Neural Computing and Applications (2023) 35:2867–2882
https://doi.org/10.1007/s00521-022-07711-6

ORIGINAL ARTICLE

Multi-objective sparse echo state network

Cuili Yang[1] · Zhanhong Wu[1]

Received: 24 November 2021 / Accepted: 5 August 2022 / Published online: 7 September 2022

Abstract
The echo state network (ESN) has been widely applied for nonlinear system modeling. However, the too large reservoir size of ESN will lead to overfitting problem and reduce generalization performance. To balance reservoir size and training performance, the multi-objective sparse echo state network (MOS-ESN) is proposed. Firstly, the ESN design problem is formulated as a two-objective optimization problem, which is solved by the decomposition-based multi-objective optimization algorithm (MOEA/D). Secondly, to accelerate algorithm convergence, the local search strategy is designed, which combines the l_1 or l_0 norm regularization and coordinate descent algorithm, respectively. Thirdly, to produce more solutions around the knee point, an adaptive weight vectors updating method is proposed, which is based on decision maker interest. Experimental results show that the MOS-ESN outperforms other methods in terms of network sparseness and prediction accuracy.

Keywords Echo state networks · MOEA/D · Local search strategy · Weight vectors updating method

1 Introduction

Time series prediction is widely existed in all aspects of life [1–4], such as forecasting the number daily discharged inpatients of hospital, wind power prediction, global financial situation prediction, and so on. Typical methods include statistical regression [5], gray prediction [6] and machine learning [7]. The autoregressive moving average [8] is a commonly used statistical regression method, but it cannot solve nonlinear problems. Gray prediction [9] is suitable for time series prediction with uncertain partial information, but it is not suitable for the static dataset. Therefore, the prediction method based on machine learning [10] has been widely concerned, which requires no assumptions about the data or model.

✉ Cuili Yang
clyang5@bjut.edu.cn

Zhanhong Wu
2650146875@qq.com

[1] Beijing Key Laboratory of Computational Intelligence and Intelligence System, Beijing Laboratory for Intelligent Environmental Protection, Faculty of Information Technology, Beijing Institute of Artificial Intelligence, Beijing University of Technology, Pingleyuan, Beijing, Beijing 100124, China

In the field of machine learning, the commonly used methods include decision tree (DT) [11], support vector machine (SVM) [12] and artificial neural networks (ANN) [13–17], among which the ANN has drawn many attentions due to its nonlinear approximation ability. The most classical ANN is feed-forward neural network (FNN) [16], which can simulate any nonlinear system. However, it is difficult for FNN to capture the hidden sequence information of time series data. Therefore, the recurrent neural network (RNN) [17] is proposed to solve complex time series problems. However, its network structure and training method may lead to low training efficiency and memory loss. Therefore, Jaeger has proposed a new type of RNN, named as echo state network (ESN) [18].

Nowadays, ESN has been successfully applied in the field of time series prediction [18–25]. Unlike the traditional RNN, the ESN uses a reservoir to store and manage information. The input weights and internal weights of ESN are generated randomly and remain unchanged, only the output weights (also called readout) should be trained. In [21], the generation of reservoirs and training of readouts are reviewed. In [22], the hierarchical ESN is proposed, which is trained by stochastic gradient descent. In [23], the ESN with leaky integrator neurons is designed, which can easily adapt to time characteristics.

In reservoir initialization phase, hundreds of sparsely connected neurons are generated, and some neurons may have little influence on training performance. If all reservoir nodes are connected with network outputs, the ESN will perform very well on training data, but not good on testing data, leading to the overfitting problem. Hence, how to design a suitable reservoir size to improve the performance of ESN has always been the focus of research. In [24], the singular value decomposition-based growing ESN is proposed, which can weaken the coupling among reservoir neurons. In [25], the reservoir pruning method is designed, in which the mutual information between reservoir states is used to delete nodes. However, the pruning method may destroy the echo state characteristics of ESN [26].

To avoid overfitting problem, the regularization techniques are widely applied to sparse the readout of ESN, rather than control the size of reservoir directly [27, 28]. In [29], the reservoir nodes are dynamically added or deleted according to their importance to network performance, the l_2 regularization is used to update the output weights. However, the l_2 regularization is not able to generate the sparse ESN. In [30], the l_1 penalty term is added into the objective function to shrink some irrelevant output weights as small values, such that the readout is sparse. In [31], the l_0 regularization is used for sparse signal recovery, which is able to reduce computation complexity and improve classification ability, simultaneously. In [32], the online sparse ESN is designed, in which the l_1 and l_0 norms are respectively used as penalty terms to control the network size, the sparse recursive least squares and sub-gradient algorithm are combined to estimate output weights. This method has shown superior performance than other ESNs in prediction accuracy and network sparseness. Hence, the l_0 and l_1 regularization are the focus of this paper.

In traditional regularization approaches [27–32], the regularization coefficient is used to introduce the penalty term into the objective function,

$$F(W^{\text{out}}) = \|T - \mathbf{H}\mathbf{W}^{\text{out}}\|_2^2 + \mu\|W^{\text{out}}\|_p \quad (1)$$

where the first term and the second term are the training error and penalty term, respectively,μ is regularization coefficient, Wout is the output weight of ESN, p = 0,1 represent the l0-norm or l1-norm, respectively. The regularization coefficient is used to balance training error and sparseness of Wout. Different μ will lead to different optimal solutions [33], and a small change of the regularization coefficient will have a great influence on the training results. Thus, it is important to choose an appropriate regularization coefficient.

To avoid choosing regularization coefficient, in this paper, the optimization of Eq. (1) is formulated as a multi-objective optimization problem (MOP), in which the two conflicting objectives can be optimized [34]. From the view point of optimization, many Pareto-optimal solutions can be obtained by multi-objective optimization algorithms, and thus it is difficult to determine which solution can obtain the best network structure and training error. To select the appreciate solution, the preferences of decision maker should be considered [35]. The knee point is proposed in [36], in which a small change of one objective will generate a big change on the other [37–39]. Although the solutions in knee points does not provide the best result for some problems, they still be Pareto solutions which has the optimal performance for MOP.

In this paper, the multi-objective sparse ESN (MOS-ESN) is proposed, in which the training error and network size are treated as two optimization objectives. The main contribution is as follows. Firstly, the MOEA/D-based multi-objective optimization algorithm is designed to optimize network structure and network performance. Secondly, to improve algorithm convergence, the l_1 or l_0 regularization and coordinate descent algorithm-based local search strategy is designed. Thirdly, the preference information of knee point is integrated into weight vectors updating method, which guides the evolution of population toward knee region. Simulation results prove that MOS-ESN can improve the training accuracy and network sparseness without involving any regularization parameters.

The paper organization is as follows. Section 2 introduces the basic description of ESN, MOP and MOEA/D. The proposed MOS-ESN is given in Sect. 3. The simulations are discussed in Sect. 4. The paper is summarized in Sect. 5

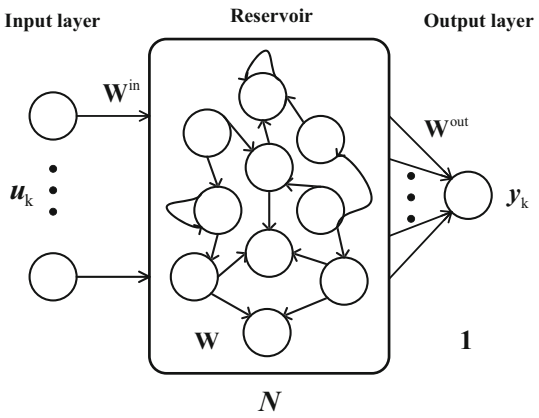

Fig. 1 Description of OESN

2 Background

2.1 Original ESN

The original ESN (OESN) in Fig. 1 is constructed with an input layer, a reservoir and an output layer. The OESN has n input neurons in input layer, N nodes in reservoir and one output node. The input layer and reservoir are connected by the input weight matrix $\mathbf{W}^{\text{in}} \in \mathbb{R}^{N\times N}$ the elements in reservoir are tied by the internal weight matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ while the input layer and reservoir are related with the output layer through the output weight matrix $\mathbf{W}^{\text{out}} \in \mathbb{R}^{(N+n)\times 1}$. Consider L distinct samples $\{\mathbf{u}(k), \mathbf{t}(k)\}$, where $\mathbf{u}(k) = [u_1(k), u_2(k), \ldots, u_n(k)]^{\text{T}} \in \mathbb{R}^{(N+n)\times 1}$ and $\mathbf{t}(k)$ are input and target, respectively, and the reservoir state $\mathbf{x}$(k) is updated as below,

$$\mathbf{x}(k) = \mathbf{g}(\mathbf{W}\mathbf{x}(k-1) + \mathbf{W}^{\text{in}}\mathbf{u}(k)) \tag{2}$$

where $\mathbf{g}(\cdot) = [g_1(\cdot), \ldots, g_N(\cdot)^{\text{T}}$ are the activation functions. The output $\mathbf{y}(k)$ is equal to $(\mathbf{W}^{\text{out}})^{\text{T}}[\mathbf{x}(k); \mathbf{u}(k)]$, where $[\mathbf{x}(k); \mathbf{u}(k)] \in$ $$ \mathbb{R}^{{1 \times \left({1 + n} \right)}} $$ is the concatenation of reservoir states and input matrix.

