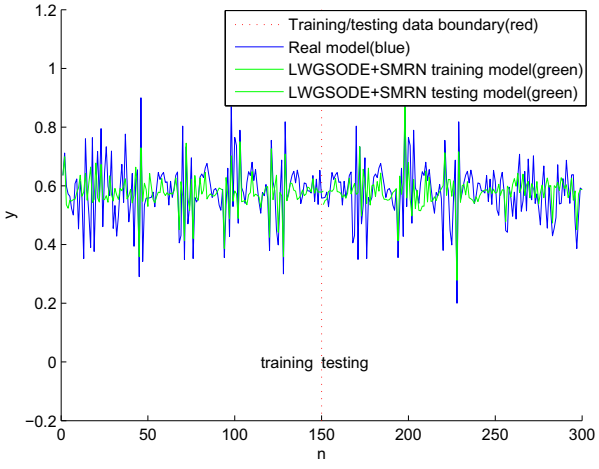

Fig. 13. The training and testing results of LWGSODE method for EEG time series prediction.

deviation values compared with other methods except PSO. Overall, the novel SMRN model trained by the proposed LWGSODE learning algorithm is effective to predict EEG time series.

From Table 15, it can be seen that SMRN combined with LWGSODE has reached the best MSE, MAD and *MAPE* values for EEG prediction series. Furthermore, the coefficient of determination of the proposed method is 0.6337 which is much better than SVR and MLR. The ANFIS has the best R^2, but its other values are inferior to the proposed method.

The mobility of average best fitness for 50 runs is presented in Fig. 12. Fig. 13 depicts the real model, the training and testing approximation for EEG time series. The approximation error for training and testing data sets has also been presented in Fig. 14.

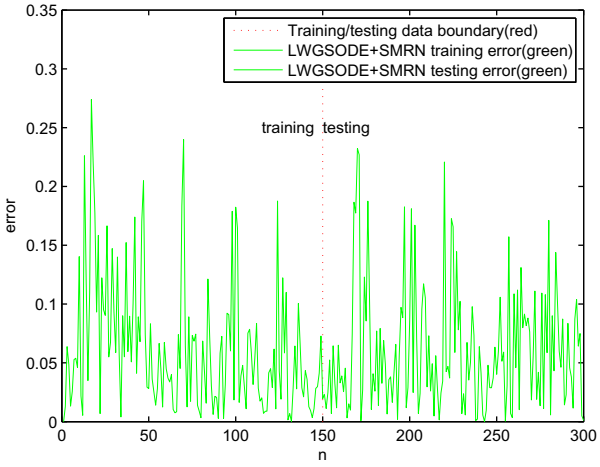

Fig. 14. The training and testing approximation error of SMRN model trained by LWGSODE algorithm for EEG time series prediction.

5. Conclusions

In order to improve the SMN's capability of time series prediction, SMRN, a variant of SMN, is firstly proposed in the paper. After that, GSO employed as a training method for SMN model has been improved significantly in allusion to its disadvantages of premature convergence and poor robustness. In the first class, a variable parameter, linearly decreasing inertia weight, is introduced into the location update formula of standard GSO to keep balance between the exploration and exploitation abilities of the original algorithm. In the second class DE is hybrided with LWGSO to enhance the swarm's memory ability and swarm diversity, thus further improving stability of agents' movements during evolution process. The numerical optimization results show LWGSODE converge faster, more precisely and more stably than GSO itself, and comparison with other state-of-the-art algorithms on several commonly used benchmark functions has also confirmed the proposed algorithm's superiority. Finally, when SMRN is trained by LWGSODE, its function approximation capability has been tested on three famous time series prediction cases. Simulation results exhibit better or at least comparative performance as compared with those of the SMN model which is trained by several other different algorithms. Thus, it can be concluded that the proposed SMRN model effectively enhances the function approximation ability of the original SMN and the combination of SMRN with LWGSODE gives a perspective approach for time series prediction. In future work, we will focus on applying this novelly combined method to solve other more complex approximation problems.

