DOI:10.1587/transfun.2023EAP1103

Publicized:2024/08/21

PAPER

Hierarchical Chaotic Wingsuit Flying Search Algorithm with Balanced Exploitation and Exploration for Optimization

Sicheng LIU†, **Kaiyu WANG**†, **Haichuan YANG**††, **Tao ZHENG**†, **Zhenyu LEI**†, **Meng JIA**†††a), ***Nonmembers***, ***and*** **Shangce GAO**†b), ***Member***

SUMMARY
Wingsuit flying search is a meta-heuristic algorithm that effectively searches for optimal solutions by narrowing down the search space iteratively. However, its performance is affected by the balance between exploration and exploitation. We propose a four-layered hierarchical population structure algorithm, multi-layered chaotic wingsuit flying search (MCWFS), to promote such balance in this paper. The proposed algorithm consists of memory, elite, sub-elite, and population layers. Communication between the memory and elite layers enhances exploration ability while maintaining population diversity. The information flow from the population layer to the elite layer ensures effective exploitation. We evaluate the performance of the proposed MCWFS algorithm by conducting comparative experiments on IEEE Congress on Evolutionary Computation (CEC) benchmark functions. Experimental results prove that MCWFS is superior to the original algorithm in terms of solution quality and search performance. Compared with other representative algorithms, MCWFS obtains more competitive results on composite problems and real-world problems.
key words: *Evolutionary algorithm, Exploration and exploitation, Population structure, Wingsuit flying search*

1. Introduction

In recent decades, evolutionary algorithms (EAs) have emerged as effective approaches for addressing high-dimensional, complex, multi-modal, and challenging optimization problems [1]. The mechanisms of EAs are derived from biological evolution theory and encompass a variety of paradigms, including differential evolution [2] [3], genetic algorithm [4], and particle swarm optimization [5]. EAs have demonstrated impressive performance in solving real-world problems, such as protein structure prediction [6] [7], dynamic evolution event processes [8], and wind farm layout optimization [9]. Consequently, an increasing number of researchers are dedicated to enhancing the performance of EAs.

Wingsuit flying search (WFS) algorithm is a population-based algorithm inspired by wingsuit flying sport [10] to solve global optimization problems. It simulates the pilot's landing point shifting to a lower area as a clearer image is obtained during landing. The process of finding the lowest point of landing is considered to be the search for the optimal solution in the minimum value optimization. The search area is confirmed by generating initial points using the Halton sequence [11] and forming a grid in the search space. The initial points present a grid distribution and are located on grid nodes. The initial points are candidate solutions and promote the grid to shrink towards the current optimal solution. WFS arrives at the global best solution by reaching the maximum number of iterations. Besides population size and maximal iteration number, WFS doesn't have to set any additional parameters. The grid formed by laying points sustains a huge area to explore the search space and find better solutions in promising areas. Many researchers have also conducted further research on WFS. Some researchers are committed to improving its performance [12] [13]. Yang et al. [13] utilized chaos perceptron to diversify the population and explore the search space. But as the search range shrinks, it suffers from convergence stagnation and losing the ability to identify other potential solution areas. Some researchers use WFS to solve real-world problems [14] [15]. Venkatesh et al. [15] proposed a hybrid strategy with joint implementation of WFS and artificial cell swarm optimization for controllable device-based load shifting to manage demand in micro-grids. Therefore, it is worthwhile to in-depthly study the performance of WFS.

Since WFS is population-based and consists of individuals, the structure of the population affects the communication and organization of individuals as well as the effectiveness and efficiency [16]. An organized population structure should have an essential impact on the performance of EAs [17] [18]. The basic and common structure is a panmictic structure, in which every individual has the possibility to interact with each other, and the selection for information communication is random. However, due to its disorganized population structure, information spreads rapidly, leading to the rapid vanishing of population diversity and causing convergence prematurely. Various population structures have been proposed to address such problems, which can be classified into four main groups: distributed, cellular, hierarchical, and other structures. The distributed structure builds upon the panmictic structure by dividing the population into several sub-populations [19]. Using a divide-and-conquer mechanism, the sub-populations iterate independently and communicate with each other through the migration of individuals among the population [20]. This helps maintain population diversity. In 2019, Luo et al. [21] proposed a

†Faculty of Engineering, University of Toyama, Toyama 930-8555, Japan.
††Graduate School of Technology, Industrial and Social Sciences, Tokushima University, Tokushima 770-8506, Japan.
†††School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, Shaanxi 710048, China.
a) E-mail: jiameng112@163.com
b) E-mail: gaosc@eng.u-toyama.ac.jp

distributed multiple population structure, dividing the population into several sub-populations and testing it on dynamic optimization problem benchmarks in IEEE CEC2009. As a result, population diversity increased significantly. The cellular structure [22] organizes the population in a grid and allows individuals to communicate only with adjacent individuals. It includes several population topologies, such as ring topology [23] and Von Neumann topology [17]. Hierarchical structures are hybrid structures [24] that combine two or more other structures hierarchically [25]. The population is often divided into several colonies, with each colony iterating using its own strategy [26]. Wang et al. [27] introduced a three-layered hierarchical gravitational search algorithm (HGSA) with a gravitational constant, which mitigates premature convergence and enhances the search capability of GSA. In 2022, another differential evolution algorithm with a three-layer hierarchical structure [28] was proposed to maintain high population diversity while promoting convergence speed. There are other types of population structures, such as small-world [29] [30], niching technique [31–33], and scale-free structure [34] [35].

WFS has significant ability in fast convergence but lacks the ability to explore new regions. Although CWFS expands the search area of WFS, it still falls into local optima and can't jump out with the search area shrinking in the iterative process. Neither WFS nor CWFS consider enhancing exploration behavior when the grid shrinks. In this paper, we propose a four-layered hierarchical chaotic wingsuit flying search, abbreviated as MCWFS. The hierarchical structure disposes the population into several layers with different targets. We divide the population into population layer, sub-elite layer, elite layer, and memory layer. The communication between each layer improves the ability of MCWFS to increase the likelihood of discovering global optima while avoiding local optima and improving population diversity.

The proposed MCWFS makes contributions to the following three aspects: 1) A four-layered hierarchical structure is firstly proposed to improve the wingsuit flying search algorithm, 2) The system of MCWFS makes it have a significant performance on balancing exploitation and exploration, and 3) The experimental results indicate MCWFS can be investigated on real-world problems.

The rest of this paper is organized as follows: Section 2 introduces previous research. MCWFS is proposed in Section 3. Experimental results are compared and analyzed with other representative algorithms in Section 4. Finally, Section 5 introduces conclusions and our future works.

