EAI Endorsed Transactions
on Energy Web
Research Article EAI.EU

Short-Term Power Load Forecasting Based on EMGCSO-Elman

Mingqing FENG[1]*, Rui DUAN[1], Yingcong WANG[2], Junwei SUN[2], Xiaoyan WANG[1] and Shuai YUAN[1]

[1]School of Information and Communication, Zhengzhou Electric Power College, Zhengzhou, Henan 450000, China
[2]School of Electrical and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou, Henan 450000, China

Abstract

INTRODUCTION: In short-term electrical load forecasting, Elman neural architectures often suffer from convergence instability and limited prediction accuracy due to random parameter initialization. To address this issue, an Elite Multi-mode Guided Chicken Swarm Optimization (EMGCSO) methodology is developed to determine the optimal weights and biases for the Elman network.
OBJECTIVES: The primary objective is to develop an EMGCSO-Elman integrated predictive framework for short-term power load forecasting. This involves designing an enhanced chicken swarm optimization algorithm that effectively guides the search process by classifying roosters based on their fitness scores into elite, failed, and ordinary categories, and leveraging optimal, random, and neighborhood optimal solution information to formulate three distinct search strategies.
METHODS: EMGCSO introduces three search strategies focusing on exploitation, exploration, and a balance between the two. These strategies are guided by elite individuals, random search, and neighborhood optima, respectively. The proposed algorithm is first validated through benchmark function tests. Subsequently, it is applied to optimize the initial parameters (weights and biases) of the Elman network, thereby constructing the EMGCSO-Elman forecasting framework.
RESULTS: Benchmark function test results demonstrate that EMGCSO achieves fast convergence and strong global search capability. When deployed for forecasting electric load in the short term, the EMGCSO-Elman model significantly outperforms BP, Elman, and CSO-Elman models in predictive accuracy together with stability.
CONCLUSION: The proposed EMGCSO-Elman model effectively overcomes the sensitivity of Elman neural networks to initial parameter settings, demonstrating clear superiority for short-term power load prediction tasks.

Keywords: Chicken swarm optimization algorithm; Short term load forecasting; Elite multi-mode guided strategy; Elman neural network

Received on 23 July 2026, accepted on 08 September 2026, published on 22 September 2026

doi: 10.4108/ew.14137

*Corresponding author. Email: fmq_1978@163.com

1. Introduction

It grants the electrical grid infrastructure the ability to maintain an equilibrium between generation and consumption while also ensuring the security of electrical energy operation [1]. Power forecasting tasks are generally categorized as medium- and long-term forecasting, short-term forecasting, extended short-term forecasting, and ultra-short-term forecasting. Among them, the goal of short-term load prediction is to estimate system demand for each time period in the next one to several days. Accurate and stable performance of short-term load forecasting directly affects the system day-ahead scheduling plan. It can effectively cut down the production costs of generating units, yielding high economic and social benefits. Moreover, it holds tremendous significance in guaranteeing that the power network operates in a secure as well as robust manner [2].

Power demand prediction for the short term is a typical example of a time series prediction problem. Often, the process is affected by different factors such as policy rules, weather, and economics [3]. Thus, the process cannot be easily modeled using traditional linear mathematical equations. As a typical algorithm, the Back Propagation (BP) neural network model is one that boasts a fairly simple structure. The algorithm can map an arbitrary nonlinear function from input to output and has found wide use in load forecasting of power. An example is the study by Wang et al. [4], which developed a two-layer optimization BP algorithm using the multi-parameters GA. This algorithm optimized the hidden layer neurons, weight, biases, and employed the stepwise increment and decrement learning factor in the correction of errors, resulting in a very precise output MAPE of 2.71%. An alternative research [5] suggested the Particle Swarm Optimization BP (PSO-BP) model, which was meant to address the issue of parameters estimation, by applying the PSO technique to optimize globally the weights and learning rates, resulting in 63.6% and 54.90% reduction of root mean square error (RMSE) in comparison to BP and LSTM, correspondingly. To tackle load fluctuations caused by the high penetration of renewable energy, a BP neural network incorporating a multi-dimensional input feature set was constructed [6], demonstrating stronger adaptability in scenarios with high proportions of wind and photovoltaic integration, and reducing mean square error (MSE) and MAPE by approximately 15% and 12% compared to traditional methods. Furthermore, based on ensemble learning theory, an Adaboost-BP ensemble framework with a dynamically adjusted weight updating scheme was introduced [7]. This framework decomposes historical load data using ensemble empirical mode decomposition and dynamically updates weak learner weights to construct a strong learner, demonstrating high engineering application value

In contrast to standard BP neural networks, the Elman network incorporates delay units positioned between its hidden and output layers. This endows it with the capability to handle time-varying characteristics and has led to its gradual application for predicting electrical loads within power grids [8-10]. An Elman model's predictive accuracy is heavily contingent upon its internal parameter configuration, and the selection of these parameters essentially constitutes an optimization problem. Hence, by leveraging effective optimization algorithms, a suitable set of parameters for the Elman network can be determined reasonably. For example, in literature [11], the connection weights and bias terms of the Elman network were taken as optimization variables, subsequently, the deviation separating predicted output from actual ground truth served as the objective criterion for the particle swarm optimization algorithm to build a PSO-Elman power load forecasting model. Literature [12] utilized chaotic mapping, a cosine-based stochastic strategy and inertia weights to improve the Gray Wolf Optimization algorithm. Subsequently, it carried out parameter optimization for the Elman network for the purpose of achieving accurate near-future electrical load forecasting. Literature [13] introduces an Elman neural network for power load forecasting based on two core modules: feature extraction using a Gaussian kernel function and parameter optimization via an improved seagull optimization algorithm. To mitigate the training convergence inconsistency caused by random initialization of weights and biases in the Elman network, literature [14] and [15] respectively developed a photovoltaic (PV) output forecasting model using an enhanced firefly algorithm and a wind energy forecasting framework based on a modified whale optimization algorithm.

Chicken Swarm Optimization (CSO) [16] represents a relatively recent Swarm Intelligence Optimization (SIO) technique, which can be regarded as a combination of elements from the synthetic foraging strategy of artificial bee colonies alongside evolutionary mutation operators [17] and PSO paradigm, furthermore, it possesses advantages including clear structure and robust global exploration capability. In this paper, the leading role associated with the dominant rooster within the canonical CSO framework is fully utilized. An elite multi-mode guided avian collective intelligent optimizer (EMGCSO) is hereby put forward to boost the optimization capability of the CSO. The effectiveness of the EMGCSO in benchmark function optimization is validated by conducting comparisons between it and six outstanding SIO algorithms on both unimodal and multimodal functions. The EMGCSO is utilized for tuning the Elman network's initial weights and biases, thereby establishing an EMGCSO-Elman predictive model dedicated to short-interval electrical load forecasting. Experimental comparisons involving the conventional BP, Elman, and CSO-Elman models confirm that the proposed forecasting framework exhibits high prediction accuracy and stability.

2. Elman neural network

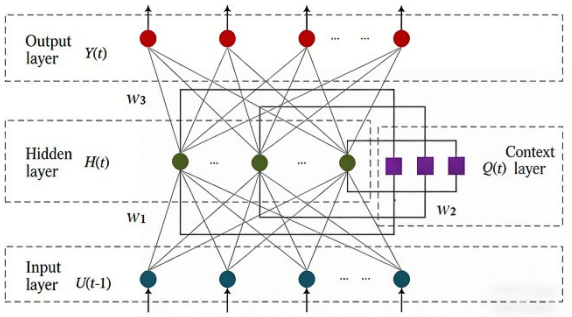

Figure 1. Architecture of the Elman network

The Elman-style dynamic neural framework constitutes a type of cyclically connected dynamic neural computational model that features regionally stored memory cells alongside localized recurrent pathways [13]. Typically, it consists of four primary components: an input layer, a hidden layer, and context layers (also known as the state layer or the memory layer), along with the output stratum. This network structure is illustrated in Figure 1.

The input layer is responsible for signal transmission. The hidden layer acts as a connection between the input and output layer. Meanwhile, a memory layer (often termed the context unit) is employed for the purpose of recording the internal state from the preceding hidden stage originating from the prior time step. The output layer computes a weighted sum of the hidden layer outputs. The Elman neural network incorporates a context structural design that follows the architecture that of the BP model. This component stores and delays the hidden-layer output and feeds it back as part of the inputs feeding into the hidden layer. Stated differently, the total input received by what is fed into the hidden layer is jointly dependent on the outputs originating from both the incoming signal layer and the recurrent memory layer. This self-association mechanism creates self-feedback within the hidden layer. As a result, the network becomes highly sensitive to historical data, which significantly enhances its capability to process time series information and realizes the objective of dynamic modeling. In the domain of power load forecasting, the context layer serves as a delay unit. This enables the hidden layer to fully leverage historical power load data from the previous time step, demonstrating excellent prediction performance for power load time series data [18].