Denote $\mathbf{T} = [t(1), t(2), \ldots, t(L)]^{\text{T}}$ as target data matrix and represent $\mathbf{H} = [\mathbf{X}(1), \mathbf{X}(2), \ldots, \mathbf{X}(L)]^{\text{T}}$ as internal state matrix as below

$$H = \begin{bmatrix} X^T(1) \\ X^T(2) \\ \vdots \\ X^T(L) \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1N+n} \\ X_{21} & X_{22} & \cdots & X_{2N+n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{L1} & X_{L2} & \cdots & X_{LN+n} \end{bmatrix} \tag{3}$$

The output weight matrix $\mathbf{W}^{\text{out}}$ can be calculated by

$$\mathbf{W}^{\text{out}} = \mathbf{H}^{\dagger}\mathbf{T} \tag{4}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose pseudoinverse of $\mathbf{H}$. $\mathbf{H}^{\dagger}$ can be computed by orthogonal projection methods [40], single-value decomposition, and so on. However, if the input data contain unknown random noise, the inverse calculation of $\mathbf{H}$ may lead to an ill-posed problem, i.e., an unstable solution is obtained.

2.2 MOPs

MOPs contain many conflicting objective functions that should be optimized at the same time. Generally speaking, the minimized MOPs can be expressed as below:

$$\text{Min}\mathbf{F}(\mathbf{W}) = [f_1(\mathbf{W}), f_2(\mathbf{W}), \ldots, f_m(\mathbf{W})]^{\text{T}} \tag{5}$$

subject to $\mathbf{W} \in \Omega$where W is the decision variable, m is the number of objective functions, Ω is the decision space, F: $\Omega \rightarrow$ Rm is consisted of m objective functions, and Rm is named as the objective space.

For two solutions $\mathbf{W}_1$ and $\mathbf{W}_2$, $\mathbf{W}_1$ is said to dominate $\mathbf{W}_2$ (denoted as $\mathbf{W}_1 \prec \mathbf{W}_2$), if and only if $f_i(\mathbf{W}_1) \leq f_i(\mathbf{W}_2)$ for each objective $i \in \{1,\ldots,m\}$, and $f_j(\mathbf{W}_1) < f_j(\mathbf{W}_2)$ for at least one value $j \in \{1,\ldots,m\}$ [40].

Furthermore, the solution $\mathbf{W}_* \in \Omega$ is defined as Pareto-optimal if there is no other feasible solution $\mathbf{W} \prec \mathbf{W}_*$. Particularly, the set of $\mathbf{W}_*$ is named as Pareto-optimal set (PS) and the union of all PS is called the Pareto-optimal front (PF) [34].

2.3 MOEA/D

MOEA/D decomposes a MOP into several single-objective subproblems by multiple weight vectors, and all the subproblems can be optimized at the same time [40]. The main steps of MOEA/D are given:

Step 1: Generate the initial population, $x_1, x_2, \ldots, x_P$, and a group of uniformly weight vectors $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_P)$, where P is the population size.

Step 2: Compute the Euclidean distance between any two weight vectors, find the nearest T vectors of each vector, which are denoted the neighborhood of λ_i and represented as $\mathbf{B}(i) = \{i_1, i_2, \ldots, i_T\}$.

Step 3: Choose two index k,l from $\mathbf{B}(i)$ randomly. Apply genetic operators on x_k and x_l to generate a new individual y.

Step 4: Update neighboring solutions. When the aggregate function value of y is smaller or equal to x_j, update $x_j = y$, where $j \in \mathbf{B}(i)$.

Step 5: Determine the non-dominated solutions of population and update the external population (EP), which saves the non-dominated solutions.

The main feature of MOEA/D is its decomposition method [40], such as the weighted sum approach, Tchebycheff approach and boundary intersection approach. In the following, the form of weighted sum approach is shown

$$\text{Ming}^{\text{ws}}(\mathbf{W}\lambda_i) = \sum_{k=1}^{m} \lambda_{i,k} f_k(\mathbf{W}) \tag{6}$$

where $\boldsymbol{\lambda}_\mathbf{i} = \{\lambda_{i,1}, \lambda_{i,2}, \ldots, \lambda_{i,m}\}$ represents the weight vector corresponding to each objective function, and it is noted that $\sum_{k=1}^{m} \lambda_{i,k} = 1$.

3 MOS-ESN

To optimize the network size and training error simultaneously, the MOS-ESN is proposed. Firstly, the design of ESN is formulated as a bi-objective optimization problem, which is solved by MOEA/D. Secondly, to improve algorithm convergence, the l1 and l0 regularization-based local

search strategy is designed. Furthermore, to find more solutions around the knee point, the decision maker preference-based weight vectors updating method is proposed.

3.1 Problem Formulation

The network size of ESN is closely related to training performance. To prove their relationship, a simple experiment is designed. Firstly, an ESN with 200 nodes is randomly initialized. Then, several sparse output weights are generated, and the corresponding training errors are recorded. Finally, the training error (denoted as f1) and the number of nonzero elements of output weights (denoted as f2) are drawn in Fig. 2. Obviously, the training error decreases as the network size increases, the too large network will lead to overfitting problem. However, if the network is too small, the training of ESN will be insufficient. Hence, how to achieve a balance between network size and training error becomes the key to research.

To solve this problem, the regularization methods are introduced by using the l_1 or l_0 norm penalty term, and then the design of ESN is realized by optimizing the following objective function

$$\min \mathrm{F}_{(\mathbf{W}^{\mathrm{out}})}(\mathrm{W}^{\mathrm{out}}) = \min\left(\|\mathbf{T}-\mathbf{H}\mathbf{W}^{\mathrm{out}}\|_2^2+\mu\|\mathbf{W}^{\mathrm{out}}\|_{0/1}\right) \tag{7}$$

where μ is regularization parameters. Actually, the selection of regularization parameters is a difficult problem, because the large μ means a small reservoir with large training error, while the small μ has the opposite effect [33].

To avoid choosing regularization coefficient, the problem in Eq. (9) is treated as a multi-objective optimization problem,

$$\min_{\mathbf{W}^{\mathrm{out}}} \mathrm{F}(\mathbf{W}^{\mathrm{out}}) = \min\left(\|\mathbf{T}-\mathbf{H}\mathbf{W}^{\mathrm{out}}\|_2^2, \|\mathbf{W}^{\mathrm{out}}\|_{0/1}\right) \tag{8}$$

where the first term is training error and the second is network size. To minimize the training error and network size simultaneously, the MOEA/D is used, in which the weighted sum approach is applied to generate a set of subproblems

$$\min_{\mathbf{W}^{out}} g^{ws}(\mathbf{W}^{out}\lambda) = \lambda_1 f_1(\mathbf{W}^{out}) + \lambda_2 f_2(\mathbf{W}^{out}) \tag{9}$$

where λ_1 and λ_2 represent the weight of f_1 and f_2, respectively, and $\lambda_1 + \lambda_2 = 1$.

3.2 Local Search Method

To accelerate the convergence speed of MOEA/D, the local search strategy is proposed, in which the l1 or l0 regularization term is applied to ensure network sparsity, and the coordinate descent algorithm is introduced to update the elements of Wout.

3.2.1 The l1 regularization-based local search method

By using the l1 regularization, the problem in Eq. (8) is formulated as below:

$$\min_{\mathbf{W}^{\mathrm{out}}} \mathrm{F}(\mathbf{W}^{\mathrm{out}}) = \min\left(\|\mathbf{T}-\mathbf{H}\mathbf{W}^{\mathrm{out}}\|_2^2+\mu\|\mathbf{W}^{\mathrm{out}}\|_1\right) \tag{10}$$

where $\|\mathrm{W}^{\mathrm{out}}\|_1=\sum_{\mathrm{i}=1}^{\mathrm{N+n}}|\mathrm{w_i}|$ represent the l_1-norm of $\mathbf{W}^{\mathrm{out}}$. The subproblem in Eq. (9) is described as

$$E(\mathbf{W}^{\mathrm{out}}) = \frac{\lambda_1}{2}\|\mathbf{T}-\mathbf{H}\mathbf{W}^{\mathrm{out}}\|_2^2+\lambda_2\|\mathbf{W}^{\mathrm{out}}\|_1 \tag{11}$$

To facilitate computational analysis, $\lambda_1/2$ is applied in Eq. (11) instead of $\lambda 1$. Because the two objectives in Eq. (11) are differentiable, the coordinate descent algorithm is selected to calculate the value of Wout, which has shown strong local search ability. Under the framework of coordinate descent algorithm, in each iteration, the ith variable wi ($i = 1, 2, \ldots, N + n$)of Wout is updated, while the other elements remain the same. Thus, Eq. (11) becomes

$$\begin{aligned}
E(\mathbf{W}^{out}(w_i)) &= \frac{\lambda_1}{2}\left\{\sum_{k=1}^{L}\left[\mathbf{t}(k)-\mathbf{X}_{ki}w_i-\sum_{j\neq i}^{N+n}\mathbf{X}_{kj}w_j\right]\right\}^2 \\
&\quad+\lambda_2\sum_{i=1}^{N+n}|w_i| \\
&= \frac{\lambda_1}{2}\sum_{k=1}^{L}(\mathbf{X}_{ki}w_i)^2 \\
&\quad-\lambda_1\left\{\left[\sum_{k=1}^{L}\left(\mathbf{t}(k)-\sum_{j\neq i}^{N+n}\mathbf{X}_{kj}w_j\right)\mathbf{X}_{ki}\right]\cdot w_i\right\} \\
&\quad+\frac{\lambda_1}{2}\sum_{k=1}^{L}\left[\left(\mathbf{t}(k)\right)-\sum_{j\neq i}^{N+n}\mathbf{X}_{kj}w_j\right)^2\Bigg] \\
&\quad+\sum_{i=1}^{N+n}\lambda_2|w_i|
\end{aligned} \tag{12}$$