2. Previous Research

2.1 Wingsuit flying search (WFS)

WFS is a population-based minimum optimization method inspired by wingsuit flying, and the process of WFS can be described as follows:

(I) Initialize N individuals: Initial points, which are candidate solutions, are generated by the Halton sequence [11] in a D-dimensional box-constrained search space, and are described as:

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_{D-1}, \mathbf{x}_D]^T \in \mathbb{R}^N, \mathbf{x}_{min} \leqslant \mathbf{x} \leqslant \mathbf{x}_{max} \quad (1)$$

In each dimension, each point $\mathbf{x}i^{(t)}$ of the current iteration t is located at a node in the grid, where $i \in 1, 2, ..., N$. The lower and upper bounds of the grid's boundary in the coordinate axis are $\mathbf{x}i, min^{(t)}$ and $\mathbf{x}i, max^{(t)}$. The distance between each point is $\Delta\mathbf{x}i^{(t)} = [\Delta\mathbf{x}1^{(t)}, \Delta\mathbf{x}2^{(t)}, ...\Delta\mathbf{x}D^{(t)}]^T$. N_0 is introduced to define the number of nodes and is set as $\sqrt[D]{N}$. The initial discretization of $\Delta\mathbf{x}i^{(t)}$ is described as:

$$\Delta\mathbf{x}_i^{(t)} = \frac{\mathbf{x}_{i,max}^{(t)} - \mathbf{x}_{i,min}^{(t)}}{N_0} \quad (2)$$

(II) Determine point's neighborhood size: In the first iteration $t = 1$, the grid includes a large area of individuals. In the first iteration, each point's neighborhood size is $P^{(t)}(max)$. For $t \geq 2$ iteration, the point is allocated with less neighborhood points $P^{(t)}(i)$. The points' objective function values of the previous iteration are sorted in ascending order. When a point increases to the $N^{(t)}$th, there will be no neighborhood points. The linear dependence of points delivered to the $(t + 1)$th iteration is defined as:

$$P^{(t)}(i) = \lceil P_{max}^{(t)}(1 - \frac{i-1}{N^{(t)} - 1})\rceil \quad (3)$$

$$N^{(t)} = \left\lceil \frac{2N}{P_{max}^{(t)}} \right\rceil \quad (4)$$

$$P_{max}^{(t)} = \left\lceil a^{(t)} \cdot N \right\rceil \quad (5)$$

$$a^{(t)} = 1 - v^{-\frac{t-1}{T-1}} \quad (6)$$

where search sharpness parameter $a^{(t)} \in (0, 1)$, and T describes the number of algorithm iterations. According to flier's velocity, the WFS's parameter is $v > 0$.

(III) Generate neighborhood points: The search range of WFS starts to shrink with iteration number increasing.

$$\Delta\mathbf{x}^{(t)} = (1 - a^{(t)}) \cdot \Delta\mathbf{x}^{(1)}, t \geq 2 \quad (7)$$

Then, the new neighborhood points are generated, and denoted as $y_j\left(\mathbf{x}_i^{(t)}\right)$, $j \in \left\{1, 2, ..., P^{(t)}(i)\right\}$. In order to determine the generated direction, the vector $\mathbf{v}_i^{(t)} = \mathbf{x}_i^{(t)} - \mathbf{x}_1^{(t)}$ is created for each point $\mathbf{x}_i^{(t)}$. And its coordinates update as:

$$S_{k,1}(\mathbf{x}_i^{(t)}) = \{\mathbf{x}_{k,i}^{(t)} - \Delta\mathbf{x}_k^{(t)}, x_{k,i}^{(t)}\}, \text{if } v_{k,i}^{(t)} < 0 \quad (8)$$

$$S_{k,2}(\mathbf{x}_i^{(t)}) = \{\mathbf{x}_{k,i}^{(t)}, \mathbf{x}_{k,i}^{(t)} + \Delta\mathbf{x}_k^{(t)}\}, \text{if } v_{k,i}^{(t)} > 0 \quad (9)$$

$$S_{k,3}(\mathbf{x}_i^{(t)}) = S_{k,1}(\mathbf{x}_i^{(t)}) \bigcup S_{k,2}(\mathbf{x}_i^{(t)}), \text{if } v_{k,i}^{(t)} = 0 \quad (10)$$

where k is a random index from $[1, 2, ..., D]$.

WFS is distinguished from other evolutionary algorithms by its simple operation. In the iteration process, $\Delta\mathbf{x}_i^{(t)}$ determines the grid boundaries, and the search space narrows down as the iteration number increases. This can lead to premature convergence and decreased search ability for the algorithm.

2.2 Wingsuit flying search improved by chaos perceptron

CWFS generates a new matrix of points by chaos perceptron in the external area to explore for identifying various solution areas. CWFS divides the population into n sub-populations, b_i, $i = 1, 2, ...n$. Each sub-population consists of two sub-components, b_i^h and b_i^l. Sub-component b_i^h updates according to the Halton sequence principle [11], and the number of individuals is defined as u_h. The other sub-component b_i^l is generated in the external area by the logistic map [36], proposed by May in 2004. The number of individuals is called u_l. The logistic map principle, u_h, and u_l are described as:

$$l_{t+1} = r \cdot l_t \cdot (1 - l_t) \tag{11}$$

$$u_l = N \cdot p \tag{12}$$

$$u_h = N \cdot (1 - p) \tag{13}$$

where l_t represents the chaotic number in generation t, $l_t \in (0, 1)$. The parameter r is set as 4 in CWFS. The p is the occupation rate of b_i^l among b_i, which is a decimal from 0 to 1, and at the same time, b_i^h occupies $1 - p$ of b_i. In CWFS, p is set as 0.2.

The updating equations of b_i^l and b_i^h are, respectively, following as:

$$l_{f,k} = (2 \cdot L_{f,k} - 1) \cdot \Delta x_{i,k}^{(t)} + x_{g,k} \tag{14}$$

$$h_{e,k} = (2 \cdot H_{e,k} - 1) \cdot \Delta x_{i,k}^{(t)} + x_{g,k} \tag{15}$$

where $l_{f,k}$ and $h_{e,k}$ are the individuals of the matrix b_i^l, $f = 1, 2, 3, ...u_l$, and b_i^h, $e = 1, 2, 3, ...u_h$, respectively. H is a matrix of all values obtained by the principle of the Halton sequence [11], while L is a matrix of all values obtained by the principle of the logistic map [36]. Both sub-populations b_i^h and b_i^l iterate around the parent individual x_g, $g = 1, 2, 3, ..., N^{(t)}$.

The individuals with the minimum objective function value in b_i^l and b_i^h are defined as X_l and X_h, respectively. The search range adjustment parameter $d = 0.1$ is used in CWFS to decrease Δx_{i+1}^t to exploit for the global optimum in the internal grid or increase Δx_{i+1}^t to explore in the external grid generated by the logistic map.

$$\Delta x_{i+1}^{(t)} = (1 - d) \cdot \Delta x_i^{(t)}, f(X_h) < f(X_l) \tag{16}$$

$$\Delta x_{i+1}^{(t)} = (1 + d) \cdot \Delta x_i^{(t)}, f(X_h) > f(X_l) \tag{17}$$

3. Four-layered hierarchical Chaotic Wingsuit Flying Search Algorithm

3.1 Motivation

In CWFS, it is notable to divide the population into two sub-components with different iteration methods. CWFS delays the convergence velocity and avoids prematurely falling into local optima. As a result, CWFS efficiently exploits the search area among $\Delta \mathbf{x}^{(t)}$, and the local search ability is enhanced. However, CWFS always iterates and generates new offspring around $\Delta \mathbf{x}^{(t)}$, limiting its ability to find a better area for an improved solution. Consequently, CWFS is excessively inclined towards exploitation, leading to premature convergence and falling into local optima. CWFS is weak in global search and exploration behavior, so that it cannot balance the search ability between exploitation and exploration.

To strike a balance between the two aspects and find a solution closer to the optimal value, we introduce a hierarchical population structure to improve CWFS. According to previous research, population topology guides the individual's evolution direction, and improving communication between individuals has proven to be significant. In this paper, we innovatively introduce a four-layered hierarchical chaotic wingsuit flying search, abbreviated as MCWFS.