The Elman neural network may be mathematically represented as follows:

$$Y(t) = g(w_3 H(t)) \tag{1}$$

$$H(t) = f(w_1 U(t-1) + w_2 Q(t)) \tag{2}$$

$$Q(t) = \sigma \cdot Q(t-1) + H(t-1) \tag{3}$$

where t denotes the sampling moment; $U(t\text{-}1)$ represents the external input signal fed into the neural network; $H(t)$ is the hidden-layer; $Q(t)$ stands for the state output from the context layer; $Y(t)$ is the final network output; w_1 ,w_2 ,and w_3 correspond to the weight matrices connecting the input layer to the hidden layer, the context layer to the hidden layer, and the hidden layer to the output layer, respectively; $f(\cdot)$ and $g(\cdot)$ denote the activation functions of neurons in the hidden and output layers, respectively; σ is the self-feedback gain.

The way errors propagate through the Elman network is described by equation (4):

$$E = \frac{1}{n}\sum_{t=1}^{T}(Y(t) - y(t))^2 \tag{4}$$

where $y(t)$ denotes the desired output.

The error of the Elman network is computed using the above formula, and the weights and biases are updated accordingly. This iterative optimization continues until the maximum number of training epochs is reached or the output deviation meets the specified accuracy requirement.

CSO algorithm

The CSO represents a relatively cutting-edge SIO inspired by collective animal behavior put forward through the work of Meng et al. [16] in 2014. This algorithm emulates the hierarchical structure along with the food-searching patterns of chickens. In CSO, chickens are classified into three categories: roosters, hens, and chicks. When addressing specific optimization problems, the following assumptions are made:

(1) Suppose the swarm contains N individuals, and each feasible candidate position resides within a D-dimensional coordinate system. Each individual within the population can be represented by its positional vector in the search space. For instance, x_i,j_t indicates the location of the i-th entity along the j-th coordinate axis at the current iterative step t.

(2) Each flock member represents a candidate solution to the optimization problem. On the basis of the respective fitness evaluation outcomes, the top N_R candidate solutions are designated to serve as roosters, while the poorest-performing N_C individuals are labeled as chicks, with the remaining N_H individuals N_H (where N_H=N—N_R—N_C) falling into the hen category.

(3) The populations are divided into N_R sub-populations. Every sub-population contains a single dominant rooster accompanied by a certain number of subordinate hens and juvenile chicks. Hens randomly select a rooster as their mate, and chicks randomly choose a hen as their mother.

(4) The rooster-hen-chick hierarchy, the rooster-hen mating associations, and the hen-chick maternal relationships remain unchanged for G successive generations and are updated every G generations, where G denotes the relationship update interval.

For any rooster individual i, its update rule for position is given by the following search equation.

$$x_{i,j}^{t+1} = x_{i,j}^{t} \times (1 + randn(0, \sigma^2)) \tag{5}$$

$$\begin{cases} \sigma^2 = exp((f_k - f_i)/(abs(f_i) + \varepsilon)), f_i > f_k \\ \sigma^2 = 1, f_i \le f_k \end{cases} \tag{6}$$

where $randn(0,\sigma^2)$ denotes a random number following a normal (Gaussian) distribution with zero mean and variance σ^2; ε is an extremely small positive constant, added to avoid division-by-zero issues; the fitness values of the ith and kth

roosters are respectively denoted as f_i and f_k, correspondingly, with the condition that $k \neq i$.
When the ith individual is a hen that follows the rooster with designation r_1 for foraging, the following search equation is used.

$$x_{i,j}^{t+1} = x_{i,j}^{t} + c_1 \times rand(x_{r_1,j}^{t} - x_{i,j}^{t}) + c_2 \times rand(x_{r_2,j}^{t} - x_{i,j}^{t}) \quad (7)$$

$$c_1 = exp((f_i - f_{r_1}) / (abs(f_i) + \varepsilon)) \quad (8)$$

$$c_2 = exp(f_{r_2} - f_i) \quad (9)$$

where c_1 and c_2 are learning factors; *rand* denotes a uniformly distributed stochastic variable over the range [-1,1]; r_1 denotes the rooster in the subpopulation where the ith hen is located; r_2 is a rooster or hen selected randomly from the entire flock, satisfying $r_1 \neq r_2 \neq i$.
For a chick individual i that follows a specific hen indexed as m during foraging, its position update follows the rule below.

$$x_{i,j}^{t+1} = x_{i,j}^{t} + FL \times (x_{m,j}^{t} - x_{i,j}^{t}) \quad (10)$$

where m denotes the mother hen associated with the ith chick; FL is a uniformly distributed random variable on [0,2]

3. EMGCSO-Elman model

3.1. Elite multi-mode guided chicken swarm optimization

In CSO, roosters forage autonomously, hens forage in the vicinity of roosters, and chicks forage around hens. This search process is illustrated in Figure 2, which can be described as a satellite-surround structure, with roosters leading the population search [19]. Roosters, as elite individuals in CSO, determine the search direction of the population. If roosters fall into local optima, hens and chicks are also susceptible to being trapped in such suboptimal solutions under the rooster-dominated guidance mechanism [20]. In swarm intelligence optimization algorithms, achieving a proper balance between exploration and exploitation is an effective way to enhance the algorithm's performance [21,22]. Therefore, three search equations are designed for roosters, one focusing on exploration, one favoring exploitation, and one balancing exploration and exploitation. Consequently, the flock can achieve the exploration-exploitation balance under the roosters' guidance. Thus, this paper proposes EMGCSO.

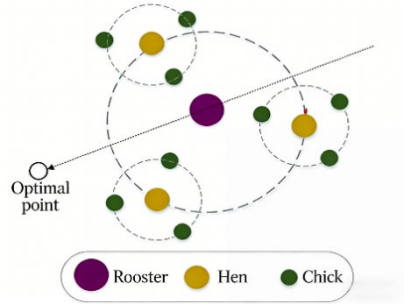

Figure 2. Chicken swarm optimization process

In EMGCSO, roosters are sorted in each generation according to fitness, and the sorting values are calculated as follows:

$$f(x_1) \leq f(x_2) \leq \cdots \leq f(x_r) \leq \cdots \leq f(x_{N_R}) \quad (11)$$

$$FR(x_r) = N_R - I_R + 1 \quad (12)$$

where N_R denotes the rooster subpopulation size; x_r represents the position associated with the rth rooster; $f(x_r)$ gives the fitness level attained by that rooster; I_R indicates the ranking of the rth rooster as determined by its fitness value; and $F_R(x_r)$ is the computed rank score assigned to the rth rooster. A rooster possessing a higher fitness value will be assigned a larger ranking value. The rooster subpopulation is divided into three categories according to their fitness ranking. Those with the largest ranking values consist of successful roosters the group with the smallest ranking values consists of failed roosters, and the remainder are ordinary roosters. To achieve a comprehensive trade-off among convergence precision, convergence rate and robustness of the proposed method, this paper configures a split ratio for high-quality, ordinary and low-quality roosters as 1:2:1.
Within the realm of SIO, the two mutually restrictive search mechanisms are known as exploration and exploitation. The former can be viewed as a global search process that does not favor any specific portion of the solution space. Excessive exploration tends to lead to disordered random search behavior. On the other hand, exploitation can be interpreted as a local search that focuses on promising regions within the solution space. Overemphasizing exploitation typically leads to getting trapped in a local optimum. Hence, incorporating stochastic solution information into the search equation can enhance the exploration ability, while introducing optimal solution information can boost the exploitation capability [23]. Moreover, considering that the value of the solution gradually diminishes from the optimal solution to the suboptimal solution and then to the stochastic solution, ordinary roosters can leverage the information from the suboptimal solution (i.e., the neighborhood optimum) during the search procedure to achieve a well-balanced interplay between broad exploration and focused exploitation. Based on the above discussion, the update rules for the three categories of roosters are presented as follows:

$$x_{i,j}^{t+1} = x_{i,j}^{t} \times (1 + randn(0, \sigma^2)) + rand(x_{best,j}^{t} - x_{i,j}^{t}) \quad (13)$$

where j denotes a stochastically chosen dimension index within the interval [1, D]; $x_{best,j}^{i}$ the jth dimension of the global optimum recorded during the tth iteration. By integrating the global optimal information into Equation (13), this update rule exhibits strong exploitation capability.

$$x_{i,j}^{t+1} = x_{nb,j}^{t} + \varphi_{i,j}(x_{nb,j}^{t} - x_{best,j}^{t}) + \psi_{i,j}(x_{nb,j}^{t} - x_{p,j}^{t}) \quad (14)$$

where $x_{nb,j}^{t}$ is the optimal solution within the neighborhood of $x_{i,j}^{t}$; p denotes a stochastically selected integer drawn from the collection [1, 2, …, N] such that $p \neq nb$; $\varphi_{i,j}^{t}$ and $\psi_{i,j}^{t}$

are uniformly distributed random values over [-1,1]. The search operation is carried out using the *k*-neighborhood approach. Here, the neighborhood radius is denoted by the parameter *k*, and its value is assigned as 10 in accordance with reference [24]. In Equation (14), the search process initiates with the optimal solution of the neighborhood of the rooster individual *i* as the starting point. By introducing both the optimal solution and random solution information, this search equation effectively balances the exploration and exploitation capabilities.

$$x_{i,j}^{t+1} = x_{i,j}^{t} \times (1 + randn(0,\sigma^2)) + rand(x_{q,j}^{t} - x_{i,j}^{t}) \quad (15)$$

where q is a random number randomly selected from [1, 2, ..., N]. By adding the random solution information in the population in Eq. (15), this search equation has good exploration capability.

