Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 21(10): 7297–7336.
DOI: 10.3934/mbe.2024322
Received: 30 May 2024
Revised: 16 September 2024
Accepted: 08 October 2024
Published: 14 October 2024

Research article

A multi-strategy improved snake optimizer and its application to SVM parameter selection

Hong Lu, Hongxiang Zhan and Tinghua Wang*

School of Mathematics and Computer Science, Gannan Normal University, Ganzhou 341000, China

***Correspondence**: Email: wangtinghua@gnnu.edu.cn.

Abstract: Support vector machine (SVM) is an effective classification tool and maturely used in various fields. However, its performance is very sensitive to parameters. As a newly proposed swarm intelligence algorithm, snake optimizer algorithm (SO) can help to solve the parameter selection problem. Nevertheless, SO has the shortcomings of weak population initialization, slow convergence speed in the early stage, and being easy to fall into local optimization. To address these problems, an improved snake optimizer algorithm (ISO) was proposed. The mirror opposition-based learning mechanism (MOBL) improved the population quality to enhance the optimization speed. The novel evolutionary population dynamics model (NEPD) was beneficial for searching accurately. The differential evolution strategy (DES) helped to reduce the probability of falling into local optimal value. The experimental results of classical benchmark functions and CEC2022 showed that ISO had higher optimization precision and faster convergence rate. In addition, it was also applied to the parameter selection of SVM to demonstrate the effectiveness of the proposed ISO.

Keyword: snake optimizer; support vector machine (SVM); parameter optimization; opposition-based learning

1. Introduction

SVM [1] is a classical method in machine learning area. Because of its excellent performance, SVM is widely used in many application scenarios, such as text classification [2–4], facial recognition [5], pedestrian detection [6,7], etc. However, what mostly affects the ability of SVM is the kernel function selection and kernel parameter choice. About the former, the commonly used kernel functions are linear kernel, polynomial kernel, sigmoid kernel, Gaussian kernel (also called radius basis kernel function), and Laplace kernel. Many studies have shown that kernel function selection is closely related to the characteristics of data. According to different data features, we choose the appropriate kernel function. Among the above five kernel functions, the Gaussian kernel is the most popularly used. As for kernel parameter choice, there are four main methods, i.e., cross validation technology, minimizing the upper bound on the error rate of algorithms, optimizing kernel function metrics, and optimization algorithms (like swarm intelligence optimization algorithms). Among them, using swarm intelligence optimization algorithms to find the best kernel parameters is an effective solution.

Swarm intelligence optimization algorithms are a class of simple, flexible and adaptive meta-heuristic algorithms, which were inspired by the social behavior of biological individuals. Generally speaking, there is no standard categorization of swarm intelligence algorithms. For convenience, we classify them into four categories: evolution-based algorithms, physical & mathematical-based algorithms, human-based algorithms, and animal & plant-based algorithms.

Evolution-based algorithms are a type of algorithms that randomly update and replace individual creatures by modeling the rules of selection, crossover, and mutation among genes in biological genetics. The primary algorithms are genetic algorithm (GA) [8], differential evolution algorithm (DE) [9], grey prediction evolution algorithm (GPE) [10], and geometric probabilistic evolutionary algorithm (GPEA) [11].

Physical & mathematical-based algorithms are constructed from real-life physical phenomena or mathematical principles. The primary algorithms are simulated annealing algorithm (SA) [12], sine cosine algorithm (SCA) [13], gravitational search algorithm (GSA) [14], atomic search optimization algorithm (ASO) [15], artificial electric field algorithm (AEFA) [16], and optical microscope algorithm (OMA) [17].

Human-based algorithms are based on human mental activity or social behavior. The primary algorithms are teaching-learning based optimization (TLBO) [18], socio evolution and learning optimization (SELO) [19], human learning optimization algorithm (HLO) [20], and student psychology based optimization algorithm (SPBO) [21].

Animal & plant-based algorithms are the most numerous swarm intelligence algorithms. These algorithms are founded on the behavior of biological populations in nature, such as predation, reproduction, and competition for territory. The primary algorithms are gray wolf optimization algorithm (GWO) [22], tree seed algorithm (TSA) [23], whale optimizer algorithm (WOA) [24], Harris hawks optimization (HHO) [25], golden jackal optimization (GJO) [26], and northern goshawk optimization (NGO) [27]. To date, these kinds of algorithms have been improved by many scholars to solve various problems. For instance, Ma et al. [28] proposed a gray wolf optimizer based on Aquila exploration method (AGWO), which can expand the search range to improve the global search ability and reduce the possibility of falling into the local optimum. Kang et al. [29] combined HHO with Brownian motion-based mutant strategy to generate a new optimization algorithm (HHOBM), which helps HHO avoid the local optimum trap problem when optimizing non-convex functions. Lou et al. [30] introduced a hybrid strategy-based golden jackal optimizer algorithm (HGJO) to balance the global and local search capabilities, and applied it to robot path planning successfully. Li et al. [31] came up with a multi-strategy enhanced northern goshawk optimization algorithm (MENGO) to solve NGO's problems of slow convergence rate and tendency to fall into local optimization in some cases. Lin et al. [32] suggested a niching hybrid heuristic whale optimization algorithm (NHWOA) to enhance convergence speed and search coverage, and experimental results showed it has good performance in the global computations.

In 2022, Hashim and Hussien proposed the SO [33], which has good optimization capability, fast convergence speed, and wide search range [34,35], and many improved SO algorithms have been developed [36–40]. SO is used in a wide range of applications. For instance, Li et al. [41] proposed a snake optimization-based variable-step multi-scale single threshold slope entropy for classifying different categories of real-world signals. Nevertheless, it also has some disadvantages such as converging slowly in the early stage and easily falling into local optimization. At present, some scholars have made improvements to SO in view of these problems. For example, Hu et al. [42] proposed a multi-strategy boosted snake-inspired optimizer (BEESO), which was formed upon three improved methods, i.e., bidirectional search, modified evolutionary population dynamics, and elite opposition-based learning. However, it still has a tendency to fall into local optimum on some test functions. Yao et al. [43] proposed an enhanced snake optimizer (ESO) that utilizes four strategies, i.e., mirror imaging strategy based on convex lens imaging, parameter dynamic update strategy, sine-cosine composite perturbation factors, and tent-chaos & Cauchy mutation. Its experimental results are much better than BEESO. However, the convergence speed in the early stage of ESO can be raised a little bit.

In this paper, we combine the advantages of BEESO and ESO to propose an ISO algorithm based on MOBL, NEPD, and DES. In order to validate the effectiveness of ISO, a classical benchmark function test experiment, CEC2022 test experiment, ablation experiment, and SVM parameter optimization experiment are conducted, respectively. The main contributions of this paper are shown as follows:

• Based on ESO, BEESO, and DE algorithms, we propose an ISO, which mainly focuses on the population initialization before the exploration phase of SO, egg hatching mode during the exploitation phase, and the final elimination process.

• To illustrate the effectiveness of ISO, we test it against 12 other algorithms for comparison on 23 classical benchmark functions and CEC2022. We also perform the ablation experiment to discuss the impact of three improvement strategies and their combinations on SO.

• We apply ISO to the problem of parameter optimization for SVM and compare it with optimization methods based on other animal & plant-based algorithms.

The remainder of this article is organized as follows. Section 2 briefly introduces the SVM and SO algorithm. Section 3 describes the improved SO algorithm, which includes the mentioned three improvement strategies above. Section 4 summarizes the performances of ISO and other swarm intelligence optimization algorithms. The conclusion is shown in Section 5.

2. Preliminaries

2.1. SVM

SVM is a machine learning method based on statistical learning theory, mainly aiming at classification and regression problems. It has attracted more and more attention from scholars, and has become a mainstream technology and standard tool in the field of machine learning. Formally, for a set of training samples $(\boldsymbol{x}_i, y_i)$, $\boldsymbol{x}_i \in R^n$, $y_i \in \{+1, -1\}$, $i = 1, \cdots, m$, if the classification surface $\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x} + b = 0$ (where $\boldsymbol{\omega}$ is the normal vector and b is the bias term) can correctly classify the training samples into two categories, then the sum of the minimum distances from the two categories to the optimal classification surface should be maximized. The optimal classification surface can be obtained by solving the following optimization problem:

$$\begin{aligned} \min\ & \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.}\ & y_i\left(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \ge 1, \\ & \xi_i \ge 0,\quad i = 1, \cdots, m \end{aligned} \tag{1}$$

where C is the penalty factor, the role of which is to strike a balance between model complexity and learning capacity, and ξ_i is the error term.

Using the Lagrange multiplier method, we can solve the above quadratic programming problem with linear constraints, and get the Wolfe dyadic problem:

$$\begin{aligned} \max \quad & \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^{m}\alpha_i y_i = 0, \\ & 0 \le \alpha_i \le C, \quad i = 1,\cdots,m \end{aligned} \tag{2}$$

where α_i is the Lagrange multiplier.

After solving the above optimization problem, we obtain the decision function:

$$f(\boldsymbol{x}) = \mathrm{sgn}(\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j + b) \tag{3}$$

For nonlinear problems, assume that there is a nonlinear mapping $\phi : X \to F$, which maps samples from the input space to a high-dimensional feature space F and is implicitly defined by a kernel function. In this article, we choose to use the Gaussian kernel function because of its better generalization ability. The definition equation takes the following form:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j) = \exp(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2}), \ \sigma>0 \tag{4}$$

where σ is the width of Gaussian kernel.

After selecting the type of kernel function, the dyadic problem becomes:

$$\begin{aligned} \max \quad & \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \kappa\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \\ \text{s.t.} \quad & \sum_{i=1}^{m}\alpha_i y_i = 0, \\ & 0 \le \alpha_i \le C, \quad i = 1,\cdots,m \end{aligned} \tag{5}$$

and the corresponding decision function is as follows:

$$f(\boldsymbol{x}) = \mathrm{sgn}(\sum_{i=1}^{m}\alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) + b) \tag{6}$$

In order to obtain a better generalization performance of SVM, we need to optimize the penalty factor C and the width of Gaussian kernel σ.

2.2. Snake optimizer

SO is inspired by the hunting and mating behavior of snakes, and its search process can be divided into two phases: exploration and exploitation. The exploration phase describes the environmental factors, i.e., temperature and food, and there is no situation in this phase where snakes only search for food in its surroundings. It ensures that SO is able to search as wide as possible. The exploitation phase consists of two transitional modes, i.e., fight mode and mate mode, which are used to improve the search efficiency of SO. In the fight mode, each male snake will battle among themselves to get the best female snake, and each female snake will select the best male snake. In the mate mode, the occurrence of mating behavior depends on the amount of food and temperature. If the mating behavior happens, the worst positions of snakes are updated for the next round of iteration.

The following mathematical model represents the basic process of SO.

(I) Population initialization

Randomly initialize the population to get the initial position:

$$U_i = U_{min} + rand \times (U_{max} - U_{min}) \tag{7}$$

where U_i is the position of ith snake, $rand$ is a random number between 0 and 1, and U_{max} and U_{min} are the upper and lower bounds for the solution problem.

For the population, we first divide them into two groups: male snake group and female snake group, then calculate the fitness of two groups and find the best individual in each group. The best individual in the male snake group is $U_{best,m}$, and the best individual in the female snake group is $U_{best,f}$. Lastly, choose the global best individual U_{best} between $U_{best,m}$ and $U_{best,f}$.