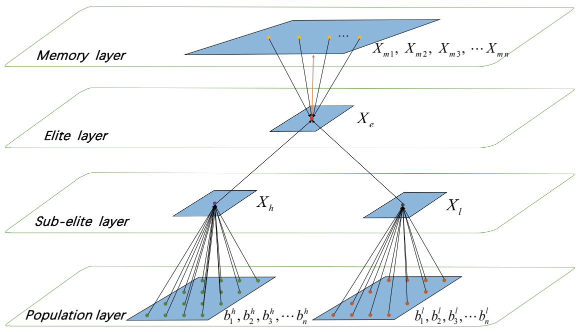
Memory layer
$X_{m1}, X_{m2}, X_{m3}, \cdots X_{mn}$
Elite layer
X_e
Sub-elite layer
X_h
X_l
Population layer
$b_1^h, b_2^h, b_3^h, \cdots b_n^h$
$b_1^l, b_2^l, b_3^l, \cdots b_n^l$

Fig. 1 The four-layered hierarchical structure of MCWFS.

3.2 Proposed MCWFS

In MCWFS, a four-layered hierarchical population structure is applied to promote information interaction and balance between exploration and exploitation. These four layers are named the population layer, sub-elite layer, elite layer, and memory layer. The four-layered hierarchical structure is shown in Fig. 1.

Population layer: This layer contains all populations representing the current iteration's individuals. It is regarded as the fundamental layer. Individuals are provided with an integrated search space for evaluation, mutation, and selection. As the iteration increases, the population layer's individuals are guided by individuals in the sub-elite layer, which describes the composition of a function assembled by numerous evolved individuals. We divide the population into an inner search area b_i^h and an outer search area b_i^l based on CWFS. Individuals in the inner search area follow the Halton sequence principle [11] for updates, while in the outer search area, the logistic map [36] generates new offspring.

Sub-elite layer: To guide the iteration of ordinary individuals, we showcase current best individuals in this layer. The current best individual generated by the Halton sequence principle is defined as X_h. Meanwhile, the current best individual generated by the logistic map is known as X_l. The sub-elite layer accelerates convergence speed towards X_h and X_l, and the local search in these areas is strengthened with each iteration. As a result, the population layer achieves complete individuals' update velocity under the guidance of the sub-elite layer. Information exchange flows one way, from the population layer to the sub-elite layer. Cooperation between these two layers enhances the exploitation behavior of the MCWFS algorithm and increases the orderliness of information interaction between individuals in the population.

Elite layer: Since some potential individuals are discovered in the sub-elite layer, the global optimal individual X_e is generated by comparing values of X_h and X_l. The flow of information between individuals is unidirectional, from the sub-elite layer to the elite layer.

$$X_e = \begin{cases} X_h, & \text{if } X_h \le X_l \\ X_l, & \text{otherwise} \end{cases} \tag{18}$$

The elite layer provides the MCWFS algorithm with two advantages: it keeps the global optimal individual attracting current best individuals from falling into local optima, and it accumulates the speed of individuals' movement, reducing the distance between individuals and the global optimum.

Memory layer: To find different areas with a variety of solutions in the search space, we establish this layer to generate new trial solutions based on the elite layer. The individuals in this layer are initialized and set as X_m, $m = \{1, 2, 3, ...N\}$. Communication between the elite and memory layers is bidirectional. On one hand, the elite layer is guided by the memory layer to update towards the increased probability of finding solutions with better objective function values. Two random individuals X_{mr2} and X_{mr1} from the memory layer are selected to generate a temporary individual $X_{e'}$. The process is defined as:

$$X_{e'} = X_e + p_r \cdot (X_{mr2} - X_{mr1}) \tag{19}$$

$$X_e(t) = \begin{cases} X_{e'}(t), & \text{if } f(X_{e'}(t)) \le f(X_e(t)) \\ X_e(t), & \text{otherwise} \end{cases} \tag{20}$$

where t denotes the algorithm's iteration numbers. Through experimentation, p_r is a fixed value set as $p_r = 0.5$. The fitness function is recorded as f.

On the other hand, the memory layer also records previous global best individuals and evolves with the iteration of the elite layer. The update process for X_m, based on greedy selection, is described as:

$$X_{mbest} = \begin{cases} X_{e'}(t), & \text{if } X_{e'}(t) \le X_{mbest} \\ X_{mbest}, & \text{otherwise} \end{cases} \tag{21}$$

where X_{mbest} is the individual with best objective function value in the memory layer.

Thus, information provided by the elite layer promotes the iteration of individuals in the memory layer. Communication between these two layers increases the probability of discovering underlying areas of solutions and avoids the cost of decline in convergence. As a result, the exploration behavior of the MCWFS algorithm is significantly improved.

3.3 Framework of MCWFS

The pseudocode for MCWFS is presented in Algorithm 1. The framework of MCWFS is divided into three main parts: 1) Initialization; 2) Information flows from the population layer to the elite layer; and 3) Information exchanges between the memory layer and the elite layer. First, initialize the parameters according to lines 3-10. The fixed value p_r is set as 0.5, and the population X is generated through $N, N_0, \triangle x$. The initial discretization step $\triangle x$ is defined in line 7. Then, lines 13-27 show how MCWFS explores the search space efficiently. The maximal number of neighborhood points P_{max} is calculated in line 13. Referred to as flier's velocity, the algorithm parameter $v > 0$ (line 15). b_i^l and b_i^h are generated around x_g by the logistic map and Halton sequence, respectively (lines 17-18). The smallest individuals X_h and X_l are generated on the sub-elite layer (lines 19-20). The global optimum X_e is also determined by comparing $f(X_h)$ and $f(X_l)$ (lines 23-27). Finally, a temporary individual $X_{e'}$ is generated by two random individuals in the memory layer to enhance exploitation behavior and avoid falling into local optima (lines 29-30). The best individuals of the elite layer and memory layer are updated by Eq. (20) and Eq. (21), respectively (lines 31-32).

4. Experiment

The results of MCWFS and other competing algorithms are evaluated by using MATLAB R2019b on a Windows 10 operating system, Intel i7-9700 with 8GB RAM. Their performance is tested on the IEEE CEC2017 and 2011 benchmark function suites under the same experimental environment. IEEE CEC competitions includes variant benchmark test sets to measure the performance of algorithms [37]. IEEE CEC2017 is a single-objective benchmark to verify the work-ability of the algorithms [38]. IEEE CEC2011 is a benchmark set to distinguish the effectiveness of the algorithms on real-world numerical optimization problems [39]. Its detailed description is shown in Supplementary File [40]. IEEE CEC2017 consists of 30 functions, including uni-modal functions (F1-F3), multi-modal functions (F4-F10), hybrid functions (F11-F20), and composite functions (F21-F30) in dimensions (i.e., D) 30, 50, and 100. IEEE CEC2011 (G1-G22) is a real-world problem benchmark function suite with 51 independent runs on 22 functions. We have chosen seven representative algorithms to validate the effectiveness of MCWFS. Initially, we selected the original algorithms WFS [10] and CWFS [13] to illustrate the advancement achieved by MCWFS. Additionally, we included CCWFSSE [12] which is a improved variant of WFS to further establish the performance of MCWFS. Simultaneously, we have also chosen a hierarchical population structure algorithm HGSA [27], a variant of the classical algorithm GLPSO [41], the IEEE CEC2020 champion algorithm

