Expert Systems With Applications 129 (2019) 135–155

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks

Yueting Xu[a], Huiling Chen[a,1,*], Ali Asghar Heidari[b,c], Jie Luo[a], Qian Zhang[a], Xuehua Zhao[d,1,*], Chengye Li[e,1,*]

[a] Department of Computer Science, Wenzhou University, Wenzhou 325035, China
[b] School of Surveying and Geospatial Engineering, University of Tehran, Tehran, Iran
[c] Department of Computer Science, School of Computing, National University of Singapore, Singapore, Singapore
[d] School of Digital Media, Shenzhen Institute of Information Technology, Shenzhen 518172, China
[e] Department of Pulmonary and Critical Care Medicine, The First Affiliated Hospital of Wenzhou Medical University, Wenzhou 325000, China

ARTICLE INFO

Article history:
Received 12 December 2018
Revised 6 March 2019
Accepted 25 March 2019
Available online 3 April 2019

Keywords:
Moth-flame optimization algorithm
Parameter optimization
Chaotic local search
Gaussian mutation
Kernel extreme learning machine

ABSTRACT

Moth-flame optimization algorithm (MFO) is a new nature-inspired meta-heuristic based on the navigation routine of moths in the environment known as transverse orientation. For some complex optimization tasks, especially high dimensional and multimodal problems, the conventional MFO may face problems in the convergence trends or be trapped into the local and deceptive optima. Therefore, in this study, two strategies have been introduced into the conventional MFO to get a more stable sense of balance between the exploration and exploitation propensities. First, Gaussian mutation is employed to increase the population diversity of MFO. Then, a chaotic local search is applied to the flame updating process of MFO for better exploiting the locality of the solutions. The proposed CLSGMFO approach was compared against a wide range of well-known classical metaheuristic algorithms (MAs) and various advanced MAs using 23 classical benchmark functions. It was shown that the designed CLSGMFO can outperform most of the popular MAs in terms of solution quality and convergence speed. Moreover, based on CLSGMFO, a hybrid kernel extreme learning machine model, which is called CLSGMFO-KELM, is established to deal with financial stress prediction scenarios. To investigate the effectiveness of the CLSGMFO-KELM model, the proposed hybrid system was tested on two widely used financial datasets and compared against a broad array of popular classifiers. The results demonstrate that the proposed learning scheme can offer a superior kernel extreme learning machine model with excellent predictive performance. Accordingly, the proposed CLSGMFO can serve as an effective and efficient computer-aided tool for financial prediction.

1. Introduction

In recent years, numerous metaheuristic algorithms (MAs) have been developed and applied to realize real-world problems (Faris et al., 2019). MAs are characterized by their simplicity, effectiveness and inexpensive computational costs. There are many new swarm-intelligence-based algorithms in the field such as Harris hawks optimizer (HHO) (Heidari, Mirjalili, et al., 2019). Well-known MAs include but are not limited to: differential evolution (DE) (Storn & Price, 1997), particle swarm optimization (PSO) (Deng, Yao, Zhao, Yang, & Li, 2019; Kennedy & Eberhart, 1995), ant colony optimization (ACO) (Deng, Xu, & Zhao, 2019; Dorigo, Birattari, & Stutzle, 2007), dragonfly algorithm (DA) (Mafarja et al., 2018; Mafarja, Heidari, Faris, Mirjalili, & Aljarah, 2020), bat algorithm (BA) (Yang, 2010), flower pollination algorithm (FPA) (Ma, Ouyang, Chen, & Zhao, 2014), sine cosine algorithm (SCA) (Mirjalili, 2016), salp swarm algorithm (SSA) (Abbassi, Abbassi, Heidari, & Mirjalili, 2019; Zhang, et al., 2019), whale optimization algorithm (WOA) (Chen, Xu, Wang, & Zhao, 2019; Heidari, Aljarah, et al., 2019), multi-verse optimization algorithm (MVO) (Aljarah, Mafarja, Heidari, Faris, & Mirjalili, 2020), bacterial foraging optimization (BFO) (Passino, 2002; Zhang et al., 2018), artificial bee colony optimization (ABC) (Karaboga & Basturk, 2007), grasshopper optimization algorithm (GOA) (Luo et al., 2018; Saremi, Mirjalili, & Lewis, 2017), grey wolf optimization (GWO) (Heidari & Pahlavani, 2017; Mirjalili, Aljarah, Mafarja, Heidari, & Faris, 2020; Zhao et al., 2019), firefly algorithm (FA) (Yang, 2009), fruit fly optimization

* Corresponding authors.
E-mail addresses: yuetingXu96@163.com (Y. Xu), chenhuiling.jlu@gmail.com (H. Chen), as_heidari@ut.ac.ir, aliasgha@comp.nus.edu.sg, t0917038@u.nus.edu (A.A. Heidari), jieluocn@foxmail.com (J. Luo), qz_zhangqian@163.com (Q. Zhang), zhaoxh@sziit.edu.cn (X. Zhao), lichengye41@126.com (C. Li).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.eswa.2019.03.043
0957-4174/

algorithm (FOA) (Pan, 2012; Shen et al., 2016), ant lion optimization (ALO) (Heidari, Faris, Mirjalili, Aljarah, & Mafarja, 2020) and moth-flame optimization algorithm (MFO) (Mirjalili, 2015). Among these algorithms, MFO has been extensively studied due to its excellent performance in tackling a variety of optimization problems. The MFO algorithm (Mirjalili, 2015), inspired by the navigation routine of moths in the environment known as transverse orientation, has been shown to perform well on various global unconstrained and constrained optimization problems.

Owing to its excellent searching ability and ease of implementation, MFO has been applied to various fields. Allam, Yousri, and Eteiba (2016) applied MFO to extract the parameter of three diode model. Li, Li, and Liu (2016) used the MFO algorithm to tune the parameters of the least squares support vector machine (SVM) model. Zhao, Zhao, and Guo (2016) developed a hybrid electricity consumption forecasting method in which MFO was used to tune the parameters in the grey model with a rolling mechanism. Aziz, Ewees, and Hassanien (2017) utilized the WOA and MFO to find the optimal threshold values. Sayed and Hassanien (2017) presented an automatic mitosis detection method by using neutrosophic sets and the MFO algorithm. Li et al. (2018) constructed a diagnostic model to predict the diagnosis of tuberculous pleural effusion in which MFO was employed to tune the key parameters of SVM. Mei, Sulaiman, Mustaffa, and Daniyal (2017) applied MFO to solve the optimal reactive power dispatch problem.

In order to improve the performance of the basic MFO in solving specific problems, researchers have developed myriad enhanced versions of MFO. Li, Zhou, Zhang, and Song (2016) presented an improved MFO via Lévy-flight strategy (LMFO). Trivedi, Kumar, Ranpariya, and Jangir (2016) used the MFO algorithm combined with the levy flight to achieve competitive results in case of both discrete and continuous control parameters. Zhang, Mistry, Neoh, and Lim (2016) proposed an evolutionary FA by combining the MFO's spiral search behavior and the attractiveness search actions of Lévy-flight FA for feature optimization. Hassanien, Gaber, Mokhtar, and Hefny (2017) applied an improved MFO to automatically detect tomato diseases. In this method, the fitness function depends on the rough sets' dependency degree and it also takes into consideration the number of selected features. Wang, Chen, Yang, et al. (2017) proposed a novel model for the kernel extreme learning machine (KELM) based on the chaotic MFO. Kumar, Reddy, and Rajasekhara Babu (2017) adopted a hybrid model with both MFO and ALO to improve the performance of cluster head selection among Internet of Things devices in Wireless Sensor Network. Khalilpourazari and Khalilpourazary (2019) hybridized MFO and the basic water cycle algorithm (WCA) to deal with the constrained problems. Apinantanakon and Sunat (2018) introduced opposition-based learning scheme into the MFO algorithm to accelerate its convergence speed. Elsakaan, El-Sehiemy, Kaddah, and Elsaid (2018) developed an enhanced MFO (EMFO) algorithm for solving the non-convex economic dispatch problem. In EMFO, the levy flight was introduced to increase the diversity of the population. Li, Wang, and Li (2018) employed a novel method called multi-objective MFO to improve the efficiency of water resource utilization using some efficient mechanisms such as opposition-based learning and indicator-based selection. Xu et al. (2018) presented an enhanced MFO technique based on cultural learning and Gaussian mutation.

However, in some practical optimization problems, MFO might not be capable of escaping from local optima (Wang, Deb, Gandomi, & Alavi, 2016), which is a limitation for its application. To mitigate this problem, an improved variant of MFO called chaotic local search and Gaussian mutation-enhanced MFO (CLSGMFO), is proposed in this study. In CLSGMFO, two efficient mechanisms (Gaussian mutation (GM) and chaotic local search (CLS)) are integrated into the original MFO. With the GM mechanism, it is more likely to generate a new offspring near the original parent because of its narrow and long tail. Using the CLS mechanism, the best current agent is given an extra chance to carry out the second search in the neighboring space. That is, the proposed CLSGMFO increases the population diversity of MFO by embedding GM-based patterns into the spiral movement of MFO, and extends the local searching space by employing the CLS to the flame updating process of MFO. The performance of CLSGMFO was evaluated and compared with several classical MAs and various advanced optimization approaches on 23 classical benchmark functions. The experimental results show that the enhanced CLSGMFO significantly outperforms the basic MFO and other MAs tested in this paper.

Based on the proposed CLSGMFO, a new hybrid KELM-based model is also established for the financial prediction problems, which is called CLSGMFO-KELM. KELM model (Huang, Zhou, Ding, & Zhang, 2012), as an expanded version of the original ELM (Huang, Zhu, & Siew, 2006) was proposed by Huang et al. The performance of KELM is known to be strongly affected by the kernel bandwidth γ and the penalty parameter C. Recently, MAs have been widely used to find the optimal parameters of KELM (Luo et al., 2018; Liu, Loo, Masuyama, & Pasupa, 2018; Wang et al., 2017a,b; Yang, Ce, & Lian, 2017). In this paper, the CLSGMFO algorithm was used to tune the two parameters for KELM. In order to illustrate the effectiveness of the CLSGMFO-KELM model, we compared the proposed method with seven commonly used classifiers in terms of Accuracy (ACC), Sensitivity, Specificity, and Matthews Correlation Coefficients (MCC). The experimental results demonstrate that the proposed method, CLSGMFO-KELM, outperforms other classifiers in terms of the four performance metrics.

The remainder of this paper is organized as follows: a brief description of MFO, GM, CLS, and KELM is given in Section 2. Section 3 outlines the enhanced MFO approach and the hybrid classification model. The experimental designs and simulation results are elaborated in Section 4. Finally, conclusions and future works are summarized in Section 5.

2. Background

2.1. Moth-flame optimization algorithm (MFO)

Moth-flame optimization algorithm (MFO) (Mirjalili, 2015) is a new nature-inspired optimization method that simulates the navigation method of the moth in the night around light sources. Maintaining a fixed angle with respect to the moon, they travel long distances in a straight line. This method is called the transverse orientation. As mentioned in the article (Khalilpourazari & Saman Khalilpourazary, 2019), the effectiveness of the mechanism strongly depends on the distance between moths and light sources. The mathematical model of MFO is mainly composed of moths and flames. In the MFO algorithm, flames are the candidate solutions and the positions of the moths in a search space are the problem variables; the moth population is the actual search agent which flies around the space, and the flames are the optimal places of moths that are reached so far. In order to find a better result, each moth searches around the corresponding flame and updates its position. In this situation, a moth never loses its best solution. S function, the main function of MFO, makes the moths move around the search space. The position of each moth is updated by a flame and this behavior is mathematically modeled with the following formula,

$$M_i = S\left(M_i, F_j\right) \tag{1}$$

where M_i is ith moth and F_j is jth flame. In order to model the behavior of the moths, the matrix is given in Eq. (2) to represent

the set of moths.

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,d} \\ m_{2,1} & \ddots & \ddots & m_{2,d} \\ \vdots & \ddots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,d} \end{bmatrix} \tag{2}$$

Here, M indicates that n moths moving in the D-dimension search space. The fitness value of each moth can be evaluated by the following function and stored in a matrix A as shown in Eq. (3).

$$A = Z(M) = [A_1 A_2 \cdots A_n]^T \tag{3}$$

The best positions obtained by moths in each dimension are flames stored in the matrix F as shown in Eq. (4) and the corresponding fitness is given in Eq. (5).

$$F = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,d} \\ f_{2,1} & \ddots & \ddots & f_{2,d} \\ \vdots & \ddots & \ddots & \vdots \\ f_{n,1} & f_{n,2} & \cdots & f_{n,d} \end{bmatrix} \tag{4}$$

$$E = Z(F) = [E_1 E_2 \cdots E_n]^T \tag{5}$$

where Z is the fitness function of the specific optimization problem. As mentioned previously, the most properly utilized approach for the transverse orientation of the moths is the logarithmic spiral function which is given in Eq. (6)

$$S(M_i, F_j) = \left|F_j - M_i\right| \cdot e^{bt} \cdot \cos(2\pi t) + F_j \tag{6}$$

Here, b is a constant, which controls the shape of the logarithmic spiral function. In this spiral equation, t indicates how close the next position of the ith moth should be to the jth flame. $|F_j - M_i|$ represents the distance of the ith moth to the jth flame. This mechanism can be realized by introducing the random number $t \in [-1, 1]$ to its spiral movement (if $t = -1$, then the moth moves toward the flame; and if $t = 1$, then the moth goes away from the flame). In addition, numbers of flames decrease gradually to assure the balance between exploration and exploitation during the searching process. Therefore, over the course of iterations, moths update their positions only based upon the best promising solution. The adaptive mechanism for numbers of flames has been proposed, as shown in Eq. (7):

$$B = \text{ceil}\left(n - g * \frac{n-1}{G}\right) \tag{7}$$

Where g presents the current iteration number of the algorithm and G shows the maximum number of the generations. The implementation of MFO can be expressed as follow:

2.2. Gaussian mutation

As proposed in Bäck and Schwefel (1993), the Gaussian mutation (GM) operation can be generally applied to improve the efficacy of MAs (Lee & Yao, 2001; Luo et al., 2018; Salgotra & Singh, 2017; Yao, Liu, & Lin, 1999). Because the GM operation is likely to generate a new and random offspring near the original parent due to its narrow and long tail, the mutation operator will take smaller steps, allowing for every corner of the search space to be explored in a much better way. To cope with the loss of diversity in global search, Gaussian mutation is incorporated into basic MAs. The Gaussian density function is given by Eq. (8):

$$f_{(0,\sigma^2)}(\vartheta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\vartheta^2}{2\sigma^2}}, \quad -\infty < x < \infty \tag{8}$$

where σ^2 is the variance for each search agent. This formula is further reduced to generate a single d-dimension random variable by setting mean to 0 and standard deviation to 1. The generated random variable is applied to MAs and given as follow:

$$X' = X_i + Gaussian(\vartheta) X_i \tag{9}$$

It should be noted that $Gaussian(\vartheta)$ is a D-dimension Gaussian step vector generated by Eq. (8) with ϑ as a Gaussian random distribution number between [0, 1]. In addition, X_i is the search agent of MA, X' is the mutated individual (Algorithm 1).

Algorithm 1 Pseudo-code of MFO.

```
Initialize the population of the moth M_i (i = 1, 2, ..., n);
  While (g ≤ G)
    Update B according to Eq. (7);
    Bring M_i back if it goes outside of the boundaries;
    If g==1
        F=sort(M_g);
        E=sort(A_g);
    else
        F = sort(M_{g-1}, M_g);
        E = sort(A_{g-1}, A_g);
    End
    Set the best search agent as Fbest;
    For i = 1: n
        For j = 1: d
            Update r, t;
            Update M_(i,j) using Eq. (6) with respect to the corresponding moth;
        End For
    End For
    g = g + 1;
  End While
  Return Fbest;
  End
```

2.3. Chaotic local search (CLS)

Chaos is a random-like phenomenon existing in non-linear and deterministic systems. Moreover, Chaos is extremely sensitive to its initial conditions and parameters. Mathematically, chaotic systems can be considered as sources of randomness (Alatas, 2010). Chaos movement is non-repetitive, and chaos can perform the search at higher speeds than a random ergodic search which depends on probability (Coelho & Mariani, 2008). Therefore, the searching technique based on the chaos theory will undoubtedly has more advantages than random search. Chaotic sequences can be easily generated and stored in that even a very long sequence requires merely a few chaotic maps and parameters. Hence, just by changing the initial conditions, an enormous number of different sequences can be generated. Furthermore, these sequences are deterministic and reproducible (Alatas, 2010). A chaotic sequence based on the logistic map is described as follow:

$$C_{i+1} = \mu \times C_i \times (1 - C_i) \; i = 1, \cdots, n-1 \tag{10}$$

Set the control parameter $\mu = 4$, set $0 < C_1 < 1$ and $C_1 \neq 0.25$, 0.5, 0.75, 1. When $\mu = 4$, we have chaos. n is the number of moth population.