(II) Exploration phase

Exploration and exploitation phase are determined by food Q and temperature $Temp$, which are defined by the following formulas:

$$\begin{aligned} Q &= c_1 \times \exp\left(\frac{t-T}{T}\right) \\ Temp &= \exp\left(\frac{-t}{T}\right) \end{aligned} \tag{8}$$

where $c_1 = 0.5$, t is the current number of iterations, and T is the maximum number of iterations.

When $Q < 0.25$, snakes start to randomly update their positions to find food.

$$\begin{aligned} U_{i,m}(t+1) &= U_{rand.m}(t) \pm c_2 \times A_m \times ((U_{max} - U_{min}) \times rand + U_{min}) \\ U_{i,f}(t+1) &= U_{rand.f}(t) \pm c_2 \times A_f \times ((U_{max} - U_{min}) \times rand + U_{min}) \end{aligned} \tag{9}$$

where $U_{i,m}$ and $U_{i,f}$ are the updated positions in the male snake group and female snake group, respectively, $U_{rand,m}$ and $U_{rand,f}$ are the random positions in the male snake group and female snake group, respectively, $c_2 = 0.05$, the sign $\pm$ can help the SO explore all possible directions and ensure a certain traversal, and A_m and A_f are the ability to find food of male snakes and female snakes, respectively.

$$\begin{aligned} A_m &= \exp\left(\frac{-f_{rand,m}}{f_{i,m}}\right) \\ A_f &= \exp\left(\frac{-f_{rand,f}}{f_{i,f}}\right) \end{aligned} \tag{10}$$

where $f_{rand,m}$ is the fitness of $U_{rand,m}$ and $f_{i,m}$ is the fitness of the ith snake in the male snake group, and $f_{rand,f}$ is the fitness of $U_{rand,f}$ and $f_{i,f}$ is the fitness of the ith snake in the female snake group.

(III) Exploitation phase

Under the condition of $Q > 0.25$, if $Temp > 0.6$, snakes do not reproduce; instead, they continue to search for food.

$$U_{i.m}(t+1)=U_{best}\pm c_3\times Temp\times rand\times\left(U_{best}-U_{i,m}(t)\right)$$
$$U_{i.f}(t+1)=U_{best}\pm c_3\times Temp\times rand\times\left(U_{best}-U_{i,f}(t)\right) \tag{11}$$

where $U_{i.m}$ and $U_{i.f}$ are the positions of male snakes and female snakes, respectively, and $c_3=2$. If $Temp<0.6$, the male and female snakes will begin to select each other and mate. Two situations exist during this period.

(i) Fight mode

The male snakes will compete with each other, as do female snakes.

$$U_{i,m}(t+1)=U_{i.m}(t)+c_3\times F_m\times rand\times\left(Q\times U_{best,f}-U_{i,m}(t)\right)$$
$$U_{i,f}(t+1)=U_{i.f}(t)+c_3\times F_f\times rand\times\left(Q\times U_{best,m}-U_{i,f}(t)\right) \tag{12}$$

where F_m and F_f are the fighting ability of male snakes and female snakes, respectively.

$$F_m=\exp\left(\frac{-f_{best,f}}{f_{i,m}}\right)$$
$$F_f=\exp\left(\frac{-f_{best,m}}{f_{i,f}}\right) \tag{13}$$

where $f_{best,f}$ is the fitness of $U_{best,f}$ and $f_{best,m}$ is the fitness of $U_{best,m}$.

(ii) Mate mode

Mating behavior begins between selected male and female snakes.

$$U_{i,m}(t+1)=U_{i.m}(t)+c_3\times M_m\times rand\times\left(Q\times U_{i,f}(t)-U_{i,m}(t)\right)$$
$$U_{i,f}(t+1)=U_{i.f}(t)+c_3\times M_f\times rand\times\left(Q\times U_{i,m}(t)-U_{i,f}(t)\right) \tag{14}$$

where M_m and M_f are the mating ability of male snakes and female snakes, respectively.

$$M_m = \exp\left(\frac{-f_{i,f}}{f_{i,m}}\right)$$
$$M_f = \exp\left(\frac{-f_{i,m}}{f_{i,f}}\right) \tag{15}$$

After mate mode was completed, SO has a certain probability to enter the egg-laying period. The session can help the worst male snake and female snake update their positions again.

The pseudo code of SO is given in Algorithm 1.

Algorithm 1: Snake optimizer

```
1.  Define the population size $N$, the maximum iteration $T$, $Dim$, $U_{max}$, and $U_{min}$.
2.  Initialize the population $U_i$ by Eq (7)
3.  while ( $t<T$ ) do
4.  Calculate food $Q$ and temperature $Temp$ by Eq (8).
5.  Choose the best individual $U_{best,m/f}$ and the global best individual $U_{best}$.
6.  if ( $Q<0.25$ )
7.  Enter the exploration phase.
```

```
8.   else if ( $Temp > 0.6$ )
9.   Enter the exploitation phase.
10.  else if ( $rand > 0.6$ )
11.  Enter the fight mode.
12.  else
13.  Enter the mate mode.
14.  if (egg == 1)
15.  Replace the worst male and female snake.
16.  end if
17   end if
18.  end if
19.  end if
20.  end while
```

3. Proposed method

Compared with other swarm intelligence algorithms, SO has good optimization ability and fast convergence speed. However, it still has some limitations. For instance, its convergence speed is a little slow in the early stage, and it has the tendency to fall into local optimization. To address these shortcomings, this section presents the improved SO algorithm based on multiple improvement strategies.

3.1. MOBL

In many cases, the problem solving process generally starts from zero or a random value and approaches toward the optimal solution. Examples include the weights of a neural network, the population parameters of a swarm intelligence algorithm, the kernel parameters of SVM, and so on. If the random value is near the optimal solution at the beginning, the problem can be solved quickly. However, there is the worst case where the random value appears opposite to the optimal solution, and the solution process will take a lot of time.

Generally, it is impossible to get a better random value initially without previous knowledge. In the perspective of logic, the solutions of a problem can be searched from all directions. If the solution produced during the search process and its opposite solution introduced together as feasible solutions to the problem, the efficiency of searching for the optimal solution will be higher. This is the core idea of opposition-based learning [44], and it can be defined as:

$$RU_i = \left(U_{max} + U_{min}\right) - U_i \tag{16}$$

where RU_i is the opposite value of U_i.

Based on this theory, Yao et al. [43] proposed a new opposition-based learning, i.e., mirror imaging strategy based on convex lens imaging. The strategy not only improves the optimization accuracy, but also ensures the convergence speed. The definition equation is given as follows:

$$MRU_i = \frac{U_{max} + U_{min}}{2} + \frac{U_{max} + U_{min}}{2q} - \frac{U_i}{q} \tag{17}$$

where q is the mirroring factor, and its definition formula is $q = 10 \times (1 - 2 \times (t/T)^2)$.

3.2. NEPD

By repeatedly testing the performance of SO, we found that the egg-laying period has some impact on optimization search accuracy, and sometimes causes SO to fall into a local optimum. Hu et al. [42] offered a solution, which modifies evolutionary population dynamics. The method provides two different optimization schemes for population.

Having sorted the population from best to worst, the top half of the individuals are recognized as the better individuals and the bottom half as the worse individuals.

(I) For the better individuals, we perform evolutionary operation.

$$NEU_i = U_i + E \times (U_{r1} - U_{r2}) \tag{18}$$

where NEU_i is the evolutionary value of U_i, U_{r1} and U_{r2} are different individuals in the current population excluding U_i, E is the scaling factor and its formula is $E = (\sin(2\pi \times freq \times t) \times (t+T) + 1)/2$, and $freq$ is the vibration frequency of the sinusoidal function, which is defined as $freq = 1/Dim$.

(II) As for the worse individuals, we also evolve them, or else eliminate them.

$$EU_i = \begin{cases} U_{best,m/f} + \mathrm{sign}(r-0.5) \times (U_{min} + (U_{max} - U_{min}) \times rand) & if\ rand < 0.5 \\ U_i + \mathrm{sign}(r-0.5) \times (U_{min} + (U_{max} - U_{min}) \times rand) & else \end{cases} \tag{19}$$

where r is a random value ranging from 0 to 1.

$$NEU_i = \begin{cases} EU_i & if\ f_{EU_i} < f_{U_i} \\ U_{min} + (U_{max} - U_{min}) \times rand & else \end{cases} \tag{20}$$

where f_{EU_i} is the fitness of EU_i and f_{U_i} is the fitness of U_i.

3.3. DES

DES is derived from the differential evolution algorithm. To be brief, through continuous evolution, the superior individuals are retained to guide the search process toward the optimal solution. The specific steps are as follows: first, we randomly select two different individuals (U_a and U_b) in the population, and subtract each other to produce the difference individual. Second, the difference individual is assigned with a weight and added with the third individual to produce the variant individual. If the fitness value of the variant individual is better than that of the parent individual, the variant individual is selected to enter the next iteration, otherwise the parent individual is retained. The defining equation is given as follows:

$$DEU_i = U_{best} + beta \times (U_a - U_b) \tag{21}$$

where $beta$ is the scale factor and its formula is $beta = beta_{max} - t \times (beta_{max} - beta_{min}) / T$, $beta_{max} = 0.8$, and $beta_{min} = 0.2$.

3.4. ISO

By introducing the above three strategies, we propose the ISO. The pseudo code of ISO is given in Algorithm 2.

Algorithm 2: Improved snake optimizer

```
1.  Define the population size $N$, the maximum iteration $T$, $Dim$, $U_{max}$, and $U_{min}$.
2.  Initialize the population $U_i$ by Eq (7)
3.  while ( $t < T$ ) do
4.  Calculate food $Q$ and temperature $Temp$ by Eq (8).
5.  Get the mirror positions $MRU_i$ by Eq (17) and select the better individuals.
6.  Choose the best individual $U_{best,m/f}$ and the global best individual $U_{best}$.
7.  if ( $Q < 0.25$ )
8.  Enter the exploration phase.
9.  else if ( $Temp > 0.6$ )
10. Enter the exploitation phase.
11. else if ( $rand > 0.6$ )
12. Enter the fight mode.
13. else
14. Enter the mate mode.
15. if (egg==1)
16. Calculate the fitness of $U_i$ and sort the fitness values.
17. for ( $i = 1 : U_i/2$ )
18.                     Get the evolutionary positions by Eq (18) and calculate the fitness.
                        Select the better individuals.
19. end for
20. for ( $i = (U_i / 2) + 1 : U_i$ )
21.                     Get the evolutionary positions by Eqs (19) and (20) and calculate
                        the fitness. Select the better individuals.
22. end for
23. end if
24. end if
25. end if
26. end if
27. Calculate the fitness of $U_i$ and update $U_{best,m/f}$.
28. for ( $i = 1 : U_i$ )
29. Calculate the variant individuals of $U_i$ by Eq (21) and select the better individuals.
30. end for
31. end while
```

The computational time complexity of SO is jointly determined by the population size N, dimension Dim and maximum number of iteration T, therefore its value is $O(N \times Dim \times T)$. As for ISO, we first analyze the time complexity involved in the three strategies. The MOBL mechanism requires time $O(2 \times Dim)$. Considering that the NEPD model requires different evolutionary operations for the first and second halves of the population, the time complexity is $O(N \times Dim)$. The computational complexity of DES is $O(N \times Dim)$. Therefore, the time complexity of ISO is $O(N \times Dim \times T)$. It is clear that there is no increase in the complexity required by ISO compared to SO.