Algorithm 1: Pseudocode of MCWFS

```
1  begin
2      /*Initialization */
3      $p = 0.5$
4      $v = rand(10, 100)$
5      $N = N^* - 2$
6      $N_0 = Ceil(N^{(1/n)})$
7      $\triangle x = (x_{max} - x_{min}) * N_0$
8      Generate and evaluate population $X$
9      while Terminal Condition do
10         /*Information flows from population layer to elite layer*/
11         $P_{max} = Ceil(\alpha * N)$
12         $N = Ceil(2 * N/P_{max})$
13         $\alpha = 1 - v^{-(t-1)/(T-1)}$
14         Sort $X$, and generate $N_{th}$ point
15         $b_i^l \leftarrow$ Generating individuals by logistic map around $x_g$ point using $\triangle x$
16         $b_i^h \leftarrow$ Generating individuals by Halton sequence around $x_g$ point using $\triangle x$
17         $f(b_i^l) = evaluate(b_i^l); f(b_i^h) = evaluate(b_i^h); f(X_l) = min[f(b_i^l)]; f(X_h) = min[f(b_i^h)]$
18         if $f(X_l) < f(X_h)$ then
19             $f(X_e) \leftarrow f(X_l)$
20         else
21             $f(X_e) \leftarrow f(X_h)$
22         end
23         /*Information exchanges between memory layer and elite layer*/
24         Initialize memory layer's individuals $X_m$ and select two random individuals $X_{mr2}$ and $X_{mr1}$.
25         $X_{e'} = X_e + p_r \cdot (X_{mr2} - X_{mr1})$
26         Update elite layer's best individual $X_e(t)$ using (20)
27         Update memory layer's best individual $X_{mbest}$ using (21)
28     end
29 end
```

IMODE [42], and a state-of-the-art algorithm SIS [43], to underscore the competitiveness of MCWFS within the domain of EAs. The experimental settings of all algorithms are obtained by its relevant paper and are listed in Table 6. Population size N is 101 for MCWFS, and the maximal number of function evaluations (MFES) of all algorithms is set as $10^4 \cdot D$ for a fair comparison. We calculate the absolute error and the optimal results for IEEE CEC2017 and CEC2011, respectively. Evaluation criteria are introduced as follows:

(1) Non-parametric statistical test: The statistical results of the Wilcoxon rank-sum test distinguish significant differences between two algorithms with a significant level of 0.05. " + " (W) ," $\approx$ " (T), and " − " (L) mean MCWFS is significantly better, draw, and worse than its competitors, respectively.

(2) Convergence curve graph: The trend of convergence over iteration can be visually represented on this graph. The horizontal axis represents MFES, while the vertical axis indicates the average error value.

(3) Box-whisker plot diagram: This graph infers information about solutions' distribution and dispersion. The values of maximum, first quartile, median, third quartile, and minimum are respectively denoted by the lines of the upper block, upper blue, red, lower blue and lower block. The red " + " represents extreme values.

Table 1 Experimental result comparisons of MCWFS and other algorithms on IEEE CEC2017.

CEC2017	D=30	MCWFS	CWFS	WFS	CCWFSSE
		+/ ≈ /−	14/15/1	30/0/0	29/1/0
		HGSA	GLPSO	IMODE	SIS
		17/4/9	25/1/4	14/0/16	22/2/6
	D=50	MCWFS	CWFS	WFS	CCWFSSE
		+/ ≈ /−	15/15/0	30/0/0	30/0/0
		HGSA	GLPSO	IMODE	SIS
		19/1/10	25/1/4	19/1/10	25/2/3
	D=100	MCWFS	CWFS	WFS	CCWFSSE
		+/ ≈ /−	13/17/0	27/1/2	30/0/0
		HGSA	GLPSO	IMODE	SIS
		18/2/10	29/0/1	18/0/12	27/1/2

Table 2 Experimental result comparisons of MCWFS and other algorithms on IEEE CEC2011.

CEC2011	MCWFS	CWFS	WFS	CCWFSSE
	+/ ≈ /−	10/12/0	20/2/0	17/1/4
	HGSA	GLPSO	IMODE	SIS
	13/4/5	18/1/3	21/0/1	18/4/0

4.1 Comparison with WFS Variants

MCWFS is adopted to verify its performance among WFS variants by comparing it with WFS [10], CWFS [13], and CCWFSSE [12]. Table 1 presents the results on IEEE CEC2017 with 30, 50, and 100 dimensions, while the results on IEEE CEC2011 are summarized in Table 2. The detailed experiment results are given in Supplementary File [40]. To directly show performance of our algorithm, its results are summarized in Table 3. From the W/T/L comparison, it is evident that MCWFS outperforms other WFS variants across different dimensions. Building on the original algorithm, MCWFS retains the advantages of CWFS and improves performance impressively. Figs. 2-7 illustrate the convergence graphs and box-and-whisker diagrams of some typical functions across various dimensions. From convergence graphs, WFS and CCWFSSE converge and stagnate at local optima quickly. However, MCWFS and CWFS continue to converge. In terms of convergence speed, MCWFS outperforms CWFS and exhibits a lower average error value. The box-and-whisker diagrams show that MCWFS obtains the minimum mean value, which indicates MCWFS has a better ability to find the global optimum than other WFS variants. Additionally, the distribution and quality of solutions are more stable. The results on IEEE CEC2011 suggest that MCWFS has great potential for solving real-world problems. Therefore, MCWFS is a superior variant of WFS.

4.2 Comparison with Competitive Algorithms

MCWFS is not only compared with WFS variants but also with other hierarchical algorithms (HGSA) [27], a variant of classical algorithm (GLPSO) [41], the first-ranked algorithm of IEEE CEC2020 (IMODE) [42], and a state-of-the-art algorithm (SIS) [43] published in 2022. The experi-

Table 3 Experimental results of MCWFS on IEEE CEC2017 and CEC2011.