It can be found that

$$\frac{\lambda_1}{2}\sum\nolimits_{\mathrm{k=1}}^{\mathrm{L}}\left[\left(\mathrm{t(k)}-\sum_{\mathrm{j\neq i}}^{\mathrm{N+n}}\mathrm{X_{kj}w_j}\right)^2\right]$$

is irrelevant to wi, thus minimizing E(Wout(wi)) in Eq. (12) is equal to minimizing Z(Wout(wi)),

$$Z(\mathbf{W}^{out}(w_i)) = \frac{\lambda_1}{2}\sum_{k=1}^{L}(\mathbf{X}_{ki}w_i)^2 - \lambda_1\left\{\left[\sum_{k=1}^{L}(\mathbf{t}(k) - \sum_{j\neq i}^{N+n}\mathbf{X}_{kj}w_j)\mathbf{X}_{ki}\right]\cdot w_i\right\} + \sum_{i=1}^{N+n}\lambda_2|w_i| \tag{13}$$

The sub-gradient of l_1-norm is shown as below

$$\partial(\|w_i\|) = \begin{cases} 1, & \text{if } w_i > 0 \\ -1, & \text{if } w_i < 0 \\ \alpha \in [-1,1], & \text{if } w_i = 0 \end{cases} \tag{14}$$

When the derivative of $Z(\mathbf{W}^{\text{out}}(\ w_i))$ respect to w_i is equal to zero, the minimize of $Z(\mathbf{W}^{\text{out}}(\ w_i))$ can be obtained. The derivative of $Z(\mathbf{W}^{\text{out}}(\ w_i))$ is given as

$$\frac{\partial Z(\mathbf{W}^{out}(w_i))}{\partial w_i} = \lambda_1\left[\sum_{k=1}^{L}(\mathbf{X}_{ki}^2 w_i) - \sum_{k=1}^{L}\left(\mathbf{t}(k) - \sum_{j\neq i}^{N+n}\mathbf{X}_{kj}w_j\right))\mathbf{X}_{ki}\right] + \lambda_2\frac{\partial\|w_i\|_1}{\partial w_i} \tag{15}$$

To simplify the calculation, two parameters D and C are introduced

$$D = \sum_{k=1}^{L}\mathbf{X}_{ki}^2 \tag{16}$$

$$C = \sum_{k=1}^{L}\left[\mathbf{t}(k) - \sum_{j\neq i}^{N+n}\mathbf{X}_{kj}w_j\mathbf{X}_{ki}\right] \tag{17}$$

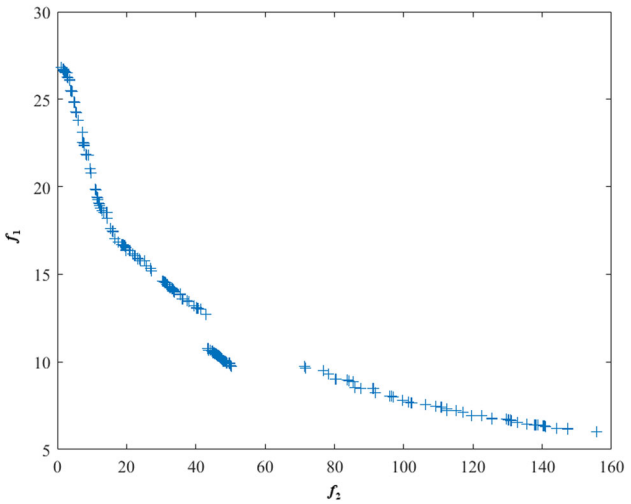

Fig. 2 Relationship between training error and network size

Thus, the derivative of $Z(\mathbf{W}^{\text{out}}(\ w_i))$ is given,

$$\frac{\partial Z(\mathbf{W}^{out}(w_i))}{\partial w_i} = \begin{cases} \lambda_1(Dw_i - C) + \lambda_2, & \text{if } w_i > 0 \\ \lambda_1(Dw_i - C) - \lambda_2, & \text{if } w_i < 0 \\ [-\lambda_1 C - \lambda_2, -\lambda_1 C + \lambda_2], & \text{if } w_i = 0 \end{cases} \tag{18}$$

By setting $\frac{\partial Z(\text{Wout}(\ \text{wi}))}{\partial \text{W}_\text{i}} = 0$, the update equation of w_i is shown in Eq. (19), the corresponding threshold function is shown in Fig. 3a.

$$w_i = \begin{cases} \dfrac{C - \frac{\lambda_2}{\lambda_1}}{D}, & if \quad C > \dfrac{\lambda_2}{\lambda_1} \\ \dfrac{C + \frac{\lambda_2}{\lambda_1}}{D} & if \quad C - \dfrac{\lambda_2}{\lambda_1} \\ 0, & if - \dfrac{\lambda_2}{\lambda_1} \le C \le \dfrac{\lambda_2}{\lambda_1} \end{cases} \tag{19}$$

In Eq. (19), w_i is related to, λ_1/λ_2 which is a fixed value and not related to $w_{i\text{-}1}$. Therefore, an adjustment is made on Eq. (19)

$$w_i = \begin{cases} \dfrac{C - \text{sgn}(C)\frac{\lambda_2}{\lambda_1}(\varepsilon + |w_{i-1}|)_+}{D} & \text{if } |C| \frac{\lambda_2}{\lambda_1}(\varepsilon + |w_{i-1}|)_+ \\ 0, & \text{if } |C| \le \frac{\lambda_2}{\lambda_1}(\varepsilon + |w_{i-1}|)_+ \end{cases} \tag{20}$$

where $w_{i\text{-}1}$ represents the weight at last iteration, ε is a small positive value and located in (0,1), and $(x)_+$ equals 1/x when $x \le 1$ and is 1 otherwise.

The advantage of above method is its modifiable threshold $\frac{\lambda_2}{\lambda_1}(\varepsilon+|\text{w}_{i-1}|)_+$. When $w_{i\text{-}1}$ is small, C has a higher probability between $-\frac{\lambda_2}{\lambda_1}(\varepsilon+|\text{w}_{i-1}|)_+$ and $\frac{\lambda_2}{\lambda_1}(\varepsilon+|\text{w}_{i-1}|)_+$. Therefore, w_i is attracted to zero with a higher possibility (show as Fig. 3b), while the increased threshold can reduce $\|\mathbf{W}^{\text{out}}\|_1$ effectively. To the contrary, if w_{i-1} is large, the threshold will decrease to avoid becoming to zero.

3.2.2 The smoothed l0 regularization-based local search method

Actually, the l_1 regularization always generates many components that are close but not equal to zero. To generate more sparse solution, the l_0 regularization is considered

$$\min_{\mathbf{W}^{\text{out}}} \text{F}(\mathbf{W}^{\text{out}}) = \min\left(\|\mathbf{T} - \mathbf{H}\mathbf{W}^{\text{out}}\|_2^2, \|\mathbf{W}^{\text{out}}\|_0\right) \tag{21}$$

However, the minimization of Eq. (21) is NP-hard. To solve it, the $\|\mathbf{W}^{\text{out}}\|_0$ is approximated by

$$||\text{Wout}||0 = \text{g}(\text{W}_{\text{out}}) = \sum_{\text{i}=1}^{\text{N+n}}\left(1 - e^{-\text{Q}|\text{w}_\text{i}|}\right)$$

where Q is an appropriate positive constant. The subdifferential of g(Wout) is as below

$$\frac{\partial g(w_i)}{\partial w_i} = sgn(w_i) \cdot Q \cdot e^{-Q|w_i|} \tag{22}$$

Transform $e^{-Q|wi|}$ by the first-order Taylor series expansion

$$e^{-Q|w_i|} \approx \begin{cases} 1 - Q|w_i|, & |w_i| \le \frac{1}{Q} \\ 0, & \text{others} \end{cases} \tag{23}$$

Similar with l_1 regularization, the subdifferential of objective function can be described as:

$$\frac{\partial Z(\mathbf{W}^{out}(w_i))}{\partial w_i} = \begin{cases} \lambda_1(Dw_i - C) - \lambda_2(Q + Q^2 w_i), -\frac{1}{Q} \le w_i < 0 \\ \lambda_1(Dw_i - C) + \lambda_2(Q - Q^2 w_i), 0 < w_i \le \frac{1}{Q} \\ [-\lambda_1 C - \lambda_2 Q, -\lambda_1 C + \lambda_2 Q], w_i = 0 \\ \lambda_1(Dw_i - C), w_i < -\frac{1}{Q} w_i < -\frac{1}{Q} or w_i > \frac{1}{Q} \end{cases} \tag{24}$$

By setting $\frac{\partial Z(\mathrm{Wout(wi)})}{\partial \mathrm{W_i}} = 0$, w_i can be obtained by Eq. (25), and the threshold function is shown in Fig. 3(c).

$$w_i = \begin{cases} \frac{\lambda_1 C + \lambda_2 Q}{\lambda_1 D - \lambda_2 Q^2}, & D > \frac{\lambda_2 Q^2}{\lambda_1} \quad \text{and} \quad -\frac{D}{Q} \le C < -\frac{\lambda_2 Q}{\lambda_1} \\ \frac{\lambda_1 C - \lambda_2 Q}{\lambda_1 D - \lambda_2 Q^2}, & D > \frac{\lambda_2 Q^2}{\lambda_1} \quad \text{and} \quad \frac{\lambda_2 Q}{\lambda_1} < C \le \frac{D}{Q} \\ 0, & -\frac{\lambda_2 Q}{\lambda_1} \le C \le \frac{\lambda_2 Q}{\lambda_1} \\ \frac{C}{D}, & \text{others} \end{cases} \tag{25}$$

To improve the zero-attraction effect, the modified l_0 regularization method is proposed

$$w_{\mathrm{i}} = \begin{cases} \frac{\lambda_1 C - \mathrm{sgn}(C) \cdot \lambda_2 Q}{\lambda_1 D - \lambda_2 Q^2}, & D > \frac{\lambda_2 Q^2}{\lambda_1} \quad \text{and} \quad \frac{\lambda_2 Q(\varepsilon + |\mathrm{w_{i-1}}|)_+}{\lambda_1} < |C| \le \frac{D}{Q} \\ 0, & \frac{\lambda_2 Q(\varepsilon + |\mathrm{w_{i-1}}|)_+}{\lambda_1} \\ \frac{C}{D}, & \text{others} \end{cases} \tag{26}$$

which can modify the threshold based on the previous value of w_i so that the small components are attracted to zeros with a higher probability (Fig. 3d).