4. Experiments

In this section, we verify the feasibility and effectiveness of ISO through four experiments, i.e., experiments on classical benchmark function, experiments on CEC2022, ablation experiment, and SVM parameter optimization experiment. ISO is compared with 12 other algorithms, including SO [33], GWO [22], GJO [26], HHO [25], NGO [27], WOA [24], ESO [43], AGWO [28], HGJO [30], HHOBM [29], MENGO [31], and NHWOA [32]. The first six are selected original algorithms because they have better optimization capabilities and are more widely used, and the last six are their improved algorithms, respectively. For a fair comparison, all algorithms use the parameter values used in the respective literature. The best results of test experiments are shown in bold format.

All experiments are conducted in an Intel Core i5-6200U, 2.4GHz CPU, and 8GB RAM laptop using the MATLAB 2021b under Windows 10 operating system.

4.1. Function test sets and UCI datasets

Table 1 shows the information of the 23 classical benchmark functions. According to their characteristics and information, we divide them into unimodal test functions (F1–F7), multimodal test functions (F8–F13), and multimodal test functions with fixed dimension (F14–F23).

Table 1. Basic information of classical benchmark functions.

Function	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
$F_5(x)=\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	30	[–30,30]	0
$F_6(x)=\sum_{i=1}^{n}[x_i+0.5]^2$	30	[–100,100]	0
$F_7(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	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 × dim
$F_9(x)=\sum_{i=1}^{n}[x_1^2-10\cos(2\pi x_i)+10]$	30	[–5.12,5.12]	0
$F_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	30	[–32,32]	0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	30	[–600,600]	0

Continued on next page

Function	Dim	Range	F_{min}
$F_{12}(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}$ $+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,s,v,w)=\begin{cases}v(x_i-s)^w,x_i>s\\0,-s<x_i<s\\v(-x_i-s)^w,x_i<-a\end{cases}$	30	[–50,50]	0
$F_{13}(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_1+1)]$ $\cdot(x_n-1)^2 1+\sin^2(2\pi x_n)\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	[–50,50]	0
$F_{14}(x)=(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^{-1}$	2	[–65.536,65.536]	0.998
$F_{15}(x)=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	4	[–5,5]	0.0003
$F_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	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
$F_{18}(x)=[1+(x_1+x_2+1)^2\cdot(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $\times[30+(2x_1-3x_2)^2\cdot(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	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	[0,1]	–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
$F_{21}(x)=-\sum_{i=1}^{5}\left[(x-a_i)(x-a_i)^T+c_i\right]^{-1}$	4	[0,10]	–10.1532
$F_{22}(x)=-\sum_{i=1}^{7}\left[(x-a_i)(x-a_i)^T+c_i\right]^{-1}$	4	[0,10]	–10.4028
$F_{23}(x)=-\sum_{i=1}^{10}\left[(x-a_i)(x-a_i)^T+c_i\right]^{-1}$	4	[0,10]	–10.5364

Table 2 lists the basic information of CEC2022. According to the characteristics and information of the CEC2022, we divide them into unimodal test function (F1), multimodal test functions (F2–F5), hybrid test functions (F6–F8), and composition test functions (F9–F12).

We select five datasets from UCI machine learning repository. Table 3 shows the information of datasets.

Table 2. Basic information of CEC2022.

ID	Function	Range	Dim	F_{min}
F1	Shifted and full Rotated Zakharov Function	[–100,100]	10/20	300
F2	Shifted and full Rotated Rosenbrock's Function	[–100,100]	10/20	400
F3	Shifted and full Rotated Rastrigin's Function	[–100,100]	10/20	600
F4	Shifted and full Rotated Non-Continuous Rastrigin's Function	[–100,100]	10/20	800
F5	Shifted and full Rotated Levy Function	[–100,100]	10/20	900
F6	Hybrid Function 1 ($N = 3$)	[–100,100]	10/20	1800
F7	Hybrid Function 2 ($N = 6$)	[–100,100]	10/20	2000
F8	Hybrid Function 3 (N = 5)	[–100,100]	10/20	2200
F9	Composition Function 1 ($N = 5$)	[–100,100]	10/20	2300
F10	Composition Function 2 ($N = 4$)	[–100,100]	10/20	2400
F11	Composition Function 3 ($N = 5$)	[–100,100]	10/20	2600
F12	Composition Function 4 ($N = 6$)	[–100,100]	10/20	2700

Table 3. Information of five datasets.

Name	Number of samples	Dimension of feature	Number of categories
Iris	150	4	3
Parkinsons	195	22	2
Fire	243	10	2
Heart	299	12	2
Ionosphere	351	34	2

4.2. SVM optimization process

The flow chart of the SVM optimization process is shown in Figure 1. Taking the ISO and Iris dataset as an example, 50% of the data samples from three types of samples (Iris setosa, Iris versicolour, and Iris virginica) are drawn as the training set and the remaining data samples are used as the test set. Next, the data in the training and test sets are normalized to the interval [0,1] to facilitate SVM model training. Then, the best combination of SVM parameters are selected by ISO, and the optimal parameters are used to train and test the SVM model.

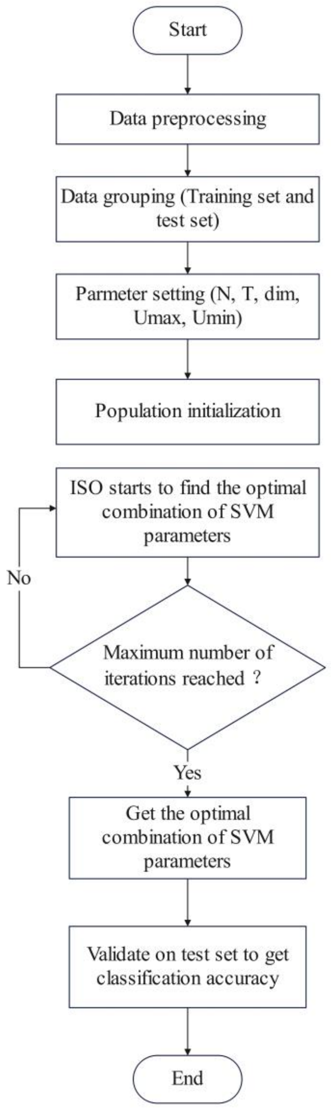

Start
Data preprocessing
Data grouping (Training set and test set)
Parmeter setting (N, T, dim, Umax, Umin)
Population initialization
ISO starts to find the optimal combination of SVM parameters
No
Maximum number of iterations reached ?
Yes
Get the optimal combination of SVM parameters
Validate on test set to get classification accuracy
End

Figure 1. Flow chart of SVM optimization using ISO.

4.3. Results and analysis

4.3.1. Experiment on classical benchmark function

In order to exclude the influence of other factors, all algorithms use uniform common parameter settings. We uniformly initialize the population, setting the population size ($N = 50$) and the maximum number of iterations ($T = 500$). Each algorithm is tested individually 30 times, and the best results are recorded. We calculate the average (Ave) and standard deviation (Std) as measure criteria from these optimal values. Eq (22) is the definition equation.

$$\begin{aligned} Ave &= \frac{1}{30}\sum_{i=1}^{30} U_{best,i} \\ Std &= \sqrt{\frac{1}{30}\sum_{i=1}^{30}(U_{best,i} - Ave)^2} \end{aligned} \tag{22}$$

The comparison results are shown in Table 4, where "−/=/+" indicates that the algorithm is lower, equal to or better than ISO, respectively. Meanwhile, we utilize statistical methods such as the Wilcoxon rank sum test to evaluate the rank of 13 algorithms.

We first discuss the performances of ISO, ESO, and SO. From Table 4, we can see that on the unimodal test functions, both ISO and ESO can get the best value on functions F1–F4. On functions F5–F7, ISO performs better than ESO and SO. On the multimodal test functions, SO performs worst of three algorithms. Although ISO and ESO obtain the same average value on function F8, ISO has a lower standard deviation. Both ISO and ESO have the same best values on functions F9–F11, and SO takes worse values. On functions F12 and F13, ISO has a significant advantage in optimization results. In the multimodal test functions with fixed dimension, ISO has the best results on functions F14, F15, F18, F20–F23 compared to ESO and SO. On functions F16, F17, F19, and F20, all three algorithms achieve the optimal mean, but they have different standard deviations. Among them, SO gets the optimal value, which is more stable than ISO and ESO on functions F16 and F19. Except that, they achieve equal results on function F17.

Compared to the other 10 algorithms, ISO is more robust on the unimodal test functions. It achieves the best optimization results on six functions, while ESO and MENGO achieve the best results only on four functions and HHOBM achieves the best result on function F5. On the multimodal test functions, the optimization ability of ISO is more significant. It can be found that ISO achieves the best results on five functions. On function F8, though the standard deviation of ISO is smaller than ESO, HHOBM is more suitable for optimization. On functions F9–F11, about half and more of these algorithms can find out the optimum. It can be seen that these three functions are easier to optimize. On the multimodal test functions with fixed dimension, ISO takes a greater advantage because it achieves the best results on seven functions, followed by NGO. On functions F14 and F20, ISO and NGO achieves the best results. Almost all algorithms can find out the best values on functions F16–F19, which indicates that the optimization search on these four benchmark functions is extremely simple. On functions F21–F23, NGO still possesses a stronger capability to find out the best value, as ISO and ESO do.

Figure 2 plots the convergence curves of the 13 algorithms on 23 classical benchmark functions. From Figure 2, the convergence rate of ISO is clearly better than ESO and SO. Only on function F14, ISO fails to converge quickly and traps in a local optimum for a longer period of time, but eventually it is able to find out the optimum value.

For other comparison algorithms, MENGO converges nearly as fast as ISO on the unimodal test functions. In particular, on functions F1–F4, MENGO converges slightly faster than ISO. However, on function F5, MENGO falls into a local optimum at the later stage, resulting in failure to reach a value near the optimum. HHOBM, on the contrary, searches for a better optimum on function F5, and maintains a higher rate. On functions F6 and F7, ISO is more competitive. On the multimodal test functions, most of the comparison algorithms converge well. Only SO lags a little behind. On function F8, HHOBM converges better in the early stage. Although HHO also performs better than ISO, it falls into a local optimum. On the other five functions, the slow convergence with SO becomes apparent. On the multimodal test functions with fixed dimension, ISO has the fastest convergence rate, except on the function F14, which falls into a local optimum, and NGO has the fastest rate of convergence at this point.

Table 4. Comparison results between ISO and other 12 algorithms on 23 classical benchmark functions.