Moreover, inertia weights serve as an effective mechanism for balancing global exploration and local exploitation in optimization frameworks. A larger inertia weight enhances exploration, while a smaller one promotes exploitation [25]. Specifically, a larger inertia weight (0.9) is assigned to failed roosters to further boost their exploratory behavior. Conversely, a smaller inertia weight (0.4) is assigned to successful roosters to strengthen their local exploitation capability.

In this paper, roosters are categorized into successful roosters, failed roosters, and ordinary roosters according to their fitness values. Subsequently, optimal solution information, random solution information, and neighborhood optimal solution information are incorporated to define search patterns for these three types of roosters. Specifically, the search pattern for successful roosters is biased towards exploitation, that for failed roosters is centered on exploration, and that for ordinary roosters strikes a middle ground that reconciles local refinement with global search. Ultimately, the chickens are led by roosters to reach a state where both global and local search capabilities are appropriately harmonized within the algorithm.

3.2. Testing of the EMGCSO algorithm

To assess the performance of EMGCSO, eight commonly used benchmark test functions are selected for numerical experiments. EMGCSO is compared with six high-performance SIO algorithms, namely: an ABC variant employing multiple search strategies (denoted as ABCX) [26]; an ABC algorithm incorporating neighborhood selection (NSABC) [27]; an ABC version guided by multiple elite individuals (MGABC) [28]; ensemble PSO (EPSO) [29]; an expanded PSO featuring multi-exemplar and forgetting mechanisms (XPSO) [30]; and a PSO with adaptive strategy dynamics (SDPSO) [31]. The parameters of all competing methods are configured according to their original references. As for the EMGCSO method proposed herein, its key parameters are configured as follows: a population size N = 100, a relationship update interval G = 10, rooster count N_R = 0.2N, hen count N_H = 0.6N, and the number of chicks determined by N_C=N—N_R—N_H. The scaling factor FL is set to a random value uniformly distributed between 0.4 and 0.9, while the maximum and minimum inertia weights are assigned as w_{max} = 0.9 and w_{min} = 0.4, respectively. Table 1 provides the equations, search spaces, functional classes, and optimum values for the eight benchmark functions. The letters *U*, *M*, *S*, and *N* used in Table 1 represent unimodal, multimodal, separable, and non-separable functions, respectively.

Table 1. Benchmark test functions

Functions	Domain	Type	Optimal Value
$f_1(\vec{x}) = \sum_{i=1}^{D} x_i^2$	$[-100,100]^D$	US	0
$f_2(\vec{x}) = \sum_{i=1}^{D} (10^6)^{(i-1)/(n-1)} x_i^2$	$[-100,100]^D$	UN	0
$f_3(\vec{x}) = max_i\{\lvert x_i \rvert, 1 \le i \le D\}$	$[-100,100]^D$	UN	0
$f_4(\vec{x}) = \sum_{i=1}^{D} i\, x_i^4$	$[-1.28,1.28]^D$	US	0
$f_5(\vec{x}) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12,5.12]^D$	MS	0
$f_6(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,600]^D$	MN	0
$f_7(\vec{x}) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi\, x_i))$	$[-50,50]^D$	MN	0
$f_8(\vec{x}) = \sum_{i=1}^{D} \lvert x_i \cdot \sin(x_i) + 0.1 \cdot x_i \rvert$	$[-10,10]^D$	MS	0

Table 2. Comparison results of EMGCSO and other algorithms

Function	[ABCX]		[NSABC]		[MGABC]		[EPSO]		[XPSO]		[SDPSO]		[EMGCSO]
	mean std		mean std		mean std		mean std		mean std		mean std		mean std
f_1	1.91E-26 1.37E-26	+	7.17E-46 ±2.63E-45	+	1.08E-92 ±2.83E-92	+	9.51E-23 ±1.40E-22	+	1.04E-52 ±3.55E-52	+	5.41E-30 ±1.23E-29	+	**5.36E-217** **0.00E+00**
f_2	6.91E-22 6.20E-22	+	6.86E-41 ±2.40E-41	+	1.52E-30 ±1.49E-30	+	4.55E-18 ±8.28E-18	+	9.16E-50 ±8.80E-50	+	1.10E-19 ±1.82E-19	+	**4.36E-300** **1.35E-289**
f_3	1.25E+0 0 3.98E- 01	+	6.76E+00 1.11E+00	+	6.03E-37 7.99E-37	+	9.39E-02 7.87E-02	+	2.28E-03 2.91E-03	+	6.00E-02 5.51E-02	+	**8.05E-110** **7.64E-110**
f_4	1.62E-02 3.58E-03	+	8.59E-02 2.34E-02	+	1.58E-03 6.35E-04	+	7.49E-03 2.81E-03	+	6.49E-03 2.33E-03	+	6.72E-03 2.99E-03	+	**1.74E-04** **6.89 E-05**
f_5	**0.00E+0 0 0.00E+0 0**	=	**0.00E+00 0.00E+00**	=	2.82E-09 1.26E-08	+	1.02E+013. 37E+00	+	4.19E+011. 26E+01	+	6.27E+001. 93E+00	+	**0.00E+00 0.00E+00**
f_6	**0.00E+0 0 0.00E+0 0**	=	1.72E-06 4.97E-06	+	**0.00E+00 0.00E+00**	=	2.69E-02 2.84E-02	+	1.06E-02 1.25E-02	+	1.02E-02 1.24E-02	+	**0.00E+00 0.00E+00**
f_7	1.12E+0 17.56E+ 00	+	3.73E-14± 4.14E-15	+	**8.88E-16± 0.00E+00**	=	4.23E-11± 6.72E-11	+	5.78E-02± 2.58E-01	+	4.48E-14± 3.55E-14	+	**8.88E-16± 0.00E+00**
f_8	2.89E-11 8.83E-12	+	2.22E-16 3.55E-16	+	8.41E-48 7.86E-48	+	1.24E-10 7.62E-09	+	3.09E-06 6.65E-06	+	2.30E-09 2.15E-09	+	**2.62E-196 3.75E-196**
+/=/-	6/2/0		7/1/0		6/2/0		8/0/0		8/0/0		8/0/0		-
Mean ranking	5.11		4.43		2.35		5.89		4.34		4.44		**1.45**

For making the comparison fair, all the algorithms have been independently tested 20 times in a $D = 30$ dimensional problem, where the termination criterion is determined such that the number of function evaluations become $3000 \times D$. The entire simulation process has been conducted on a Windows 10 system using MATLAB R2017b software, where the computer system specifications include an Intel Core i5-5350 CPU processor operating at 1.8 GHz with 8 GB RAM.In order to carry out the comparison experiments, the algorithms have been analyzed from three different viewpoints as follows: (1) The performance of the algorithms is evaluated by analyzing the mean value and the standard deviation of the optimal solutions. (2) Statistical tests such as the Wilcoxon signed-rank test and the Friedman test are performed in SPASS software environment. While the former one helps us identify whether there is any significant difference between two algorithms, the latter is used to analyze the optimization ability of the algorithms. (3) Convergence curves are plotted to illustrate how quickly each algorithm reaches optimal solutions across different test functions. Table 2 lists a performance comparison involving EMGCSO and several baseline algorithms on eight test functions. The comparison not only involves assessing solution quality using mean and standard deviation values but also a comprehensive analysis based on nonparametric tests. In Table 2, "mean" and "std" represent the average and the corresponding standard deviation calculated from 20 independent executions of each algorithm. The best-performing results are highlighted in bold. Given that the test functions are formulated as minimization problems, a smaller mean and standard deviation signify better quality of the resulting solution. As evident from Table 2, the EMGCSO algorithm yields superior solutions for five of the functions and attains results equivalent to the optimal solutions among the other algorithms for the remaining three functions.