3.3 Weight Vectors Updating Algorithm

By using MOEA/D, many non-dominated solutions can be obtained. However, only one solution is chosen to realize the nonlinear modeling problem. Generally speaking, the ESN with too large training error (λ_1 is too small) or with too large network size (λ_2 is too small) will not be chosen. As described in [41], the knee point is able to make a tradeoff between two objects. Thus, the knee point is selected as the final solution. In order to generate more solutions around the knee point, the weight vectors updating method is proposed, in which the information of knee point is incorporated.

3.3.1 Knee point

In this part, the distance-based method is introduced to find the knee point [41]. For a bi-objective optimization problem, a line L can be defined as $ax + by + c = 0$, where a, b, and c are determined by the two solutions that has minimize f_1 and f_2, respectively. Then, the distances d between solutions in PF and L can be calculated as

$$d = \frac{|ax + by + c|}{\sqrt{a^2 + b^2}} \tag{27}$$

Considering the minimization problem in this paper, only the solutions in the convex region are of interesting. Thus, the above equation can be modified as

$$d = \begin{cases} \frac{|ax + by + c|}{\sqrt{a^2 + b^2}}, & ax + by + c < 0 \\ -\frac{|ax + by + c|}{\sqrt{a^2 + b^2}}, & \text{others} \end{cases} \tag{28}$$

According to Eq. (28), the solution farthest from L is defined as knee point. For example, in Fig. 4, points A and B can determine the line L. By calculating the distance d between each point and L, the point E is the knee point obviously.

3.3.2 Weight Vectors Updating Method

The weight vectors are updated according to the information of knee point and decision maker preference. As shown in Fig. 5, the red point K is the projection of the knee point on the line L_w and A and B are the boundary points where the updated weight vectors should be located, i.e., the line AB is divided into two subintervals by K. Then, the same number of weight vectors $\frac{P}{2}$ will be generated in each subinterval. The distance between two weight vectors is called step size, which is calculated as

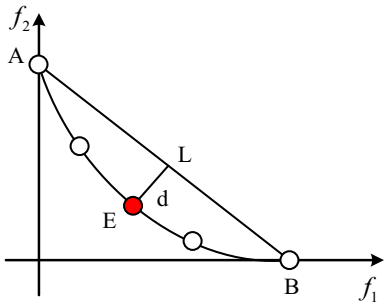

Fig. 4 Example of knee point

$$d_1 = \sqrt[\alpha]{2}\left(\frac{x - x_1}{1^\alpha + 2^\alpha + \ldots + \left\lceil \frac{P}{2} \right\rceil^\alpha}\right)^{\frac{1}{\alpha}} \tag{29}$$

Algorithm 1 Framework of weight vectors updating method

Input: P: population size; EP: non-domination solutions; lamdaMat: weight matrix ($\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \ldots, \boldsymbol{\lambda_{POP}}$, $\boldsymbol{\lambda_1}=(\lambda_{1,1},\lambda_{1,2})$; α: step size of weight vector

Output: The updated weight matrix

Step 1: Find the point A, which minimizes f_1.

Step 2: Find the point B, which minimizes f_2.

Step 3: Find the line L by using points A and B.

Step 4: Calculate the distance between each solution in EP and line L by Eq. (28).

Step 5: Rank the individuals in EP in descending order according to the distance values. The individual with the largest distance is the knee point, record its corresponding weight vectors.

Step 6: Update weight vectors in lamdaMat according to Eq. (31) and Eq. (32).

Step 7: Return the updated weight matrix.

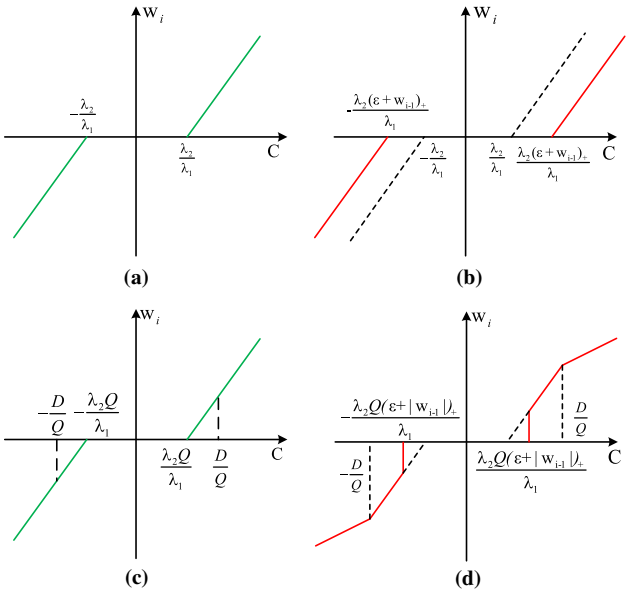

Fig. 3 The soft thresholding function and the modified one

$$d_2 = \sqrt[\alpha]{2}\left(\frac{x_2 - x}{1^\alpha + 2^\alpha + \ldots + \left\lfloor \frac{P}{2} \right\rfloor^\alpha}\right)^{\frac{1}{\alpha}} \tag{30}$$

where d_1 is the step size in line KA, d_2 is the step size in line KB, and $\alpha > 0$ is the step size parameter. The value of weight vector λ_i in line KA is,

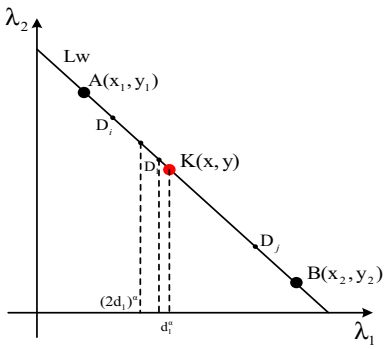

Fig. 5 The weight vectors updating method

$$\lambda_{i,1} = x - d_1^{\alpha} - (2d_1)^{\alpha} - \ldots - (id_1)^{\alpha}$$
$$\lambda_{i,2} = 1 - \lambda_{i.,1} \tag{31}$$

Similarly, the value of weight vector λ_j in line KB is

$$\lambda_{j,1} = x - d_2^{\alpha}(2d_2)^{\alpha} - \ldots - (jd_2)^{\alpha}$$
$$\lambda_{j,2} = 1 - \lambda_{j,1} \tag{32}$$

In the above weight vectors updating method, the weight vectors have a denser distribution near the knee point and are sparser at the boundary. Hence, more weight vectors will be generated near the knee point, and fewer at the boundary. Moreover, the determination of points A and B can be made by decision maker preference, which makes the algorithm converge to the region of interest.

The weight vectors updating algorithm is presented in Algorithm 1. Firstly, the two solutions which minimize f_1 and f_2 are selected and the line L can be calculated. Then, the distance between each solution to L is computed to find the knee point, and the weight vectors corresponding to knee point are chosen. Finally, the weight vectors are updated according to Eqs. (31) and (32).

Algorithm 2 MOS-ESN

Input: P: population size; T: neighborhoods size; Pc: crossover probability; Pm: mutation probability; $gen_{\max}$: maximum number of iterations; l_{oca}: local search iterations.

Output: $\mathbf{W}^{\text{out}}$

Step 1: **Initialization**: Generate the initial population $\mathbf{W}^{\text{out}}$ and find the T closest neighborhoods. Generate P weight vectors $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_{\text{POP}}$. Set $gen = 1$.

while not reach $gen_{\max}$ **do**

for i =1: POP **do**

Step 2: **Crossover operator**: Two individuals are selected randomly from the population as the parent, and the crossover operation is performed with crossover probability to obtain the offspring individual $\mathbf{P}_{\text{c}}^{\text{t}}$.

Step 3: **Mutation operation**: The mutation operation is performed on $\mathbf{P}_{\text{c}}^{\text{t}}$ according to the mutation probability, obtain the offspring $\mathbf{P}_{\text{m}}^{\text{t}}$.

Step 4: **Update neighborhoods**: For each $j \in \text{B}(i)$, if $\text{g}^{\text{WS}}(\mathbf{P}_{\text{m}}^{\text{t}} | w^j) < \text{g}^{\text{WS}}(\mathbf{W}_{\text{j}}^{\text{out}} | w^j)$, set $\mathbf{W}_{\text{j}}^{\text{out}} = \mathbf{P}_{\text{m}}^{\text{t}}$.

end for

Step 5: **Implement local search**: Perform l_{oca} iterations of local search on each individual in population. For each individual, choose w_i from $\mathbf{W}^{\text{out}}$ randomly, then update by Eq. (20) or Eq. (26).

Step 6: **Update the external population and obtain EP**.

Step 7: **Implement weight vectors updating method** as Eq. (31) and Eq. (32) to update weight vectors $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_{\text{POP}}$.

gen = gen + 1.

end while

3.4 Framework of MOS-ESN

The pseudo code of MOS-ESN is described in Algorithm 2. In Step 1, the population is randomly initialized. In Steps 2 and 3, two individuals are randomly chosen to generate the offspring, the uniform crossover operation and polynomial mutation operator are applied. In Step 4, the neighborhoods of each weight vector are updated. In Step 5, the local search is operated l_{oca} iterations to improve algorithm convergence. In Step 6, the knee point is selected from EP. In Step 7, the weight vectors are updated by the weight vectors updating method.