References

[1] Z.H. Guo, J. Wu, H.Y. Lu, J.Z. Wang, A case study on a hybrid wind speed forecasting method using BP neural network, Knowl.-Based Syst. 24 (2011) 1048–1056.
[2] J.P. Donatea, P. Cortez, Evolutionary optimization of sparsely connected and time-lagged neural networks for time series forecasting, Appl. Soft Comput. 23 (2014) 432–443.
[3] R. Chao, N. An, J.Z. Wang, L. Li, B. Hu, D. Shang, Optimal parameters selection for BP neural network based on particle swarm optimization: a case study of wind speed forecasting, Knowl.-Based Syst. 56 (2014) 226–239.
[4] W. Shen, X.P. Guo, C. Wu, D.S. Wu, Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm, Knowl.-Based Syst. 24 (2011) 378–385.
[5] P.J. Angeline, G.M. Sauders, An evolutionary algorithm that constructs recurrent neural networks, IEEE Trans. Neural Networks 5 (1994) 54–65.
[6] Y. Ji, X. Li, S. Wang, An improved particle swarm optimization for evolving feedforward artificial neural networks, Neural Process. Lett. 26 (3) (2007) 217–231.
[7] J.R. Zhang, J. Zhang, T.M. Lok, M.R. Lyu, A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training, Appl. Math. Comput. 185 (2007) 1026–1037.
[8] R.K. AlSeyab, Y. Cao, Differential recurrent neural network based predictive control, Comput. Chem. Eng. 32 (2008) 1533–1545.
[9] R.K. AlSeyab, Y. Cao, Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation, Process Control 18 (2008) 568–581.
[10] M.M. Noel, B.J. Pandian, Control of a nonlinear liquid level system using a new artificial neural network based reinforcement learning approach, Appl. Soft Comput. 23 (2014) 444–451.
[11] X.L. Deng, P.F. Zhou, Nonlinear identification based on diagonal recurrent neural network and particle filter, Nat. Comput. (2009) 217–221.
[12] R.N. Yadav, P.K. Kalra, J. John, Time series prediction with single multiplicative neuron model, Appl. Soft Comput. 7 (2007) 1157–1163.
[13] W.C. Yeh, Orthogonal simplified swarm optimization for the series-parallel redundancy allocation problem with a mix of components, Knowl.-Based Syst. 64 (2014) 1–12.
[14] E.M. Iyoda, H. Nobuhara, K. Hirota, A solution for the N-bit parity problem using a single multi-plicative neuron, Neural Process. Lett. 18 (2003) 233–238.
[15] A. Yadav, D. Mishra, R.N. Yadav, S. Ray, P.K. Kalra, Time series prediction with single integrate-and-fire neuron, Appl. Soft Comput. 7 (2007) 739–745.
[16] M. Schmitt, VC dimension bounds for higher-order neurons, in: Proceedings of the Ninth International Conference on Artificial Neural Networks, 1999, pp. 563–568.
[17] Y. Zhou, X. Li, L. Gao, A differential evolution algorithm with intersect mutation operator, Appl. Soft Comput. 13 (1) (2013) 390–401.
[18] H. Karshenas, R. Santana, C. Bielza, P. Larrafinaga, Regularized continuous estimation of distribution algorithms, Appl. Soft Comput. 13 (5) (2013) 2412–2432.
[19] Q. Cai, D.F. Zhang, W. Zheng, S.C.H. Leung, A new fuzzy time series forecasting model combined with ant colony optimization and auto-regression, Knowl.-Based Syst. 74 (2015) 61–68.
[20] D. Karaboga, B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, Lect. Notes Comput. Sci. 4529 (2007) 789–798.
[21] K.N. Krishnanand, D. Ghose, Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions, Swarm Intell. 3 (2) (2009) 87–124.
[22] C.H. Dai, W.R. Chen, Y.H. Song, Seek optimization algorithm: a novel stochastic search algorithm for global numerical optimization, Syst. Eng. Electron. 21 (2) (2010) 300–311.
[23] M. Basu, Artificial immune system for dynamic economic dispatch, Electron. Power Energy Syst. 33 (1) (2011) 131–136.
[24] A. Chatterjee, S.P. Ghoshal, V. Mukherjee, A maiden application of gravitational search algorithm with wavelet mutation for the solution of economic load dispatch problems, Bio-Inspired Comput. 4 (1) (2012) 33–46.
[25] X. Han, X. Chang, A chaotic digital secure communication based on a modified gravitational search algorithm filter, Inform. Sci. 208 (2) (2012) 14–27.
[26] G. Sun, A. Zhang, A hybrid genetic algorithm and gravitational search algorithm for image segmentation using multilevel thresholding, Pattern Recognit. Image Anal. (2013) 707–714.
[27] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[28] E.D. Taillard, Few guidelines for analyzing methods, in: The Sixth Metaheuristics International Conference, 2005, pp. 1–10.
[29] K.N. Krishnanand, D. Ghose, Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications, Multiagent Grid Syst. 35 (2006) 209–222.
[30] K.N. Krishnanand, D. Ghose, Theoretical foundations for rendezvous of glowworm-inspired agent swarms at multiple locations, Robot. Auton. Syst. (2008) 549–569.
[31] K.N. Krishnanand, D. Ghose, Glowworm swarm optimization: a new method for optimizing multi-modal functions, Comput. Intell. Stud. 1 (1) (2009) 93–119.
[32] R.C. Eberhart, Y.H. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: IEEE Congress on Evolutionary Computation, 2000, pp. 84–88.
[33] Y.H. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Congress on Evolutionary Computation, 1999.
[34] G. Quarantaa, G. Carlo Maranob, R. Grecob, G. Monti, Parametric identification of seismic isolators using differential evolution and particle swarm optimization, Appl. Soft Comput. 22 (2014) 458–464.
[35] A. Kyprianou, K. Worden, M. Panet, Identification of hysteretic systems using the differential evolution algorithm, Sound Vib. 248 (2001) 289–314.
[36] S. Salehi, A. Selamat, M.R. Mashinchi, H. Fujita, The synergistic combination of particle swarm optimization and fuzzy sets to design granular classifier, Knowl.-Based Syst. 76 (2015) 200–218.
[37] H. Tang, S. Xue, C. Fan, Differential evolution strategy for structural system identification, Comput. Struct. 86 (21–22) (2008) 2004–2012.
[38] F.Z. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Appl. Math. Comput. 186 (1) (2007) 340–356.
[39] L.Z. Wang, H. Geng, P. Liu, K. Lu, J. Kolodziej, R. Ranjan, A.Y. Zomaya, Particle swarm optimization based dictionary learning for remote sensing big data, Knowl.-Based Syst. 79 (2015) 43–50.