F(x)		GWO	GJO	HHO	NGO	WOA	AGWO	HGJO	HHOBM	MENGO	NHWOA	SO	ESO	ISO
F1	Ave	3.59×10^{-33}	1.64×10^{-61}	1.36×10^{-100}	7.52×10^{-89}	6.91×10^{-65}	1.12×10^{-161}	**0**	1.60×10^{-100}	**0**	5.16×10^{-87}	4.53×10^{-97}	**0**	**0**
	Std	7.23×10^{-33}	7.74×10^{-61}	6.14×10^{-100}	1.75×10^{-88}	3.78×10^{-64}	3.55×10^{-161}	**0**	8.66×10^{-100}	**0**	2.83×10^{-86}	7.05×10^{-97}	**0**	**0**
	−/=/+	–	–	–	–	–	–	**=**	–	**=**	–	–	**=**	/
	Rank	13	12	6	9	11	5	**1**	7	**1**	10	8	**1**	**1**
F2	Ave	8.35×10^{-20}	2.61×10^{-36}	8.07×10^{-53}	5.93×10^{-46}	1.23×10^{-47}	2.37×10^{-91}	4.71×10^{-264}	5.92×10^{-51}	**0**	2.59×10^{-56}	2.04×10^{-45}	**0**	**0**
	Std	5.26×10^{-20}	3.98×10^{-36}	4.15×10^{-52}	3.73×10^{-46}	3.99×10^{-47}	3.71×10^{-91}	**0**	2.22×10^{-50}	**0**	5.74×10^{-56}	4.21×10^{-45}	**0**	**0**
	−/=/+	–	–	–	–	–	–	–	–	**=**	–	–	**=**	/
	Rank	13	12	7	10	9	5	4	8	**1**	6	11	**1**	**1**
F3	Ave	4.04×10^{-8}	5.12×10^{-21}	1.16×10^{-80}	3.15×10^{-23}	3.59×10^{4}	2.97×10^{-87}	**0**	1.35×10^{-79}	**0**	9.96×10^{3}	4.62×10^{-58}	**0**	**0**
	Std	8.83×10^{-8}	1.22×10^{-20}	6.35×10^{-80}	1.19×10^{-22}	4.21×10^{3}	1.51×10^{-86}	**0**	7.40×10^{-79}	**0**	6.01×10^{3}	1.47×10^{-57}	**0**	**0**
	−/=/+	–	–	–	–	–	–	**=**	–	**=**	–	–	**=**	/
	Rank	11	10	6	9	13	5	**1**	7	**1**	12	8	**1**	**1**
F4	Ave	3.48×10^{-8}	9.13×10^{-19}	4.39×10^{-52}	7.55×10^{-38}	22.2	1.52×10^{-65}	5.14×10^{-251}	3.18×10^{-51}	**0**	25.1	8.60×10^{-42}	**0**	**0**
	Std	2.97×10^{-8}	1.69×10^{-18}	1.18×10^{-51}	4.43×10^{-38}	23.3	6.13×10^{-65}	**0**	1.63×10^{-50}	**0**	22.6	2.18×10^{-41}	**0**	**0**
	−/=/+	–	–	–	–	–	–	–	–	=	–	–	**=**	/
	Rank	11	10	6	9	12	5	4	7	1	13	8	**1**	**1**
F5	Ave	26.9	27.3	4.87×10^{-3}	25.4	27.6	27.5	28.9	$\mathbf{6.27\times10^{-5}}$	27	27.4	19.7	1.97	1.21
	Std	0.688	0.66	5.28×10^{-3}	0.353	0.401	0.501	9.36×10^{-2}	$\mathbf{1.09\times10^{-4}}$	0.958	0.466	11.9	6.72	4.61
	−/=/+	–	–	+	–	–	–	–	**+**	–	–	–	–	/
	Rank	7	9	2	6	12	11	13	**1**	8	10	5	4	3
F6	Ave	0.445	2.21	5.50×10^{-5}	8.00×10^{-7}	5.41×10^{-2}	3.01	6.08	2.15×10^{-5}	1.48	4.62×10^{-2}	0.134	4.35×10^{-3}	$\mathbf{1.51\times10^{-8}}$
	Std	0.314	0.488	8.19×10^{-5}	1.70×10^{-7}	3.28×10^{-2}	0.29	0.413	3.10×10^{-5}	0.45	5.67×10^{-2}	7.45×10^{-2}	1.05×10^{-2}	$\mathbf{4.29\times10^{-8}}$
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	/
	Rank	9	11	4	2	7	12	13	3	10	6	8	5	**1**

Continued on next page

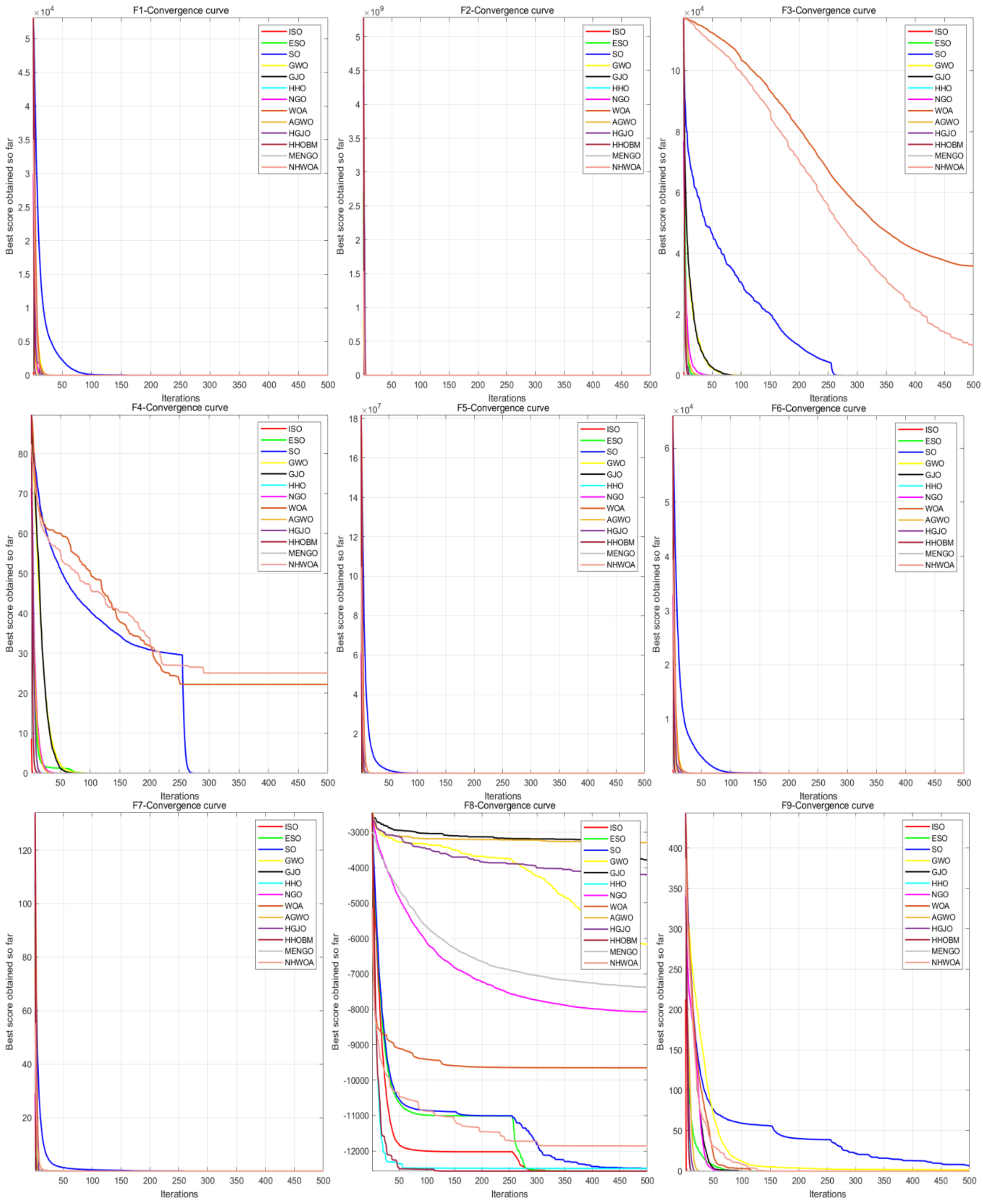
F(x)		GWO	GJO	HHO	NGO	WOA	AGWO	HGJO	HHOBM	MENGO	NHWOA	SO	ESO	ISO
F7	Ave	1.34×10^{-3}	3.28×10^{-4}	7.50×10^{-5}	4.66×10^{-4}	2.52×10^{-3}	1.15×10^{-4}	4.35×10^{-5}	8.97×10^{-5}	2.18×10^{-5}	1.60×10^{-3}	1.72×10^{-4}	1.31×10^{-4}	$\mathbf{1.98\times10^{-5}}$
	Std	8.61×10^{-4}	2.47×10^{-4}	8.76×10^{-5}	1.89×10^{-4}	3.72×10^{-3}	1.04×10^{-4}	4.56×10^{-5}	7.45×10^{-5}	2.33×10^{-5}	1.72×10^{-3}	1.55×10^{-4}	1.08×10^{-4}	$\mathbf{1.81\times10^{-5}}$
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	/
	Rank	11	9	4	10	13	6	3	5	2	12	8	7	**1**
F8	Ave	-6.16×10^{3}	-3.79×10^{3}	-1.25×10^{4}	-8.07×10^{3}	-9.65×10^{3}	-3.30×10^{3}	-4.20×10^{3}	$-\mathbf{1.26\times10^{4}}$	-7.38×10^{3}	-1.19×10^{4}	-1.25×10^{4}	$-\mathbf{1.26\times10^{4}}$	$-\mathbf{1.26\times10^{4}}$
	Std	7.72×10^{2}	9.86×10^{2}	3.94×10^{2}	3.72×10^{2}	1.49×10^{2}	3.59×10^{2}	5.94×10^{2}	$\mathbf{1.90\times10^{-2}}$	4.05×10^{2}	1.15×10^{3}	1.89×10^{2}	1.12	0.371
	−/=/+	–	–	–	–	–	–	–	+	–	–	–	–	/
	Rank	10	12	5	8	7	13	11	**1**	9	6	4	3	2
F9	Ave	1.53	**0**	**0**	**0**	1.89×10^{-15}	**0**	**0**	**0**	**0**	**0**	6.51	**0**	**0**
	Std	3.23	**0**	**0**	**0**	1.04×10^{-14}	**0**	**0**	**0**	**0**	**0**	9.97	**0**	**0**
	−/=/+	–	=	=	=	–	=	=	=	=	=	–	=	/
	Rank	12	**1**	**1**	**1**	11	**1**	**1**	**1**	**1**	**1**	13	**1**	**1**
F10	Ave	4.25×10^{-14}	6.69×10^{-15}	$\mathbf{8.88\times10^{-16}}$	5.27×10^{-15}	4.80×10^{-15}	4.56×10^{-15}	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	4.20E-15	4.44×10^{-15}	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$
	Std	2.82×10^{-15}	1.74×10^{-15}	**0**	1.53×10^{-15}	2.53×10^{-15}	6.49×10^{-16}	**0**	**0**	**0**	2.46×10^{-15}	**0**	**0**	**0**
	−/=/+	–	–	=	–	–	–	=	=	=	–	–	=	/
	Rank	13	12	**1**	11	10	9	**1**	**1**	**1**	7	8	**1**	**1**
F11	Ave	5.01×10^{-3}	**0**	**0**	**0**	4.59×10^{-3}	**0**	**0**	**0**	**0**	1.02×10^{-2}	1.54×10^{-2}	**0**	**0**
	Std	7.63×10^{-3}	**0**	**0**	**0**	2.51×10^{-2}	**0**	**0**	**0**	**0**	4.58×10^{-2}	3.02×10^{-2}	**0**	**0**
	−/=/+	–	=	=	=	–	=	=	=	=	–	–	=	/
	Rank	11	**1**	**1**	**1**	10	**1**	**1**	**1**	**1**	12	13	**1**	**1**
F12	Ave	2.59×10^{-2}	0.181	3.23×10^{-6}	1.91×10^{-7}	5.69×10^{-3}	0.197	0.915	1.22×10^{-7}	3.68×10^{-2}	3.98×10^{-3}	0.112	1.42×10^{-4}	$\mathbf{2.71\times10^{-9}}$
	Std	1.65×10^{-2}	5.93×10^{-2}	4.23×10^{-6}	1.27×10^{-7}	3.42×10^{-3}	5.04×10^{-2}	0.172	1.27×10^{-7}	2.49×10^{-2}	2.60×10^{-3}	0.343	1.35×10^{-4}	$\mathbf{7.51\times10^{-9}}$
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	/
	Rank	8	11	4	3	7	12	13	2	9	6	10	5	**1**