4 4 Simulation

In this section, the proposed MOS-ESN models are tested on two simulated benchmark problems and one practical system modeling problem, including the Rossler chaotic time series prediction [29], the nonlinear system modeling problem [42] and the effluent ammonia nitrogen (NH4-N) prediction in wastewater treatment process (WWTP) [28]. It is noted that the MOS-ESN with l0 regularization is named as MOS-ESN-l0, while the MOS-ESN with l1 regularization is termed as MOS-ESN-l1.The MOS-ESN models are compared with OESN [29], the OESN with l1 norm regularization (OESN-l1) [33], the OESN with l0 norm regularization (OESN-l0) [32], the OESN whose output weight is updated by coordinate descent and l1 or l0 norm (CD-ESN-l1 [43], CD-ESN-l0), as well as the OESN whose output weights are directly calculated by MOEA/D (OESN-MOEA/D). For each algorithm, 50 independent runs are carried out in the MATLAB 2018b environment on a personal computer with i7 core 8.0 GB memory.

The training and testing RMSE values are applied to evaluate the learning and testing performance of ESNs. Furthermore, the sparsity degree (SP) of the output weight matrix [28] is also introduced. The SP and RMSE are defined as follows:

$$\mathrm{SP} = \frac{\|\mathbf{W}^{\mathrm{out}}\|_0}{N+n} \times 100\% \tag{33}$$

$$\mathrm{RMSE} = \sqrt{\sum_{k=1}^{L} \frac{[\mathbf{y}(k) - \mathbf{t}(k)]^2}{L}} \tag{34}$$

where $\mathbf{W}^{\mathrm{out}}$ is the output weights matrix and $\mathbf{y}(k)$ and $\mathbf{t}(k)$ stand for actual and target output, respectively. A smaller RMSE means a better training or testing accuracy. Meanwhile, the smaller SP means, the ESN has the sparser structure.

To evaluate the searching ability of a multi-objective optimization, the C-matrix [45] is introduced, which can measure the ration of the non-dominated solutions in **P** that are not dominated by any other solutions in **P***,

$$C(\mathbf{P}, \mathbf{P}^*) = \frac{\mathrm{size}(\mathbf{P} - \{\mathrm{x} \in \mathbf{P} | \exists \mathrm{y} \in \mathbf{P}^* : \mathrm{y} \prec \mathrm{x}\})}{\mathrm{size}(\mathbf{P})} \tag{35}$$

The larger C(**P, P***) value means a better non-dominated solution set **P**.

The parameters setting of MOS-ESN-l_0 and MOS-ESN-l_1 are as below: the reservoir size, the population size, and the neighborhood size are set as 1000, 200, 15 in each test instance, which is suggested in [44]. The optimal value of l_{oca}, α is selected by the grid search method, the number of local search operations l_{ocal} is varied from 0 to 5 by the step of 1, and the step size α is set from 0 to 3 by the step of 1.

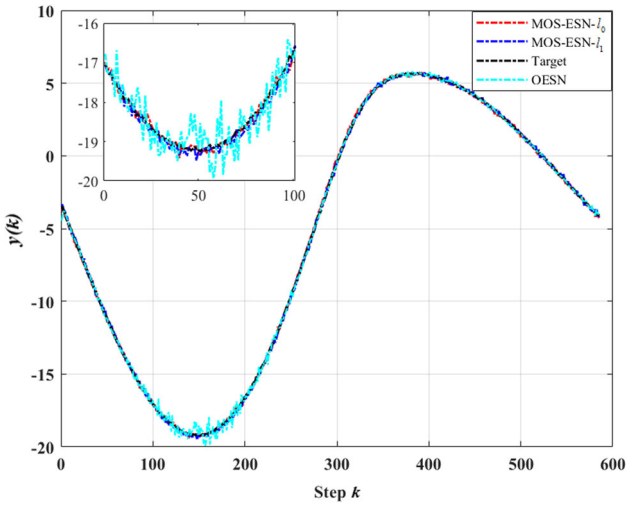

Fig. 6 The prediction outputs for Rossler chaotic time series prediction

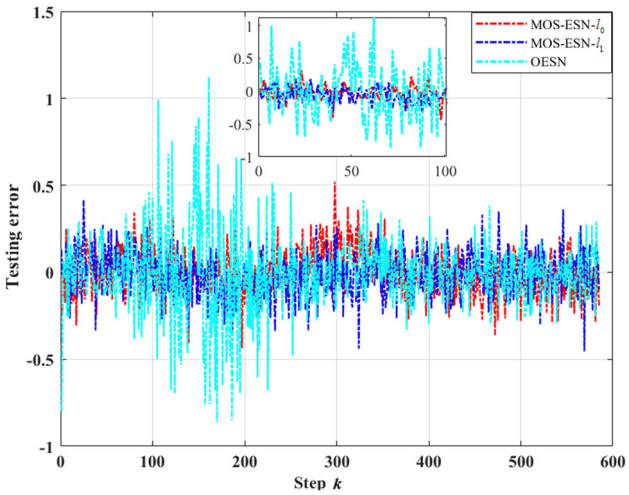

Fig. 7 The prediction errors for Rossler chaotic time series prediction

For other algorithms, their corresponding parameter settings are described in Appendix.

4.1 Rossler chaotic time series prediction

To study the performance of MOS-ESN, the Rossler chaotic time series [29], a typical chaotic dynamical time series, is introduced as below:

$$\begin{aligned} \frac{dx}{dt} &= -y - z \\ \frac{dy}{dt} &= x + \alpha y \\ \frac{dz}{dt} &= \beta + z(x - \gamma) \end{aligned} \tag{36}$$

where $\alpha = 0.2$, $\beta = 0.4$, $\gamma = 5.7$. There are 2000 samples in the experiment, in which 1400 are used for training and the

Table 1 Simulation results for Rossler chaotic series prediction

Approaches	SP	Time(s)	Mean RMSE of training	Std. RMSE of training	Mean RMSE of testing	Std. RMSE of testing
OESN	100%	**6.27**	**0.0612**	**0.0031**	0.3347	0.0239
OESN-l_1	69.32%	19.28	0.1556	0.0112	0.1573	**0.0115**
OESN-l_0	60.01%	20.12	0.1780	0.0205	0.1785	0.0215
OESN-MOEA/D	60.83%	421.58	0.1932	0.0194	0.2005	0.0209
CD-ESN-l_1	58.76%	128.47	0.1488	0.0190	0.1501	0.0202
CD-ESN-l_0	55.34%	196.19	0.1395	0.0185	0.1404	0.0199
MOS-ESN-l_1	51.88%	832.77	0.1302	0.0218	0.1311	0.0224
MOS-ESN-l_0	**46.87%**	856.81	0.1285	0.0202	**0.1296**	0.0211

The best results are marked in bold

rest 600 are applied for testing. The White Gaussian noise, which has the signal-to-noise ratio (SNR) of 20 dB, is added into the original training and testing datasets.

The testing outputs and prediction error of MOS-ESN-l_0, MOS-ESN-l_1, OESN are illustrated in Figs. 6 and 7, respectively, in which the red, blue, black and cyan lines are the trends of MOS-ESN-l_0, MOS-ESN-l_1, target and OESN, respectively. It is easily found that both MOS-ESN-l_0 and MOS-ESN-l_1 can predict the trends of testing output, while the OESN shows missing outputs in a partial enlargement. Furthermore, the RMSE values of MOS-ESN-l_0 are concentrated in [− 0.3,0.3], which is smaller than other methods, demonstrating its stable performance.

Simulation results of all methods are presented in Table 1, including sparsity, CUP running time for one operation, the mean RMSE and standard deviation (Std. for short) RMSE of training and testing of 50 independent runs. It is easily found that the OESN has the smallest running time, while it has the largest testing RMSE, implying its poor generalization ability. Both OESN-l_1 and OESN-l_0 have smaller SP, which implies the regularization method could generate the sparse output weight matrix. Besides, the proposed MOS-ESN-l_0 has the smallest testing RMSE and SP among all ESN models, which proves its effectiveness in terms of network sparseness and prediction accuracy.

4.1.1 Effect of loca

As introduced in Sect. 3.4, l_{oca} decides how many local search operators are conducted on each individual. The effects of l_{oca} on network performance are investigated through 50 independent experiments. By setting $l_{oca} = 0$, $l_{oca} = 1$ and $l_{oca} = 3$, the obtained non-dominated solutions of MOS-ESN-l_1 and MOS-ESN-l_0 are plotted in Figs. 8 and 9, respectively. The x-coordinate and y-coordinate are the objective function $f_2 = \|W^{out}\|_{1/0}$ and $f_1 = \|T - HW^{out}\|_2^2$, respectively. Obviously, the algorithm with $l_{oca} = 3$ can always generate more non-dominated solutions than the algorithms with $l_{oca} = 0$ or