[40] A. Salman, A.P. Engelbrecht, M.G.H. Omran, Empirical analysis of self-adaptive differential evolution, Oper. Res. Int. J. 183 (2) (2007) 785–804.
[41] R.L. Becerra, C.A. Coello, Cultured differential evolution for constrained optimization, Comput. Methods Appl. Mech. Eng. 195 (2006) 4303–4322.
[42] K. Zielinski, R. Laur, Constrained single-objective optimization using differential evolution, in: IEEE Congress on Evolutionary Computation, 2006, pp. 223–230.
[43] X.C. Zhao, A perturbed particle swarm algorithm for numerical optimization, Appl. Soft Comput. 10 (1) (2010) 119–124.
[44] X. Yuan, J. Zhao, Y. Yang, Y. Wang, Hybrid parallel chaos optimization algorithm with harmony search algorithm, Appl. Soft Comput. 17 (2014) 12–22.
[45] Y. Wang, J. Zeng, Z. Cui, X. He, A novel constraint multi-objective artificial physics optimization algorithm and its convergence, Innovative Comput. Appl. 3 (2) (2011) 61–70.
[46] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232–2248.
[47] X. Han, X. Chang, A chaotic digital secure communication based on a modified gravitational search algorithm filter, Inform. Sci. 208 (2) (2012) 14–27.
[48] A. Chatterjee, S.P. Ghoshal, V. Mukherjee, A maiden application of gravitational search algorithm with wavelet mutation for the solution of economic load dispatch problems, Bio-Inspired Comput. 4 (1) (2012) 33–46.
[49] L. Zhao, Y. Yang, PSO-based single multiplicative neuron model for time series prediction, Expert Syst. Appl. 36 (2009) 2805–2812.
[50] H. Liu, Z.X. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Appl. Soft Comput. 10 (2010) 629–640.
[51] H.S. Wang, X.F. Yan, Optimizing the echo state network with a binary particle swarm optimization algorithm, Knowl.-Based Syst. 86 (2015) 182–193.