The time spent in chaos optimization is acceptable on a small scale. But the time cost will be difficult to bear after the search space becomes large. In order to solve the above problem, a chaotic local search (CLS) (Jia, Zheng, & Khurram Khan, 2011) can be integrated into intelligent optimization algorithms. The integration of CLS mechanism into the MFO algorithm can not only enhance its searching ability but also make it enable to avoid falling into a local optimum. The candidate solution CS of the target position T generated by CLS is formulated as follows:

$$CS = (1 - s) \times T + s \times C_i' \; i = 1, \cdots, n \tag{11}$$

The contraction factor s is determined as follows:

$$s = (G - g + 1)/G \tag{12}$$

$$C_i^{'} = lb + C_i \times (ub - lb) \tag{13}$$

The chaotic variable C_i in Eq. (13) was generated by Eq. (10), which maps the chaotic vector C_i' into the domain of definition [*lb*, *ub*], *lb* and *ub* represent the lower and upper bounds of the initial solution, respectively. The chaotic vector C_i' is linearly combined with the target position *T* to generate the candidate solution *CS*.

2.4. Kernel extreme learning machine (KELM)

KELM was developed on the basis of extreme learning machine (ELM). First introduced by Huang, Zhu, and Siew (2004), ELM, which is equivalent to a new type of single-hidden layer feed-forward neural network (SLFN) (Han, Huang, Zhu, & Rong, 2006; Huang et al., 2006; Huang, Zhou, Ding, & Zhang, 2012; Kaya & Uyar, 2013; Marques & Graña, 2012), has become an important machine learning technique. Unlike traditional neural network learning algorithms, the output weight of network in ELM only calculates one time, which greatly improves the generalization performance and accelerates training speeds.

For given N samples $\{(x_i, t_i)|x_i \in R^q, t_i \in R^z\}_{i=1}^N$ where x_i is a $q \times 1$ input feature vector and t_i is a $z \times 1$ target vector. The output function of ELM with p hidden neurons and an activation function $h(x)$ is given as:

$$y_j = \sum_{i=1}^{p} \beta_i h(\omega_i \cdot x_i + b_i), \quad j = 1, 2, \ldots, N \tag{14}$$

where ω_i is the weight vector connecting ith hidden neuron to output neurons, the bias of the ith neuron in the hidden layer is b_i and $h(x)$ is the activation function, β_i is the weight vector between hidden nodes and ith output node. And the standard SLFNs with p hidden neurons within an activation function $h(x)$ can approximate these samples with zero error, as follow:

$$\sum_{i=1}^{N} \|y_i - t_i\| = 0 \tag{15}$$

There exist β_i, ω_i and b_i such that

$$t_j = \sum_{i=1}^{p} \beta_i h(\omega_i \cdot x_i + b_i), \quad j = 1, 2, \ldots, N \tag{16}$$

The above N formulas can also be written compactly as follow:

$$H\beta = T \tag{17}$$

where $\beta = [\beta_1, \ldots, \beta_p]^T$ is the weight vector between the hidden nodes and output nodes, $H = [h_1(x), \ldots, h_p(x)]$ is the output matrix of the hidden layer of the neural network with respect to the input *x*. *T* is the output matrix. Consequently, based on this condition, the output weight can be calculated by finding the least squares solution under the linear system. Huang et al. (2004, 2006) showed that the input weights and the hidden layer's biases of SLFNs requisite not to be attuned whatsoever and can be arbitrarily specified. Under this assumption, the output weights need to be methodically calculated by finding the least square solution β^* of the linear system $H\beta = T$:

$$\beta^* = H^+ T \tag{18}$$

where H^+ is the Moore Penrose generalized inverse of the output matrix H (Huang et al., 2004). The Moore Penrose generalized inverse can be obtained by $H^+ = H^T(HH^T)^{-1}$. In order to improve the generalization ability of the ELM model, $1/C$ as a positive value is added to the diagonal elements of HH^T(C is the penalty parameter). Therefore, the output function can be formulated as follow:

$$f(x) = H\beta = HH^T\left(I/C + HH^T\right)^{-1} T \tag{19}$$

If the prior knowledge of feature mapping $h(x)$ is known to users, the following equation can be utilized to define the kernel matrix of ELM,

$$\begin{cases} \Omega_{ELM} = HH^T \\ \Omega_{ELMij} = h(x_i) \cdot h\left(x_j\right) = K\left(x_i, x_j\right) \end{cases} \tag{20}$$

where $K(x_i, x_j)$ is a kernel function, which always adopts a radial-basis function (a RBF kernel function) for ELM. Thus, the output function of the KELM model can be represented as follows:

$$f(x) = h(x)H^T\left(I/C + HH^T\right)^{-1} = \begin{bmatrix} K(x, x_1) \\ \cdots \\ K(x, x_N) \end{bmatrix}^T = (I/C + \Omega_{ELM})^{-1} T \tag{21}$$

The stable kernel matrix Ω_{ELM} replaces the random matrix HH^T, thus the output weight of the model is more stable. The common form of a RBF kernel is given as follow:

$$K(x, x_i) = \exp\left(-\gamma \|x - x_i\|^2\right) \tag{22}$$

Based on the above formula, the two main parameters mentioned in KELM are the penalty factor C and the kernel parameter γ. The parameter C is responsible for balancing fitting error minimization and complexity for that model. The parameter γ defines the nonlinear kernel mapping directly from the input space to a high-dimensional feature space.

3. Proposed method

3.1. The CLS and GM enhanced MFO algorithm

In this subsection, the developed CLS and GM enhanced MFO (CLSGMFO) will be described in detail. In CLSGMFO, two main strategies, called CLS and GM, are introduced to keep a more suitable balance between exploration and exploitation during the searching process of moths.

The diversity of the population is known to be crucial for MAs because it can enhance the global-searching capability of search agents. In the proposed method, GM is applied to improve the population diversity of MFO. This strategy assists in searching the solution space more effectively and enhances the exploration ability of MFO. The GM mechanism is used to generate the mutated individual corresponding to the current search agent after the position of each moth in the population has been updated, and it can be generated by the following formula:

$$M_i^{g'} = M_i^g + \text{Gaussian}(\vartheta) M_i^g \tag{23}$$

In order to ensure the quality of the moth population, the superior moths in each generation can continue to be greatly enriched for the next generations (M_i^{g+1}) and the inferior ones are discarded.

The original MFO algorithm updates its search agents based on the positions of flames. However, the population of MFO is still inclined to stagnation in local optima in some cases. Problems of immature convergence of MFO can still be experienced. In order to mitigate the phenomenon of stagnation and to obtain more precise results, CLS is embedded in the MFO algorithm. For the efficient incorporation of CLS into the MFO algorithm, the search agents that will undergo the CLS process should be properly selected. Hence, only the fittest flame of the whole population is selected for using CLS. Then, the CLS mechanism chooses the position of the fittest flame as the base and then forms the new position of the best flame. The candidate solution of the best flame's position can be generated by the following formula:

$$CS = (1 - s) * Fbest + s * C_i^{'} \tag{24}$$

Where *Fbest* is the best flame of the whole population of MFO. Note, the quality of the candidate solution is enhanced, and thus

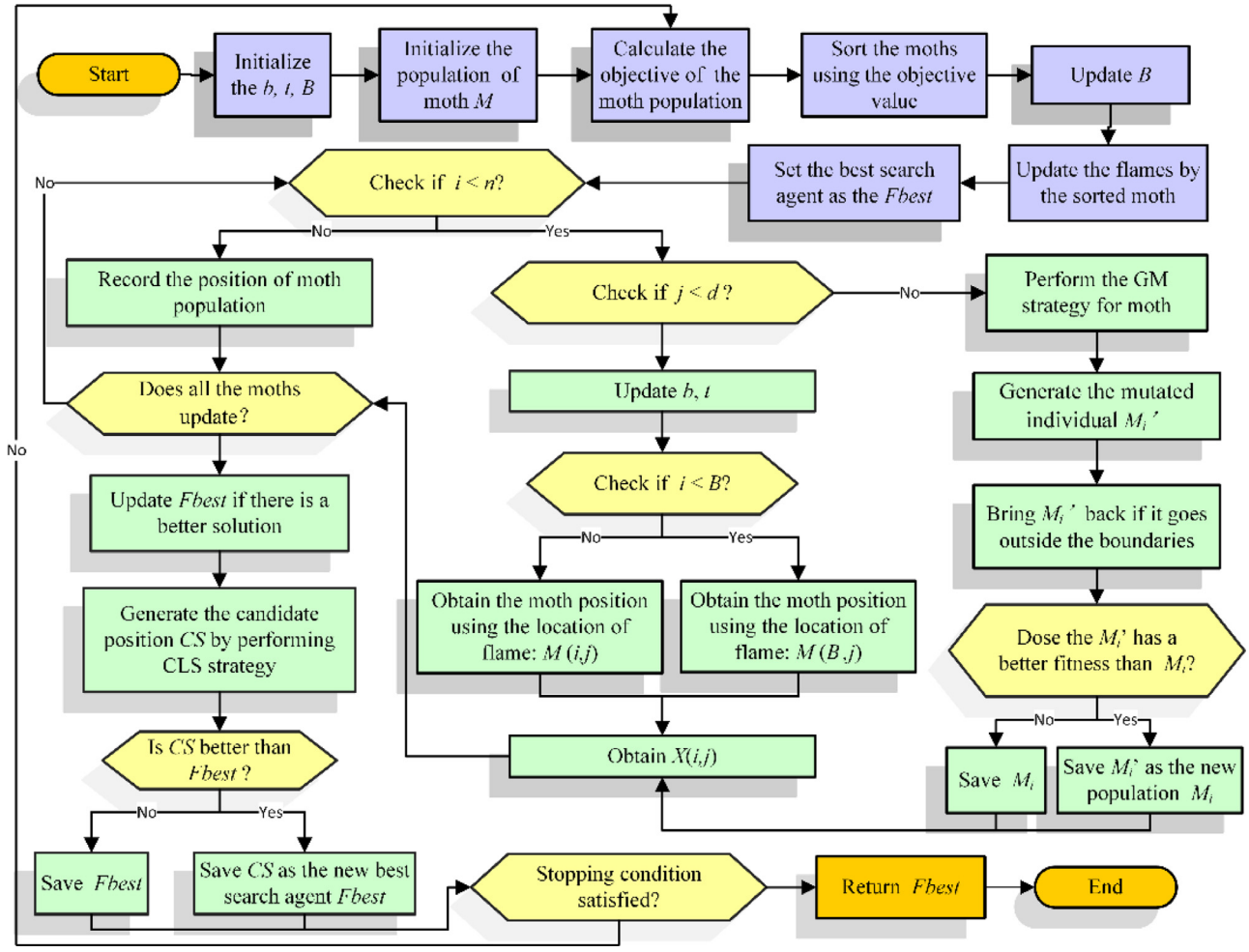

Fig. 1. Flowchart of the proposed CLSGMFO.

the stagnation problem can be relieved. In this manner, it can be ensured that the CLSGMFO algorithm can handle local search more efficiently. Therefore, the achieved balance between exploration and exploitation assists CLSGMFO in converging to a better-quality solution. The CLSGMFO can be divided into several steps as follows:

Step 1. Parameter initialization

Initialize the generation counter g, the number of maximum iterations G, the boundary of the searching area [lb, ub], the population size n, the dimensionality of the space d.

Step 2. Population M initialization randomly within the searching space

Step 3. Calculate the fitness of moths

Step 4. Update flames

Update population of the flame by selecting the best n search agents from M^{g-1} and M^{g}, then update the position of *Fbest*

Step 5. Update moths

Update the moth population M according to the corresponding mechanism Eqs. (6) and (7)

Step 6. Gaussian mutation

Obtain the variant M' by Performing the Gaussian mutation for M by Eq. (23), then update M if the fitness of M' is better than that of M. Bring M' back if it goes outside the boundaries;

Step 7. Chaotic local search

Choose the *Fbest* as the base position, and perform CLS strategy according to Eq. (24) to generate the candidate position *CS*. Bring *CS* back if it goes out of the boundaries. Then, update *Fbest* if the fitness of *CS* is better than that of *Fbest*.

Step 8. Stop or continue the iterative process

Repeat step 3 to step 8 until the exit condition is satisfied, then return *Fbest*.

As mentioned above, the flowchart of CLSGMFO is illustrated in Fig. 1.

The main time complexity of the proposed CLSGMFO depends on steps 3–8. Considering that the calculation of time complexity is dependent on the specific optimization problem, the main focus of the time complexity analysis is on the other five procedures. Therefore, the overall computational complexity is $O(CLSGMFO) = O\ (random\ \ initialization) + G \times (O(fitness\ \ evaluations) + O(flames\ updating) + O(moth\ \ updating) + O(the\ \ GM\ \ strategy) + O(the\ \ CLS\ mechanism))$. Considering the swarm with n moths, the computational complexity of initialization is $O(n \times d)$. Updating the position of flame is $O(G \times (2n)^2)$. The computational complexity of updating positions of all moths is $O(G \times (nd))$. The computational complexity of the GM strategy and the CLS mechanism is $O(G \times (nd + nd + 4n)) + O(G \times (nd + nd + n^2))$. Therefore, The computational complexity of the original MFO is $O(nd) + O(G \times (4n^2 + nd))$ and the final computational complexity of the proposed method is $O(CLSGMFO) \approx O(nd) + O(G \times (5n^2 + 5nd + 4n))$. The complexity of the proposed *CLSGMFO* and the classical MFO based on big o notation is both $O(n^2)$.

3.2. Proposed CLSGMFO-KELM

In this subsection, we seek to construct a new and effective learning system based on the proposed CLSGMFO algorithm. In the

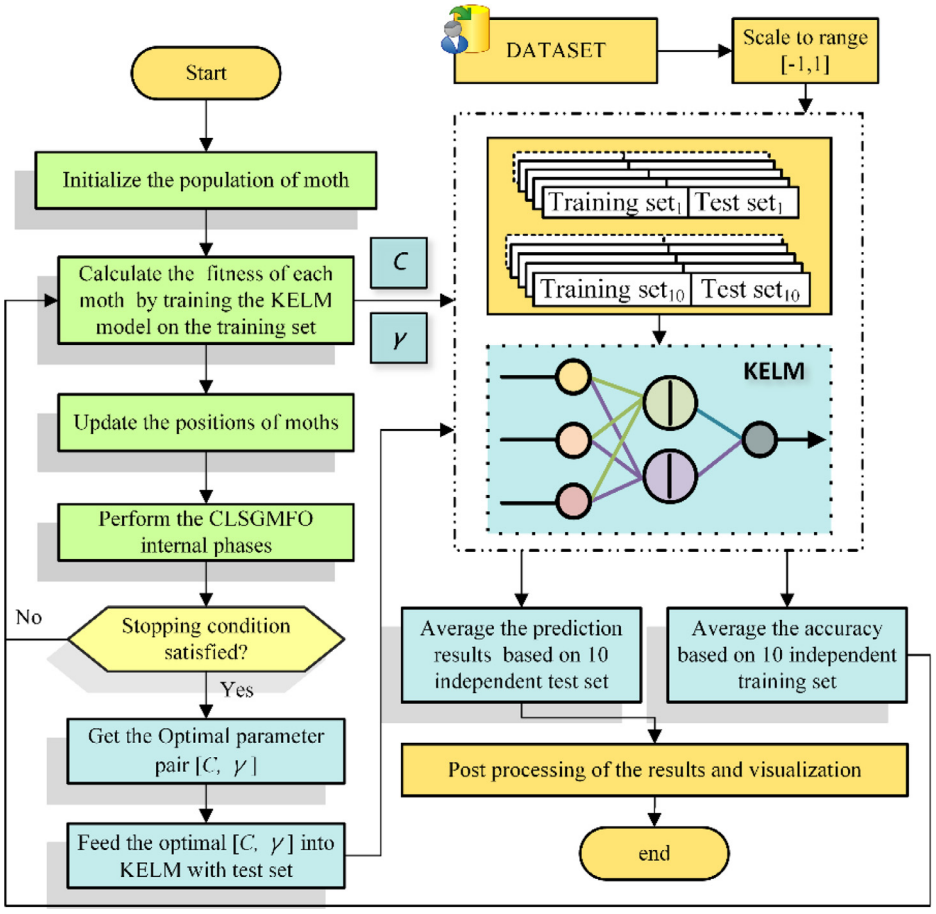

Fig. 2. Flowchart of the proposed CLSGMFO-KELM system.

proposed system, CLSGMFO is used to tune the two parameters of the KELM model. In other words, KELM was chosen to execute the classification task after getting the optimal key main parameter pair (C, γ). For the parameter optimization process, CLSM is suitable for searching better solutions (parameter pair) by using its ergodicity and stochasticity, while GM is adopted to raise the chances of finding the optimal parameter pair. During the procedure of parameter optimization, the key parameters for the whole set are tuned by the CLSGMFO strategy via the k-fold cross validation (CV) analysis (Salzberg, 1997). In this study, the value of k is set as 10. Then, when the process of parameter optimization is terminated, the obtained optimal parameter pair is fed to carry out the classification procedure in the outer loop through use of the k-fold CV analysis. Classification accuracy is considered to design the fitness function as shown in Eq. (25).

$$\text{fitness} = \text{avgACC} = \frac{\sum_{i=1}^{k} \text{testACC}_i}{k} \tag{25}$$

where avgACC means the average classification accuracy obtained by the KELM classifier via the k-fold CV analysis. Hence, the first part of the proposed hybrid system is to standardize the experimental data. The second piece is to obtain the parameter pair obtained by CLSGMFO. The third prong of the hybrid system is to assess parameter pair by KELM model based on a k-fold independent test set, then get the optimal parameter pair by *Fbest*. The fourth part is that feed the optimal parameter into KELM to predict the testing data based on k-fold CV. As mentioned above, the detailed description and the general framework of the proposed CLSGMFO-KELM can be illustrated by Fig. 2 and Algorithm 2, respectively.