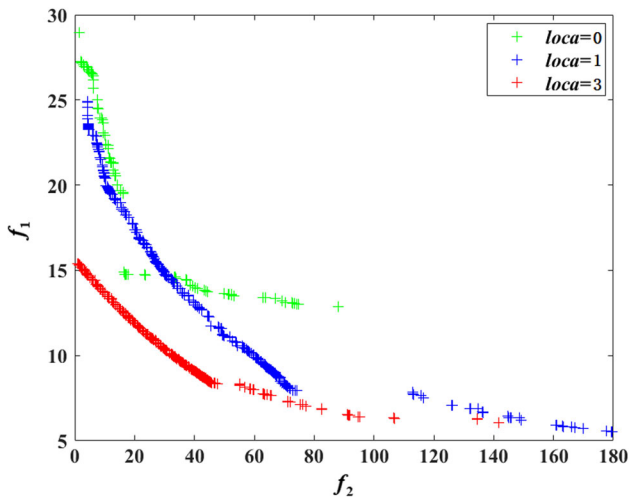

Fig. 8 Effect of l_{oca} on MOS-ESN-l_1

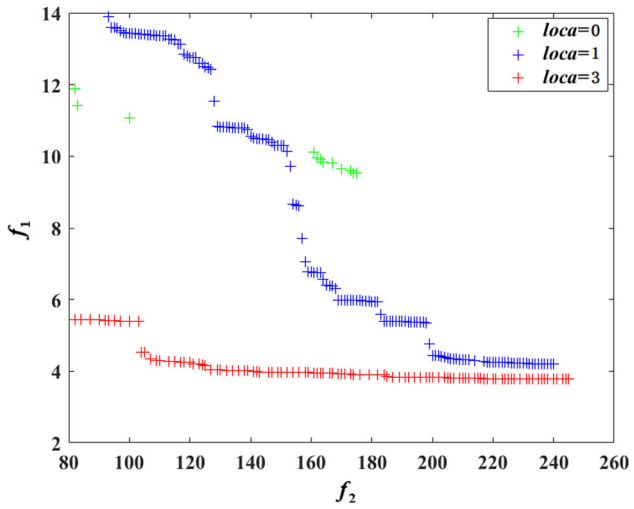

Fig. 9 Effect of l_{oca} on MOS-ESN-l_0

Table 2 Simulation results of different value of l_{oca} on MOS-ESN-l_1

Parameter	C-metrix	SP	Time(s)	Mean RMSE of training	Std. RMSE of training	Mean RMSE of testing	Std. RMSE of testing
MOS-ESN-l_1 (l_{oca} = 0)	0	68.12%	**400.81**	0.1941	**0.0142**	0.1961	**0.0163**
MOS-ESN-l_1 (l_{oca} = 1)	0.25	57.29%	522.81	0.1805	0.0203	0.1857	0.0259
MOS-ESN-l_1 (l_{oca} = 2)	0.50	54.36%	637.44	0.1432	0.0174	0.1476	0.0181
MOS-ESN-l_1 (l_{oca} = 3)	0.85	**50.12%**	867.26	**0.1311**	0.0198	**0.1329**	0.0205
MOS-ESN-l_1 (l_{oca} = 4)	**0.91**	51.87%	979.73	0.1458	0.0196	0.1462	0.0210
MOS-ESN-l_1 (l_{oca} = 5)	0.86	51.99%	1082.01	0.1599	0.0189	0.1605	0.0192

The best results are marked in bold

Table 3 Simulation results of different value of l_{oca} on MOS-ESN-l_0

Parameter	C-metrix	SP	Time(s)	Mean RMSE of training	Std RMSE of training	Mean RMSE of testing	Std RMSE of testing
MOS-ESN-l_0 (l_{oca} = 0)	0	56.62%	**398.69**	0.1942	**0.0169**	0.1953	0.0161
MOS-ESN-l_0 (l_{oca} = 1)	0.53	48.69%	**482.20**	0.1889	0.0202	0.1894	0.0259
MOS-ESN-l_0 (l_{oca} = 2)	0.62	48.13%	610.24	0.1492	0.0183	0.1507	0.0197
MOS-ESN-l_0 (l_{oca} = 3)	0.84	**46.21%**	869.01	**0.1231**	0.0187	**0.1239**	**0.0159**
MOS-ESN-l_0 (l_{oca} = 4)	**0.84**	46.53%	973.18	0.1401	0.0174	0.1428	0.0181
MOS-ESN-l_0 (l_{oca} = 5)	0.82	46.99%	1082.76	0.1437	0.0191	0.1511	0.0202

The best results are marked in bold

l_{oca} = 1, which implies that the local search algorithm can accelerate the algorithm convergence speed.

By setting l_{oca} = {1, 2, 3, 4, 5} and α = 2, the statistics results of training time, training and testing RMSE values, C-matrix and SP of MOS-ESN-l_1 and MOS-ESN-l_0 are reported in Tables 2 and 3, respectively. It is noted that during the calculation process of C-matrix, $\mathbf{P}^* = (\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n)$, where $\mathbf{P}_i$ is the non-dominated solution set by the algorithm with $l_{oca} = i$ (i = 1, …, 5). It is easily found that when l_{oca} is set as a small value, such as 0 or 1, the small value of C-matrix is obtained, which means the worse non-dominated solutions is obtained. When l_{oca} is set as a moderate value (l_{oca} = 3), the larger value of C-matrix, lower SP and testing RMSE values can be obtained. On the contrary, if l_{oca} is set as a too large value (l_{oca} = 5), the training and testing RMSE values are not best among all the models, because the too large value of l_{oca} may have a risk of converging to local regain. Furthermore, the too large l_{oca} will increase the computational complexity or training time.

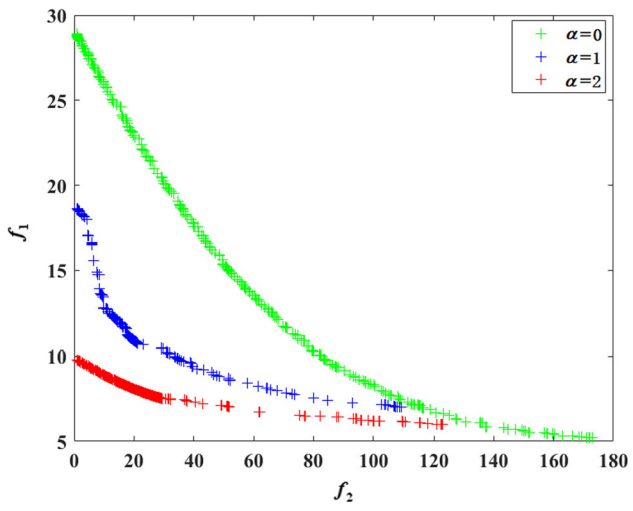

Fig. 10 Effect of α on MOS-ESN-l_1

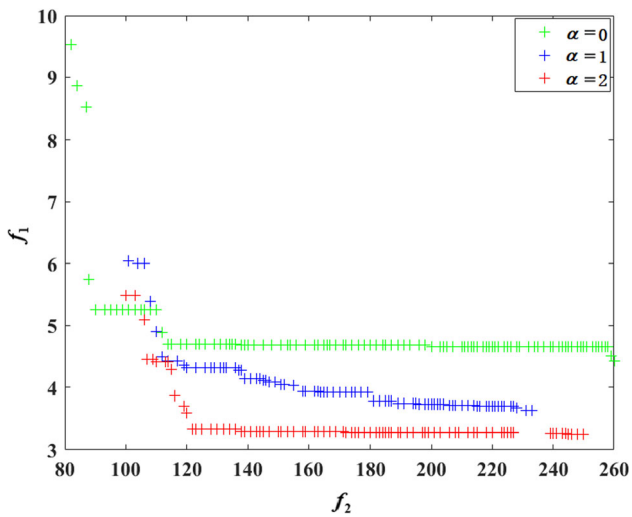

Fig. 11 Effect of α on MOS-ESN-l_0

Table 4 Simulation results of different value of α on MOS-ESN-l_1

Parameter	C-matrix	SP	Time(s)	Mean RMSE of training	Std. RMSE of training	Mean RMSE of testing	Std. RMSE of testing
MOS-ESN-l_1 (α = 0)	0	43.86%	**860.11**	0.1848	**0.0181**	0.1851	0.0230
MOS-ESN-l_1 (α = 1)	0.54	45.11%	862.13	0.1727	0.0204	0.1730	0.0199
MOS-ESN-l_1 (α = 2)	0.75	**48.39%**	863.09	**0.1402**	0.0197	**0.1422**	0.0201
MOS-ESN-l_1 (α = 3)	**0.74**	47.12%	860.61	0.1589	0.0201	0.1601	**0.0192**

The best results are marked in bold

Table 5 Simulation results of different value of α on MOS-ESN-l_0

Parameter	C-matrix	SP	Time(s)	Mean RMSE of training	Std. RMSE of training	Mean RMSE of testing	Std. RMSE of testing
MOS-ESN-l_1 (α = 0)	0	**38.77%**	**858.32**	0.1901	**0.0175**	0.1914	**0.0183**
MOS-ESN-l_1 (α = 1)	0.41	57.75%	860.01	0.1856	0.0202	0.1860	0.0210
MOS-ESN-l_1 (α = 2)	**0.68**	40.34%	859.18	**0.1378**	0.0216	**0.1402**	0.0189
MOS-ESN-l_1 (α = 3)	0.66	42.33%	861.26	0.1749	0.0230	0.1755	0.0216

The best results are marked in bold

4.1.2 Effect of α

For ESN design, the network with too small training error or too small network sparseness is not preferred, while the solution at the knee point maybe a good choice, which is a tradeoff between two objects. To help the algorithm converge to the knee point, the weight vectors updating algorithm is proposed, in which the weight updating step α is applied. A larger α implies that the updated weight vector is closer to the corresponding weight vector of knee point.

To show the influence of α on network performance, by setting α = {0, 1, 2}, the obtained non-dominated solution sets of MOS-ESN-l_1 and MOS-ESN-l_0 are compared in Figs. 10 and 11, respectively, and the x-coordinate and y-coordinate are the objective function $f_2 = \|W^{out}\|_{1/0}$ and $f_1 = \|T - HW^{out}\|_2^2$, respectively. It is easily found that the non-dominated solutions sets with α = 1 or α = 2 are better than that with α = 0, which implies the effectiveness of weight vectors updating algorithm in terms of algorithm convergence.

With α = {0, 1, 2, 3} and loca = 3, the statistic results of 50 independent experiments of MOS-ESN-l1 and MOS-ESN-l0 are listed in Tables 4 and 5, respectively. Obviously, when α = 2, the obtained ESN has the most sparse network structure and the best testing RMSE values. However, when α = 3, the corresponding testing RMSE values become larger. Hence, the too large or too small value of α is not preferred.