In Table 2, the symbols "+", "-", and "=" represent the outcomes of the Wilcoxon signed-rank test conducted at a significance level of 0.05. These symbols indicate that EMGCSO is significantly better, significantly worse, or not significantly different from the corresponding benchmark algorithm on each function, respectively. Table 2 shows that EMGCSO substantially outperforms EPSO, XPSO, and SDPSO across all functions. Additionally, it significantly outperforms ABCX, NSABC, and MGABC on 6, 7, and 6 functions, respectively, while showing no significant difference on the remaining functions. The "Mean ranking" in Table 2 is the result of the Friedman test. A smaller mean rank indicates better overall optimization performance. It is clear that EMGCSO ranks first, followed by MGABC, while EPSO ranks last. Figure 3 further illustrates the convergence curves for each algorithm across the test functions. From Figure 3, it can be clearly observed that EMGCSO achieves both faster convergence rate and

higher convergence precision across all types of benchmark functions. For instance, although ABCX, NSABC, and EMGCSO all converge to the theoretical optimum value of 0 for function f5, EMGCSO achieves this with the fastest convergence speed. Similarly, for function f6, ABCX, MGABC, and EMGCSO all reach the theoretical optimum of 0, yet EMGCSO exhibits the quickest convergence rate.

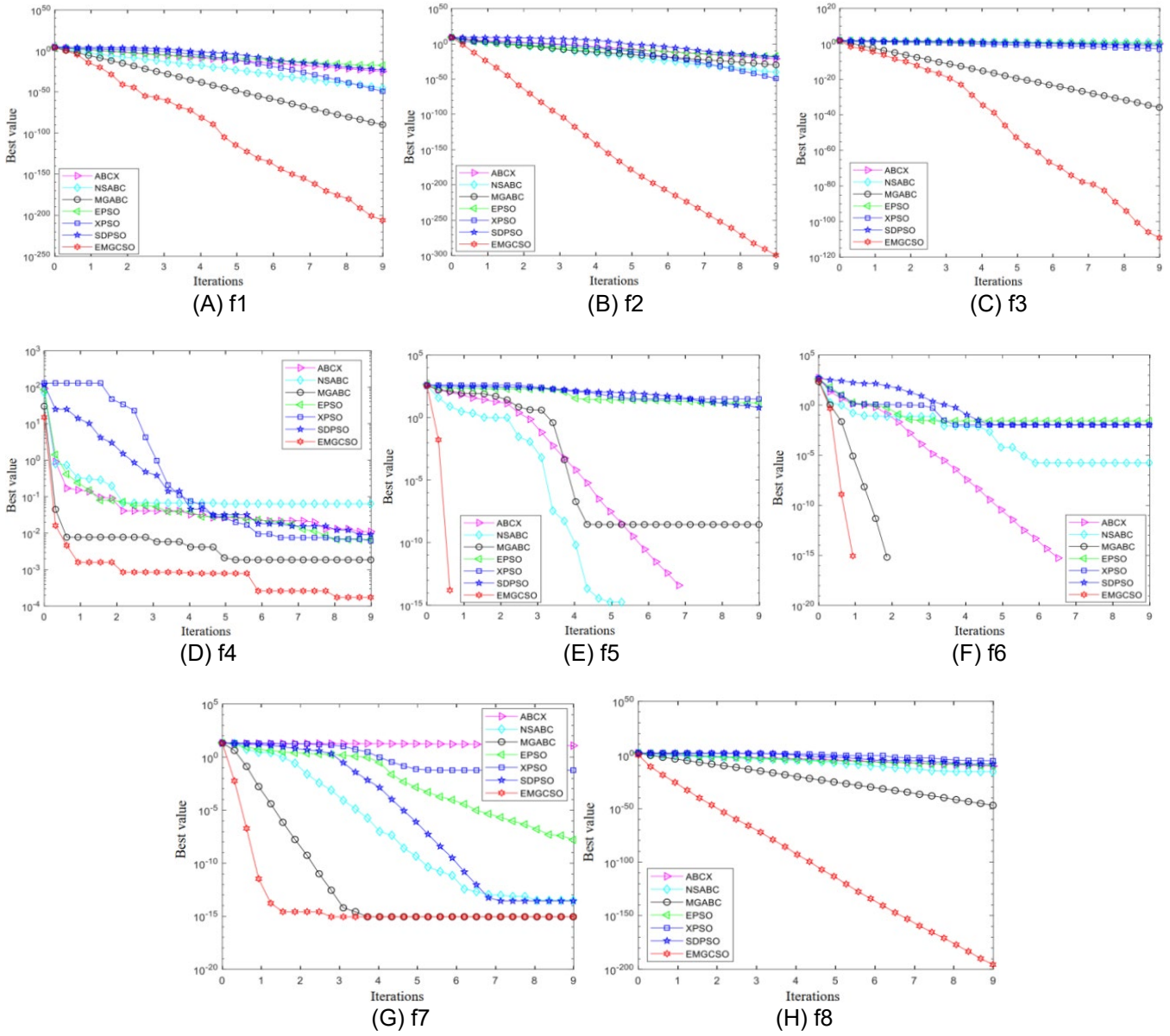

Figure 3. Convergence curves for all algorithm

Table 3. Ablation study results of EMGCSO variants on benchmark functions

Function	[EMGCSO-w/o-EG] (mean±std)	[EMGCSO-w/o-RS] (mean±std)	[EMGCSO-w/o-NG] (mean±std)	[EMGCSO] (mean±std)
f_1	1.23E-45±2.34E-45	2.45E-67±4.56E-67	5.67E-89±1.23E-88	**5.36E-217±0.00E+00**
f_2	3.45E-78±5.67E-78	6.78E-102±1.23E-101	1.23E-145±2.34E-145	**4.36E-300±1.35E-289**
f_3	2.34E-32±4.56E-32	5.67E-48±1.23E-47	1.23E-56±2.34E-56	**8.05E-110±7.64E-110**
f_4	2.45E-03±5.67E-04	1.23E-03±3.45E-04	5.67E-04±1.23E-04	**1.74E-04±6.89 E-05**
f_5	1.23E-02±2.34E-02	2.34E-04±4.56E-04	5.67E-06±1.23E-05	**0.00E+00±0.00E+00**
f_6	2.34E-03±4.56E-03	1.23E-04±2.34E-04	5.67E-06±1.23E-05	**0.00E+00±0.00E+00**
f_7	3.45E-15±5.67E-15	2.34E-15±3.45E-15	1.23E-15±2.34E-15	**8.88E-16±0.00E+00**

f_8	1.23E-78±2.34E-78	2.34E-102±4.56E-102	5.67E-134±1.23E-133	**2.62E-196±3.75E-196**

To quantitatively evaluate the contribution of each improvement strategy in the proposed EMGCSO algorithm, we conduct ablation experiments on the eight benchmark functions. Three variants of EMGCSO are designed: without elite guidance (EMGCSO-w/o-EG), without random search (EMGCSO-w/o-RS), and without neighborhood optimum guidance (EMGCSO-w/o-NG). Table 3 reports the mean and standard deviation of the optimal solutions obtained by each variant and the complete EMGCSO. As shown in Table 3, the complete EMGCSO consistently achieves the best performance on all functions. In particular, elimination of elite guidance (w/o-EG) results in considerable deterioration of the performance on unimodal problems (f_1-f_3). This finding confirms the importance of elite guidance for exploitation and fast convergence. Elimination of random search (w/o-RS) influences significantly multimodal problems (f_5-f_8). The findings demonstrate that random search increases the diversity of population and assists in avoiding local optimums. The negative impact of neighborhood guidance (w/o-NG) on all problems is moderate. Thus, all three strategies are effective for the EMGCSO algorithm.

EMGCSO optimization of Elman neural network

Elman neural networks are highly suitable for the modelling and forecasting of sequential data. Nevertheless, specifying the initial connection weights and biases among nodes of this network often depends on the user's prior knowledge. This approach lacks a solid theoretical foundation and has the potential to result in inaccurate prediction outcomes [32,33]. In this paper, EMGCSO is utilized to identify the optimal initial weights and biases of the Elman network Consequently, an EMGCSO-Elman prediction model is proposed, aiming to achieve accurate short-term predictions of electricity loads.

The schematic illustration outlining the entire workflow for the developed EMGCSO-Elman model is presented in Figure 4, and the core steps of the forecasting procedure are outlined below.