The pseudo-code of the whole procedure is given below.

Algorithm 2 Pseudo-code of CLSGMFO-KELM.

Start
  Divide the dataset using k-fold CV
  Initialize *lb*, *ub*, and G;
  Initialize the population of moth M_i ($i = 1, 2, \ldots, n$) based on *lb*, *ub*;
  Set the KELM parameters with the initialized moths $\boldsymbol{C} = M_{(i,1)}$ and $\boldsymbol{\gamma} = M_{(i,2)}$;
  Fbest = the position of the best flame;
  $g = 1$;
While ($g \leq G$)
  Update B according to Eq. (7);
  Classify the training dataset by using the position of M with k-fold CV analysis, and the *avgACC* is achieved as the fitness of moth;
  Update *Fbest* if there is a better solution;
  For $i <$ n
    Update M_i using Eq. (6) with respect to the corresponding moth;
    Perform the GM strategy according to Eq. (23);
    Calculate the fitness of M_i' by training the KELM model on the training set;
    Bring M_i' back if it goes outside the boundaries;
    Update M_i if M_i' is better than M_i
  End for
  Generate the candidate position *CS* by performing CLS strategy according to Eq.(24);
  Update *Fbest* if *CS* is better than *Fbest*
  $g = g + 1$;
End While
Obtain optimal parameters from the best position *Fbest*;
Return ***bestC*** = *Fbest*(1);
Return ***best*γ** = *Fbest*(2);
Classify the testing dataset by using the optimal parameter pair (*bestC*, *best*γ) with k-fold CV analysis, and the average result is achieved as a final
performance measure;
End

4. Experimental designs and results

In this section, experiments were conducted on many functions and bankruptcy classification problems to rigorously examine various aspects of the proposed CLSGMFO algorithm. All results of the proposed CLSGMFO are recorded and compared to several well-known classical MAs and various advanced MAs. All experiments were conducted on a Windows Server 2008 R2 operating system with Intel (R) Xeon (R) CPU E5-2660 v3 (2.60 GHz) and 16 GB of RAM. All algorithms are coded on the MATLAB R2014b software.

4.1. Benchmark function validation

In this subsection, the results of CLSGMFO on 23 benchmark functions are described in detail. Table 1 presents the detailed descriptions of test functions, which can be divided into three classes: unimodal functions ($f_1 - f_7$); multimodal functions

Table 1
Classical benchmark functions.

Function equation	Dim	Range	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100,100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[−10,10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30	[−100,100]	0
$f_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	30	[−100,100]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[−30,30]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[−100,100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	[−1.28,1.28]	0
$f_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[−500,500]	$-418.9829 \times n$
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[−5.12,5.12]	0
$f_{10}(x) = -20\exp\left\{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i}\right\} - exp\left\{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right\} + 20 + e$	30	[−32,32]	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[−600,600]	0
$f_{12}(x) = \frac{\pi}{n}\left\{10\sin(ay_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2 + \sum_{i=1}^{n}\mu(x_i, 10, 100, 4)\right\}$ $y_i = 1 + \frac{x_i + 1}{4}$ $\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	[−50,50]	0
$f_{13}(x) = 0.1\{sin^2(3\pi x_i) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + sin^2(2\pi x_n)] + \sum_{i=1}^{n}\mu(x_i, 5, 100, 4)$	30	[−50,50]	0
$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	[−65,65]	1
$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	[−5,5]	0.00030
$f_{16}(x) = 4x_1^2 - 2.1x_i^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[−5,5]	−1.0316
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	[−5,5]	0.398
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[−2,2]	3
$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_{j-} p_{ij})^2\right)$	3	[1,3]	−3.86
$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_{j-} p_{ij})^2\right)$	6	[0,1]	−3.32
$f_{21}(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	−10.1532
$f_{22}(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	−10.4028
$f_{23}(x) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	−10.5363

($f_8 - f_{13}$); and fixed-dimension multimodal functions ($f_{14} - f_{23}$). Therefore, the 23 benchmark functions can thoroughly measure the performance of the developed CLSGMFO.

All algorithms in this paper were conducted under the same conditions. All algorithms were uniformly randomly initialized and the number of individuals in all algorithms was set to 50, and the maximum number of iterations was set to 1000. The dimensions (Dim) of the above benchmark functions are shown in Table 1. In order to decrease the impact of randomness, the proposed CLSGMFO and other MAs were run 30 times on each function.

Tables 2 and 3 present the comparative results obtained by MAs on the 23 classical benchmark problems, where the Avg. index of each function means the average value and the Std. index represents the standard deviation. Additionally, the Friedman test (Alcalá-Fdez et al., 2009) is also utilized to rank the average performance of the proposed method and all the selected algorithms, then the average ranking value is expressed as ARV in the following simulation experiments. For clarity, best results for each test function are marked in boldface in each table.

Table 2
Results obtained by CLSGMFO and 16 other algorithms on 23 benchmark functions.

	F1		F2		F3	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**0.00E+00**	0.00E+00	**2.43E−295**	0.00E+00	**0.00E+00**	0.00E+00
MFO	2.00E+03	4.07E+03	3.20E+01	2.15E+01	1.52E+04	1.05E+04
BA	1.42E+01	1.49E+00	1.67E+01	1.22E+00	5.20E+01	1.17E+01
DA	6.06E+02	3.15E+02	1.24E+01	5.77E+00	7.33E+03	5.49E+03
FA	2.91E−03	1.12E−03	1.19E−01	6.88E−02	6.44E+02	3.02E+02
FPA	1.73E+02	5.59E+01	2.20E+01	7.90E+00	3.26E+02	1.01E+02
GOA	9.50E−01	9.61E−01	6.96E+00	1.71E+01	1.56E+03	9.16E+02
GWO	9.36E−66	1.35E−65	2.60E−38	2.83E−38	1.82E−17	6.04E−17
SCA	1.28E−02	3.40E−02	5.32E−06	8.50E−06	3.69E+03	4.63E+03
SSA	9.21E−09	2.22E−09	6.01E−01	7.02E−01	1.11E+02	8.26E+01
WOA	5.80E−164	0.00E+00	6.60E−107	2.77E−106	1.47E+04	8.13E+03
MVO	2.31E−01	5.10E−02	3.33E−01	7.69E−02	2.41E+01	7.95E+00
FOA	2.32E−07	2.03E−09	2.64E−03	1.18E−05	7.32E−05	6.16E−07
DE	3.82E−12	2.00E−12	4.40E−08	1.30E−08	2.48E+04	4.04E+03
PSO	1.22E+02	1.70E+01	7.93E+01	4.42E+01	3.67E+02	6.29E+01
ABC	4.29E−11	5.16E−11	1.11E−06	5.79E−07	1.22E+04	2.57E+03
BFO	8.30E−03	4.55E−03	3.38E−01	8.18E−02	3.14E−13	5.01E−13
	F4		F5		F6	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**3.17E−282**	0.00E+00	**4.25E−04**	5.66E−04	**1.05E−15**	3.76E−15
MFO	**6.05E+01**	1.15E+01	2.68E+06	1.46E+07	3.30E+02	1.81E+03
BA	1.71E+00	2.26E−01	3.88E+03	1.11E+03	1.37E+01	2.20E+00
DA	1.91E+01	7.20E+00	7.18E+04	7.07E+04	8.56E+02	4.29E+02
FA	5.66E−02	1.37E−02	1.74E+02	3.47E+02	2.69E−03	8.98E−04
FPA	1.63E+01	2.20E+00	5.77E+03	3.08E+03	1.72E+02	5.30E+01
GOA	9.11E+00	3.62E+00	3.74E+02	3.88E+02	1.45E+00	1.83E+00
GWO	3.29E−16	4.14E−16	2.65E+01	8.72E−01	4.38E−01	3.09E−01
SCA	1.37E+01	9.57E+00	1.60E+02	2.97E+02	4.42E+00	4.59E−01
SSA	5.79E+00	2.37E+00	1.13E+02	1.18E+02	1.05E−08	2.60E−09
WOA	3.19E+01	2.40E+01	2.68E+01	4.82E−01	1.10E−02	1.01E−02
MVO	7.99E−01	3.12E−01	3.62E+02	6.10E+02	2.34E−01	7.66E−02
FOA	8.80E−05	3.29E−07	2.87E+01	8.79E−05	7.50E+00	8.79E−06
DE	1.96E+00	3.30E−01	4.43E+01	2.21E+01	4.52E−12	1.86E−12
PSO	4.42E+00	3.46E−01	1.34E+05	3.48E+04	1.27E+02	1.84E+01
ABC	2.28E+01	3.76E+00	2.38E+00	2.09E+00	3.10E−11	3.45E−11
BFO	2.87E−02	4.38E−03	6.55E+04		1.61E−01	2.71E−02
	F7		F8		F9	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**6.81E−05**	8.69E−05	**−1.26E+04**	1.76E−06	0.00E+00	0.00E+00
MFO	1.74E+00	3.51E+00	−8.85E+03	8.67E+02	1.55E+02	3.83E+01
BA	1.01E+01	8.11E+00	−7.23E+03	6.88E+02	2.58E+02	2.23E+01
DA	2.49E−01	1.57E−01	−5.67E+03	6.06E+02	1.49E+02	3.20E+01
FA	4.56E−03	1.74E−03	−6.81E+03	6.76E+02	3.28E+01	9.23E+00
FPA	1.36E−01	6.65E−02	−8.14E+03	2.72E+02	1.16E+02	2.09E+01
GOA	1.42E−02	6.85E−03	−7.60E+03	7.34E+02	8.69E+01	2.89E+01
GWO	6.54E−04	4.22E−04	−6.10E+03	8.89E+02	2.87E−01	1.57E+00
SCA	3.00E−02	3.40E−02	−3.96E+03	2.37E+02	1.75E+01	2.71E+01
SSA	7.39E−02	2.86E−02	−7.89E+03	6.91E+02	5.56E+01	2.39E+01
WOA	1.51E−03	2.06E−03	−1.15E+04	1.34E+03	0.00E+00	0.00E+00
MVO	1.44E−02	4.82E−03	−7.91E+03	6.96E+02	1.09E+02	2.69E+01
FOA	9.51E−05	4.44E−05	−1.08E+02	4.55E+01	4.59E−05	2.80E−07
DE	2.50E−02	4.35E−03	−1.25E+04	2.09E+02	5.92E+01	4.90E+00
PSO	1.01E+02	2.73E+01	−6.87E+03	9.54E+02	3.71E+02	1.86E+01
ABC	1.24E−01	3.60E−02	−1.21E+04	1.35E+02	4.19E−01	5.96E−01
BFO	3.24E−03	2.60E−03	−2.77E+03	5.59E+02	**−2.88E+02**	4.83E−01

(*continued on next page*)

Table 2
(*continued*)