	CEC2017 *D*=30		CEC2017 *D*=50		CEC2017 *D*=100			CEC2011	
	Mean	Std	Mean	Std	Mean	Std		Mean	Std
F1	4.276.E+03	4.200.E+03	7.205.E+03	6.452.E+03	1.631.E+05	7.890.E+04	G1	1.773.E+01	4.551.E+00
F2	1.016.E+06	4.190.E+06	1.042.E+13	5.386.E+13	1.000.E+30	2.843.E+14	G2	-1.907.E+01	3.378.E+00
F3	2.336.E-01	2.783.E-01	6.751.E+00	2.415.E+00	1.513.E+03	7.281.E+02	G3	1.151.E-05	2.249.E-14
F4	8.043.E+01	2.464.E+01	1.223.E+02	4.548.E+01	2.699.E+02	4.843.E+01	G4	1.692.E+01	2.841.E+00
F5	6.645.E+01	1.702.E+01	1.359.E+02	2.768.E+01	4.262.E+02	5.135.E+01	G5	-3.382.E+01	1.767.E+00
F6	3.159.E+00	2.555.E+00	1.002.E+01	4.584.E+00	3.144.E+01	5.729.E+00	G6	-2.422.E+01	3.023.E+00
F7	9.825.E+01	1.457.E+01	2.039.E+02	3.239.E+01	6.194.E+02	5.643.E+01	G7	6.830.E-01	1.007.E-01
F8	6.307.E+01	1.348.E+01	1.313.E+02	2.112.E+01	4.228.E+02	4.756.E+01	G8	2.272.E+02	9.827.E+00
F9	9.448.E+00	9.583.E+00	9.700.E+02	9.093.E+02	1.450.E+04	3.088.E+03	G9	1.574.E+05	5.913.E+04
F10	2.371.E+03	4.400.E+02	4.212.E+03	6.525.E+02	1.088.E+04	1.124.E+03	G10	-1.913.E+01	2.427.E+00
F11	1.009.E+02	3.197.E+01	1.777.E+02	4.343.E+01	1.124.E+03	1.390.E+02	G11	5.188.E+04	5.637.E+02
F12	1.786.E+05	2.096.E+05	1.825.E+06	1.263.E+06	1.481.E+07	6.317.E+06	G12	2.248.E+07	4.894.E+05
F13	2.015.E+04	1.241.E+04	3.598.E+04	1.595.E+04	4.720.E+04	1.576.E+04	G13	1.548.E+04	2.185.E+01
F14	1.967.E+02	4.116.E+01	6.352.E+02	3.988.E+02	4.816.E+04	2.599.E+04	G14	1.907.E+04	1.206.E+02
F15	6.704.E+03	5.253.E+03	1.116.E+04	6.935.E+03	3.085.E+04	1.007.E+04	G15	3.308.E+04	9.287.E+01
F16	4.530.E+02	1.733.E+02	7.773.E+02	1.845.E+02	2.189.E+03	3.864.E+02	G16	1.336.E+05	2.883.E+03
F17	1.459.E+02	7.120.E+01	6.670.E+02	1.544.E+02	1.688.E+03	3.764.E+02	G17	1.920.E+06	1.420.E+04
F18	2.428.E+04	1.406.E+04	5.694.E+04	2.707.E+04	1.675.E+05	6.138.E+04	G18	9.434.E+05	2.552.E+03
F19	7.107.E+03	1.163.E+04	1.878.E+04	1.938.E+04	1.314.E+05	9.160.E+04	G19	9.773.E+05	3.055.E+04
F20	2.436.E+02	9.637.E+01	5.289.E+02	1.657.E+02	1.646.E+03	3.334.E+02	G20	9.436.E+05	2.544.E+03
F21	2.597.E+02	1.314.E+01	3.270.E+02	2.243.E+01	6.649.E+02	5.354.E+01	G21	1.531.E+01	2.837.E+00
F22	1.000.E+02	1.525.E-02	3.798.E+03	1.929.E+03	1.254.E+04	1.249.E+03	G22	2.010.E+01	3.349.E+00
F23	4.157.E+02	1.555.E+01	5.698.E+02	3.489.E+01	1.055.E+03	7.272.E+01			
F24	4.735.E+02	1.452.E+01	6.266.E+02	2.781.E+01	1.378.E+03	7.467.E+01			
F25	3.873.E+02	2.454.E+00	5.056.E+02	2.809.E+01	7.989.E+02	5.564.E+01			
F26	1.476.E+03	4.442.E+02	2.619.E+03	4.509.E+02	7.872.E+03	6.258.E+02			
F27	5.180.E+02	1.354.E+01	6.206.E+02	5.209.E+01	7.759.E+02	6.236.E+01			
F28	3.614.E+02	4.767.E+01	4.695.E+02	1.786.E+01	6.112.E+02	4.052.E+01			
F29	5.965.E+02	7.869.E+01	9.624.E+02	2.068.E+02	3.097.E+03	4.447.E+02			
F30	1.346.E+05	1.151.E+05	1.219.E+07	3.024.E+06	4.538.E+06	2.110.E+06			

Table 4 Running time of MCWFS and other algorithms on IEEE CEC2017.

Algorithm	MCWFS	CWFS	WFS	CCWFSSE	HGSA	GLPSO	IMODE	SIS
Running time	102.17	187.40	301.50	61.16	403.51	1022.36	89.62	92.75

Table 5 Friedman test ranking of MCWFS.

MCWFS	p_r=0.01	p_r=0.1	p_r=0.2	p_r=0.3	p_r=0.4	p_r=0.5	p_r=0.6	p_r=0.7	p_r=0.8	p_r=0.9	p_r=1
Score	8.4000	7.1667	6.7000	5.4333	5.1333	4.7667	5.1333	5.1000	5.8667	5.4333	6.8667
Ranking	10	9	7	5	3	1	3	2	6	5	8

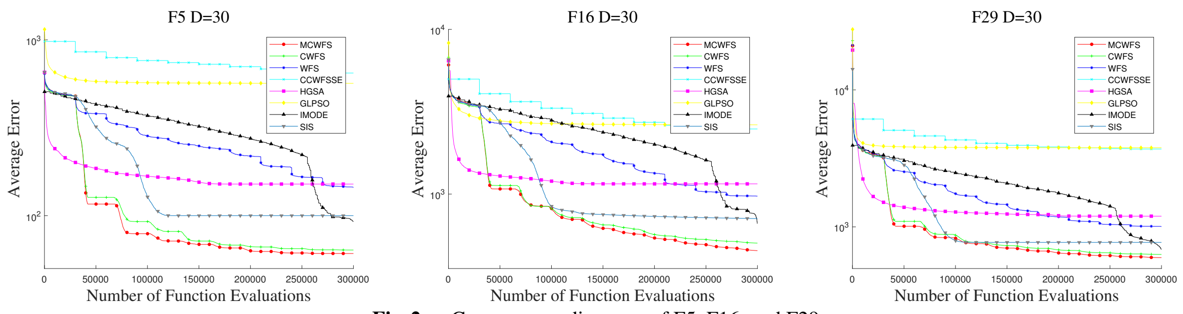

Fig. 2 Convergence diagrams of F5, F16, and F29.

ment results are detailly showed in Supplementary File [40]. From the W/T/L comparison, MCWFS is significantly better than GLPSO, and SIS. Based on the experimental data, the performance of MCWFS on IEEE CEC2017 is highly competitive with that of HGSA, and outperforms in addressing more complex problems. Furthermore, MCWFS exhibits superior performance over HGSA in real-world problems, particularly demonstrating significant advantages over HGSA in addressing the economic load dispatch problems on IEEE CEC2011 [44]. The convergence graphs reveal that although HGSA has a higher convergence speed than MCWFS, it stagnates and falls into local optima quickly, resulting in a worse average error value than MCWFS. Comparisons with IMODE indicate that MCWFS performs better on higher dimensions but still achieves competitive results with IMODE on 30 dimensions. Moreover, results on IEEE CEC2011 reveal that MCWFS is more suitable than IMODE for solving real-world problems. Thus, MCWFS is

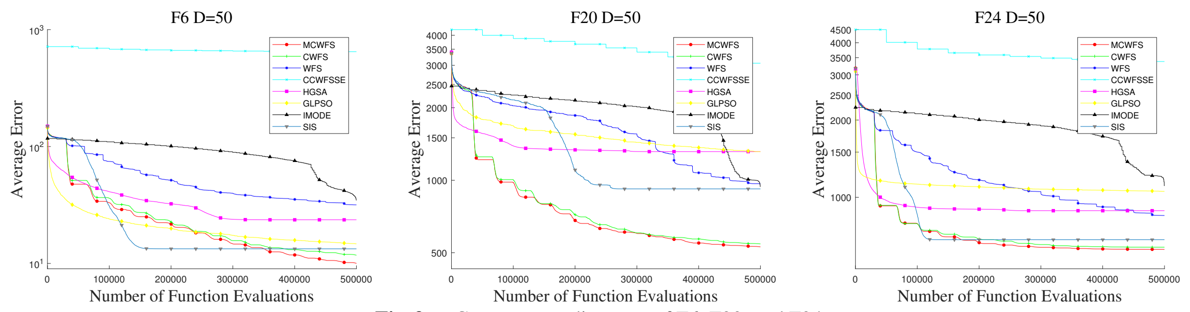

Fig. 3 Convergence diagrams of F6, F20, and F24.