Continued on next page

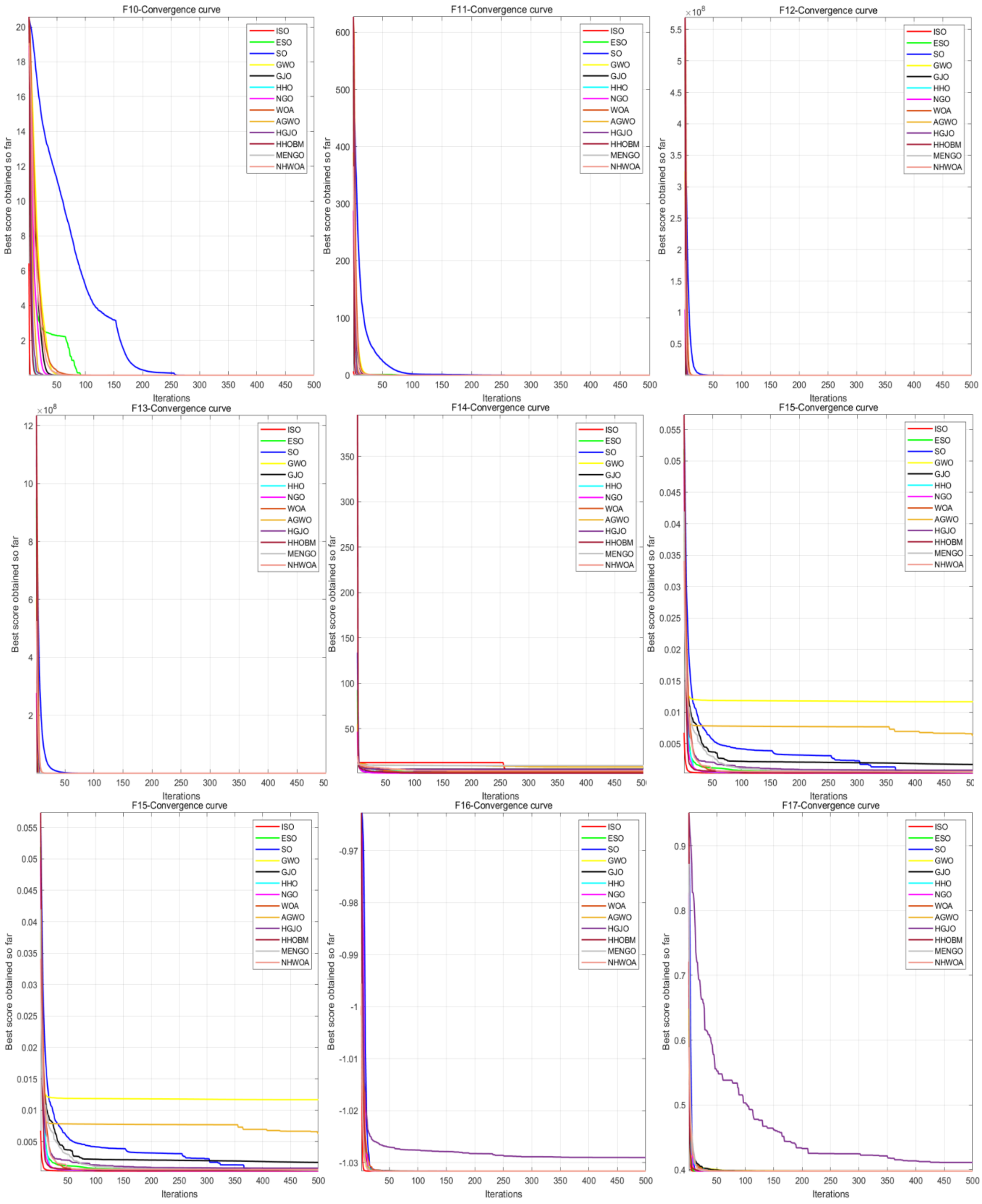
F(x)		GWO	GJO	HHO	NGO	WOA	AGWO	HGJO	HHOBM	MENGO	NHWOA	SO	ESO	ISO
F13	Ave	0.39	1.48	3.51×10^{-5}	6.62×10^{-2}	0.116	1.99	2.85	1.95×10^{-6}	2.75	0.155	0.216	1.02×10^{-3}	**5.74×10^{-8}**
	Std	0.223	0.16	1.11×10^{-4}	6.55×10^{-2}	7.45×10^{-2}	0.188	9.18×10^{-2}	1.80×10^{-6}	0.566	0.107	0.439	2.06×10^{-3}	**2.22×10^{-7}**
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	**/**
	Rank	9	10	3	5	6	11	13	2	12	7	8	4	**1**
F14	Ave	4.49	5.37	1.03	**0.998**	2.24	7.77	5.10	1.03	9.39	3.71	1.03	**0.998**	**0.998**
	Std	4.09	4.39	0.181	**0**	3.05	4.33	4.12	0.181	4.03	4.09	0.181	1.08×10^{-8}	**0**
	−/=/+	–	–	–	**=**	–	–	–	–	–	–	–	–	**/**
	Rank	9	11	4	**1**	7	12	10	4	13	8	4	3	**1**
F15	Ave	1.17×10^{-2}	1.69×10^{-3}	3.42×10^{-4}	3.08×10^{-4}	4.72×10^{-4}	6.32×10^{-3}	7.82×10^{-4}	3.62×10^{-4}	4.46×10^{-4}	5.61×10^{-4}	4.13×10^{-4}	3.11×10^{-4}	**3.07×10^{-4}**
	Std	1.01×10^{-2}	5.08×10^{-3}	2.30×10^{-5}	2.81×10^{-7}	1.30×10^{-4}	8.89×10^{-3}	5.13×10^{-4}	1.64×10^{-4}	9.98×10^{-5}	3.89×10^{-4}	1.35×10^{-4}	1.58×10^{-5}	**1.03×10^{-18}**
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	**/**
	Rank	13	11	4	2	8	12	10	5	7	9	6	3	**1**
F16	Ave	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**
	Std	5.22×10^{-9}	9.70×10^{-8}	9.53×10^{-11}	6.71×10^{-16}	9.24×10^{-12}	8.22×10^{-8}	8.34×10^{-3}	4.17×10^{-12}	6.45×10^{-16}	4.86×10^{-11}	**5.68×10^{-16}**	5.83×10^{-16}	6.78×10^{-16}
	−/=/+	–	–	–	+	–	–	–	–	+	–	**+**	+	/
	Rank	10	12	9	4	7	11	13	6	3	8	**1**	2	5
F17	Ave	**0.398**	**0.398**	**0.398**	**0.398**	**0.398**	**0.398**	0.411	**0.398**	**0.398**	**0.398**	**0.398**	**0.398**	**0.398**
	Std	1.44×10^{-7}	4.27×10^{-6}	7.95×10^{-7}	**0**	1.72×10^{-6}	3.93×10^{-6}	1.60×10^{-2}	3.52×10^{-10}	**0**	4.28×10^{-7}	**0**	**0**	**0**
	−/=/+	–	–	–	**=**	–	–	–	–	**=**	–	**=**	**=**	**/**
	Rank	7	12	9	**1**	10	11	13	6	**1**	8	**1**	**1**	**1**
F18	Ave	**3**	**3**	**3**	**3**	**3**	**3**	3.01	**3**	**3**	**3**	15.6	3.9	**3**
	Std	1.28×10^{-5}	1.67×10^{-6}	5.48×10^{-8}	**1.16×10^{-15}**	4.27×10^{-5}	4.88×10^{-7}	7.33×10^{-3}	1.90×10^{-14}	1.37×10^{-15}	9.09×10^{-6}	13.7	4.93	1.32×10^{-15}
	−/=/+	–	–	–	+	–	–	–	–	–	–	–	–	/
	Rank	9	7	5	**1**	10	6	11	4	3	8	13	12	2

Continued on next page

F(x)		GWO	GJO	HHO	NGO	WOA	AGWO	HGJO	HHOBM	MENGO	NHWOA	SO	ESO	ISO
F19	Ave	**–3.86**	**–3.86**	**–3.86**	**–3.86**	**–3.86**	**–3.86**	–3.8	**–3.86**	**–3.86**	**–3.86**	**–3.86**	**–3.86**	**–3.86**
	Std	2.12×10^{-3}	3.63×10^{-3}	1.99×10^{-3}	2.71×10^{-15}	4.78×10^{-3}	2.84×10^{-3}	2.19×10^{-2}	2.63×10^{-5}	2.63×10^{-15}	2.78×10^{-3}	**2.61×10^{-15}**	2.63×10^{-15}	2.71×10^{-15}
	−/=/+	–	–	–	=	–	–	–	–	+	–	**+**	+	/
	Rank	8	11	7	4	12	10	13	6	2	9	**1**	2	4
F20	Ave	**–3.32**	–3.25	–3.13	**–3.32**	–3.31	**–3.32**	–2.18	–3.22	**–3.32**	**–3.32**	**–3.32**	**–3.32**	**–3.32**
	Std	3.40×10^{-6}	0.122	0.112	**1.36×10^{-15}**	3.46×10^{-2}	7.14×10^{-4}	0.206	8.06×10^{-2}	1.35×10^{-14}	5.24×10^{-4}	6.94×10^{-15}	3.83×10^{-11}	**1.36×10^{-15}**
	−/=/+	–	–	–	**=**	–	–	–	–	–	–	–	–	/
	Rank	6	11	12	**1**	10	8	13	9	4	7	3	5	**1**
F21	Ave	–7.1	–8.03	–5.05	**–10.2**	–5.90	–6.07	–1.70	–6.07	–10.1	–9.28	–10	**–10.2**	**–10.2**
	Std	2.54	2.67	1.37×10^{-3}	1.55×10^{-8}	1.93	2.07	0.948	2.07	4.23×10^{-2}	1.93	0.359	1.72×10^{-5}	**7.07×10^{-15}**
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	/
	Rank	8	7	12	2	11	9	13	9	4	6	5	3	**1**
F22	Ave	**–10.4**	–9.34	–5.09	**–10.4**	–5.09	–10.1	–1.12	–7.39	**–10.4**	–7.05	–10.3	**–10.4**	**–10.4**
	Std	1.62×10^{-4}	2.43	1.39×10^{-3}	3.28×10^{-7}	2.36	1.39	0.498	2.68	3.01×10^{-2}	3.67	0.303	1.46×10^{-4}	**1.23×10^{-15}**
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	/
	Rank	4	8	11	2	12	7	13	9	5	10	6	3	**1**
F23	Ave	–10	–10.1	–5.04	**–10.5**	–6.32	–5.67	–2.62	–6.39	–10.3	–7.82	–10.4	**–10.5**	**–10.5**
	Std	2.06	1.76	0.496	2.40×10^{-15}	2.83	4.03	1.06	2.33	1.02	3.68	0.399	2.56×10^{-15}	**1.89**$\times10^{-15}$
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	/
	Rank	7	6	12	2	10	11	13	9	5	8	4	3	**1**
−/=/+		23/0/0	21/2/0	19/3/1	15/6/2	23/0/0	21/2/0	18/5/0	18/3/2	13/8/2	22/1/0	20/1/2	14/8/1	/
Average rank		9.52	9.39	5.87	4.52	10.30	8.39	9.30	4.91	4.52	8.22	7.13	3.13	**1.48**
Total rank		11	12	6	4	13	9	10	5	3	8	7	2	**1**