	F10		F11		F12	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	8.88E − 16	0.00E + 00	**0.00E + 00**	0.00E + 00	**2.37E − 17**	1.07E − 16
MFO	1.37E + 01	8.38E + 00	1.20E + 01	3.12E + 01	3.00E − 01	5.15E − 01
BA	4.75E + 00	2.56E + 00	5.53E − 01	6.28E − 02	1.15E + 01	5.68E + 00
DA	7.83E + 00	1.37E + 00	6.07E + 00	2.80E + 00	7.77E + 01	2.75E + 02
FA	1.32E − 02	2.00E − 03	4.09E − 03	3.70E − 03	2.43E − 05	1.08E − 05
FPA	9.49E + 00	1.35E + 00	2.50E + 00	5.16E − 01	4.27E + 00	8.36E − 01
GOA	3.50E + 00	1.11E + 00	5.39E − 01	1.27E − 01	5.31E + 00	2.37E + 00
GWO	1.39E − 14	2.69E − 15	1.62E − 03	4.32E − 03	3.13E − 02	2.14E − 02
SCA	1.42E + 01	8.49E + 00	2.85E − 01	2.99E − 01	1.48E + 00	1.87E + 00
SSA	1.97E + 00	9.14E − 01	7.31E − 03	8.46E − 03	5.59E + 00	3.07E + 00
WOA	4.68E − 15	2.46E − 15	1.49E − 02	2.82E − 02	4.48E − 03	6.13E − 03
MVO	1.27E + 00	6.87E − 01	4.94E − 01	9.36E − 02	1.01E + 00	1.02E + 00
FOA	3.52E − 04	1.58E − 06	1.55E − 08	1.21E − 10	1.67E + 00	1.96E − 06
DE	5.56E − 07	1.62E − 07	1.02E − 10	2.02E − 10	5.54E − 13	3.57E − 13
PSO	8.40E + 00	4.06E − 01	1.03E + 00	5.29E − 03	4.65E + 00	5.03E − 01
ABC	2.07E − 05	1.20E − 05	6.17E − 04	2.38E − 03	9.99E − 13	1.58E − 12
BFO	**− 9.34E + 12**	4.67E + 11	3.92E − 03	2.39E − 03	2.43E − 11	5.77E − 11
	F13		F14		F15	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**4.89E − 17**	9.85E − 17	**9.98E − 01**	0.00E + 00	**3.42E − 04**	1.67E − 04
MFO	2.56E − 01	7.35E − 01	2.31E + 00	2.19E + 00	1.08E − 03	4.81E − 04
BA	2.20E + 00	2.99E − 01	3.75E + 00	3.30E + 00	4.25E − 03	7.33E − 03
DA	7.17E + 03	1.90E + 04	1.03E + 00	1.81E − 01	1.30E − 03	4.44E − 04
FA	4.09E − 04	2.41E − 04	1.41E + 00	5.54E − 01	7.24E − 04	5.82E − 05
FPA	2.15E + 01	5.70E + 00	9.98E − 01	7.15E − 07	4.35E − 04	9.62E − 05
GOA	1.22E + 01	1.74E + 01	9.98E − 01	3.63E − 16	5.33E − 03	8.08E − 03
GWO	3.53E − 01	2.24E − 01	2.99E + 00	3.42E + 00	1.71E − 03	5.08E − 03
SCA	5.44E + 01	1.92E + 02	1.39E + 00	8.07E − 01	8.92E − 04	3.76E − 04
SSA	6.98E − 02	3.41E − 01	9.98E − 01	2.39E − 16	1.47E − 03	3.58E − 03
WOA	1.05E − 01	9.04E − 02	2.05E + 00	2.50E + 00	7.57E − 04	4.45E − 04
MVO	5.05E − 02	3.49E − 02	9.98E − 01	8.08E − 12	3.95E − 03	7.46E − 03
FOA	5.78E − 01	8.29E − 02	1.27E + 01	2.09E − 15	7.80E − 04	3.31E − 04
DE	2.38E − 12	1.61E − 12	1.03E + 00	1.81E − 01	1.29E − 03	3.60E − 03
PSO	2.14E + 01	3.89E + 00	2.22E + 00	2.02E + 00	1.16E − 03	1.87E − 04
ABC	6.39E − 11	8.99E − 11	**9.98E − 01**	7.14E − 17	7.44E − 04	2.55E − 04
BFO	9.64E − 02	8.47E − 03	1.63E + 00	9.57E − 01	4.85E − 04	9.39E − 05
	F16		F17		F18	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**− 1.03E + 00**	6.78E − 16	**3.98E − 01**	0.00E + 00	**3.00E + 00**	2.18E − 15
MFO	**− 1.03E + 00**	6.78E − 16	**3.98E − 01**	0.00E + 00	3.00E + 00	2.19E − 15
BA	− 1.03E + 00	3.94E − 04	3.98E − 01	2.72E − 04	3.03E + 00	2.69E − 02
DA	− 1.03E + 00	9.32E − 06	3.98E − 01	5.42E − 08	3.00E + 00	9.93E − 06
FA	− 1.03E + 00	1.44E − 09	3.98E − 01	3.35E − 10	3.00E + 00	1.36E − 08
FPA	− 1.03E + 00	1.94E − 11	3.98E − 01	3.01E − 15	3.00E + 00	5.13E − 12
GOA	− 1.03E + 00	1.85E − 14	3.98E − 01	1.88E − 14	5.70E + 00	1.48E + 01
GWO	− 1.03E + 00	3.90E − 09	3.98E − 01	1.62E − 07	5.70E + 00	1.48E + 01
SCA	− 1.03E + 00	2.14E − 05	3.98E − 01	5.63E − 04	3.00E + 00	2.25E − 05
SSA	− 1.03E + 00	7.01E − 15	3.98E − 01	2.21E − 15	3.00E + 00	1.16E − 13
WOA	− 1.03E + 00	8.16E − 11	3.98E − 01	4.58E − 07	3.00E + 00	2.57E − 05
MVO	− 1.03E + 00	8.62E − 08	3.98E − 01	2.46E − 08	3.00E + 00	4.64E − 07
FOA	− 1.72E − 01	1.62E − 01	1.37E + 00	9.41E − 01	6.00E + 02	5.22E − 04
DE	**− 1.03E + 00**	6.78E − 16	**3.98E − 01**	0.00E + 00	3.00E + 00	1.23E − 15
PSO	− 1.03E + 00	1.16E − 03	3.98E − 01	5.42E − 04	3.09E + 00	1.00E − 01
ABC	**− 1.03E + 00**	5.98E − 16	**3.98E − 01**	0.00E + 00	3.00E + 00	2.25E − 06
BFO	− 1.03E + 00	1.03E − 06	3.98E − 01	5.33E − 07	3.00E + 00	9.34E − 05
	F19		F20		F21	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**− 3.86E + 00**	2.65E − 15	− 3.27E + 00	6.03E − 02	**− 1.02E + 01**	6.96E − 15
MFO	**− 3.86E + 00**	2.71E − 15	− 3.23E + 00	5.63E − 02	− 6.81E + 00	3.50E + 00
BA	− 3.85E + 00	1.02E − 02	− 2.90E + 00	1.20E − 01	− 5.14E + 00	2.76E + 00
DA	− 3.86E + 00	4.00E − 04	− 3.27E + 00	7.60E − 02	− 7.78E + 00	2.56E + 00
FA	− 3.86E + 00	3.67E − 10	− 3.30E + 00	4.84E − 02	− 8.66E + 00	3.04E + 00
FPA	− 3.86E + 00	1.42E − 11	− 3.32E + 00	3.39E − 05	− 1.02E + 01	1.05E − 04
GOA	− 3.80E + 00	2.09E − 01	− 3.27E + 00	6.04E − 02	− 4.72E + 00	2.70E + 00
GWO	− 3.86E + 00	1.65E − 03	− 3.30E + 00	4.90E − 02	− 9.65E + 00	1.55E + 00
SCA	− 3.85E + 00	2.17E − 03	− 2.89E + 00	3.32E − 01	− 3.20E + 00	2.04E + 00
SSA	− 3.86E + 00	1.36E − 14	− 3.24E + 00	5.74E − 02	− 8.48E + 00	2.90E + 00

(*continued on next page*)

Table 2
(*continued*)

	F19		F20		F21	
	Avg	Std	Avg	Std	Avg	Std
WOA	−3.86E+00	2.17E−03	−3.25E+00	8.74E−02	−9.22E+00	2.15E+00
MVO	−3.86E+00	6.63E−07	−3.26E+00	6.05E−02	−7.63E+00	2.81E+00
FOA	−3.63E+00	2.73E−01	−1.73E+00	4.12E−01	−3.92E+00	7.89E−01
DE	**−3.86E+00**	2.71E−15	−3.32E+00	7.73E−04	−1.02E+01	1.18E−07
PSO	−3.85E+00	8.17E−03	−2.83E+00	2.25E−01	−4.83E+00	1.35E+00
ABC	**−3.86E+00**	2.26E−15	**−3.32E+00**	1.23E−15	−1.02E+01	4.91E−14
BFO	−3.86E+00	3.82E−04	−3.29E+00	1.31E−02	−1.01E+01	6.97E−03

	F22		F23		Overall Rank	
	Avg	Std	Avg	Std	Rank	ARV
CLSGMFO	**−1.04E+01**	1.19E−15	**−1.05E+01**	1.75E−15	**1**	**1.827536**
MFO	−7.45E+00	3.30E+00	−8.48E+00	3.26E+00	13	8.85942
BA	−6.19E+00	3.16E+00	−5.24E+00	2.99E+00	16	13.44928
DA	−8.34E+00	2.79E+00	−8.30E+00	2.81E+00	15	12.4087
FA	−1.01E+01	1.40E+00	−1.05E+01	7.38E−07	4	7.782609
FPA	−1.04E+01	1.66E−04	−1.05E+01	8.24E−04	10	10.59783
GOA	−6.34E+00	3.69E+00	−6.94E+00	3.94E+00	12	10.36449
GWO	−1.02E+01	9.70E−01	−1.00E+01	1.98E+00	5	6.931159
SCA	−4.60E+00	1.38E+00	−3.82E+00	1.69E+00	14	12.05942
SSA	−9.29E+00	2.58E+00	−9.22E+00	2.74E+00	8	7.636957
WOA	−9.69E+00	1.84E+00	−8.99E+00	2.93E+00	7	7.666667
MVO	−8.50E+00	3.02E+00	−9.73E+00	2.14E+00	9	9.295652
FOA	−3.73E+00	8.49E−01	−3.65E+00	7.74E−01	11	11.07101
DE	−1.04E+01	1.07E−02	−1.05E+01	3.17E−12	3	4.598551
PSO	−4.86E+00	1.21E+00	−5.28E+00	1.25E+00	17	14.24348
ABC	−1.04E+01	7.00E−06	−1.05E+01	9.32E−03	2	5.402899
BFO	−1.04E+01	6.46E−03	−1.05E+01	6.92E−03	6	8.804348

4.1.1. Comparison of CLSGMFO with traditional MAs

CLSGMFO was compared with a comprehensive set of classical MAs including BA, DA, FA, FPA, GOA (Heidari, Faris, Aljarah, & Mirjalili, 2018), GWO, SCA, SSA (Aljarah et al., 2018; Faris et al., 2018; Faris, Mirjalili, Aljarah, Mafarja, & Heidari, 2020), WOA, MVO, FOA, DE, PSO, ABC, BFO, and the original MFO. Table 2 presents the comparative results of the developed CLSGMFO versus the above popular MAs on $f_1 - f_{23}$ problems.

From Table 2, it can be seen that CLSGMFO obtains the best solutions on all unimodal functions ($f_1 - f_7$). Hence, it can be concluded that CLSGMFO is better than other MAs in dealing with the 7 unimodal functions. In addition, the proposed CLSGMFO obtains the best results on f_8, f_{11}, f_{12} and f_{13}, and achieved the second-best results on f_9 and f_{10}. Moreover, the ranking results for all multimodal functions ($f_8 - f_{13}$) also show that the proposed CLSGMFO approach can provide the fittest solution among all in terms of the mean index. In dealing with the ten fixed-dimension multimodal functions, CLSGMFO obtains the best solutions on all the functions except f_{20}, as shown in Table 2. However, even on f_{20}, CLSGMFO still obtains a better solution than the original MFO. The average performance of the proposed CLSGMFO and other MAs is also ranked by the non-parameters statistical Friedman test. It can be seen in the final column in Table 2, according to the ARV index, the developed CLSGMFO achieves the lowest average ranking in dealing with three different types of functions, followed by the traditional DE, ABC, GWO, SSA, WOA, FA, BFO, MFO, MVO, GOA,

Table 3
Results obtained by CLSGMFO and 7 other algorithms on 23 benchmark functions.

	F1		F2		F3	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**0.00E+00**	0.00E+00	**2.02E−298**	0.00E+00	**0.00E+00**	0.00E+00
ALCPSO	6.46E−18	9.96E−18	6.54E−08	2.31E−07	2.36E+02	1.49E+02
BLPSO	1.43E+03	2.60E+02	1.46E+01	1.37E+00	1.05E+04	2.00E+03
CLPSO	3.68E+02	7.88E+01	9.51E+00	1.55E+00	1.49E+04	3.41E+03
TVPSO	1.06E−09	3.32E−09	5.26E−03	9.87E−03	3.16E+00	4.30E+00
JADE	1.87E−31	6.43E−31	2.79E−15	9.51E−15	1.10E−03	5.14E−03
JDE	3.33E−17	3.83E−17	5.10E−11	4.88E−11	9.21E+01	7.13E+01
SADE	9.96E−16	4.78E−15	7.00E−12	8.68E−12	4.97E+01	3.85E+01

	F4		F5		F6	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**6.93E−280**	0.00E+00	**3.44E−04**	5.64E−04	6.26E−16	1.41E−15
ALCPSO	4.55E+00	1.75E+00	4.38E+01	4.13E+01	3.82E−17	1.22E−16
BLPSO	2.51E+01	2.18E+00	3.02E+05	9.57E+04	1.46E+03	2.90E+02
CLPSO	3.98E+01	3.12E+00	5.45E+04	2.52E+04	3.43E+02	8.90E+01
TVPSO	2.98E−01	1.68E−01	3.62E+01	2.41E+01	1.49E−08	7.68E−08
JADE	1.66E−03	1.98E−03	1.18E+01	1.57E+01	**4.59E−31**	1.65E−30
JDE	1.08E+01	4.96E+00	3.35E+01	2.11E+01	1.78E−17	1.49E−17
SADE	3.31E+00	1.92E+00	5.63E+01	3.95E+01	5.76E−17	1.55E−16

(*continued on next page*)

Table 3
(*continued*)

	F7		F8		F9	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**8.96E − 05**	1.14E − 04	− 1.26E + 04	1.46E − 06	**0.00E + 00**	0.00E + 00
ALCPSO	6.98E − 02	2.84E − 02	− 1.01E + 04	4.93E + 02	5.87E + 01	1.78E + 01
BLPSO	4.90E − 01	1.30E − 01	− 4.84E + 03	3.21E + 02	1.95E + 02	1.33E + 01
CLPSO	1.64E − 01	4.11E − 02	− 8.59E + 03	4.13E + 02	1.22E + 02	1.28E + 01
TVPSO	3.28E − 02	1.60E − 02	− 7.04E + 03	8.20E + 02	5.09E + 01	1.20E + 01
JADE	6.49E − 03	2.48E − 03	− 1.24E + 04	1.27E + 02	1.71E − 04	1.52E − 04
JDE	1.36E − 02	4.49E − 03	− 1.25E + 04	5.74E + 01	4.66E − 04	1.02E − 03
SADE	1.23E − 02	4.92E − 03	**− 1.26E + 04**	3.60E − 06	6.02E + 00	5.14E + 00
	F10		F11		F12	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**8.88E − 16**	0.00E + 00	**0.00E + 00**	0.00E + 00	1.09E − 17	2.84E − 17
ALCPSO	6.59E − 01	7.74E − 01	1.52E − 02	1.48E − 02	1.52E − 01	3.86E − 01
BLPSO	9.15E + 00	6.23E − 01	1.43E + 01	1.54E + 00	4.94E + 02	1.06E + 03
CLPSO	7.17E + 00	5.01E − 01	4.11E + 00	6.02E − 01	1.47E + 01	3.64E + 00
TVPSO	5.86E − 01	6.68E − 01	2.22E − 02	2.75E − 02	8.44E − 11	2.39E − 10
JADE	1.31E − 14	2.46E − 14	2.87E − 03	7.85E − 03	1.73E − 02	7.74E − 02
JDE	1.01E − 09	7.78E − 10	1.48E − 17	4.82E − 17	**1.75E − 18**	2.82E − 18
SADE	4.92E − 01	5.89E − 01	8.09E − 03	1.52E − 02	4.15E − 02	1.21E − 01
	F13		F14		F15	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	1.61E − 16	5.02E − 16	**9.98E − 01**	0.00E + 00	3.08E − 04	2.03E − 06
ALCPSO	3.66E − 03	5.27E − 03	**9.98E − 01**	1.01E − 16	1.87E − 03	5.04E − 03
BLPSO	1.21E + 05	7.71E + 04	**9.98E − 01**	0.00E + 00	6.01E − 04	4.46E − 05
CLPSO	1.25E + 03	1.80E + 03	**9.98E − 01**	0.00E + 00	6.16E − 04	9.84E − 05
TVPSO	4.03E − 03	8.89E − 03	1.03E + 00	1.81E − 01	4.72E − 04	3.02E − 04
JADE	**5.45E − 24**	2.58E − 23	**9.98E − 01**	0.00E + 00	3.01E − 03	6.92E − 03
JDE	3.66E − 04	2.01E − 03	**9.98E − 01**	0.00E + 00	**3.07E − 04**	2.09E − 19
SADE	3.66E − 04	2.01E − 03	**9.98E − 01**	0.00E + 00	**3.07E − 04**	2.14E − 19
	F16		F17		F18	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**− 1.03E + 00**	6.78E − 16	**3.98E − 01**	0.00E + 00	3.00E + 00	1.37E − 15
ALCPSO	**− 1.03E + 00**	5.98E − 16	**3.98E − 01**	0.00E + 00	3.00E + 00	1.66E − 15
BLPSO	**− 1.03E + 00**	6.78E − 16	**3.98E − 01**	0.00E + 00	3.00E + 00	1.36E − 15
CLPSO	**− 1.03E + 00**	5.18E − 16	**3.98E − 01**	0.00E + 00	3.00E + 00	1.33E − 15
TVPSO	**− 1.03E + 00**	6.71E − 16	**3.98E − 01**	0.00E + 00	3.00E + 00	6.55E − 16
JADE	**− 1.03E + 00**	6.78E − 16	**3.98E − 01**	0.00E + 00	**3.00E + 00**	1.82E − 15
JDE	**− 1.03E + 00**	6.78E − 16	**3.98E − 01**	0.00E + 00	3.00E + 00	1.19E − 15
SADE	**− 1.03E + 00**	6.78E − 16	**3.98E − 01**	0.00E + 00	**3.00E + 00**	2.01E − 15
	F19		F20		F21	
	Avg	Std	Avg	Std	Avg	Std
CLSGMFO	**− 3.86E + 00**	2.68E − 15	− 3.29E + 00	5.54E − 02	**− 1.02E + 01**	1.07E − 07
ALCPSO	**− 3.86E + 00**	2.60E − 15	− 3.27E + 00	5.92E − 02	− 7.63E + 00	3.21E + 00
BLPSO	**− 3.86E + 00**	2.71E − 15	− 3.31E + 00	3.75E − 02	− 9.61E + 00	1.61E + 00
CLPSO	**− 3.86E + 00**	2.71E − 15	− 3.32E + 00	3.78E − 07	− 1.02E + 01	6.37E − 03
TVPSO	**− 3.86E + 00**	2.71E − 15	− 3.26E + 00	6.03E − 02	− 8.05E + 00	2.88E + 00
JADE	**− 3.86E + 00**	2.71E − 15	− 3.29E + 00	5.11E − 02	− 9.14E + 00	2.06E + 00
JDE	**− 3.86E + 00**	2.71E − 15	− 3.27E + 00	5.92E − 02	− 9.64E + 00	1.55E + 00
SADE	**− 3.86E + 00**	2.71E − 15	**− 3.32E + 00**	1.28E − 15	− 9.66E + 00	1.90E + 00
	F22		F23		Overall Rank	
	Avg	Std	Avg	Std	Rank	ARV
CLSGMFO	**− 1.04E + 01**	1.28E − 15	**− 1.05E + 01**	2.06E − 15	**1**	**2.802174**
ALCPSO	− 9.42E + 00	2.28E + 00	− 8.29E + 00	3.36E + 00	5	5.158696
BLPSO	**− 1.04E + 01**	1.28E − 15	**− 1.05E + 01**	1.81E − 15	8	6.314493
CLPSO	− 1.04E + 01	1.76E − 03	− 1.05E + 01	2.89E − 03	6	6.554348
TVPSO	− 9.37E + 00	2.41E + 00	− 9.23E + 00	2.71E + 00	6	4.967391
JADE	− 9.88E + 00	1.61E + 00	− 1.03E + 01	1.40E + 00	2	2.937681
JDE	− 1.02E + 01	1.22E + 00	**− 1.05E + 01**	1.81E − 15	3	3.599275
SADE	− 1.02E + 01	1.22E + 00	**− 1.05E + 01**	1.78E − 15	3	3.665942