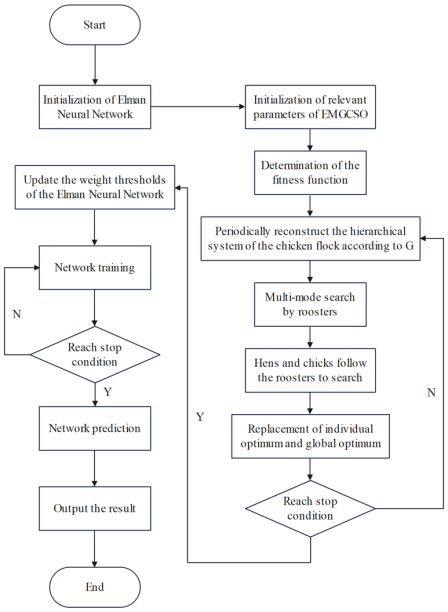

Figure 4. Flowchart of EMGCSO-Elman

(1) Preprocess the power load data to identify outliers and missing values, and use the Laida criterion for enhancing load data accuracy.

(2) Initialize the weights, biases, and training parameters for the Elman network, then determine the node count within the implicit layer using an empirical trial-and-error approach, whose empirical formula is:

$$l = \sqrt{m+n} + \alpha \quad (16)$$

where l denotes the number of hidden-layer neurons, m and n denote the numbers of input- and output-layer neurons, respectively, and α is a random value within [1,10]. Candidate hidden-layer sizes obtained from Equation (16) were further examined by trial-and-error, and l=8 was retained as the final configuration. This hidden-layer size was fixed for all subsequent forecasting experiments. Initialize the relevant parameters of EMGCSO, including N, N_R, N_H, N_C, G, FL, MAX_FES, L_b, and U_b. The search dimension is determined by $D=ml+l^2+ln+l+n$, where ml, l^2, and ln correspond to the input-hidden, context-hidden, and hidden-output connection weights, while $l+n$ denotes the hidden- and output-layer thresholds. For the 6-8-1 Elman topology used in the experiments, $D=6\times8+8^2+8\times1+8+1=129$."

(3) The error function associated with the Elman network is defined to serve as the fitness function for the EMGCSO as shown in equation (17). The optimization objective consists of identifying the initial weights and biases which yield the smallest error function value.

$$Fitness = E = \frac{1}{n}\sum_{t=1}^{T}(Y(t) - y(t))^2 \quad (17)$$

(4) Randomly generate the number of N flock individuals in the optimization space, establish the flock hierarchy, and classify the

roosters into three types: successful roosters, ordinary roosters, and failed roosters according to the fitness value.
(5) The roosters execute different search equations depending on their type, while hens and chicks follow the roosters within their corresponding subpopulations during the search process.
(6) Keep updating the best positions found by each individual and by the entire swarm iteratively, and output the position of the optimal flock individual after reaching the EMGCSO stopping condition, i.e., the optimal initial weights and biases of the Elman network.
(7) The discovered optimal initial weights and biases are then assigned to the Elman network, which undergoes training repeatedly until its termination criteria are satisfied. Once training is complete, the resulting model is employed to make predictions.

4. Simulation outcomes and evaluation

4.1. Data Preprocessing

The data used in this study are sourced from the 2017 electric load data of a full-service restaurant located in Zhengzhou, Henan Province, China, which constituted a portion of the data in the "China Electrical Engineering Society Cup Mathematical Modeling Competition". The historical data have a temporal resolution of one hour, i.e., 24 data points per day. The data preprocessing procedure consists of the following steps.
(1) Missing value handling: Missing values in the dataset are identified and filled using linear interpolation, which estimates missing values based on the values of adjacent time points.
(2) Outlier detection using the Laida criterion (3σ rule): Outliers are detected using the Laida criterion (also known as the 3σ rule). For each time series, the mean (μ) and standard deviation (σ) are calculated. Any data point that deviates from the mean by more than 3σ (i.e., $|x - \mu| > 3\sigma$) is considered an outlier and is replaced with the boundary value ($\mu \pm 3\sigma$).
(3) Normalization: To eliminate the influence of different scales, all input features are normalized to the range [0, 1] using min-max normalization.
(4) Seasonal division using K-means clustering: The annual load curve is divided into two typical load profiles: off-season and peak-season. This is achieved by applying the K-means clustering algorithm to daily load curves (24-dimensional vectors). The number of clusters is set to $k = 2$ based on the elbow method, which identifies the optimal k by plotting the within-cluster sum of squares as a function of k. The elbow point at $k = 2$ indicates that two clusters best capture the distinct seasonal patterns in the load data.
(5) Training and testing data partition: The historical load data selected for the analysis are as follows. Off-season: March 6 to July 23, 2017 (20 weeks in total). Peak-season: January 2 to February 26, 2017 and October 2 to December 24, 2017 (20 weeks in total).
For load forecasting, the input variable is the load value at the same specific moment six days prior to the forecast date, and the output variable is the load value at that same moment on the forecast date. A rolling forecasting strategy is adopted for the 7-day test period (July 17–23, 2017 for off-season and December 18–24, 2017 for peak-season). All compared models use identical chronological training/test partitions, preprocessing results, and input-output samples.
A comprehensive assessment of the proposed model's forecasting ability is carried out by comparing the EMGCSO-Elman model against the BP, Elman, CSO-Elman, LSTM, and XGBoost models. To guarantee a fair comparison across all models, the input layer is configured with 6 neurons. Based on the architecture selection described above, the hidden and output layers contain 8 and 1 neurons, respectively. The activation function employed by the network is the sigmoid function. The maximum training epoch of the network is configured as 1000, and the learning rate and the minimum training error are respectively set to 0.01 and 0.00001. For the EMGCSO algorithm, the swarm size is set to N=40, and the total number of function evaluations is capped at MAX_FEs=1000×D. The remaining algorithm parameters and simulation environment are kept unchanged. Ten independent runs are conducted using a common seed sequence s_r=2024+r, r=1,…,10, corresponding to seeds 2025–2034. For each run r, the same seed s_r, chronological data partition, preprocessing results, and input-output samples are used across all compared models, while model-specific hyperparameters remain fixed throughout the ten runs. The reported statistics are calculated from these 10 runs. In this study, three evaluation metrics are adopted: Mean Squared Error (MSE), Relative Error (RE), and Mean Absolute Percentage Error (MAPE). The closer the values of these metrics are to zero, the better the forecasting performance. Assuming that n represents the quantity of prediction samples, let $Y(t)$ stand for the forecasted load values and $y(t)$ for the ground-truth measurements, respectively, the mathematical expressions of the three evaluation indexes are as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y(i) - y(i))^2 \tag{18}$$

$$RE = abs(\frac{Y(i) - y(i)}{Y(i)}) \cdot 100\% \tag{19}$$

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n} abs(\frac{Y(i) - y(i)}{Y(i)}) \tag{20}$$

4.2. Analysis of off-season load forecasting results

The load data from March 6 to July 16, 2017 are used as training samples. A rolling forecasting strategy is adopted over a 7-day test period (July 17–23, 2017). Specifically, the model uses the actual load values of the previous six days to predict the next day, and this window slides forward day by day over the entire test period.
Figure 5 depicts the load forecasting curves of all the six models during the off-season 7-day testing period. The figures have been arranged in a 2× 4 grid, whereby the first seven subplots represent Days 1−7 respectively while the eighth subplot contains the legend. From Figure 5, it is evident that the newly introduced EMGCSO-Elman model

has the highest degree of fitness compared to the actual load data, especially at the peak and valley points. LSTM and XGBoost models also have good performance while BP has the least accuracy. At load mutation points, the EMGCSO-Elman model maintains high fitting accuracy, indicating strong generalization capability.

Figure 6 presents the average relative error (RE) of all six models over the 7-day test period. The x-axis represents the hour of the day (0–23), and the y-axis represents the RE (%). For each hour, the RE is averaged across the 7 test days. As can be observed from Figure 6, the proposed EMGCSO-Elman model consistently maintains RE values at a relatively low level compared with other models across most hours, indicating its robust prediction performance.

Table 4 reports the mean, standard deviation, best, and worst values of MSE and MAPE over 10 independent runs for all six models. The proposed EMGCSO-Elman achieves the lowest mean MSE (82.2296) and the lowest mean MAPE (5.2755%), indicating superior average accuracy. Moreover, EMGCSO-Elman achieves the lowest best MSE (66.5676) and the lowest best MAPE (4.8621%), demonstrating its potential to achieve excellent performance under favorable conditions. Its worst MSE (95.8328) and worst MAPE (5.6154%) are slightly higher than those of XGBoost (94.1829 and 5.5207%, respectively) but are substantially lower than those of BP, Elman, CSO-Elman, and LSTM, indicating that the proposed model maintains competitive performance even in the worst-case scenarios.