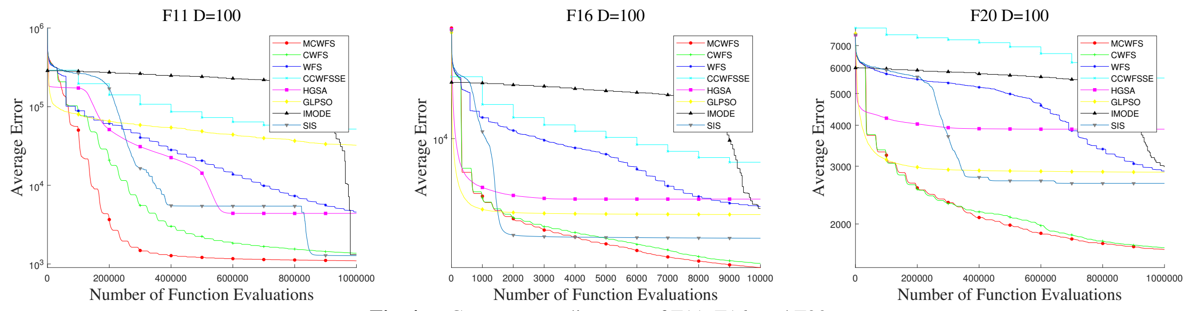

Fig. 4 Convergence diagrams of F11, F16, and F20.

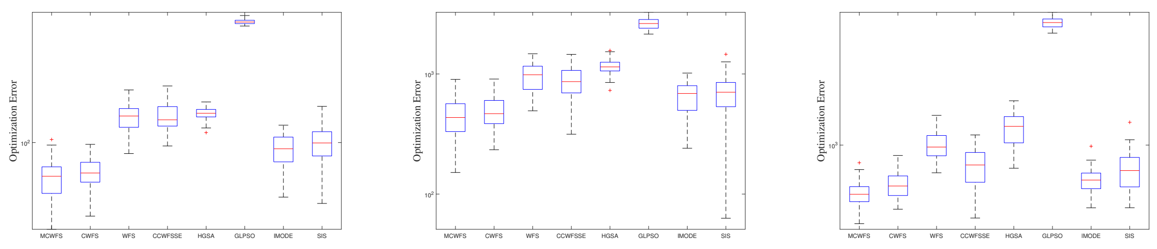

Fig. 5 Box-and-whisker plot diagrams of F5, F16, and F29.

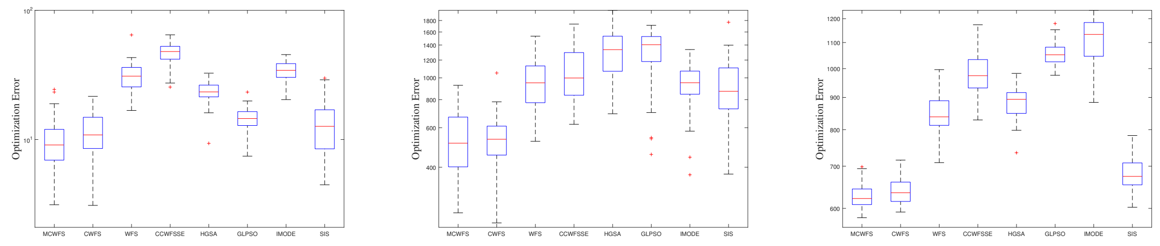

Fig. 6 Box-and-whisker plot diagrams of F6, F20, and F24.

Table 6 Parameter settings of MCWFS and other algorithms.

Algorithm	Parameters
MCWFS	$N = 101$, $p = 0.2$, $p_r = 0.5$
CWFS	$N = 101$, $p = 0.2$
WFS	$N = 100$
CCWFSSE	$N = 100$, $v \sim \mathrm{U}(10, 100)$
HGSA	$L = 100, G_{(0)} = 100, w_1(t) = 1 - t^6/T^6$, $w_2(t) = 1 - t^6/T^6, K \in [1, 2]$
GLPSO	$\omega = 0.9 \sim 0.4$, $c_1 = c_2 = 2.0$
IMODE	$N^{init} = 6 \times D^2, N^{min} = 4, \psi = 20, p = 0.1 \times D$
SIS	$N = 100, b \in (0, 0.1]$

not only a superior WFS variant but also a competitive algorithm in the series of EAs. Besides, we analyze the running time of all algorithms in Table 4, where the test is adjusted to run once on 30 IEEE CEC2017 functions with 30 dimensions. It is observed that MCWFS is competitive. It is better than almost algorithms.

4.3 Population Diversity

To verify the features of MCWFS and validate its ability to balance exploration and exploitation, the population diversity is introduced as follows:

$$Div(x) = \frac{1}{N}\sum_{i=1}^{N} \|x_i - \overline{x}\| / max_{1\leq i,j\leq N} \left\|x_i - x_j\right\| \tag{22}$$

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{23}$$

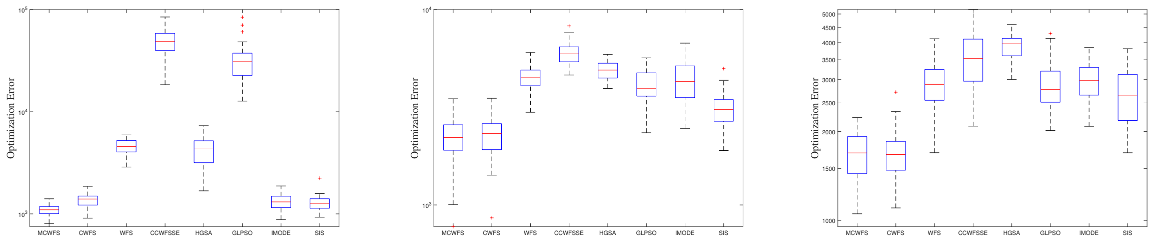

Fig. 7 Box-and-whisker plot diagrams of F11, F16, and F20.

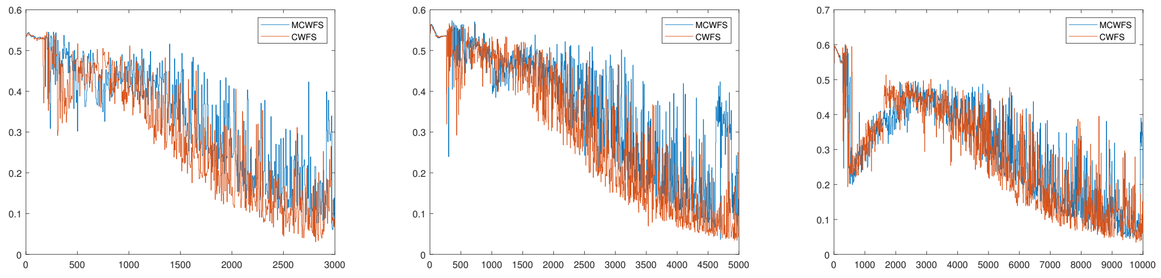

Fig. 8 Population diversity diagrams of (a) F5 D=30, (b) F17 D=50, and (c) F23 D=100.

where N represents the population size, while $\overline{x}$ is the average point. High population diversity causes the algorithm to prefer to explore, whereas low population diversity leads to focus on exploitation [1]. Three kinds of functions, including uni-modal functions, multi-modal functions, and hybrid functions, are selected to analyze the population diversity of MCWFS with 30, 50, and 100 dimensions, as shown in Fig. 8. Both MCWFS's and CWFS's population diversities decline rapidly at the beginning because the information flows from the population layer to the elite layer and the exploitation behavior is emphasized. However, the population diversity of CWFS exhibits an irreversible downward trend as the number of iterations increases, ultimately falling into a local optimum. In contrast, under the cooperation of the memory layer and elite layer, MCWFS has the ability to maintain population diversity at a high level, especially showing an upward trend towards the end of the iteration.