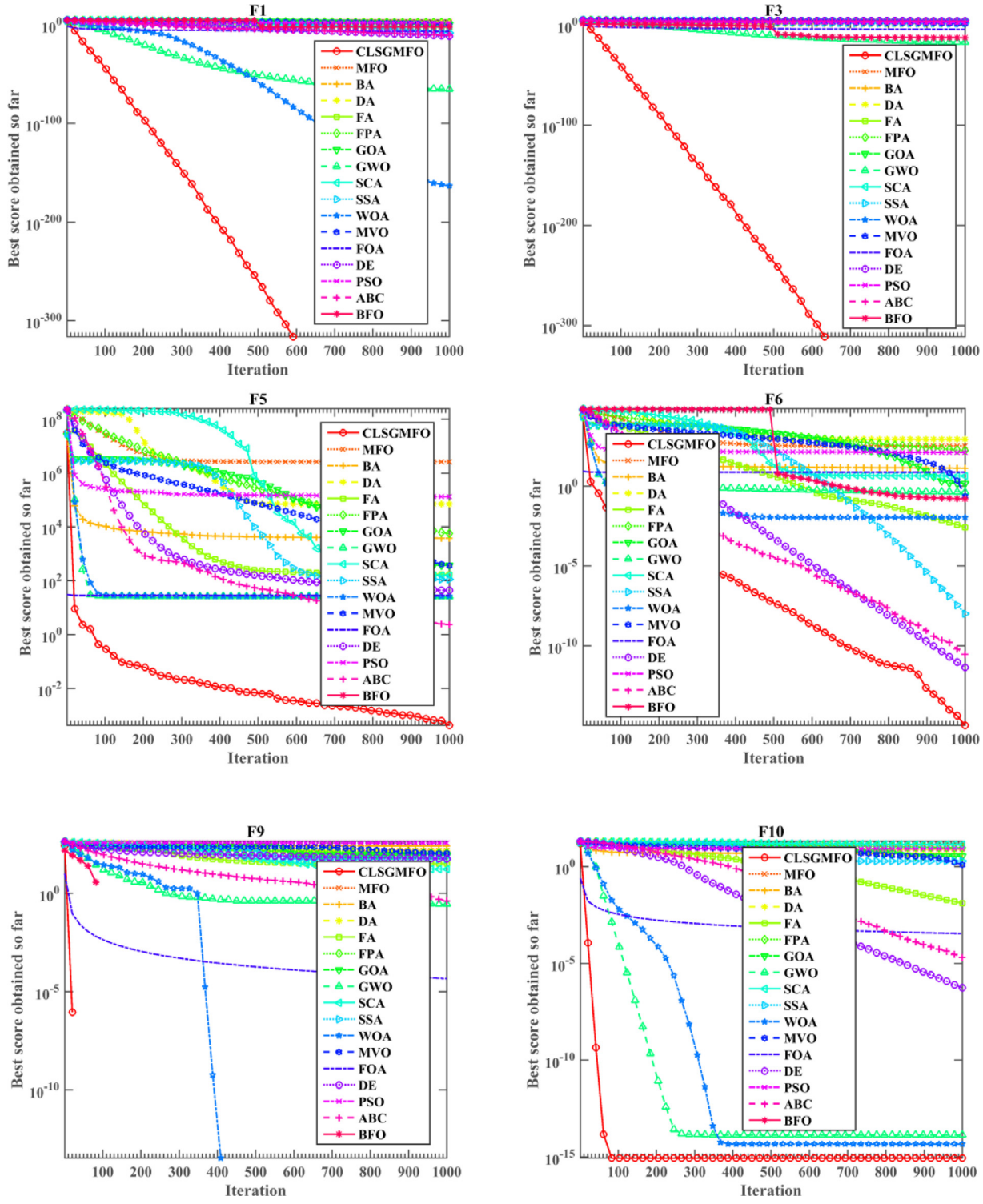

Fig. 3. Convergence curves of CLSGMFO, MFO, BA, DA, FA, FPA, GOA, GWO, SCA, SSA, WOA, MVO, FOA, DE, PSO, ABC, and BFO on F1, F3, F5, F6, F9 and F10 benchmark functions.

FPA, FOA, SCA, DA, BA, and PSO methods. The experimental observations indicate that the improved CLSGMFO can still be the best algorithm in tackling 23 benchmark tasks under 30 independent runs.

From Table 2, it can be seen that CLSGMFO obtains better results than the original MFO in dealing with all 23 benchmark functions. This finding demonstrates that the GM operator and the CLS strategy utilized in the CLSGMFO algorithm have improved the efficacy of the original MFO in a significant manner. Based on the overall rank, CLSGMFO is followed successively by ABC, DE, FA, GWO, BFO, WOA, SSA, MVO, FPA, FOA, GOA, MFO, SCA, DA, BA, and PSO.

Moreover, to visually compare the performance of CLSGMFO and the other optimizers, convergence curves of CLSGMFO, BA, DA,

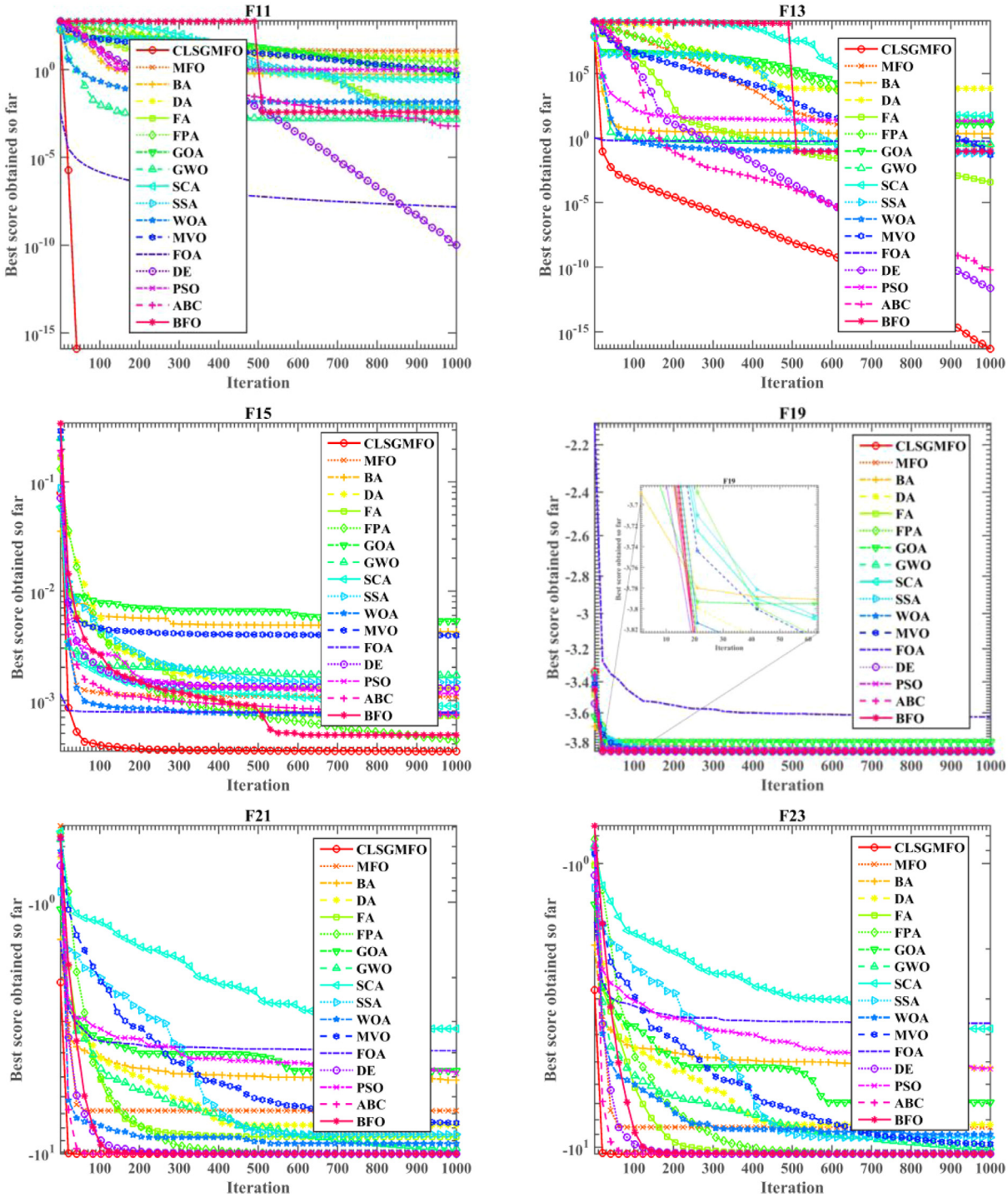

Fig. 4. Convergence curves of CLSGMFO, MFO, BA, DA, FA, FPA, GOA, GWO, SCA, SSA, WOA, MVO, FOA, DE, PSO, ABC, and BFO on F11, F13, F15, F19, F21, and F23 benchmark functions.

FA, FPA, GOA, GWO, SCA, SSA, WOA, MVO, FOA, DE, PSO, ABC, BFO and the original MFO on the 12 selected classical benchmark functions are generated, as shown in Figs. 3 and 4.

As can be seen from Fig. 3, CLSGMFO obtains the optimal solutions on f_1 and f_3. In addition, on f_5 and f_6, CLSGMFO obtains the best solutions and its convergence curve is far better than those of the other optimizers. Therefore, it can be concluded that CLSGMFO is much better in performance than the other algorithms on unimodal functions. As can be seen from Figs. 3 and 4, on f_9, CLSGMFO has the fastest convergence speed, while WOA and GWO are the second fittest methods due to their similar convergence trends; however, BA, DA, FA, FPA, GOA, SCA, SSA, MVO, FOA, DE, PSO, ABC, BFO and the original MFO cannot show a better trend. On f_{11}, CLSGMFO shows the best convergence value in the early stages, while the other algorithms all fall into local optimal solutions because of their weak searching capability. The proposed CLSGMFO converges very fast on f_{10} and f_{13}. Therefore, it can be concluded that CLSGMFO outperforms the other optimizers on multimodal functions.

As can be seen from Fig. 4, on f_{15}, CLSGMFO has a faster convergence speed than the other optimizers, and on f_{19}, all the approaches obtain competitive results but CLSGMFO obtains the best results among all. On f_{21} and f_{23}, there is a close rivalry between ABC and CLSGMFO. However, it can be seen that CLSGMFO converges in a few searching steps and obtains better results than ABC.

A comparison of convergence speed among CLSGMFO and the 16 other optimizers demonstrates that CLSGMFO can search the solution space more effectively. According to Figs. 3 and 4, CLSGMFO outperforms the original MFO in terms of convergence speed on f_1, f_3, f_5, f_6, f_9, f_{10}, f_{11}, f_{13}, f_{15}, f_{19}, f_{21} and f_{23}. Accordingly, it can be concluded that the chaotic local search strategy and the Gaussian mutation operation lead to the improved searching capability of CLSGMFO.

4.1.2. Comparison of CLSGMFO with advanced MAs

In this subsection, CLSGMFO was compared with 7 famous advanced MAs, namely, ALCPSO (Chen et al., 2013), BLPSO (Chen, Tianfield, Mei, Du, & Liu, 2017), CLPSO (Liang, Qin, Suganthan, & Baskar, 2006), TVPSO (Ratnaweera, Halgamuge, & Watson, 2004), JADE (Zhang & Sanderson, 2009), JDE (Brest, Greiner, Boskovic, Mernik, & Zumer, 2006) and SADE (Qin & Suganthan, 2005). TVPSO is an improved time variant particle swarm optimization algorithm. ALCPSO is an enhanced PSO with aging leader and challengers which overcomes the drawback of premature convergence suffered by the original PSO. CLPSO uses a novel comprehensive learning strategy to preserve the diversity of the swarm to avoid premature convergence. In CLPSO, particles update the flying velocity according to all the personal best historical information. And BLPSO uses a biogeography-based learning strategy in place of the comprehensive learning strategy in CLPSO. In this method, each particle updates itself by using the combination of its own personal fittest position and personal best positions of all other particles through biogeography-based optimization. JDE is a novel version of the DE algorithm for obtaining self-adaptive control parameter settings. SADE is a new self-adaptive differential evolution algorithm. During the evolution process, the proper learning technique and parameter updating are gradually self-adapted according to the learning experience. JADE is a DE algorithm with a new mutation and optional external archive and adaptive modes. In the experiments, parameters of CLSGMFO were the same as those used in the above experiments. Parameters, except the maximum number of iterations and the population size of other algorithms, were the same as those used in the original references for fair comparison. The detailed comparative results are shown in Table 3 and Figs. 5 and 6.

As can be seen from Table 3, CLSGMFO obtains the best solutions on 16 out of 23 benchmark functions. Specifically, CLSGMFO significantly outperforms TVPSO on all the 23 benchmark functions and performs better than ALCPSO, BLPSO, CLPSO, JADE, JDE, and SADE on $f_1, f_2, f_3, f_4, f_5, f_7, f_9, f_{10}, f_{11}, f_{14}, f_{16}, f_{17}, f_{19}, f_{21}, f_{22}, f_{23}$. It can also be seen that ALCPSO outperforms CLSGMFO on f_6, BLPSO and CLPSO outperform CLSGMFO on f_{20}, JADE outperforms CLSGMFO on four functions (f_6, f_{13}, f_{18} and f_{20}), JDE outperforms CLSGMFO on three functions (f_6, f_{12} and f_{15}), and SADE outperforms CLSGMFO on f_8, f_{15}, f_{18} and f_{20}. However, based on the overall rank in Table 3, CLSGMFO takes the first place. Also, the proposed CLSGMFO provides the best ranking of 2.802174.

The average convergence patterns are compared in a logarithmic scale in Figs. 5 and 6. As can be seen from Figs. 5 and 6, the proposed CLSGMFO obtains better solutions and has faster convergence speeds than the 7 other advanced competitors in dealing with the benchmark functions. Therefore, from results and convergence trends, it is can be concluded that CLSGMFO obtains the fittest results over 30 independent runs.

4.2. Application to the financial prediction problem

To investigate the effectiveness of the CLSGMFO for evolving KELM in the context of corporate the real-world optimization problem, the established CLSGMFO-KELM model was evaluated on two widely used financial datasets, the Wieslaw dataset (Pietruszkiewicz, 2008) and the Australian credit dataset (Chen et al., 2011; Wang, Chen, Li, et al., 2017), and was compared against various popular classifiers including k-Nearest Neighbor (KNN), Classification and Regression Tree (CART), Back propagation (BP), Adaptive Boosting algorithm (Adaboost), Support vector machine (SVM), random forest (RF)[1] and the basic MFO based KELM model (MFO-KELM). The BP with the Levenberg-Marquardt training algorithm in the MATLAB neural network toolbox was employed. ClassificationTree (fitctree) algorithm and fitensemble (AdaBoostM1) in the MATLAB Statistics toolbox were used for CART and Adaboost, respectively. For SVM, the LIBSVM developed by (Chang & Lin, 2011) was applied. KNN and MFO-KELM were developed from scratch using MATLAB.