Figures 7 and 8 represent the box plots of the MSE and MAPE distributions, respectively, through the 10 different runs. It can be seen from Figure 7 that the MSE distribution for EMGCSO-Elman is tightly clustered around lower values, indicating the consistency of its prediction accuracy. In the case of BP, Elman, and CSO-Elman models, however, there is more spread out data along with higher medians and outliers. As shown in Figure 8, for MAPE values, EMGCSO-Elman model shows more concentration around low MAPE values. This clearly confirm the strength and stability of the EMGCSO-Elman model.

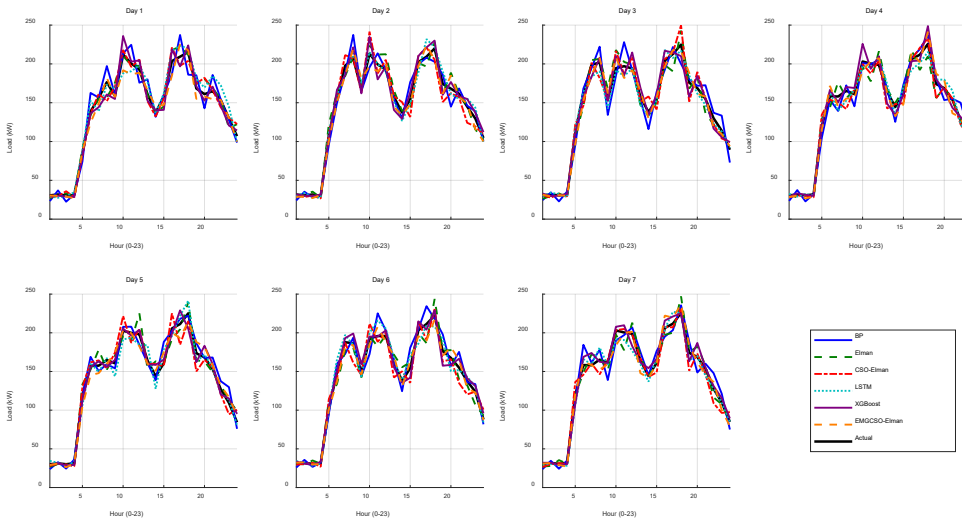

Figure 5. Off-season load prediction curves

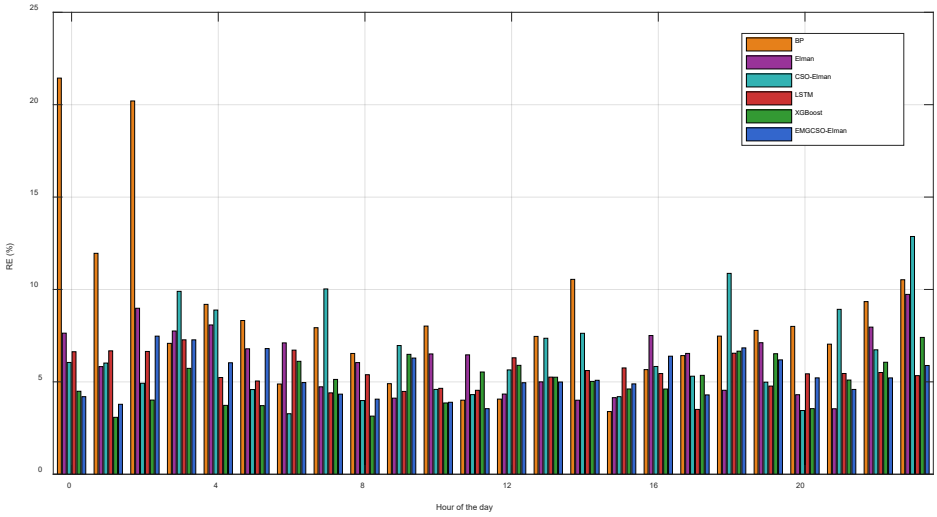

Figure 6. Average relative error over 7-day test period (off-season)

Table 4. Comparison of forecasting results in off-season

Model	MSE				MAPE			
	mean	std	best	worst	mean	std	best	worst
BP	164.9684	18.2735	133.0659	194.5511	8.1038%	0.5041%	7.2700%	8.8277%
Elman	115.6649	12.1290	96.9676	132.4403	6.2883%	0.3487%	5.8058%	6.7918%
CSO-Elman	136.5097	13.8248	108.8703	152.2368	6.5411%	0.3767%	5.7619%	6.9300%
LSTM	93.5325	8.9819	82.1076	108.1785	5.4180%	0.3035%	5.0761%	5.9222%
XGBoost	87.9948	**4.7331**	81.8091	**94.1829**	5.3258%	**0.1652%**	5.0454%	**5.5207%**
EMGCSO-Elman	**82.2296**	8.3421	**66.5676**	95.8328	**5.2755%**	0.2655%	**4.8621%**	5.6154%

Table 5 presents the Wilcoxon signed-rank test results for MSE differences. The proposed EMGCSO-Elman significantly outperforms BP, Elman, CSO-Elman, and LSTM ($p < 0.05$ in all cases), confirming its superior accuracy over these models. No significant difference is observed between EMGCSO-Elman and XGBoost ($p = 0.084 > 0.05$), indicating that the proposed method is competitive with this strong baseline.

Table 5. Statistical significance of MSE differences in off-season

Comparison	p-value	Significant
EMGCSO-Elman vs BP	0.0020	Yes
EMGCSO-Elman vs Elman	0.0020	Yes
EMGCSO-Elman vs CSO-Elman	0.0020	Yes
EMGCSO-Elman vs LSTM	0.0195	Yes
EMGCSO-Elman vs XGBoost	0.0840	No

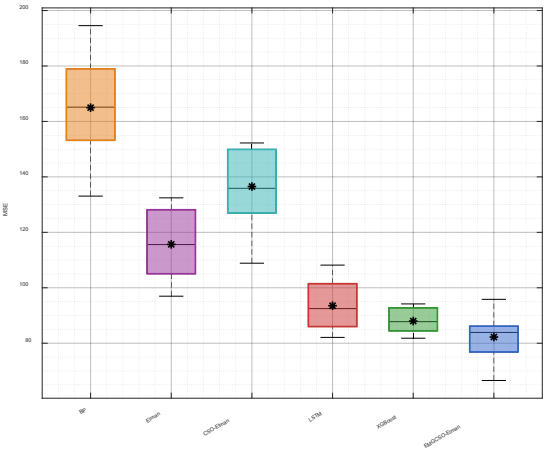

Figure 7. MSE distribution of six models over 10 independent runs (off-season)

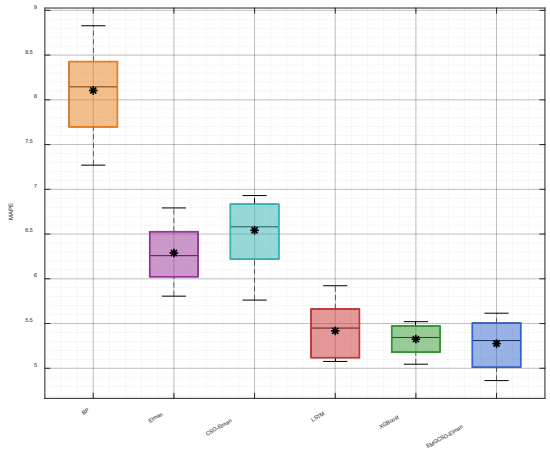

Figure 8. MAPE distribution of six models over 10 independent runs (off-season)

4.3. Analysis of peak -season load forecasting results

The load data from January 2 to February 26, 2017 and from October 2 to December 17, 2017 are used as training samples. The same rolling forecasting strategy described in Section 5.2 is adopted over a 7-day test period (December 18–24, 2017).

Figure 9 displays the load prediction graphs of the six models during the 7 days' peak season testing. These are displayed in a 2 × 4 grid form where the first seven plots represent the prediction of each day from Days 1 to 7, and the last plot is for the legends. From Figure 9, it can be seen that the EMGCSO-Elman model performs better than other models with respect to the actual load values especially at peak and valley times. The performances of LSTM and XGBoost are good, while CSO-Elman does not work well at some points. The Elman model shows large deviations from the actual load, and the BP model has the lowest accuracy.

Figure 10 depicts the mean relative error (RE) during the 7-day peak season test period. In Figure 10, the horizontal axis denotes the time of day (0-23), while the vertical axis denotes the RE (%) for every hour. For every hour, the mean RE is calculated based on 7 test days. It can be seen from Figure 10 that the proposed EMGCSO-Elman model sustains low RE values when compared to other models for most of the hours.

Table 6 shows the mean, standard deviation, best, and worst values of MSE and MAPE for 10 different runs of all the models. It is observed that the proposed EMGCSO-Elman provides the minimum mean MSE value (317.5829) and the minimum mean MAPE value (5.1815%) as well as minimum best MAPE value (4.7025%) and minimum worst MAPE value (5.3876%). Its best MSE (283.9781) and worst MSE (366.7191) are quite similar to those of the LSTM model (283.7970 and 363.9120, respectively), and much smaller compared to BP, Elman, and CSO-Elman.