4.4 Parameters Sensitivity Analysis

In MCWFS, we set a new parameter $p_r \in (0, 1]$ to adjust information feedback from the memory layer to the elite layer. We test $p_r = \{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ on 30 benchmark functions in CEC2017. The variance is measured using the Friedman test, the ranking score and final ranking of parameters are given in Table 5. The parameter p_r demonstrates the variation range of the temporary individual $X_{e'}$. From results of the Friedman test, we know that the performance of MCWFS lacks of competitive results when p_r is close to 0 or 1. The top ranking among all parameters is $p_r = 0.5$. It shows that the information from memory layer to elite layer plays an important role in improving WFS. If the temporary individual is generated around X_e, MCWFS can't find another potential area and jump out from local optimum. The exploration behavior is not enhanced. On the other hand, if we generate $X_{e'}$ too far away from X_e, excessive exploration makes the algorithm lose the ability to explore, resulting in MWFS not being able to converge to the optimal. Therefore, $p_r = 0.5$ is the optimal parameter obtained from the experiment, which not only improves the exploration ability of the algorithm but also maintains the characteristic of fast convergence.

5. Conclusions

In this paper, a four-layered hierarchical chaotic wingsuit flying search algorithm is proposed. The information exchange between the memory layer and the elite layer promotes exploration behavior, while the information flow from the population layer to the elite layer enhances the ability of exploitation. In general, the four-layered hierarchical structure proposed significantly improves MCWFS's performance in balancing exploration and exploitation. MCWFS is verified for its superiority by comparing it with seven representative algorithms on 30 IEEE CEC2017 benchmark functions. Additionally, its performance on real-world optimization is confirmed on 22 IEEE CEC2011 benchmark functions. In the future, MCWFS is worth investigating for its potential applications and improvements in real-world problems, such as structural optimization [45], solar parameter estimation problem [46], and wind farm layout optimization problem [47].

Acknowledgments

This research was partially supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP22H03643, Japan Science and Technology Agency (JST) Support for Pioneering Research Initiated by the Next Generation (SPRING) under Grant JPMJSP2145, and JST through the Establishment of University Fellowships towards the Creation of Science Technology Innovation under

Grant JPMJFS2115, supported by the Key Research and Development Project of Shaanxi Province (No. 2023-YBGY-222); Beilin District Science and Technology Project (No. GX2246).

References

[1] K. Wang, Y. Wang, S. Tao, Z. Cai, Z. Lei, and S. Gao, "Spherical search algorithm with adaptive population control for global continuous optimization problems," Applied Soft Computing, vol.132, p.109845, 2023.

[2] K.R. Opara and J. Arabas, "Differential evolution: A survey of theoretical analyses," Swarm and Evolutionary Computation, vol.44, pp.546–558, 2019.

[3] K. Wang, S. Gao, M. Zhou, Z.H. Zhan, and J. Cheng, "Fractional order differential evolution," IEEE Transactions on Evolutionary Computation, 2024. doi: 10.1109/TEVC.2024.3382047.

[4] M. Squires, X. Tao, S. Elangovan, R. Gururajan, X. Zhou, and U.R. Acharya, "A novel genetic algorithm based system for the scheduling of medical treatments," Expert Systems with Applications, vol.195, p.116464, 2022.

[5] R. Kuo and S.S. Li, "Applying particle swarm optimization algorithm-based collaborative filtering recommender system considering rating and review," Applied Soft Computing, p.110038, 2023.

[6] Y. Zhang, S. Gao, P. Cai, Z. Lei, and Y. Wang, "Information entropy-based differential evolution with extremely randomized trees and lightgbm for protein structural class prediction," Applied Soft Computing, vol.136, p.110064, 2023.

[7] Z. Lei, S. Gao, Z. Zhang, M. Zhou, and J. Cheng, "MO4: A many-objective evolutionary algorithm for protein structure prediction," IEEE Transactions on Evolutionary Computation, vol.26, no.3, pp.417–430, 2021.

[8] G. Yuan, J. Cheng, M. Zhou, S. Cheng, S. Gao, C. Jiang, and A. Abusorrah, "A dynamic evolution method for autonomous vehicle groups in an urban scene," IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022.

[9] Z. Lei, S. Gao, Y. Wang, Y. Yu, and L. Guo, "An adaptive replacement strategy-incorporated particle swarm optimizer for wind farm layout optimization," Energy Conversion and Management, vol.269, p.116174, 2022.

[10] N. Covic and B. Lacevic, "Wingsuit flying search—a novel global optimization algorithm," IEEE Access, vol.8, pp.53883–53900, 2020.

[11] J.H. Halton, "Algorithm 247: Radical-inverse quasi-random point sequence," Communications of the ACM, vol.7, no.12, pp.701–702, 1964.

[12] J. Yang, Y. Zhang, Z. Wang, Y. Todo, B. Lu, and S. Gao, "A cooperative coevolution wingsuit flying search algorithm with spherical evolution," International Journal of Computational Intelligence Systems, vol.14, pp.1–19, 2021.

[13] H. Yang, S. Tao, Z. Zhang, Z. Cai, and S. Gao, "Spatial information sampling: another feedback mechanism of realising adaptive parameter control in meta-heuristic algorithms," International Journal of Bio-Inspired Computation, vol.19, no.1, pp.48–58, 2022.

[14] A. Karami, B. Ranjbar, M. Rahimi, and F. Mohammadi, "Novel hybrid neuro-fuzzy model to anticipate the heat transfer in a heat exchanger equipped with a new type of self-rotating tube insert," The European Physical Journal E, vol.45, no.11, p.92, 2022.

[15] B. Venkatesh, P. Sankaramurthy, B. Chokkalingam, and L. Mihet Popa, "Managing the demand in a micro grid based on load shifting with controllable devices using hybrid wfs2acso technique," Energies, vol.15, no.3, p.790, 2022.

[16] T. Zheng, H. Zhang, B. Zhang, Z. Cai, K. Wang, Y. Todo, and S. Gao, "Umbrellalike hierarchical artificial bee colony algorithm," IEICE Transactions on Information and Systems, vol.106, no.3, pp.410–418, 2023.

[17] N. Lynn, M.Z. Ali, and P.N. Suganthan, "Population topologies for particle swarm optimization and differential evolution," Swarm and Evolutionary Computation, vol.39, pp.24–35, 2018.

[18] J. Yang, K. Wang, Y. Wang, J. Wang, Z. Lei, and S. Gao, "Dynamic population structures-based differential evolution algorithm," IEEE Transactions on Emerging Topics in Computational Intelligence, 2024.

[19] W. Deng, H. Liu, J. Xu, H. Zhao, and Y. Song, "An improved quantum-inspired differential evolution algorithm for deep belief network," IEEE Transactions on Instrumentation and Measurement, vol.69, no.10, pp.7319–7327, 2020.

[20] Y.J. Gong, W.N. Chen, Z.H. Zhan, J. Zhang, Y. Li, Q. Zhang, and J.J. Li, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," Applied Soft Computing, vol.34, pp.286–300, 2015.

[21] X.W. Luo, Z.J. Wang, R.C. Guan, Z.H. Zhan, and Y. Gao, "A distributed multiple populations framework for evolutionary algorithm in solving dynamic optimization problems," IEEE Access, vol.7, pp.44372–44390, 2019.

[22] J.G. Falcón-Cardona, R.H. Gómez, C.A.C. Coello, and M.G.C. Tapia, "Parallel multi-objective evolutionary algorithms: A comprehensive survey," Swarm and Evolutionary Computation, vol.67, p.100960, 2021.

[23] V. Giammarino, S. Baldi, P. Frasca, and M.L. Delle Monache, "Traffic flow on a ring with a single autonomous vehicle: An interconnected stability perspective," IEEE Transactions on Intelligent Transportation Systems, vol.22, no.8, pp.4998–5008, 2020.