For a fair comparison, all the implementations were conducted under the same conditions. The detailed parameter settings for CLSGMFO-KELM and MFO-KELM are given as follows. The range for kernel parameter C and kernel band width γ was set to $[2^{-5}, 2^5]$. The population size and the maximum iteration number were set to 30 and 100, respectively. The parameters for the other classifiers were set as follows. The value of the nearest neighbor in KNN was set to 1, and the Euclidean distance metric was used in this study. In the SVM classifier, the Gaussian kernel function was utilized in this work, and the penalty coefficient and the kernel function parameter of SVM were optimized by using a grid search. And the parameter range of these two parameters was the same as that for KELM. For BP, 8 hidden neurons were taken for training, the maximum number of generations was set to 200, the learning rate was set to 0.01, and the mean squared error goal was set to 0.001. The value of ensemble learning cycle in the AdaBoost classifier was set to 500. Parameters for CART and RF were set by default, as provided by the above software package and the Classification Tree toolbox, respectively.

4.2.1. Data description

As mentioned above, two real-world datasets, the Wieslaw dataset, and the Australian credit dataset, were used to evaluate the performance of the hybrid CLSGMFO-MFO system. The Wieslaw dataset was obtained from Pietruszkiewicz (2008) and it contains corporate bankruptcy-related information. The dataset, which collects information from 240 companies (112 bankrupt enterprises and 128 non-bankrupt cases), was compiled during a five-year period in Poland. A total of 30 financial features, as described in Table 4, were used for prediction.

The Australian credit dataset is publicly obtainable from the UCI Repository and it contains 690 samples (307 'good credit' cases and 383 'bad credit' cases) with a total of 14 features. Each sample consists of three general categories: 6 nominal attributes, 8 numeric attributes, and 1 class attribute. For the sake of information security, the Australian dataset provider replaced all attribute names with meaningless symbols.

The data were normalized into the interval of [0, 1] prior to classification. In addition, in order to enhance the reliability of the generalization results, the k-fold CV analysis was used to estimate the classification results. Specifically, the dataset was divided into k subsets, and then one of the k subsets was used as the testing set while the rest of the subsets were put together to form a training set. This procedure was conducted for k trials and

[1] the software package at http://code.google.com/p/randomforest-matlab

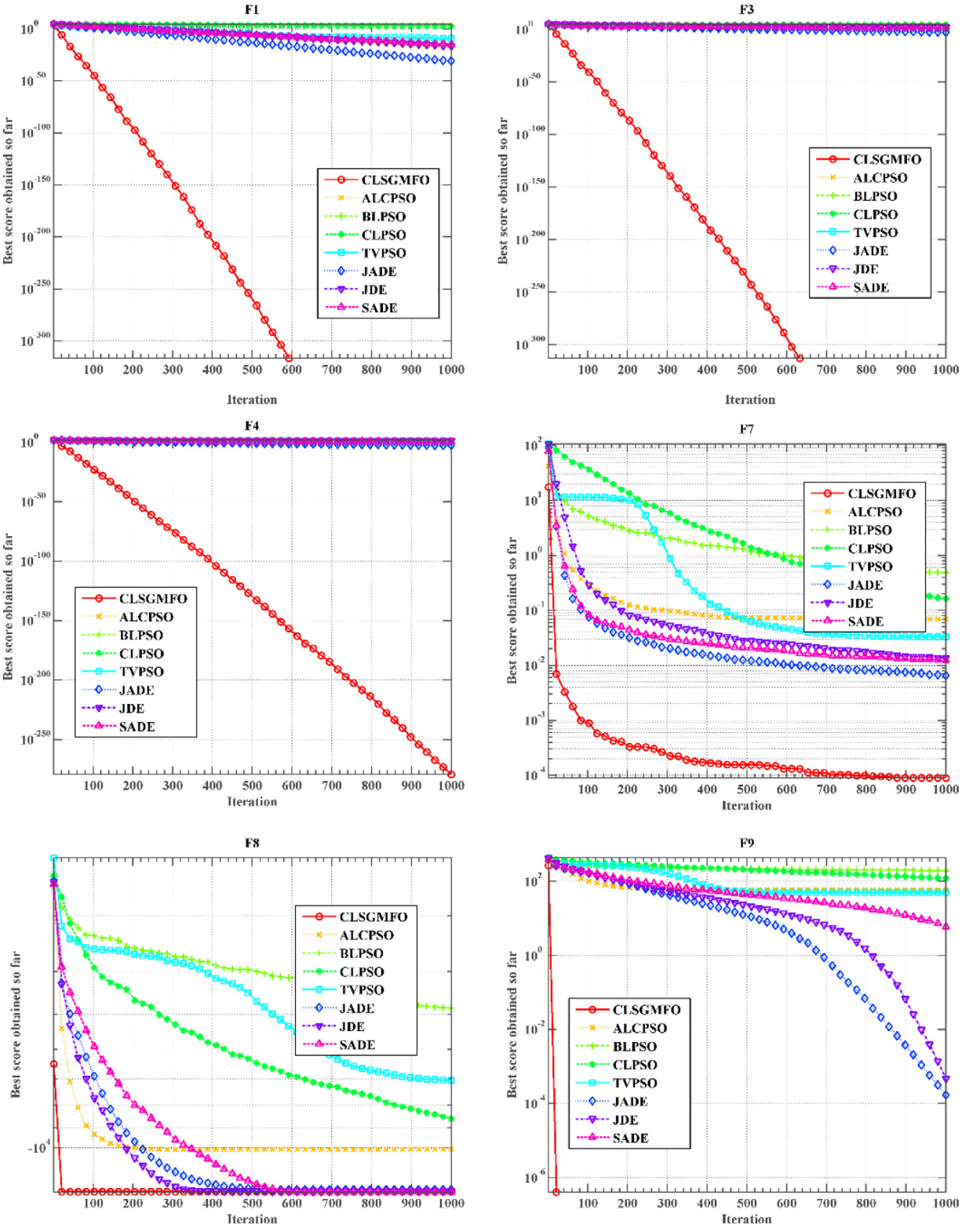

Fig. 5. Convergence curves of CLSGMFO, ALCPSO, BLPSO, CLPSO, TVPSO, JADE, JDE, and SADE on F1, F3, F4, F7, F8, and F9 benchmark functions.

the average value of the test results was considered as the final result.

4.2.2. Measure metrics for performance evaluation

In order to measure the performance of the developed hybrid system, four common criteria are used in the classification field. The criteria are derived from the confusion matrix which is an original source for evaluating the performance of a binary classification model. According to the confusion matrix shown in Fig. 7, these measure metrics used in this work are defined below.

ACC is the percentage of correctly classified instances and it is an essential measure of classification performance.

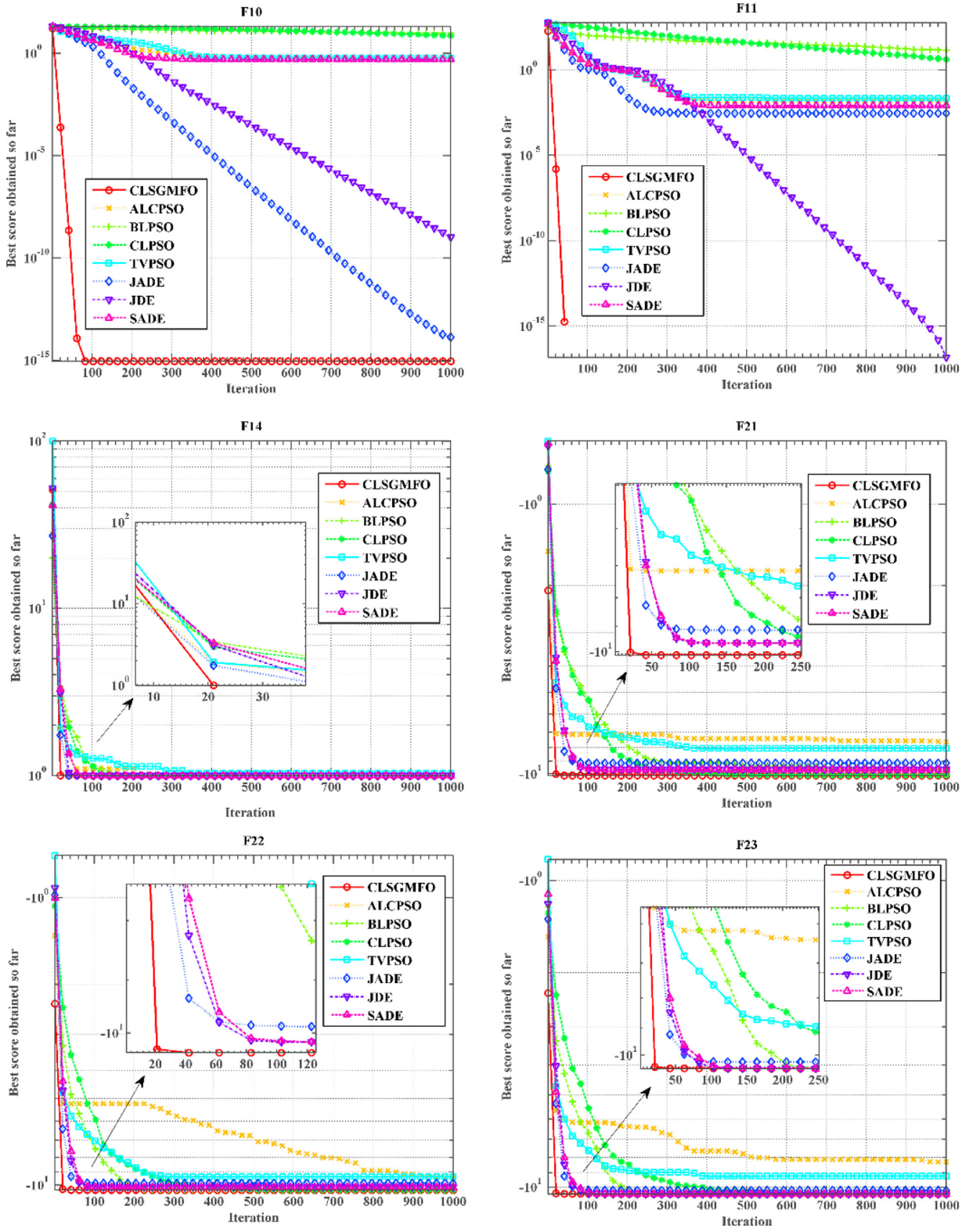

Fig. 6. Convergence curves of CLSGMFO, ALCPSO, BLPSO, CLPSO, TVPSO, JADE, JDE, and SADE on F10, F11, F14, F21, F22, and F23 benchmark functions.

$$ACC = \frac{TP + TN}{TP + FP + FN + TN}$$

TP is the number of samples of class *POSITIVE* which has been correctly classified (Bankruptcy predicted as abnormal instances). TN is the number of samples of class *NEGATIVE* which has been correctly classified (Non-fault-prone instances predicted as normal instances). FN is the number of samples of class *POSITIVE* which has been misclassified as *NEGATIVE* (Bankruptcy instances predicted as normal instances). FP is the number of samples of class *NEGATIVE* which has been misclassified as *POSITIVE* (Non-fault-prone instances predicted abnormal instances).

Table 4
Financial ratios of the Wieslaw dataset.

Symbol	Financial ratios	Symbol	Financial ratios
s_1	Cash/current liabilities	s_{16}	Sales/receivables
s_2	Cash/total assets	s_{17}	Sales/total assets
s_3	Current assets/current liabilities	s_{18}	Sales/current assets
s_4	Current assets/total assets	s_{19}	(365 receivables)/sales
s_5	Working capital/total assets	s_{20}	Sales/total assets
s_6	Working capital/sales	s_{21}	Liabilities/total income
s_7	Sales/inventory	s_{22}	Current liabilities/total income
s_8	Sales/receivables	s_{23}	Receivables/liabilities
s_9	Net profit/total assets	s_{24}	Net profit/sales
s_{10}	Net profit/current assets	s_{25}	Liabilities/total assets
s_{11}	Net profit/sales	s_{26}	Liabilities/equity
s_{12}	Gross profit/sales	s_{27}	Long term liabilities/equity
s_{13}	Net profit/liabilities	s_{28}	Current liabilities/equity
s_{14}	Net profit/equity	s_{29}	EBIT/total assets
s_{15}	Net profit/(equity + long term liabilities)	s_{30}	Current assets/sales

		Actual	
		Positive	Negative
Predicted	Positive	True Positive TP	False Positive FP
	Negative	False Negative FN	True Negative TN

Fig. 7. The confusion matrix.

Sensitivity, also known as the TP rate, is an index which measures how well a classifier can recognize abnormal records,

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

Specificity measures how well a binary classifications model can recognize normal instances also referred to as the TN rate,

$$\text{Specificity} = \frac{TN}{FP+TN}$$

MCC is utilized to evaluate the performance of the proposed classifier model. *MCC*, which uses all four criteria (TP, TN, FP, FN), may provide a better evaluation of prediction than, for instance, the percentages [49],

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}}$$

4.2.3. Simulation results on the Wieslaw dataset

The average performance of the proposed CLSGMFO-KELM system and other competitors has been compared in Fig. 8. It can be clearly seen that the average values of the four indexes of the proposed method were the highest. In the Wieslaw dataset, the developed CLSGMFO-KELM system obtains the average results of 85% ACC, 86.37% Sensitivity, 86.32% Specificity, and 0.7104 MCC. Regarding the ACC metric, CLSGMFO-KELM obtains the best results, which are much better than results yielded by RF, MFO-KELM, KNN, and SVM, whereas BP, CART, and Adaboost produce inferior results. As for the Sensitivity metric, CLSGMFO-KELM achieves the best results, followed successively by MFO-KELM, RF, KNN, SVM, BP, CART, and Adaboost classifiers. As for the Specificity metric, CLSGMFO-KELM obtains the best results and RF yields similar results, whereas BP yields the worst results. In terms of the MCC metric, CLSGMFO-KELM obtains the best results, whereas BP again produces the worst results.

To further illustrate the advantages of the proposed method, the Wilcoxon rank sign test (García, Fernández, Luengo, & Herrera, 2010; García, Molina, Lozano, & Herrera, 2009) with 5% significance is used to judge whether the improvements achieved by CLSGMFO are significant over the other optimizers. When p-value is greater than 0.05, it can be determined that the improvement is not statistically significant; Otherwise, the improvement is determined to be statistically significant. Experimental results are presented in Table 5 with p values. In order to make the comparison results clearly, a p-value greater than 0.05 is in bold. As shown in the table, p-values for ACC and MCC are all less than 0.05. In terms of the 'p-value', the results obtained by different classifiers are significantly different. Overall, we observed that the results of CLSGMFO are significantly better than those obtained by the well-known CART, BP, Adaboost, SVM and the original MFO-KELM according to four matrixes in dealing with the financial dataset of Wieslaw. The developed CLSGMFO-KELM is superior to KNN and RF on 3 and 2 out of 4 indicators, and similar to them on 1 and 2 out of 4 ones, respectively. In other words, the average performance of the developed CLSGMFO-KELM outperforms that of KNN while the proposed method is insignificantly better on Sensitivity index compared to KNN. Similarly, the average performance of CLSGMFO-KELM model is better than that of RF, but it is not statistically significant in Sensitivity and Specificity metrics under the statistical test.

The CLSGMFO-KELM has the best performance among all these selected algorithms from a statistical point of view. In addition, as can be seen from Fig. 8, CLSGMFO-KELM is superior to MFO-KELM in terms of the four evaluation metrics (ACC, Sensitivity, Specificity, and MCC), indicating that the searching capability of MFO is greatly enhanced by embedding with two mechanisms (GM and CLS). In addition, it also indicates that the two key parameters of the KELM model have a significant influence on classification performance. Based on the comprehensive evaluation of the experimental results above, it can be concluded that CLSGMFO-KELM can be regarded as the most effective model in dealing with the Wieslaw dataset.