Continued on next page



Continued on next page

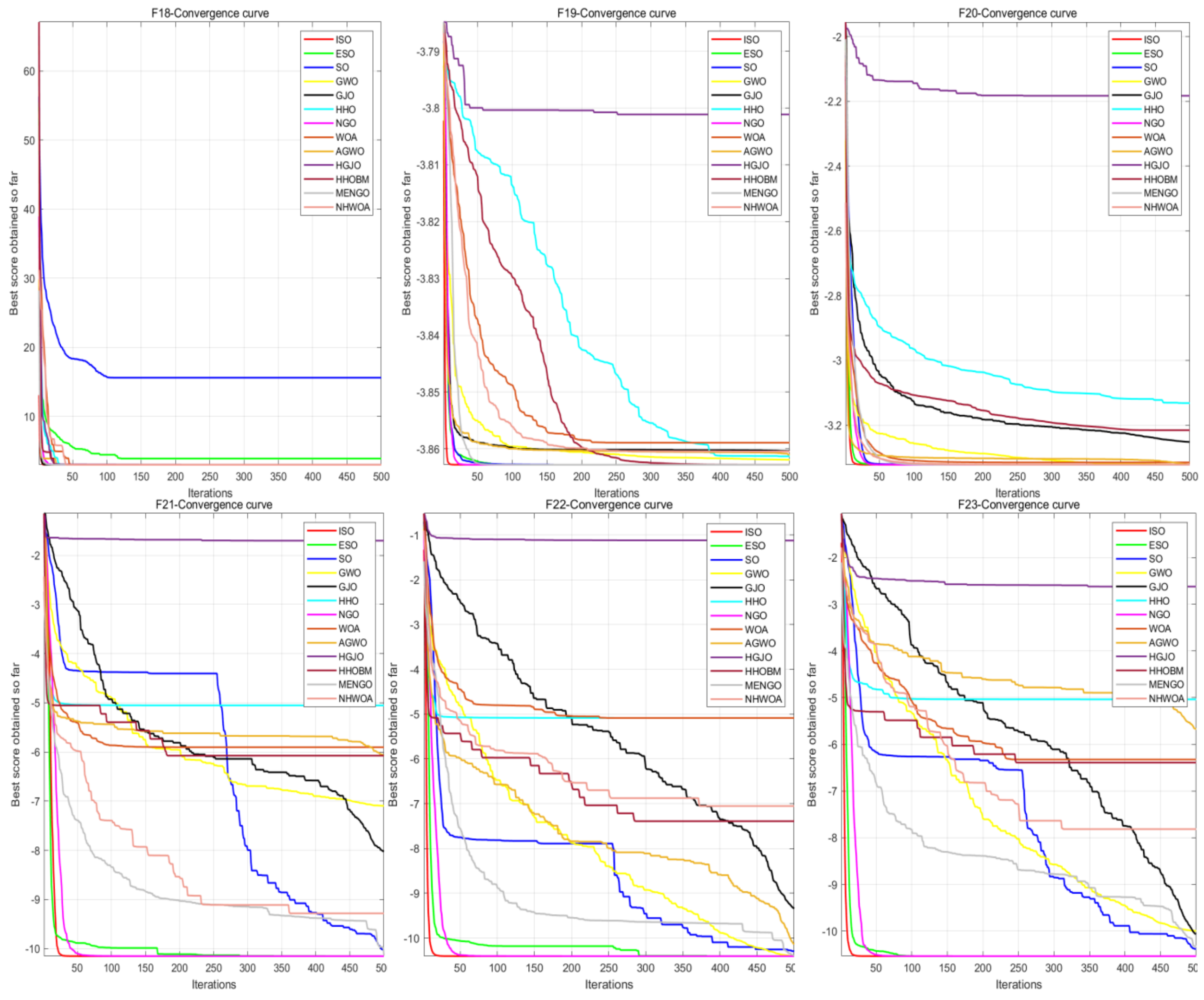


Figure 2. Convergence curves of 13 algorithms on 23 classical benchmark functions.

4.3.2. Experiment on CEC2022

CEC2022 is a newer test set for evaluating swarm intelligence algorithms. In this experiment, we also set the population size ($N = 50$) and the maximum number of iterations ($T = 500$), and run each algorithm 30 times individually and the corresponding optimal values are recorded. Tables 5 and 6 show the experimental results of 13 algorithms on these functions with different dimensions, where "−/=/+" indicates that the algorithm is lower, equal to, or better than ISO, respectively. Likewise, we use the Wilcoxon's rank sum test to evaluate the rank of these algorithms.

According to Table 5, ISO ranks second among all algorithms with an average rank of 2.5, while NGO ranks first with an average rank of 2.33. The main reason for ISO lagging behind NGO may be that ISO ranks 10th on function F4, which is far behind other algorithms due to its poor optimization on this function. The second reason is that ISO ranks fourth on functions F2 and F10. Nevertheless, ISO achieves the best results on other nine functions.

Table 5. Comparison results between ISO and other 12 algorithms on CEC2022 (Dim = 10).

F(x)		GWO	GJO	HHO	NGO	WOA	AGWO	HGJO	HHOBM	MENGO	NHWOA	SO	ESO	ISO
F1	Ave	1.36×10^{3}	3.26×10^{3}	4.62×10^{2}	3.14×10^{2}	1.35×10^{4}	5.21×10^{3}	1.27×10^{4}	4.01×10^{2}	5.25×10^{2}	1.17×10^{4}	5.47×10^{2}	3.10×10^{2}	$\mathbf{3.00\times10^{2}}$
	Std	1.40×10^{3}	1.83×10^{3}	1.84×10^{2}	2.12×10^{2}	5.02×10^{3}	2.66×10^{3}	3.72×10^{3}	1.38×10^{2}	2.22×10^{2}	4.89×10^{3}	2.52×10^{2}	21.8	$\mathbf{6.70\times10^{-4}}$
	−/=/+	−	−	−	−	−	−	−	−	−	−	−	−	/
	Rank	8	9	5	3	13	10	12	4	6	11	7	2	**1**
F2	Ave	4.20×10^{2}	4.39×10^{2}	4.41×10^{2}	$\mathbf{4.01\times10^{2}}$	4.60×10^{2}	4.36×10^{2}	9.29×10^{2}	4.32×10^{2}	4.14×10^{2}	4.46×10^{2}	4.05×10^{2}	4.02×10^{2}	4.08×10^{2}
	Std	18.5	22.3	36.6	**2.24**	34.2	26.8	2.54×10^{2}	32.1	20.1	33.8	9.05	3.73	2.73
	−/=/+	−	−	−	**+**	−	−	−	−	−	−	+	+	/
	Rank	6	9	10	**1**	12	8	13	7	5	11	3	2	4
F3	Ave	6.01×10^{2}	6.08×10^{2}	6.24×10^{2}	**6.00**$\times10^{2}$	6.22×10^{2}	6.12×10^{2}	6.57×10^{2}	6.25×10^{2}	6.03×10^{2}	6.25×10^{2}	$\mathbf{6.00\times10^{2}}$	$\mathbf{6.00\times10^{2}}$	$\mathbf{6.00\times10^{2}}$
	Std	0.911	6.24	7.26	0.128	6.85	4.69	10.1	7.15	3.54	7.39	0.135	8.94×10^{-2}	$\mathbf{2.28\times10^{-5}}$
	−/=/+	−	−	−	−	−	−	−	−	−	−	−	−	/
	Rank	5	7	10	3	9	8	13	11	6	12	4	2	**1**
F4	Ave	8.15×10^{2}	8.24×10^{2}	8.20×10^{2}	8.09×10^{2}	8.39×10^{2}	8.20×10^{2}	8.53×10^{2}	8.19×10^{2}	8.17×10^{2}	8.39×10^{2}	8.11×10^{2}	$\mathbf{8.07\times10^{2}}$	8.27×10^{2}
	Std	7.45	9.64	5.26	**2.25**	11.3	6.06	9.92	5.5	5.72	12.1	4.52	3.90	6.11
	−/=/+	+	+	+	**+**	−	+	−	+	+	−	+	+	/
	Rank	4	9	7	2	11	8	13	6	5	12	3	**1**	10
F5	Ave	9.08×10^{2}	9.76×10^{2}	1.54×10^{3}	**9.00**$\times10^{2}$	1.54×10^{3}	9.87×10^{2}	1.51×10^{3}	1.49×10^{3}	9.54×10^{2}	1.47×10^{2}	9.08×10^{2}	$\mathbf{9.00\times10^{2}}$	$\mathbf{9.00\times10^{2}}$
	Std	15.8	74.8	1.21×10^{2}	0.116	3.57×10^{2}	49.8	23.2	13.8	73.7	3.41×10^{2}	19.6	3.26×10^{-2}	$\mathbf{3.96\times10^{-12}}$
	−/=/+	−	−	−	−	−	−	−	−	−	−	−	−	/
	Rank	4	7	12	3	13	8	11	10	6	9	5	2	**1**
F6	Ave	4.81×10^{3}	7.90×10^{3}	3.33×10^{3}	$\mathbf{1.90\times10^{3}}$	3.30×10^{3}	1.51×10^{4}	4.07×10^{7}	2.44×10^{3}	2.44×10^{3}	2.61×10^{3}	3.74×10^{3}	2.25×10^{3}	1.99×10^{3}
	Std	2.53×10^{3}	2.30×10^{3}	1.41×10^{3}	**32.7**	1.61×10^{3}	9.85×10^{3}	2.21×10^{7}	6.94×10^{2}	1.09×10^{3}	9.15×10^{2}	1.64×10^{3}	3.92×10^{2}	3.38×10^{2}
	−/=/+	−	−	−	**+**	−	−	−	−	−	−	−	−	/
	Rank	10	11	8	**1**	7	12	13	4	5	6	9	3	2