[24] J.C.L. López, E. Solares, and J.R. Figueira, "An evolutionary approach for inferring the model parameters of the hierarchical electre iii method," Information Sciences, vol.607, pp.705–726, 2022.

[25] X. Xue and J. Zhang, "Matching large-scale biomedical ontologies with central concept based partitioning algorithm and adaptive compact evolutionary algorithm," Applied Soft Computing, vol.106, p.107343, 2021.

[26] P. Pławiak, M. Abdar, J. Pławiak, V. Makarenkov, and U.R. Acharya, "Dghnl: A new deep genetic hierarchical network of learners for prediction of credit scoring," Information Sciences, vol.516, pp.401–418, 2020.

[27] Y. Wang, Y. Yu, S. Gao, H. Pan, and G. Yang, "A hierarchical gravitational search algorithm with an effective gravitational constant," Swarm and Evolutionary Computation, vol.46, pp.118–139, 2019.

[28] Z. Zhou, J. Abawajy, M. Shojafar, and M. Chowdhury, "Dehm: an improved differential evolution algorithm using hierarchical multistrategy in a cybertwin 6g network," IEEE Transactions on Industrial Informatics, vol.18, no.7, pp.4944–4953, 2022.

[29] N. Chen, T. Qiu, Z. Lu, and D.O. Wu, "An adaptive robustness evolution algorithm with self-competition and its 3d deployment for internet of things," IEEE/ACM Transactions on Networking, vol.30, no.1, pp.368–381, 2021.

[30] Q. Li, Z. Cao, W. Ding, and Q. Li, "A multi-objective adaptive evolutionary algorithm to extract communities in networks," Swarm and Evolutionary Computation, vol.52, p.100629, 2020.

[31] Z. Liao, W. Gong, and L. Wang, "Memetic niching-based evolutionary algorithms for solving nonlinear equation system," Expert Systems with Applications, vol.149, p.113261, 2020.

[32] W. Sheng, X. Wang, Z. Wang, Q. Li, Y. Zheng, and S. Chen, "A differential evolution algorithm with adaptive niching and k-means operation for data clustering," IEEE Transactions on Cybernetics, vol.52, no.7, pp.6181–6195, 2020.

[33] Z. Hu, T. Zhou, Q. Su, and M. Liu, "A niching backtracking search algorithm with adaptive local search for multimodal multiobjective optimization," Swarm and Evolutionary Computation, vol.69, p.101031, 2022.

[34] Y. Yu, S. Gao, M. Zhou, Y. Wang, Z. Lei, T. Zhang, and J. Wang, "Scale-free network-based differential evolution to solve function optimization and parameter estimation of photovoltaic models," Swarm and Evolutionary Computation, vol.74, p.101142, 2022.

[35] T. Qiu, J. Liu, W. Si, and D.O. Wu, "Robustness optimization scheme with multi-population co-evolution for scale-free wireless sensor networks," IEEE/ACM Transactions on Networking, vol.27, no.3, pp.1028–1042, 2019.

[36] R.M. May, "Simple mathematical models with very complicated dynamics," Nature, vol.261, no.5560, pp.459–467, 1976.

[37] I. Fister, J. Brest, A. Iglesias, A. Galvez, and S. Deb, "On selection of a benchmark by determining the algorithms' qualities," IEEE Access, vol.9, pp.51166–51178, 2021.

[38] V. Stanovov, S. Akhmedova, and E. Semenkin, "Lshade algorithm with rank-based selective pressure strategy for solving cec 2017 benchmark problems," 2018 IEEE congress on evolutionary computation (CEC), pp.1–8, IEEE, 2018.

[39] S.M. Elsayed, R.A. Sarker, and D.L. Essam, "Differential evolution with multiple strategies for solving cec2011 real-world numerical optimization problems," 2011 IEEE Congress of Evolutionary Computation (CEC), pp.1041–1048, IEEE, 2011.

[40] S. Liu, K. Wang, H. Yang, T. Zheng, Z. Lei, M. Jia, and S. Gao, "https://github.com/liusc1996/supplementary-file-for-mcwfs-paper."

[41] Y.J. Gong, J.J. Li, Y. Zhou, Y. Li, H.S.H. Chung, Y.H. Shi, and J. Zhang, "Genetic learning particle swarm optimization," IEEE Transactions on Cybernetics, vol.46, no.10, pp.2277–2290, 2015.

[42] K.M. Sallam, S.M. Elsayed, R.K. Chakrabortty, and M.J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," 2020 IEEE Congress on Evolutionary Computation (CEC), pp.1–8, IEEE, 2020.

[43] H. Yang, Y. Yu, J. Cheng, Z. Lei, Z. Cai, Z. Zhang, and S. Gao, "An intelligent metaphor-free spatial information sampling algorithm for balancing exploitation and exploration," Knowledge-Based Systems, vol.250, p.109081, 2022.

[44] S. Das and P.N. Suganthan, "Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems," Jadavpur University, Nanyang Technological University, Kolkata, pp.341–359, 2010.

[45] J. Liu, S. Li, C. Xu, Z. Wu, N. Ao, and Y.F. Chen, "Automatic and optimal rebar layout in reinforced concrete structure by decomposed optimization algorithms," Automation in Construction, vol.126, p.103655, 2021.

[46] S. Gao, K. Wang, S. Tao, T. Jin, H. Dai, and J. Cheng, "A state-of-the-art differential evolution algorithm for parameter estimation of solar photovoltaic models," Energy Conversion and Management, vol.230, p.113784, 2021.

[47] H. Yang, S. Gao, Z. Lei, J. Li, Y. Yu, and Y. Wang, "An improved spherical evolution with enhanced exploration capabilities to address wind farm layout optimization problem," Engineering Applications of Artificial Intelligence, vol.123, p.106198, 2023.

Sicheng LIU received the M.S. degree from University of Toyama, Toyama, Japan, in 2022. He is currently pursuing the Ph.D. degree of artificial intelligence with the University of Toyama, Toyama, Japan. His current research interests include computational intelligence and real-world applications.

Kaiyu WANG received the M.E. degree from the University of Toyama, Toyama, Japan, in 2022, where he is currently pursuing the Ph.D. degree. His current interests include computational intelligence and neural networks for real-world applications.

Haichuan YANG received Ph.D. degree from the University of Toyama, Toyama, Japan in 2022 and 2023 respectively. He is currently an assistant professor at Graduate School of Technology, Industrial and Social Sciences of Tokushima University. His current research interests lie in computational intelligence and complex systems.

Tao ZHENG received the M.S. degree from University of Toyama, Toyama, Japan, in 2023. He is currently pursuing the Ph.D. degree at University of Toyama, Toyama, Japan. His current research interest is computational intelligence.

Zhenyu LEI received the Ph.D. degree in Science and Engineering from the University of Toyama, Toyama, Japan, in 2023. He is currently an Assistant Professor with the Faculty of Engineering, University of Toyama, Japan. His current research interests include evolutionary computation, machine learning, and neural network for real-world applications and optimization problems.

Meng JIA received Ph.D. degrees from Xidian University, Xi'an, China, in 2016. She is currently a Lecturer with the School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, and also a member with the Shaanxi Key Laboratory of Network Computing and Security Technology, Xi'an. Her research interests include deep neural network and pattern recognition.

Shangce GAO received his Ph.D. degree in Innovative Life Science from University of Toyama, Toyama, Japan in 2011. He is currently a Professor with the Faculty of Engineering, University of Toyama, Japan. His current research interests include nature-inspired technologies, machine learning, and neural networks for real-world applications. He serves as an Associate Editor for IEEE Transactions on Neural Networks and Learning Systems.