Figure 11 and Figure 12 show box plots of the distribution of MSE and MAPE, respectively, on the basis of the 10 independent experiments performed. Figure 11 illustrates that the distribution of MSE of EMGCSO-Elman model has low values and a narrow interquartile range, which indicates the stable prediction accuracy of EMGCSO-Elman. In addition, BP, Elman, and CSO-Elman have larger distribution ranges and high medians, with the presence of many outliers. Figure 12 illustrates the similar pattern of MAPE, in which EMGCSO-Elman has narrow distribution at low error rates. These results further confirm the robustness and stability of the proposed EMGCSO-Elman model under peak-season conditions.

Table 7 presents the Wilcoxon signed-rank tests for MSE differences. EMGCSO-Elman achieves significantly lower MSE than BP, Elman, CSO-Elman, and XGBoost ($p<0.05$), while no significant difference is observed between EMGCSO-Elman and LSTM ($p=0.2754>0.05$). Notably, LSTM obtains a lower mean MSE than Elman (333.9212 vs. 369.5994) but a higher mean MAPE (5.8660% vs. 5.7328%), indicating different error sensitivity of the two metrics under peak-season fluctuations. MSE emphasizes large absolute deviations, whereas MAPE reflects proportional errors.

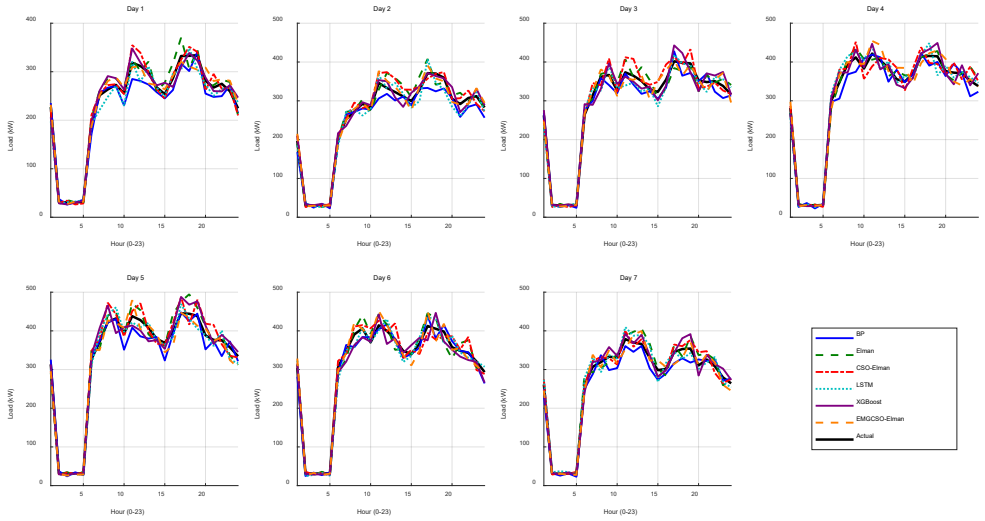

Figure 9. Peak-season load prediction curves

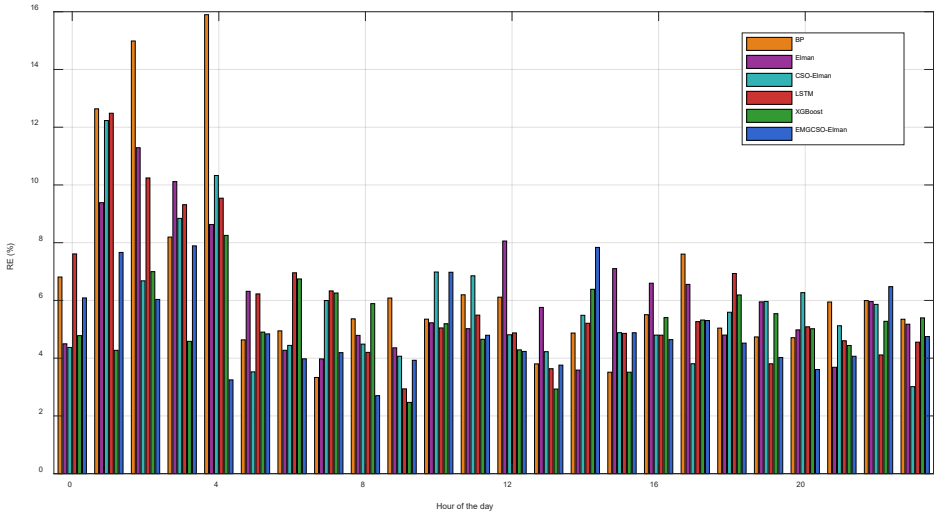

Figure 10. Average relative error over 7-day test period (peak-season)

Table 6. Comparison of forecasting results in peak-season

Model	MSE				MAPE			
	mean	std	best	worst	mean	std	best	worst
BP	380.7811	29.3156	321.3544	418.0084	6.4455%	0.1931%	6.0887%	6.6043%
Elman	369.5994	32.2180	320.6002	419.2287	5.7328%	0.3224%	5.1570%	6.1559%
CSO-Elman	374.7454	23.2565	343.9555	411.3850	5.7976%	0.1984%	5.4517%	6.0811%
LSTM	333.9212	25.8591	**283.7970**	**363.9120**	5.8660%	0.2697%	5.4531%	6.3568%
XGBoost	346.5921	**18.3063**	319.8182	380.5751	5.1972%	**0.1718%**	4.9487%	5.5069%
EMGCSO-Elman	**317.5829**	27.9296	283.9781	366.7191	**5.1815%**	0.2200%	**4.7025%**	**5.3876%**

Table 7. Statistical significance of MSE differences in peak-season

Comparison	p-value	Significant
EMGCSO-Elman vs BP	0.0020	Yes
EMGCSO-Elman vs Elman	0.0039	Yes
EMGCSO-Elman vs CSO-Elman	0.0020	Yes
EMGCSO-Elman vs LSTM	0.2754	No

EMGCSO-Elman vs XGBoost	0.0039	Yes

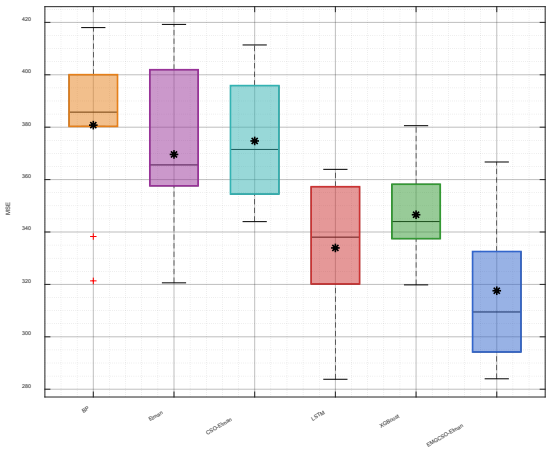

Figure 11. MSE distribution of six models over 10 independent runs (peak-season)

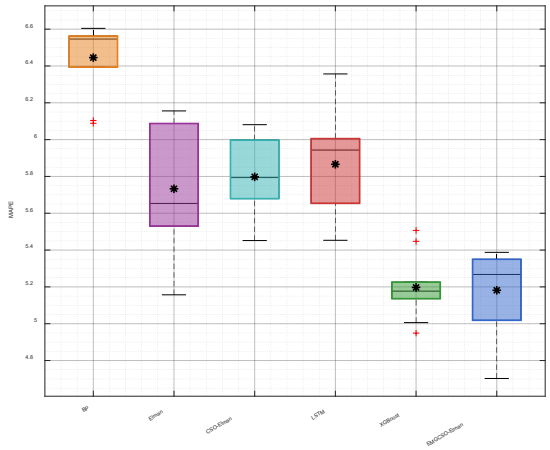

Figure 12. MAPE distribution of six models over 10 independent runs (peak-season)

4.4 Computational Complexity Analysis

This subsection analyzes the computational time of the proposed EMGCSO-Elman model in comparison with standard Elman and CSO-Elman. All models are implemented under the same experimental conditions, and the computational time (including optimization time and training time) is recorded over 10 independent runs. The results are presented in Table 8.