Continued on next page

F(x)		GWO	GJO	HHO	NGO	WOA	AGWO	HGJO	HHOBM	MENGO	NHWOA	SO	ESO	ISO
F7	Ave	2.03×10^{3}	2.04×10^{3}	2.09×10^{3}	2.02×10^{3}	2.08×10^{3}	2.06×10^{3}	2.13×10^{3}	2.08×10^{3}	2.03×10^{3}	2.09×10^{3}	2.02×10^{3}	2.02×10^{3}	**2.01×10^{3}**
	Std	9.34	14.5	35.5	**6.63**	32	14.2	25.7	33.9	9.49	41. 6	8.02	10.3	8.93
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	/
	Rank	5	7	11	2	9	8	13	10	6	12	3	4	**1**
F8	Ave	2.23×10^{3}	2.23×10^{3}	2.23×10^{3}	2.22×10^{3}	2.23×10^{3}	2.23×10^{3}	2.34×10^{3}	2.23×10^{3}	2.23×10^{3}	2.23×10^{3}	2.22×10^{3}	2.22×10^{3}	**2.21×10^{3}**
	Std	4.94	3.79	2.34	3.64	7.73	2.21	97.2	9.24	**2.01**	3.65	3.94	6.48	9.89
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	/
	Rank	10	9	6	8	13	5	12	11	4	7	2	3	**1**
F9	Ave	2.55×10^{3}	2.59×10^{3}	2.63×10^{3}	**2.53**$\times10^{3}$	2.62×10^{3}	2.60×10^{3}	2.77×10^{3}	2.62×10^{3}	**2.53**$\times10^{3}$	2.64×10^{3}	**2.53×10^{3}**	**2.53×10^{3}**	**2.53×10^{3}**
	Std	18.9	41	42.4	3.79×10^{-11}	45.8	30.3	48.9	40.4	1.1×10^{-3}	52.3	0.192	2.98×10^{-4}	**0**
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	**/**
	Rank	6	7	11	2	10	8	13	9	4	12	5	3	**1**
F10	Ave	2.61×10^{3}	2.63×10^{3}	2.64×10^{3}	**2.55×10^{3}**	2.60×10^{3}	2.62×10^{3}	2.65×10^{3}	2.62×10^{3}	2.61×10^{3}	2.63×10^{3}	2.57×10^{3}	2.58×10^{3}	2.60×10^{3}
	Std	5.3	9.05	40.5	54.5	67.9	**7.13**	79.7	51	42.6	45.5	62.6	49.1	33
	−/=/+	–	–	–	+	–	–	–	–	–	–	+	+	/
	Rank	6	10	12	**1**	5	8	13	9	7	11	2	3	4
F11	Ave	2.97×10^{3}	3.14×10^{3}	2.90×10^{3}	**2.62×10^{3}**	2.90×10^{3}	3.06×10^{3}	3.30×10^{3}	2.88×10^{3}	2.78×10^{3}	2.91×10^{3}	2.65×10^{3}	2.67×10^{3}	**2.62×10^{3}**
	Std	1.31×10^{2}	1.58×10^{2}	**63.5**	65.2	1.09×10^{2}	89.2	5.29×10^{2}	97.9	1.32×10^{2}	83.1	82.1	1.16×10^{2}	76.1
	−/=/+	–	–	–	+	–	–	–	–	–	–	–	–	/
	Rank	10	12	**7**	**1**	8	11	13	6	5	9	3	4	2
F12	Ave	2.87×10^{3}	2.87×10^{3}	2.92×10^{3}	**2.86×10^{3}**	2.87×10^{3}	2.87×10^{3}	2.99×10^{3}	2.92×10^{3}	**2.86×10^{3}**	2.88×10^{3}	2.87×10^{3}	2.87×10^{3}	**2.86×10^{3}**
	Std	5.17	7.8	35.8	1.43	8.54	12.8	45.9	31.1	1.71	19.2	3.00=	**1.09**	1.55
	−/=/+	–	–	–	+	–	–	–	–	–	–	–	–	/
	Rank	6	7	12	**1**	8	9	13	11	3	10	5	4	2
−/=/+		11/0/1	11/0/1	11/0/1	6/0/6	12/0/0	11/0/1	12/0/0	11/0/1	11/0/1	12/0/0	9/0/3	9/0/3	/
Average rank		6.67	8.67	9.25	**2.33**	9.83	8.58	12.67	8.17	5.17	10.17	4.25	2.75	2.5
Total rank		6	9	10	**1**	11	8	13	7	5	12	4	3	2

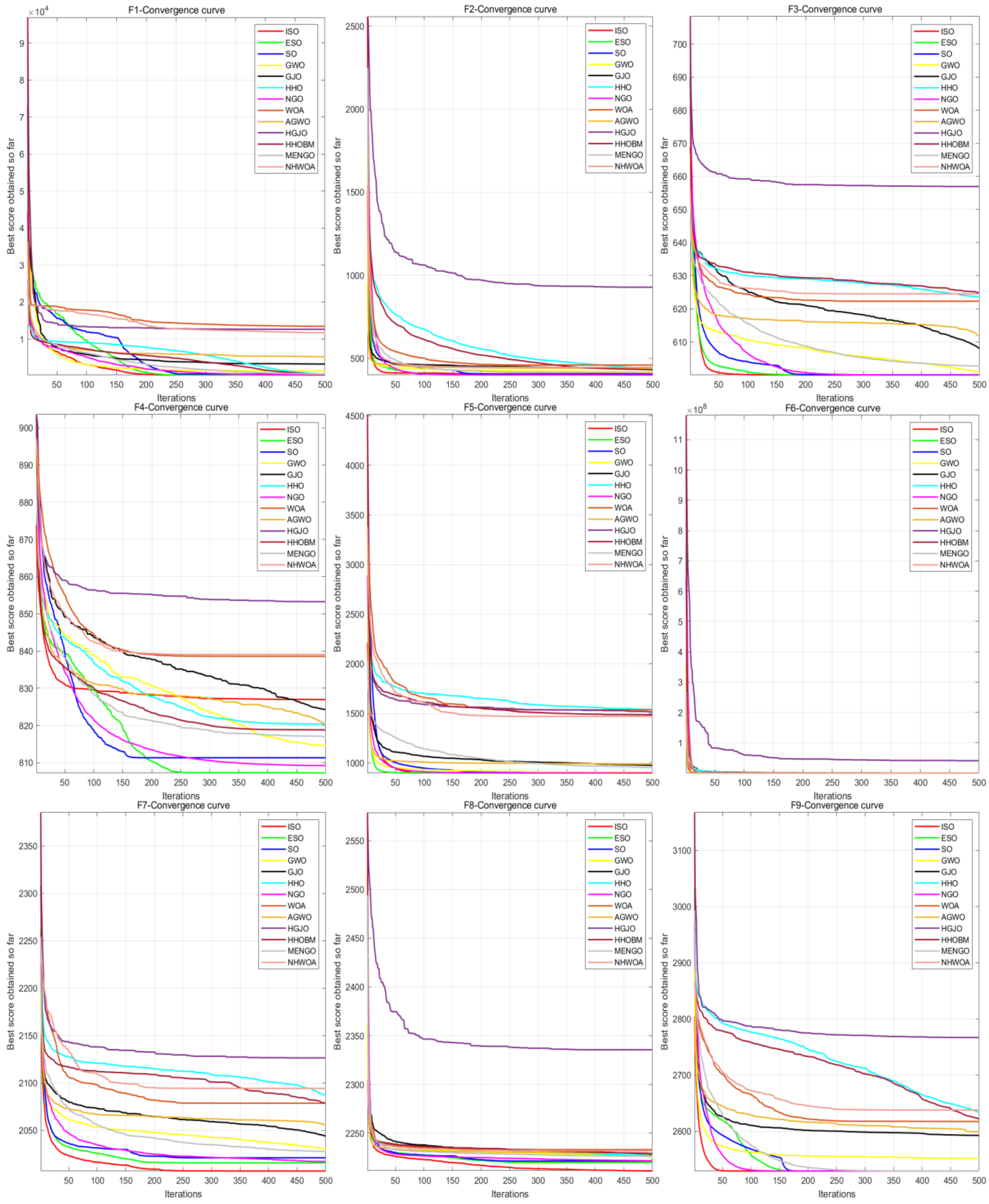
Table 6. Comparison results between ISO and other 12 algorithms on CEC2022 (Dim = 20).

F(x)		GWO	GJO	HHO	NGO	WOA	AGWO	HGJO	HHOBM	MENGO	NHWOA	SO	ESO	ISO
F1	Ave	1.23×10^4	1.55×10^4	2.49×10^4	1.48×10^4	3.33×10^4	1.92×10^4	5.32×10^4	1.66×10^4	1.75×10^4	3.08×10^4	1.65×10^4	1.36×10^4	**5.04×10^3**
	Std	3.41×10^3	6.28×10^3	1.15×10^4	2.42×10^4	1.16×10^4	3.58×10^3	1.85×10^4	7.82×10^3	3.95×10^3	1.08×10^4	3.15×10^3	4.39×10^3	**2.01×10^3**
	−/=/+	−	−	−	−	−	−	−	−	−	−	−	−	/
	Rank	2	5	10	4	12	9	13	7	8	11	6	3	**1**
F2	Ave	5.21×10^2	6.17×10^2	5.85×10^2	4.60×10^2	6.49×10^2	6.94×10^2	1.98×10^3	5.70×10^2	5.86×10^2	6.24×10^2	4.59×10^2	4.64×10^2	**4.50**$\times10^2$
	Std	45.3	87	32.9	12.8	30.3	56.6	4.27×10^2	37.9	38.5	43.1	**11.5**	18.3	12.2
	−/=/+	−	−	−	−	−	−	−	−	−	−	−	−	/
	Rank	5	9	7	3	11	12	13	6	8	10	2	4	**1**
F3	Ave	6.05×10^2	6.24×10^2	6.63×10^2	6.04×10^2	6.72×10^2	6.47×10^2	6.94×10^2	6.64×10^2	6.28×10^2	6.73×10^2	6.01×10^2	6.01×10^2	**6.00×10^2**
	Std	2.28	8.01	5.72	3.35	8.82	7.3	7.72	7	16.5	6.87	0.828	0.624	**0.169**
	−/=/+	−	−	−	−	−	−	−	−	−	−	−	−	/
	Rank	5	6	9	4	11	8	13	10	7	12	3	2	**1**
F4	Ave	8.56×10^2	8.93×10^2	8.95×10^2	8.68×10^2	9.32×10^2	8.99×10^2	9.76×10^2	8.96×10^2	8.84×10^2	9.24×10^2	**8.41×10^2**	8.44×10^2	8.82×10^2
	Std	31.7	27.2	13.7	9.68	29.1	19.1	14.9	15.1	16.6	28.7	21.2	12.1	**3.78**
	−/=/+	−	−	−	+	−	−	−	−	−	−	+	+	/
	Rank	3	7	8	4	12	10	13	9	6	11	**1**	2	5
F5	Ave	1.19×10^3	1.83×10^3	2.81×10^3	1.38×10^3	3.76×10^3	1.73×10^3	3.74×10^3	2.84×10^3	2.17×10^3	3.64×10^2	9.87×10^2	**9.29×10^2**	1.55×10^3
	Std	3.40×10^2	4.23×10^2	2.23×10^2	1.69×10^2	9.36×10^2	3.35×10^2	2.78×10^2	1.99×10^2	2.84×10^2	9.9×10^2	69.9	**54.8**	5.52×10^2
	−/=/+	+	−	−	+	−	−	−	−	−	−	+	**+**	/
	Rank	3	7	9	4	13	6	12	10	8	11	2	**1**	5
F6	Ave	2.52×10^5	1.63×10^7	1.56×10^5	3.53×10^3	4.01×10^6	8.98×10^6	8.44×10^8	1.24×10^5	4.81×10^3	2.15×10^6	5.56×10^3	3.79×10^3	**3.11×10^3**
	Std	5.66×10^5	1.73×10^7	8.46×10^4	1.18×10^3	4.34×10^6	1.28×10^7	5.87×10^8	6.76×10^4	1.22×10^3	3.73×10^6	2.53×10^3	1.6×10^3	**9.69×10^2**
	−/=/+	−	−	−	−	−	−	−	−	−	−	−	−	/
	Rank	8	12	7	2	10	11	13	6	4	9	5	3	**1**