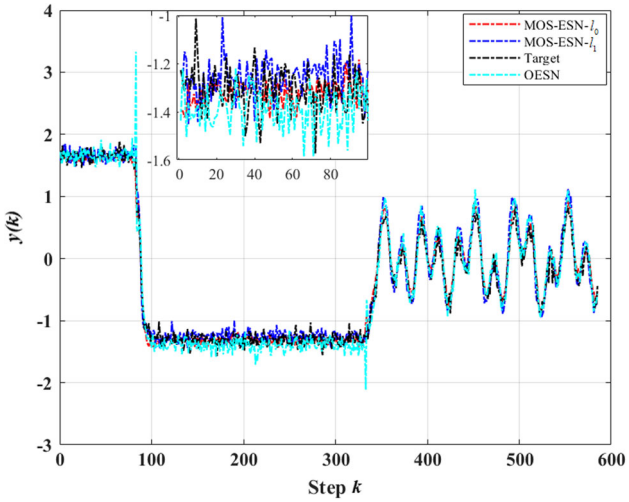

Fig. 12 The prediction outputs for nonlinear dynamic system

4.2 Nonlinear dynamic system modeling

The proposed method is performed on the nonlinear dynamic system as below

$$y(k+1) = 0.72y(k) + 0.025y(k-1)u(k-1) + 0.01u^2(k-2) + 0.2u(k-3) \quad (37)$$

where u(k) and y(k) are input and output, respectively. y($k+1$) is predicted by y(k), y($k-1$), u($k-1$), u($k-2$), u($k-3$). In the training phase, u(k) is 1.05sin(k/45). In the testing phase, u(k) is given as

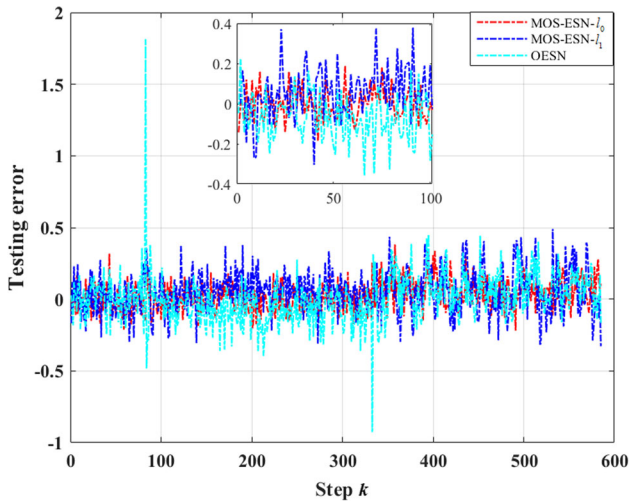

Fig. 13 The prediction errors for nonlinear dynamic system

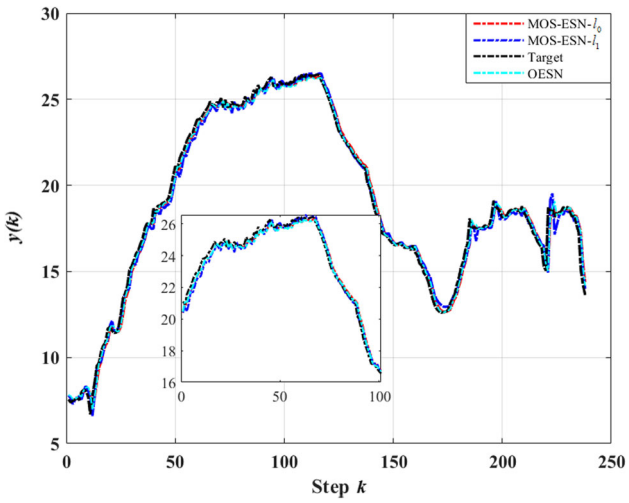

Fig. 14 Prediction output of effluent NH_4-N

$$u(k) = \begin{cases} \sin\dfrac{\pi k}{25}, & 0 < k \le 250 \\ 1.0, & 250 < k \le 500 \\ -1.0, & 500 < k \le 750 \\ 0.3\sin\dfrac{\pi k}{25} + 0.1\sin\dfrac{\pi k}{32} & \\ 0.6\sin\dfrac{\pi k}{10}, & 750 < k \le 1000 \end{cases} \tag{38}$$

In this experiment, 2000 samples are generated by Eq. (38). The first 1400 points are used in training stage, and the remaining 600 are used in testing phase. In addition, the 20-dB Gaussian noise is added to generate the noisy environment.

The prediction output and testing error of the resulted MOS-ESN-l_1, MOS-ESN-l_0 and OESN are plotted in Figs. 12 and 13, respectively. Actually, all the algorithms show similar predictive trend on nonlinear dynamic system. However, the prediction error of MOS-ESN-l_0 is limited in [-0.4,0.4], which is smaller than other methods. Thus, the proposed MOS-ESN-l_0 has the best prediction effect among all compared algorithms.

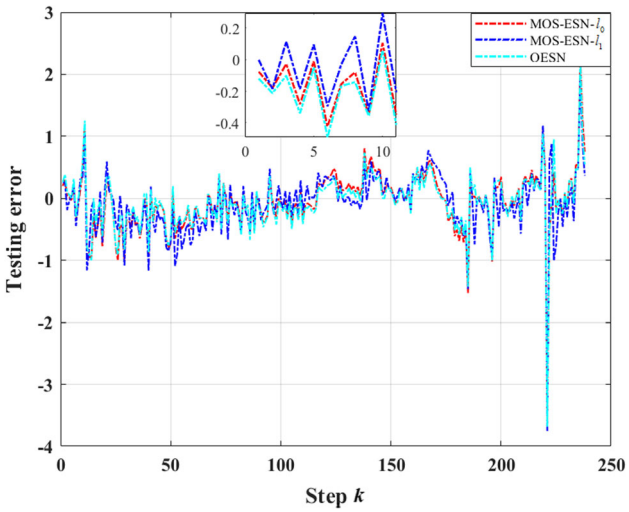

Fig. 15 Prediction errors of effluent NH_4-N model

The statistic results of 50 independent runs of compared algorithms are summarized in Table 6. Obviously, the OESN has shortest training time and smallest training

Table 6 Simulation results for nonlinear dynamic system modeling

Approaches	SP	Time(s)	Mean RMSE of training	Std. RMSE of training	Mean RMSE of testing	Std. RMSE of testing
OESN	100%	**6.65**	**0.0464**	**0.0058**	0.1799	0.0191
OESN-l_1	72.35%	26.09	0.1438	0.0104	0.1451	**0.0117**
OESN-l_0	70.68%	28.32	0.0915	0.0141	0.1068	0.0144
OESN-MOEA/D	71.92%	260.01	0.1579	0.0137	0.1585	0.0140
CD-ESN-l_1	68.12%	106.99	0.1238	0.0127	0.1243	0.0138
CD-ESN-l_0	66.76%	121.47	0.1227	0.0130	0.1235	0.0135
MOS-ESN-l_1	59.17%	633.15	0.1179	0.0134	0.1183	0.0128
MOS-ESN-l_0	**50.01%**	654.31	0.0913	0.0150	**0.1027**	0.0158

The best results are marked in bold

Table 7 Simulation results for Effluent $NH_4 - N$ model in WWTP

Approaches	SP	Time(s)	Mean MSE of training	Std. MSE of training	Mean MSE of testing	Std. MSE of testing
OESN	100%	**5.65**	**0.0056**	**0.0197**	0.7134	0.0504
OESN-l_1	78.77%	220.56	0.6986	0.0375	0.6998	0.0254
OESN-l_0	75.43%	227.99	0.5631	0.0346	0.5687	0.0358
OESN-MOEA/D	64.93%	170.06	0.5532	0.0305	0.5635	0.0314
CD-ESN-l_1	63.01%	301.22	0.3698	0.0297	0.3704	0.0302
CD-ESN-l_0	60.88%	329.88	0.3682	0.0252	0.3697	0.0288
MOS-ESN-l_1	56.48%	772.33	0.3589	0.0232	0.3603	0.0258
MOS-ESN-l_0	**48.17%**	774.87	0.3172	0.0205	**0.3181**	**0.0224**

error, but its testing error is largest, which indicates overfitting problem. Furthermore, the MOS-ESN-l_0 obtains the smallest testing RMSE and SP values, which means the MOS-ESN-l_0 has better prediction accuracy, sparser reservoir for nonlinear dynamic system modeling.

4.3 Effluent NH4 − N model in WWTP

Recently, the discharge of industrial and domestic wastewater has also increased sharply, and the phenomenon of water quality exceeding standard in the wastewater treatment process (WWTP) is serious. In WWTP, the excessive NH_4 -N will lead to eutrophication of water body and affect human health. Thus, predicting NH_4-N accurately is critical. However, the WWTP is a complex system with nonlinear, uncertainty, it is difficult to predict NH_4-N. To solve this problem, the laboratory analytical techniques are used. However, these methods always require long time.

In this section, the proposed MOS-ESN models are applied to predict NH_4-N in WWTP. This experiment contains 641 sets of data, which are collected from Chaoyang, Beijing in 2016. The first 400 groups are treated as training data and the rest 241 are set as testing data. The inputs of ESN include T, ORP, DO, TSS and pH, which are described in [29].

The prediction result of effluent NH4-N models of MOS-ESN-l_0, MOS-ESN-l_1 and OESN is demonstrated in Fig. 14, and the corresponding prediction error is shown in Fig. 15. Obviously, all the algorithms achieve the similar prediction accuracy. As compared with MOS-ESN-l_0 and MOS-ESN-l_1, it can be found that the l_0 regularization can get sparser structure, and thus the MOS-ESN-l_0-based effluent NH4-N model has smaller prediction error.

The comparison results of different models are shown in Table 7, including the network sparsity SP, training time, the mean and standard deviation of training and testing RMSE values of 50 independent experiments. Obviously, the OESN has small training but large testing RMSE values, which implies the overfitting problem occurs. Thus, the OESN has difficulty in predicting NH_4-N in WWTP. In OESN-l_1 and OESN-l_0, the regularization technique is applied to make the network structure sparse, but the testing RMSE value is still large. In CD-ESN-l_0 and CD-ESN-l_1, the regularization technique and coordinate descent are used to updates the output weights, which can obtain better prediction performance than OESN-l_1 and OESN-l_0. As compared with OESN-MOEA/D, the local search and weight vectors updating algorithm are applied in MOS-ESN-l_0 and MOS-ESN-l_1, which helps to improve solution performance. Particularly, the MOS-ESN-l_0 has the smallest testing error and the sparsest network structure, which can effectively predict NH_4-N of WWTP.