Table 8. Comparison of computational time

Model	Optimization time	Training time	Total time
Elman	—	3.82 ± 0.07	3.82 ± 0.07
CSO-Elman	16.82 ± 1.01	3.77 ± 0.23	20.59 ± 1.20
EMGCSO-Elman	21.48 ± 1.27	4.99 ± 0.31	26.47 ± 1.52

As shown in Table 8, the training time of Elman (3.82 s) is comparable to that of CSO-Elman (3.77 s) and EMGCSO-Elman (4.99 s), as all models use the same backpropagation training algorithm. The additional time in CSO-Elman and EMGCSO-Elman comes primarily from the optimization phase (16.82 s and 21.48 s, respectively). Compared with CSO-Elman, EMGCSO-Elman requires approximately 29% more total time (26.47 s vs. 20.59 s). The increase in optimization time (27.7%) is expected due to the additional multi-mode guidance strategies. The training time of EMGCSO-Elman is also slightly longer (4.99 s vs. 3.77 s), which may be because the optimal parameters found by EMGCSO lead to a more thorough training process. Overall, this additional computational cost is justified by the substantial improvement in prediction accuracy: compared with CSO-Elman, MSE is reduced by 39.8% in off-season and 15.3% in peak-season. Wilcoxon signed-rank tests (Tables 5 and 7) confirm that these improvements are statistically significant ($p < 0.05$), demonstrating a favorable accuracy-to-time trade-off.

5. Conclusion

To overcome the deficiencies of traditional short-term power load forecasting models in terms of accuracy and stability, this paper proposes a prediction model based on an Elman neural network optimized by the elite multi-mode guided chicken swarm optimization (EMGCSO) algorithm. EMGCSO introduces an elite multi-mode guidance strategy to design differentiated search behaviors for roosters based on their fitness. Benchmark function tests demonstrate that EMGCSO delivers fast convergence and strong global search capability. By optimizing the initial weights and biases of the Elman network, the EMGCSO-Elman forecasting framework is constructed.

Experimental results on the tested restaurant load dataset confirm that the proposed model outperforms BP, Elman, CSO-Elman, LSTM, and XGBoost in terms of prediction accuracy and robustness, with statistically significant improvements ($p<0.05$). These results demonstrate the effectiveness of the proposed model for short-term power load forecasting on the evaluated dataset.

It is important to note that all these results have been derived from only one load dataset from the restaurant. The generalization of the developed approach for other power system dispatching situations needs to be tested through more diverse and larger-scale datasets. Future research will be concerned with making the algorithm more computationally efficient, incorporating various features from multiple sources (such as weather and economics), and evaluating the algorithm under different loading conditions to further enhance its practicality and generalization capability.

References

[1] HABBAK H, MAHMOUD M, METWALLY K, et al. Load forecasting techniques and their applications in smart grids[J]. Energies, 2023, 16(3): 1480.

[2] JALALIFAR R, DELAVAR M R, GHADERI S F. SAC-ConvLSTM: A novel spatio-temporal deep learning-based approach for a short term power load forecasting[J]. Expert Systems with Applications, 2024, 237: 121487.

[3] PINHEIRO M G, MADEIRA S C, FRANCISCO A P. Short-term electricity load forecasting—A systematic approach from system level to secondary substations[J]. Applied Energy, 2023, 332: 120493.

[4] WANG J, WU K, WEN Y, et al. Short-term electric load forecasting based on enhanced GA-BP multi-parameter algorithm[J]. Measurement, 2025: 117855.

[5] WU H, ZHU X. Short-term electric load forecasting model based on PSO-BP[C]//2023 4th International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE). IEEE, 2023: 224-228.

[6] SU H, LI Z, LI J, et al. Short term load forecasting based on BP network model[C]//2025 6th International Conference on Energy Power and Automation Engineering (ICEPAE). IEEE, 2025: 81-84.

[7] LIN W, ZHANG B, LI H, et al. Short-term load forecasting based on EEMD-Adaboost-BP[J]. Systems Science & Control Engineering, 2022, 10(1): 846-853.

[8] AIHMOUD L, NAWAFLEH Q. Short-term load forecasting for Jordan power system based on NARX-ELMAN neural network and ARMA model[J]. IEEE Canadian Journal of Electrical and Computer Engineering, 2021, 44(3): 356-363.

[9] ISMAEL L, ISMAEL A I. Prediction the data consumption for power demands by Elman neural network[J]. International Journal of Electrical and Computer Engineering (IJECE), 2019, 9(5): 4003-4009.

[10] MAHMOUD I A, MUHAMMAD U J, KAWU S J, et al. Enhancing Energy Demand Prediction Using Elman Neural Network and Support Vector Machine Model: A Case Study in Lagos State, Nigeria[J]. Artificial Intelligence Evolution, 2024: 39-51.

[11] LIU J C, YIN Y. Power load forecasting considering climate factors based on IPSO-Elman method in China. Energies, 2022, 15(3), 1236.

[12] HAO J, ZHU C S, GUO X T. A new CIGWO-Elman hybrid model for power load forecasting. Journal of Electrical Engineering & Technology, 2022, 17: 1319-1333.

[13] LIU Z L, NING D Y, HOU J Y. A novel Elman neural network based on Gaussian kernel and improved SOA and its applications. Expert Systems with Applications, 2024, 249: 123453.

[14] MA X Y, ZHANG X H. A short-term prediction model to forecast power of photovoltaic based on MFA-Elman. Energy Reports, 2022, 8: 495-507.

[15] Narayanan V L, Narayan J, Dhaked D K, et al. Elman Neural Network with Customized Particle Swarm Optimization for Hydraulic Pitch Control Strategy of Offshore Wind Turbine[J]. Processes, 2025, 13(3): 808.

[16] MENG X, LIU Y, GAO X, et al. A new bio-inspired algorithm: chicken swarm optimization[C]//, Fifth International Conference on Swarm Intelligence, Hefei, China, 17 October 2014: 86-94.

[17] Oguntoye J P, Ajagbe S A, Adedeji O T, et al. An improved chicken swarm optimization techniques based on cultural algorithm operators for biometric access control[J]. Computers, Materials & Continua, 2025, 84(3): 5713-5732.

[18] Abbas M, Che Y, Maqsood S, et al. Self-adaptive evolutionary neural networks for high-precision short-term electric load forecasting[J]. Scientific Reports, 2025, 15(1): 21674.

[19] Nagarajan B, Svn S K, Selvi M, et al. A fuzzy based chicken swarm optimization algorithm for efficient fault node detection in Wireless Sensor Networks[J]. Scientific Reports, 2024, 14(1): 27532.

[20] DEB S, GAO X, TAMMI K, et al. Recent studies on chicken swarm optimization algorithm: a review (2014–2018)[J]. Artificial Intelligence Review, 2020, 53(4):1737-1765.

[21] REZAEI F, SAFAVI H R. GU A. SPSO: a new approach to hold a better exploration-exploitation balance in PSO algorithm[J]. Soft Computing, 2020, 24(7): 4855-4875.

[22] ASKARZADEH A, ALIPOUR M A. E2: A basic optimization method using exploration-exploitation concept[J]. Soft Computing, 2025, 29(19): 5519-5539.

[23] KIRAN M S, HAKLI H, GUNDUZ M, et al. Artificial bee colony algorithm with variable search strategy for continuous optimization[J]. Information Sciences, 2015, 300: 140-157.

[24] WANG H, WANG W J, XIAO S Y, et al. Improving artificial bee colony algorithm using a new neighborhood selection mechanism[J]. Information Sciences, 2020, 527: 227-240.

[25] SEKYERE Y O M, EFFAH F B, OKYERE P Y. Hyperbolic tangent-based adaptive inertia weight particle swarm optimization[J]. Jurnal Nasional Teknik Elektro, 2023, 12(2): 89-96.

[26] HAKLI H, KIRAN M S. An improved artificial bee colony algorithm for balancing local and global search behaviors in continuous optimization[J]. International Journal of Machine Learning and Cybernetics, 2020 11(9): 2051-2076.

[27] ZHOU X, LU J, HUANG J, et al. Enhancing artificial bee colony algorithm with multi-elite guidance[J]. Information Sciences, 2021, 543: 242-258.

[28] LYNN N, SUGANTHAN P N. Ensemble particle swarm optimizer[J]. Applied Soft Computing, 2017, 55: 533-548.

[29] XIA X, GUI L, HE G, et al. An expanded particle swarm optimization based on multi-exemplar and forgetting ability[J]. Information Sciences, 2020, 508: 105-120.

[30] LIU Z, NISHI T. Strategy dynamics particle swarm optimizer[J]. Information Sciences, 2022, 582: 665-703.

[31] ALI B, LASHARI S A, SHARIF W, et al. An efficient learning weight of elman neural network with chicken swarm optimization algorithm[J]. Procedia Computer Science, 2021, 192: 3060-3069.

[32] MAHDI Q A M N, ALI M, ATTA M N, et al. Training learning weights of elman neural network using salp swarm optimization algorithm[J]. Procedia Computer Science, 2023, 225: 1974-1986.