4.2.4. Simulation results on the Australian credit dataset

Fig. 9 presents intuitively the comparative results of 8 classifiers in terms of ACC, sensitivity, specificity, and MCC on the Australian dataset. As shown in the table, CLSGMFO-KELM obtains the average results of 87.25% ACC, 86.45% sensitivity, 88.85% specificity and 0.7461 MCC. As can be seen from Fig. 9, the hybrid CLSGMFO-KELM system obtains the highest accuracies in terms of the ACC metric, followed successively by RF, SVM, MFO-KELM, Adaboost, CART, BP, and KNN. According to the Sensitivity metric, RF obtains the best results while KNN still yields the worst results. As for the

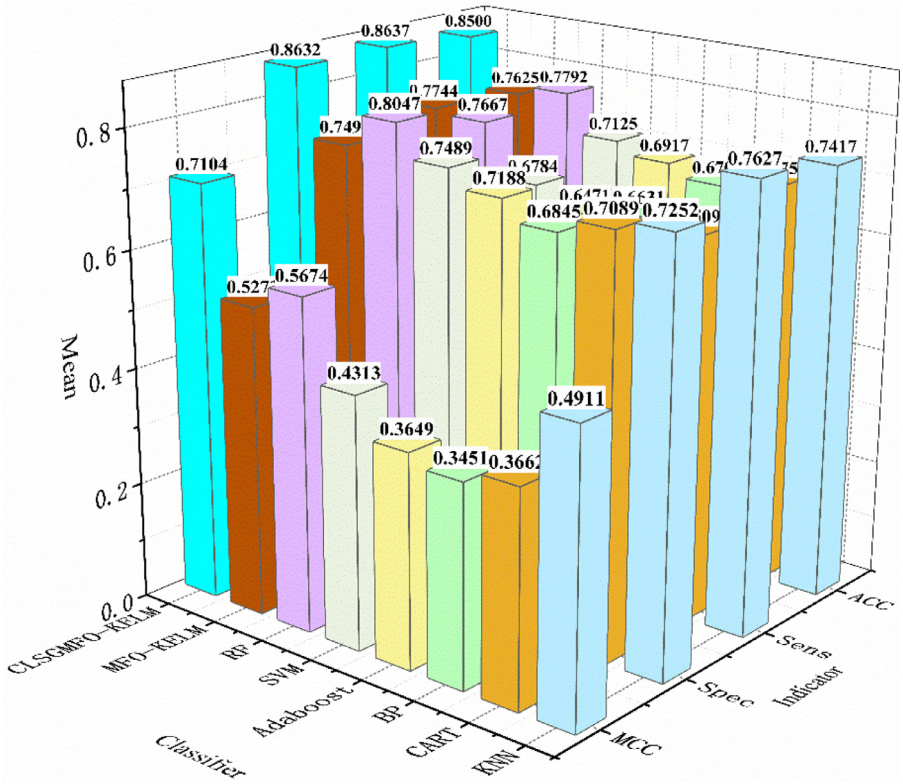

Fig. 8. The average ACC, Sensitivity, Specificity, and MCC obtained by CLSGMFO-KELM, MFO-KELM, RF, SVM, Adaboost, BP, CART and KNN on the Wieslaw dataset.

Table 5
Comparison between CLSGMFO – KELM and the other different classifiers in dealing with the Wieslaw dataset.

Algrithm	Indicator	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	p – value
KNN	ACC	0.708	0.833	0.708	0.750	0.542	0.792	0.750	0.750	0.750	0.833	2.81E – 04
CART		0.792	0.667	0.750	0.667	0.792	0.542	0.750	0.708	0.625	0.583	2.10E – 03
BP		0.625	0.625	0.708	0.625	0.750	0.708	0.583	0.708	0.583	0.792	8.09E – 04
Adaboost		0.625	0.708	0.625	0.625	0.667	0.667	0.833	0.750	0.708	0.708	4.07E – 04
SVM		0.667	0.583	0.792	0.625	0.750	0.750	0.667	0.833	0.750	0.708	3.96E – 03
RF		0.750	0.750	0.792	0.750	0.833	0.750	0.833	0.708	0.833	0.792	2.52E – 02
MFO-KELM		0.708	0.792	0.792	0.792	0.708	0.708	0.708	0.750	0.833	0.833	9.33E – 04
CLSGMFO-KELM		**0.917**	**0.875**	**0.833**	**0.875**	**0.708**	**0.833**	**0.833**	**0.833**	**0.917**	**0.875**	~
KNN	Sensitivity	0.500	0.875	0.583	0.750	0.636	0.727	0.800	1.000	0.846	0.909	**1.13E – 01**
CART		0.700	0.500	0.667	0.615	0.818	0.769	0.727	0.545	0.500	0.667	3.42E – 03
BP		0.667	0.583	0.667	0.545	0.750	0.727	0.667	0.778	0.462	0.786	7.55E – 03
Adaboost		0.583	0.615	0.500	0.333	0.750	0.727	0.889	0.692	0.667	0.714	1.12E – 02
SVM		0.545	0.583	0.800	0.364	0.875	0.786	0.538	0.700	0.900	0.692	1.97E – 02
RF		0.556	0.800	1.000	0.769	1.000	0.692	0.833	0.615	0.818	0.583	**1.15E – 01**
MFO-KELM		0.833	0.900	0.909	0.889	0.546	0.636	0.667	0.769	0.929	0.929	4.52E – 02
CLSGMFO-KELM		**0.875**	**0.923**	**0.800**	**1.000**	**0.875**	**0.769**	**0.769**	**0.750**	**1.000**	**0.875**	~
KNN	Specificity	0.857	0.813	0.833	0.750	0.462	0.846	0.714	0.571	0.636	0.769	5.69E – 03
CART		0.857	0.833	0.800	0.727	0.769	0.273	0.769	0.846	0.714	0.500	4.83E – 02
BP		0.583	0.667	0.733	0.692	0.750	0.692	0.533	0.667	0.727	0.800	4.58E – 03
Adaboost		0.667	0.818	0.714	0.722	0.583	0.615	0.800	0.818	0.750	0.700	8.49E – 04
SVM		0.769	0.583	0.786	0.846	0.688	0.700	0.818	0.929	0.643	0.727	5.26E – 03
RF		0.867	0.667	0.706	0.727	0.765	0.818	0.833	0.818	0.846	1.000	**1.41E – 01**
MFO-KELM		0.733	0.714	0.692	0.733	0.846	0.769	0.750	0.727	0.700	0.833	2.56E – 02
CLSGMFO-KELM		**0.938**	**0.818**	**0.889**	**0.824**	**0.625**	**0.909**	**0.909**	**1.000**	**0.846**	**0.875**	~
KNN	MCC	0.387	0.657	0.430	0.478	0.099	0.580	0.507	0.598	0.497	0.678	5.80E – 04
CART		0.567	0.354	0.467	0.343	0.585	0.048	0.497	0.414	0.218	0.169	1.14E – 03
BP		0.251	0.251	0.393	0.240	0.500	0.418	0.194	0.430	0.194	0.580	4.04E – 04
Adaboost		0.251	0.438	0.218	0.053	0.338	0.343	0.669	0.510	0.418	0.410	4.68E – 04
SVM		0.324	0.167	0.580	0.241	0.530	0.486	0.367	0.657	0.543	0.418	2.45E – 03
RF		0.450	0.467	0.642	0.497	0.698	0.510	0.667	0.438	0.664	0.642	2.83E – 02
MFO-KELM		0.393	0.608	0.608	0.603	0.414	0.410	0.418	0.497	0.657	0.667	6.74E – 04
CLSGMFO-KELM		**0.813**	**0.749**	**0.669**	**0.759**	**0.473**	**0.678**	**0.678**	**0.707**	**0.846**	**0.730**	~

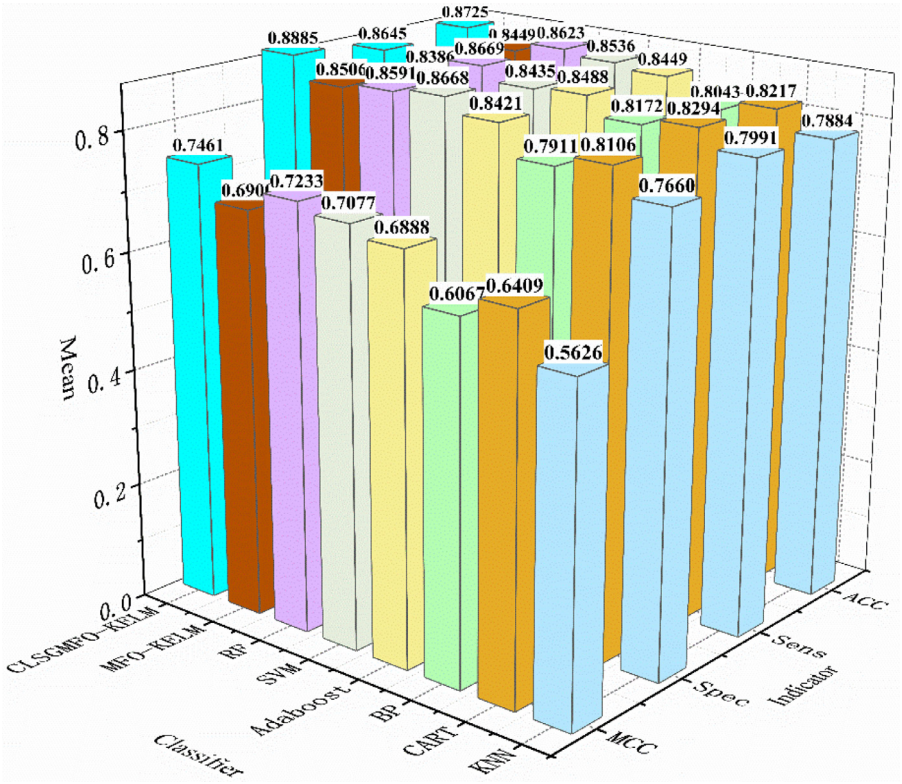

Fig. 9. The average ACC, Sensitivity, Specificity, and MCC obtained by CLSGMFO-KELM, MFO-KELM, RF, SVM, Adaboost, BP, CART, and KNN on the Australian credit dataset.

Specificity metric, CLSGMFO-KELM obtains the best results, which are slightly better than the results yielded by SVM, RF, MFO-KELM, Adaboost, CART, and BP, and again, KNN obtains the worst results. Based on the MCC metric, the CLSGMFO-KELM model is much better than the other classifiers. Table 6 shows the detailed comparative results of the 8 classifiers on the Australian credit dataset. As shown in the table, the developed CLSGMFO-KELM is superior to KNN, CART, and BP on 3 out of 4 indicators, and similar to them on 1 out of 4 ones. Similarly, the developed CLSGMFO-KELM is significantly better than Adaboost on 2 out of 4 indicators, and

Table 6
Comparison between CLSGMFO-KELM and the other different classifiers on the Australian credit dataset.

Algorithm	Indicator	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	p-value
KNN	ACC	0.768	0.812	0.812	0.841	0.739	0.841	0.797	0.681	0.783	0.812	4.32E – 03
CART		0.812	0.797	0.826	0.884	0.841	0.841	0.710	0.855	0.841	0.812	2.00E – 02
BP		0.768	0.826	0.696	0.855	0.826	0.826	0.797	0.841	0.826	0.783	2.66E – 03
Adaboost		0.841	0.884	0.797	0.826	0.855	0.826	0.812	0.913	0.855	0.841	**5.19E – 02**
SVM		0.855	0.870	0.841	0.841	0.812	0.884	0.855	0.884	0.855	0.841	**2.21E – 01**
RF		0.899	0.870	0.899	0.812	0.870	0.899	0.768	0.899	0.841	0.870	**5.82E – 01**
MFO-KELM		0.841	0.797	0.812	0.855	0.884	0.797	0.855	0.855	0.913	0.841	**8.50E – 02**
CLSGMFO-KELM		**0.870**	**0.884**	**0.870**	**0.884**	**0.928**	**0.826**	**0.884**	**0.899**	**0.826**	**0.855**	~
KNN	Sensitivity	0.758	0.826	0.842	0.833	0.737	0.778	0.898	0.676	0.816	0.828	**5.64E – 02**
CART		0.833	0.857	0.791	0.907	0.838	0.868	0.649	0.943	0.765	0.844	**2.64E – 01**
BP		0.700	0.850	0.718	0.947	0.838	0.795	0.841	0.825	0.889	0.769	**1.52E – 01**
Adaboost		0.857	0.884	0.784	0.762	0.917	0.853	0.825	0.897	0.829	0.881	**4.47E – 01**
SVM		0.857	0.884	0.865	0.810	0.861	0.853	0.775	0.897	0.800	0.833	**2.70E – 01**
RF		0.881	0.912	0.861	0.795	0.850	0.907	0.811	0.895	0.886	0.872	**9.24E – 01**
MFO-KELM		0.914	0.756	0.750	0.816	0.919	0.886	0.784	0.842	0.900	0.818	**3.14E – 01**
CLSGMFO-KELM		**0.854**	**0.821**	**0.884**	**0.865**	**0.970**	**0.871**	**0.889**	**0.892**	**0.775**	**0.826**	~
KNN	Specificity	0.778	0.783	0.774	0.846	0.742	0.958	0.550	0.688	0.742	0.800	1.46E – 02
CART		0.778	0.704	0.885	0.846	0.844	0.806	0.781	0.765	0.914	0.784	2.51E – 02
BP		0.862	0.793	0.667	0.742	0.813	0.867	0.720	0.862	0.786	0.800	4.26E – 03
Adaboost		0.824	0.885	0.813	0.926	0.788	0.800	0.793	0.933	0.882	0.778	2.81E – 02
SVM		0.853	0.846	0.813	0.889	0.758	0.914	0.966	0.867	0.912	0.852	**4.18E – 01**
RF		0.926	0.829	0.939	0.833	0.897	0.885	0.719	0.903	0.794	0.867	**3.29E – 01**
MFO-KELM		0.765	0.857	0.897	0.903	0.844	0.640	0.938	0.871	0.931	0.861	**1.41E – 01**
CLSGMFO-KELM		**0.893**	**0.967**	**0.846**	**0.906**	**0.889**	**0.790**	**0.879**	**0.906**	**0.897**	**0.913**	~
KNN	MCC	0.535	0.592	0.618	0.677	0.477	0.702	0.482	0.362	0.560	0.621	3.44E – 03
CART		0.607	0.569	0.656	0.753	0.680	0.677	0.431	0.720	0.688	0.626	1.61E – 02
BP		0.556	0.643	0.383	0.713	0.650	0.656	0.561	0.680	0.659	0.565	2.14E – 03
Adaboost		0.681	0.758	0.595	0.671	0.713	0.653	0.615	0.826	0.712	0.663	3.30E – 02
SVM		0.710	0.725	0.679	0.684	0.623	0.769	0.733	0.764	0.715	0.675	**2.09E – 01**
RF		0.794	0.742	0.801	0.623	0.739	0.786	0.533	0.796	0.683	0.736	**5.38E – 01**
MFO-KELM		0.688	0.602	0.639	0.715	0.767	0.550	0.723	0.710	0.825	0.680	**7.38E – 02**
CLSGMFO-KELM		**0.737**	**0.781**	**0.725**	**0.769**	**0.859**	**0.657**	**0.768**	**0.797**	**0.663**	**0.706**	~

similar to them on 2 out of 4 ones. In addition, the hybrid CLSGMFO-KELM system is similar to RF, MFO-KELM and SVM on 4 out of 4 indexes. Although the proposed method is insignificantly better on four indexes compared to RF, MFO-KELM and SVM, the average performance of the developed CLSGMFO-KELM outperforms them according to the measure metrics used in this work.

Overall, as can be seen from Fig. 9, the average performance of CLSGMFO-KELM outperforms MFO-KELM in terms of the four evaluation metrics, suggesting that CLSGMFO can find more suitable parameters for KELM than the original MFO. Comprehensive evaluation of the results demonstrates that CLSGMFO-KELM is much better than MFO-KELM in dealing with the financial stress prediction problem.

5. Conclusions and future works

In this study, two mechanisms (Gaussian mutation and chaotic local search) are introduced into the original MFO algorithm and have been shown to significantly improve the global and local exploration capabilities of the original MFO. This is because the Gaussian mutation operation can increase the diversity of the moth population during the exploration phase and the chaotic local search mechanism can provide the original MFO with the ability to jump out of local optima. Firstly, the effectiveness of the proposed CLSGMFO method was validated by comparing with a comprehensive set of well-known algorithms, including BA, DA, FA, FPA, GOA, GWO, SCA, SSA, WOA, MVO, FOA, DE, PSO, ABC, BFO, and the original MFO. The comparative results show that CLSGMFO can explore more precise solutions and significantly outperforms all the other competitors. This superior performance of CLSGMFO can also be seen in convergence trends. Secondly, CLSGMFO was compared with multiple advanced MAs including ALCPSO, BLPSO, CLPSO, TVPSO, JADE, JDE and SADE. The experimental results show that CLSGMFO obtains better solutions and has faster convergence speeds than the other optimizers in dealing with the benchmark problems. Finally, CLSGMFO was used to tackle the parameter optimization problem of the KELM model. The proposed CLSGMFO-KELM system was compared with the KNN, CART, BP, Adaboost, SVM, RF and MFO-KELM on two real-world financial prediction problems. The experimental results show that CLSGMFO outperforms the other well-known machine learning models in terms of the four evaluation metrics. Therefore, it can be concluded that the proposed CLSGMFO can be used as a powerful tool in dealing with both classic benchmark problems and real-world applications.