5 Conclusion

In this paper, the multi-objective sparse ESN is proposed, in which the training error and network structure are optimized simultaneously. Firstly, instead of searching the regularization parameters, the design of ESN is treated as a bi-objective optimization problem. Secondly, to improve algorithm convergence performance, the local search strategy is designed, which incorporates the l_1 or l_0 norm regularization and coordinate descent algorithm. Furthermore, to make the algorithm converge to the region of interest, the weight vectors updating method is designed, which applies the information of knee point. The effectiveness and usability of the proposed algorithm are evaluated by experimental results. In future work, this method will be applied in other practical engineering fields, such as garbage classification and image recognition.

Appendix

The parameters setting of different algorithms is given:

- OESN: The reservoir has 1000 nodes. The reservoir sparsity s_1 is chosen from the set (0.01, 0.015, …, 0.6),

and the spectral radius of reservoir s_2 is chosen from the set (0.1, 0.15, …, 0.95).

- OESN-l_1: The reservoir has same parameters as OESN. The regularization parameter λ_1 is selected by (LASSO) method [33].
- OESN-l_0: The reservoir has same parameters as OESN. The regularization parameter λ_0 is adaptively calculated [32].
- CD-ESN-l_1: The reservoir has same parameters as OESN. The regularization parameter λ is chosen from the set (0.05, 0.10, 0.15, …, 0.9) as suggested in [43].
- CD-ESN-l_0: The reservoir has same parameters as OESN. The regularization parameter λ is chosen from the set (0, 0.05, 0.15, …, 0.95).

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (61973010, 61890930–5,62021003, 61533002), in part by the National Natural Science Foundation of Beijing (4202006), and in part by the National Key Research and Development Project (2021ZD0112302, 2019YFC1906002, 2018YFC1900802

Data availability statement The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work in this paper.

References

1. Zhu T, Luo L, Zhang XL et al (2017) Time-series approaches for forecasting the number of hospital daily discharged inpatients. IEEE J Biomed Health Inform 21(2):515–526
2. Zhang H, Cao X, John H, Tommy C (2017) Object-level video advertising: an optimization framework. IEEE Trans Industr Inf 13(2):520–531
3. Safari N, Chung CY, Price G (2018) A novel multi-step short-term wind power prediction framework based on chaotic time series analysis and singular spectrum analysis. IEEE Trans Power Syst 33(1):590–601
4. Lee R (2020) Chaotic type-2 transient-fuzzy deep neuro-oscillatory network (CT2TFDNN) for worldwide financial prediction. IEEE Trans Fuzzy Syst 28(4):731–745
5. Li JD, Tang H, Wu Z et al (2019) A stable autoregressive moving average hysteresis model in flexure fast tool servo control. IEEE Trans Autom Sci Eng 16(3):1484–1493
6. Zhou D, Al-Durra A, Zhang K et al (2019) A robust prognostic indicator for renewable energy technologies: a novel error correction grey prediction model. IEEE Trans Industr Electron 66(12):9312–9325
7. Ciprian C, Masychev K, Ravan M et al (2020) A machine learning approach using effective connectivity to predict response to clozapine treatment. IEEE Trans Neural Syst Rehabil Eng 28(12):2598–2607
8. Park YM, Moon UC, Lee KY (1996) A self-organizing power system stabilizer using fuzzy auto-regressive moving average (FARMA) model. IEEE Trans Energy Convers 11(2):442–448
9. Xie N, Liu S (2015) Interval grey number sequence prediction by using non-homogenous exponential discrete grey forecasting model. J Syst Eng Electron 26(1):96–102
10. Zhang K, Liu Z, Zheng L (2020) Short-term prediction of passenger demand in multi-zone level: temporal convolutional neural network with multi-task learning. IEEE Trans Intell Transp Syst 21(4):1480–1490
11. Kuang W, Chan YL, Tsang SH et al (2019) Machine learning-based fast intra mode decision for HEVC screen content coding via decision trees. IEEE Trans Circuits Syst Video Technol 30(5):1481–1496
12. Han SJ, Bae KY, Park HS et al (2016) Solar power prediction based on satellite images and support vector machine. IEEE Transactions on Sustainable Energy 7(3):1255–1263
13. Liu YT, Lin YY, Wu SL et al (2015) Brain dynamics in predicting driving fatigue using a recurrent self-evolving fuzzy neural network. IEEE Transactions on Neural Networks and Learning Systems 27(2):1–14
14. Zhang HJ, Li JX, Ji YZ, Yue H (2017) Subtitle understanding by character-level sequence-to-sequence learning. IEEE Trans Industr Inf 13(2):616–624
15. Zsuzsa P, Radu EP, Jozsef KT et al (2006) Use of multi-parametric quadratic programming in fuzzy control systems. Acta Polytechnica Hungarica 3(3):29–43
16. Rizvi SA, Wang LC (1997) Nonlinear vector prediction using feed-forward neural networks. IEEE Trans Image Process 6(10):1431–1436
17. Shi Z, Liang H, Dinavahi V (2017) Direct interval forecast of uncertain wind power based on recurrent neural networks. IEEE Transactions on Sustainable Energy 9(3):1177–1187
18. Jaeger H, Hass H (2004) Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science 304:78–80
19. Wu Z, Li Q, Zhang HJ (2022) Chain-structure echo state network with stochastic optimization: methodology and application. IEEE Trans Neural Netw Learn Syst 33(5):1974–1985
20. Wu Z, Li Q, Xia XH (2021) Multi-timescale forecast of solar irradiance based on multi-task learning and echo state network approaches. IEEE Trans Industr Inf 17(1):300–310
21. Mantas L, Jaeger H (2009) Reservoir computing approaches to recurrent neural network training. Computer science review 3:127–149
22. Jaeger H (2007) Discovering multiscale dynamical features with hierarchical echo state networks. Jacobs University Bremen, Bremen
23. Jaeger H, Lukosevicius M, Popovici D et al (2007) Optimization and applications of echo state networks with leaky integrator neurons. Neural Netw 20(2007):335–352
24. Qiao J, Li F, Han H et al (2017) Growing echo-state network with multiple subreservoirs. IEEE Trans Neural Netw Learn Syst 28(2):391–404
25. Wang HS, Ni CJ, Yan XF (2017) Optimizing the echo state network based on mutual information for modeling fed-batch bioprocesses. Neurocomputing 225:111–118
26. Xu M, Han M (2017) Adaptive elastic echo state network for multivariate time series prediction. IEEE Trans Cybern 46(10):2173–2183
27. Yang C, Nie K, Qiao J et al (2022) Robust echo state network with sparse online learning. Inf Sci 594:95–117
28. Luo X, Chang X, Ban X (2016) Regression and classification using extreme learning machine based on l_1-norm and l_2-norm. Neurocomputing 174:179–186

29. Yang CL, Qiao JF, Wang L et al (2019) Dynamical regularized echo state network for time series prediction. Neural Comput Appl 31(10):6781–6794
30. Han M, Ren W, Xu M (2014) An improved echo state network via l_1-norm regularization. Acta Automatica Sinica 40(11):2428–2435
31. Dzati A, Ramli, et al (2017) Fast kernel sparse representation classifier using improved smoothed-l_0 norm. Proc Comput Sci 112:494–503
32. Yang CL, Qiao JF, Ahmad Z et al (2019) Online sequential echo state network with sparse RLS algorithm for time series prediction. Neural Netw 118:32–42
33. Qiao JF, Wang L, Yang CL (2018) Adaptive lasso echo state network based on modified Bayesian information criterion for nonlinear system modeling. Neural Comput Appl 31(10):6163–6177
34. Huang HZ, Gu YK, Du X (2006) An interactive fuzzy multiobjective optimization method for engineering design. Eng Appl Artif Intell 19(5):451–460
35. Zhang HJ, Sun YF, Zhao MB et al (2020) Bridging user interest to item content for recommender systems: an optimization model. IEEE Transactions on Cybern 50(10):4268–4280
36. Lin L, Yao X, Stolkin R et al (2014) An evolutionary multiobjective approach to sparse reconstruction. IEEE Trans Evol Comput 18(6):827–845
37. Rachmawati L, Srinivasan D (2009) Multiobjective evolutionary algorithm with controllable focus on the knees of the pareto front. IEEE Trans Evol Comput 13(4):810–824
38. Branke J, Deb K, Dierolf H et al (2004) Finding knees in multiobjective optimization. In: International Conference on Parallel Problem Solving from Nature, LNCS 3242:722–731
39. Das I (1999) On characterizing the 'knee' of the pareto curve based on normal-boundary intersection. Struct Multidiscip Optimiz 18(2):107–115
40. Zhang Q (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Transa Evolut Comput 11(6):712–731
41. Deb K, Gupta S (2011) Understanding knee points in bicriteria problems and their implications as preferred solution principles. Eng Optim 43(11):1175–1204
42. Weitian C, Brian DOA (2012) A combined multiple model adaptive control scheme and its application to nonlinear systems with nonlinear parameterization. IEEE Trans Autom Control 57(7):1778–1782
43. Yang CL, Wu ZH, Qiao JF (2020) Design of echo state network with coordinate descent method and l_1 regularization. Commun Comput Inf Sci 1265:357–367
44. Dong ZM, Wang XP, Tang LX (2020) MOEA/D with a self-adaptive weight vector adjustment strategy based on chain segmentation. Inf Sci 521:209–230
45. Ishibuchi H, Yoshida T, Murata T (2003) Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans Evol Comput 7(2):20