There are many aspects worth exploring in future research. For example, the developed CLSGMFO can be mixed with other well-known MAs to further improve its performance. In addition, a binary version of CLSGMFO can be designed for discrete optimization problems. Moreover, CLSGMFO can also be applied to tackle some special industrial tasks with more large-scale data.

Credit authorship contribution statement

Yueting Xu: Methodology, Software, Validation, Formal analysis, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Huiling Chen:** Conceptualization, Methodology, Project administration, Funding acquisition. **Ali Asghar Heidari:** Writing - original draft, Writing - review & editing. **Jie Luo:** Resources. **Qian Zhang:** Investigation. **Xuehua Zhao:** Supervision, Funding acquisition. **Chengye Li:** Project administration, Funding acquisition.

Acknowledgments

This research is supported by the Science and Technology Plan Project of Wenzhou, China (ZG2017019), Zhejiang Provincial Natural Science Foundation of China (LY17F020012), the Graduate Scientific Research Foundation of Wenzhou University (3162018024), Guangdong Natural Science Foundation (2018A030313339), MOE (Ministry of Education in China) Youth Fund Project of Humanities and Social Sciences (17YJCZH261) and National Natural Science Foundation of China (61471133, 61871475), Medical and Health Technology Projects of Zhejiang Province (2019RC207).

References

Abbassi, R., Abbassi, A., Heidari, A. A., & Mirjalili, S. (2019). An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models. *Energy Conversion and Management, 179*, 362–372.

Alatas, B. (2010). Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications, 37*, 5682–5687.

Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M. J., Ventura, S., Garrell, J. M., et al. (2009). KEEL: A software tool to assess evolutionary algorithms for data mining problems. *Soft Computing, 13*, 307–318.

Aljarah, I., Mafarja, M., Heidari, A. A., Faris, H., & Mirjalili, S. (2020). Multi-verse optimizer: theory, literature review, and application in data clustering. In S. Mirjalili, J. Song Dong, & A. Lewis (Eds.), *Nature-inspired optimizers: Theories, literature reviews and applications* (pp. 123–141). Cham: Springer International Publishing.

Aljarah, I., Mafarja, M., Heidari, A. A., Faris, H., Zhang, Y., & Mirjalili, S. (2018). Asynchronous accelerating multi-leader salp chains for feature selection. *Applied Soft Computing, 71*, 964–979.

Allam, D., Yousri, D. A., & Eteiba, M. B. (2016). Parameters extraction of the three diode model for the multi-crystalline solar cell/module using Moth-Flame Optimization Algorithm. *Energy Conversion and Management, 123*, 535–548.

Apinantanakon, W., & Sunat, K. (2018). OMFO: A new opposition-based moth-flame optimization algorithm for solving unconstrained optimization problems. *Advances in Intelligent Systems and Computing, 566*, 22–31.

Aziz, M. A. E., Ewees, A. A., & Hassanien, A. E. (2017). Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. *Expert Systems with Applications, 83*, 242–256.

Bäck, T., & Schwefel, H.P. (1993). An overview of evolutionary algorithms for parameter optimization. In (pp. 1–23).

Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation, 10*, 646–657.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology, 2*, 1–27.

Chen, H.-L., Yang, B., Wang, G., Liu, J., Xu, X., Wang, S.-J., et al. (2011). A novel bankruptcy prediction model based on an adaptive fuzzy k-nearest neighbor method. *Knowledge-Based Systems, 24*, 1348–1359.

Chen, H., Xu, Y., Wang, M., & Zhao, X. (2019). A balanced whale optimization algorithm for constrained engineering design problems. *Applied Mathematical Modelling, 71*, 45–59.

Chen, W. N., Zhang, J., Lin, Y., Chen, N., Zhan, Z. H., Chung, H. S. H., et al. (2013). Particle swarm optimization with an aging leader and challengers. *IEEE Transactions on Evolutionary Computation, 17*, 241–258.

Chen, X., Tianfield, H., Mei, C., Du, W., & Liu, G. (2017). Biogeography-based learning particle swarm optimization. *Soft Computing, 21*, 7519–7541.

Coelho, L. d. S., & Mariani, V. C. (2008). Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. *Expert Systems with Applications, 34*, 1905–1913.

Deng, W., Xu, J., & Zhao, H. (2019). An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. *IEEE Access, 7*, 20281–20292.

Deng, W., Yao, R., Zhao, H., Yang, X., & Li, G. (2019). A novel intelligent diagnosis method using optimal LS-SVM with improved PSO algorithm. *Soft Computing, 23*, 2445–2462.

Dorigo, M., Birattari, M., & Stutzle, T. (2007). Ant colony optimization. *IEEE Computational Intelligence Magazine, 1*, 28–39.

Elsakaan, A. A., El-Sehiemy, R. A., Kaddah, S. S., & Elsaid, M. I. (2018). An enhanced moth-flame optimizer for solving non-smooth economic dispatch problems with emissions. *Energy, 157*, 1063–1078.

Faris, H., Ala'M, A.-Z., Heidari, A. A., Aljarah, I., Mafarja, M., Hassonah, M. A., et al. (2019). An intelligent system for spam detection and identification of the most relevant features based on evolutionary Random Weight Networks. *Information Fusion, 48*, 67–83.

Faris, H., Mafarja, M. M., Heidari, A. A., Aljarah, I., Ala'M, A.-Z., Mirjalili, S., et al. (2018). An efficient binary Salp Swarm Algorithm with crossover scheme for feature selection problems. *Knowledge-Based Systems, 154*, 43–67.

Faris, H., Mirjalili, S., Aljarah, I., Mafarja, M., & Heidari, A. A. (2020). Salp swarm algorithm: theory, literature review, and application in extreme learning machines. In S. Mirjalili, J. Song Dong, & A. Lewis (Eds.), *Nature-inspired optimizers: Theories, literature reviews and applications* (pp. 185–199). Cham: Springer International Publishing.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences, 180*, 2044–2064.

García, S., Molina, D., Lozano, M., & Herrera, F. (2009). A study on the use of non–parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 Special Session on Real Parameter Optimization. *Journal of Heuristics, 15*, 617–644.

Han, F., Huang, D. E. S., Zhu, Z. H., & Rong, T. H. (2006). The forecast of the postoperative survival time of patients suffered from non-small cell lung cancer based on PCA and extreme learning machine. *International Journal of Neural Systems, 16*, 39–46.

Hassanien, A. E., Gaber, T., Mokhtar, U., & Hefny, H. (2017). An improved moth flame optimization algorithm based on rough sets for tomato diseases detection. *Computers and Electronics in Agriculture, 136*, 86–96.

Heidari, A. A., Aljarah, I., Faris, H., Chen, H., Luo, J., & Mirjalili, S. (2019). An enhanced associative learning-based exploratory whale optimizer for global optimization. *Neural Computing and Applications*. doi:10.1007/s00521-019-04015-0.

Heidari, A. A., Faris, H., Aljarah, I., & Mirjalili, S. (2018). An efficient hybrid multilayer perceptron neural network with grasshopper optimization. *Soft Computing*, 1–18. doi:10.1007/s00500-018-3424-2.

Heidari, A. A., Faris, H., Mirjalili, S., Aljarah, I., & Mafarja, M. (2020). Ant lion optimizer: Theory, literature review, and application in multi-layer perceptron neural networks. In S. Mirjalili, J. Song Dong, & A. Lewis (Eds.), *Nature-inspired optimizers: Theories, literature reviews and applications* (pp. 23–46). Cham: Springer International Publishing.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems, 97*, 849–872.

Heidari, A. A., & Pahlavani, P. (2017). An efficient modified grey wolf optimizer with Lévy flight for optimization tasks. *Applied Soft Computing Journal, 60*, 115–134.

Huang, G. B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 42*, 513–529.

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2004). Extreme learning machine: A new learning scheme of feedforward neural networks. In *IEEE international conference on neural networks – Conference proceedings: 2* (pp. 985–990).

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing, 70*, 489–501.

Jia, D., Zheng, G., & Khurram Khan, M. (2011). An effective memetic differential evolution algorithm based on chaotic local search. *Information Sciences, 181*, 3175–3187.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 39*, 459–471.

Kaya, Y., & Uyar, M. (2013). A hybrid decision support system based on rough set and extreme learning machine for diagnosis of hepatitis disease. *Applied Soft Computing Journal, 13*, 3429–3438.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *IEEE international conference on neural networks – Conference proceedings: Vol. 4* (pp. 1942–1948).

Khalilpourazari, S., & Khalilpourazary, S. (2019). An efficient hybrid algorithm based on Water Cycle and Moth-Flame Optimization algorithms for solving numerical and constrained engineering optimization problems. *Soft Computing, 23*, 1699–1722.

Lee, C. Y., & Yao, X. (2001). Evolutionary algorithms with adaptive lévy mutations. In *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC: 1* (pp. 568–575).

Li, C., Hou, L., Sharma, B. Y., Li, H., Chen, C., Li, Y., et al. (2018). Developing a new intelligent system for the diagnosis of tuberculous pleural effusion. *Computer Methods and Programs in Biomedicine, 153*, 211–225.

Li, C., Li, S., & Liu, Y. (2016). A least squares support vector machine model optimized by moth-flame optimization algorithm for annual power load forecasting. *Applied Intelligence, 45*, 1166–1178.

Li, W. K., Wang, W. L., & Li, L. (2018). Optimization of water resources utilization by multi-objective moth-flame algorithm. *Water Resources Management, 32*, 3303–3316.

Li, Z., Zhou, Y., Zhang, S., & Song, J. (2016). Lévy-flight moth-flame algorithm for function optimization and engineering design problems. *Mathematical Problems in Engineering, 2016*.

Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation, 10*, 281–295.

Liu, Z., Loo, C. K., Masuyama, N., & Pasupa, K. (2018). Recurrent kernel extreme reservoir machine for time series prediction. *IEEE Access, 6*, 19583–19596.

Luo, J., Chen, H., Zhang, Q., Xu, Y., Huang, H., & Zhao, X. (2018). An improved grasshopper optimization algorithm with application to financial stress prediction. *Applied Mathematical Modelling, 64*, 654–668.

Ma, C., Ouyang, J., Chen, H.-L., & Zhao, X.-H. (2014). An efficient diagnosis system for Parkinson's disease using kernel-based extreme learning machine with subtractive clustering features weighting approach. *Computational and Mathematical Methods in Medicine, 2014*, 1–14.

Mafarja, M., Aljarah, I., Heidari, A. A., Faris, H., Fournier-Viger, P., Li, X., et al. (2018). Binary dragonfly optimization for feature selection using time-varying transfer functions. *Knowledge-Based Systems, 161*, 185–204.

Mafarja, M., Heidari, A. A., Faris, H., Mirjalili, S., & Aljarah, I. (2020). Dragonfly algorithm: theory, literature review, and application in feature selection. In S. Mirjalili, J. Song Dong, & A. Lewis (Eds.), *Nature-inspired optimizers: Theories, literature reviews and applications* (pp. 47–67). Cham: Springer International Publishing.

Marques, I., & Graña, M. (2012). Face recognition with lattice independent component analysis and extreme learning machines. *Soft Computing, 16*, 1525–1537.

Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.

Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems, 96*, 120–133.

Mirjalili, S., Aljarah, I., Mafarja, M., Heidari, A. A., & Faris, H. (2020). Grey wolf optimizer: Theory, literature review, and application in computational fluid dynamics problems. In S. Mirjalili, J. Song Dong, & A. Lewis (Eds.), *Nature-inspired optimizers: Theories, literature reviews and applications* (pp. 87–105). Cham: Springer International Publishing.

Mei, Ng Shin, R. , Sulaiman, H. , M., Mustaffa, Z., & Daniyal, H. (2017). Optimal reactive power dispatch solution by loss minimization using moth-flame optimization technique. *Applied Soft Computing Journal, 59*, 210–222.

Pan, W. T. (2012). A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems, 26*, 69–74.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems, 22*, 52–67.

Pietruszkiewicz, W. (2008). Dynamical systems and nonlinear Kalman filtering applied in classification. *2008 7th IEEE international conference on Cybernetic Intelligent Systems, CIS 2008*.

Kumar, Praveen, Reddy, M., & Rajasekhara Babu, M. (2017). A hybrid cluster head selection model for internet of things. *Cluster Computing*, 1–13.

Qin, A. K., & Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In *The 2005 IEEE congress on evolutionary computation, 2005: Vol. 1782* (pp. 1785–1791).

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation, 8*, 240–255.

Salgotra, R., & Singh, U. (2017). Application of mutation operators to flower pollination algorithm. *Expert Systems with Applications, 79*, 112–129.

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery, 1*, 317–328.

Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: theory and application. *Advances in Engineering Software, 105*, 30–47.

Sayed, G. I., & Hassanien, A. E. (2017). Moth-flame swarm optimization with neutrosophic sets for automatic mitosis detection in breast cancer histology images. *Applied Intelligence, 47*, 397–408.

Shen, L., Chen, H., Yu, Z., Kang, W., Zhang, B., Li, H., et al. (2016). Evolving support vector machines using fruit fly optimization for medical data classification. *Knowledge-Based Systems, 96*, 61–75.

Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359.

Trivedi, I. N., Kumar, A., Ranpariya, A. H., & Jangir, P. (2016). Economic Load Dispatch problem with ramp rate limits and prohibited operating zones solve using Levy flight Moth-Flame optimizer. In *2016 International Conference on Energy Efficient Technologies for Sustainability, ICEETS 2016* (pp. 442–447).

Wang, G. G., Deb, S., Gandomi, A. H., & Alavi, A. H. (2016). Opposition-based krill herd algorithm with Cauchy mutation and position clamping. *Neurocomputing, 177*, 147–157.

Wang, M., Chen, H., Li, H., Cai, Z., Zhao, X., Tong, C., et al. (2017a). Grey wolf optimization evolving kernel extreme learning machine: Application to bankruptcy prediction. *Engineering Applications of Artificial Intelligence, 63*, 54–68.

Wang, M., Chen, H., Yang, B., Zhao, X., Hu, L., Cai, Z., et al. (2017b). Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. *Neurocomputing, 267*, 69–84.

Xu, L., Li, Y., Li, K., Beng, G. H., Jiang, Z., Wang, C., et al. (2018). Enhanced moth-flame optimization based on cultural learning and Gaussian mutation. *Journal of Bionic Engineering, 15*, 751–763.

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In *Lecture notes in computer science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 5792* (pp. 169–178). LNCS.

Yang, X. S. (2010). A new metaheuristic Bat-inspired algorithm. *Studies in Computational Intelligence, 284*, 65–74.

Yang, Z., Ce, L., & Lian, L. (2017). Electricity price forecasting by a hybrid model, combining wavelet transform, ARMA and kernel-based extreme learning machine methods. *Applied Energy, 190*, 291–305.

Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation, 3*, 82–102.

Zhang, J., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation, 13*, 945–958.

Zhang, L., Mistry, K., Neoh, S. C., & Lim, C. P. (2016). Intelligent facial emotion recognition using moth-firefly optimization. *Knowledge-Based Systems, 111*, 248–267.

Zhang, Q., Chen, H., Heidari, A. A., Zhao, X., Xu, Y., Wang, P., et al. (2019). Chaos-induced and mutation-driven schemes boosting salp chains-inspired optimizers. *IEEE Access* 1-1. doi:10.1109/ACCESS.2019.2902306.

Zhang, Q., Chen, H., Luo, J., Xu, Y., Wu, C., & Li, C. (2018). Chaos enhanced bacterial foraging optimization for global optimization. *IEEE Access, 6*, 64905–64919.

Zhao, H., Zhao, H., & Guo, S. (2016). Using GM (1, 1) optimized by MFO with rolling mechanism to forecast the electricity consumption of inner mongolia. *Applied Sciences (Switzerland), 6*, 1–18.

Zhao, X., Zhang, X., Cai, Z., Tian, X., Wang, X., Huang, Y., et al. (2019). Chaos enhanced grey wolf optimization wrapped ELM for diagnosis of paraquat-poisoned patients. *Computational Biology and Chemistry, 78*, 481–490.