Continued on next page

F(x)		GWO	GJO	HHO	NGO	WOA	AGWO	HGJO	HHOBM	MENGO	NHWOA	SO	ESO	ISO
F7	Ave	2.08×10^{3}	2.14×10^{3}	2.25×10^{3}	2.08×10^{3}	2.23×10^{3}	2.15×10^{3}	2.26×10^{3}	2.23×10^{3}	2.15×10^{3}	2.26×10^{3}	2.07×10^{3}	**2.06×10^{3}**	**2.06×10^{3}**
	Std	27.6	68.6	93.1	15.9	58.7	22.7	70.7	84.2	77.1	63.6	19.8	**14.8**	56.4
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	**+**	/
	Rank	5	6	11	4	9	7	13	10	8	12	3	**1**	2
F8	Ave	2.24×10^{3}	2.24×10^{3}	2.26×10^{3}	**2.23×10^{3}**	2.27×10^{3}	2.24×10^{3}	2.43×10^{3}	2.26×10^{3}	2.24×10^{3}	2.27×10^{3}	**2.23×10^{3}**	**2.23×10^{3}**	**2.23×10^{3}**
	Std	36.1	9.26	12.6	2.16	15	7.66	1.32×10^{2}	12.5	5.86	20.5	6.41	2.89	**1.32**
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	**/**
	Rank	8	7	10	2	11	6	13	9	5	12	4	3	**1**
F9	Ave	2.52×10^{3}	2.58×10^{3}	2.51×10^{3}	**2.48×10^{3}**	2.58×10^{3}	2.58×10^{3}	2.97×10^{3}	2.51×10^{3}	**2.48×10^{3}**	2.55×10^{3}	**2.48×10^{3}**	**2.48×10^{3}**	**2.48×10^{3}**
	Std	35	38.9	13.7	8.21×10^{-3}	36.9	36.4	1.90×10^{2}	14	2.02	30.8	3.77×10^{-2}	0.492	**2.70×10^{-5}**
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	**/**
	Rank	8	12	6	2	11	10	13	7	5	9	3	4	**1**
F10	Ave	3.83×10^{3}	4.12×10^{3}	3.98×10^{3}	2.52×10^{3}	5.11×10^{3}	4.61×10^{3}	6.01×10^{3}	3.85×10^{3}	2.56×10^{3}	5.19×10^{3}	2.74×10^{3}	2.60×10^{3}	**2.50×10^{3}**
	Std	8.40×10^{2}	1.55×10^{3}	5.82×10^{2}	**46.2**	7.89×10^{2}	1.22×10^{3}	1.45×10^{3}	6.28×10^{2}	1.53×10^{2}	4.03×10^{2}	2.18×10^{2}	1.14×10^{2}	81.3
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	**/**
	Rank	6	9	8	2	11	10	13	7	3	12	5	4	**1**
F11	Ave	3.40×10^{3}	3.72×10^{3}	3.31×10^{3}	2.99×10^{3}	3.46×10^{3}	3.79×10^{3}	7.07×10^{3}	3.29×10^{3}	3.31×10^{3}	3.39×10^{3}	**2.91×10^{3}**	2.98×10^{3}	2.94×10^{3}
	Std	1.84×10^{2}	2.75×10^{2}	59.4	41.8	94.9	2.07×10^{2}	8.73×10^{2}	95.7	80.1	72.2	**6.18**	1.17×10^{2}	49
	−/=/+	–	–	–	–	–	–	–	–	–	–	+	–	/
	Rank	9	11	6	4	10	12	13	5	7	8	**1**	3	2
F12	Ave	2.97×10^{3}	3.01×10^{3}	3.22×10^{3}	2.95×10^{3}	3.10×10^{3}	3.09×10^{3}	3.82×10^{3}	3.20×10^{3}	2.97×10^{3}	3.13×10^{3}	3.00×10^{3}	2.96×10^{3}	**2.94×10^{3}**
	Std	19.4	38.8	95.1	7.95	80.1	49.1	2.37×10^{2}	1.09×10^{2}	19.5	86.6	20.7	9.4	**4.27**
	−/=/+	–	–	–	–	–	–	–	–	–	–	–	–	/
	Rank	4	7	12	2	9	8	13	11	5	10	6	3	**1**
−/=/+		11/0/1	12/0/0	12/0/0	10/0/2	12/0/0	12/0/0	12/0/0	12/0/0	12/0/0	12/0/0	9/0/3	9/0/3	/
Average rank		5.5	8.17	8.58	3.08	10.83	9.08	12.91	8.08	6.17	10.58	3.42	2.75	**1.83**
Total rank		5	8	9	3	12	10	13	7	6	11	4	2	**1**

Figure 3 plots the convergence curves of the 13 algorithms on CEC2022 when the dimension is set to 10. From Figure 3, we can see that ISO converges at a faster rate than other algorithms, except for the poor performance on functions F4 and F10. In particular, ESO has the best optimization capability on function F4, and NGO achieves the best results on function F10. On function F2, although ISO converges quickly in the early stage, the optimization ability decreases in the late stage.



Continued on next page

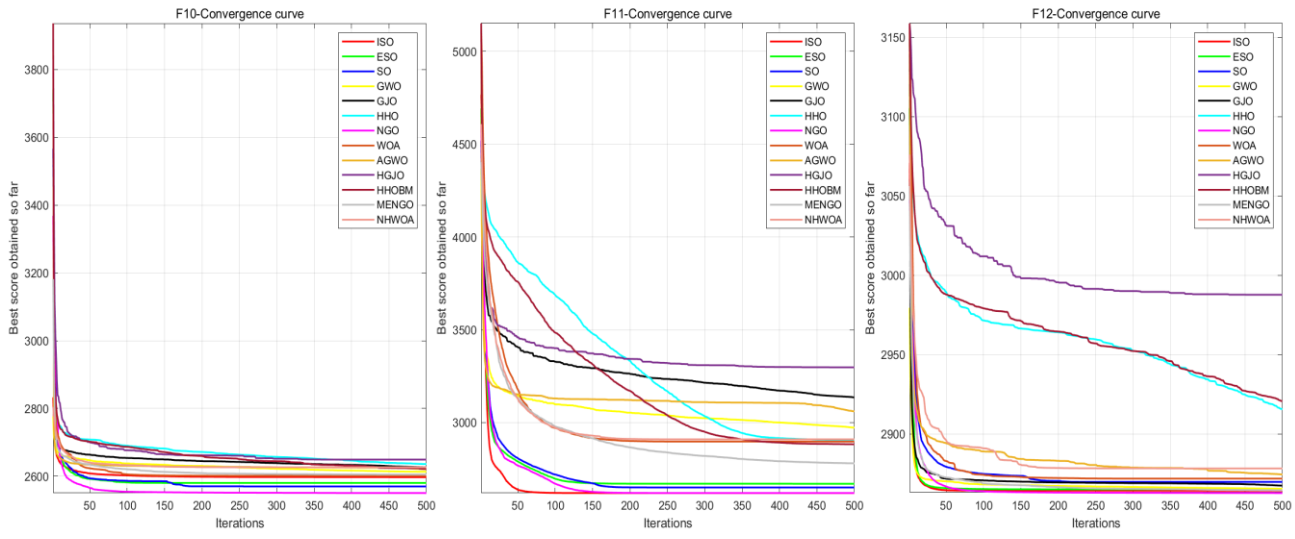
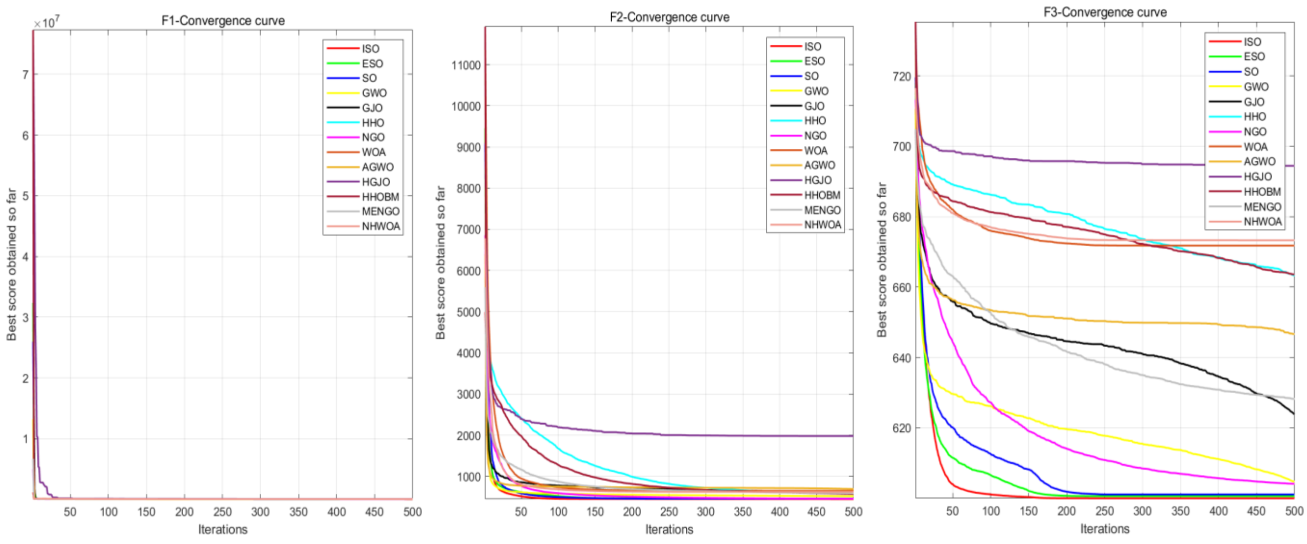


Figure 3. Convergence curves of 13 algorithms on CEC2022 (Dim = 10).

From Table 6, when the dimension of CEC2022 functions is set to 20, NGO shows weakness and ISO gradually gets the upper hand. Generally, ISO obtains the first rank with an average rank of 1.83, followed by ESO, and NGO ranks third. Among the 12 CEC2022 functions, ISO ranks fifth on functions F4 and F5, while ranking in the top two on the other 10 functions. It can be seen that as the dimension increases, the performance of ISO improves.

Figure 4 plots the convergence curves of the 13 algorithms on CEC2022, when the dimension is set to 20. As can be seen from Figure 4, ISO falls into a local optimum on functions F4 and F5, and SO converges faster than ISO on both functions. On function F7, ESO performs the best and ISO is slightly weaker in the later iterations. All in all, ISO keeps a faster rate to seek out the optimal value.



Continued on next page

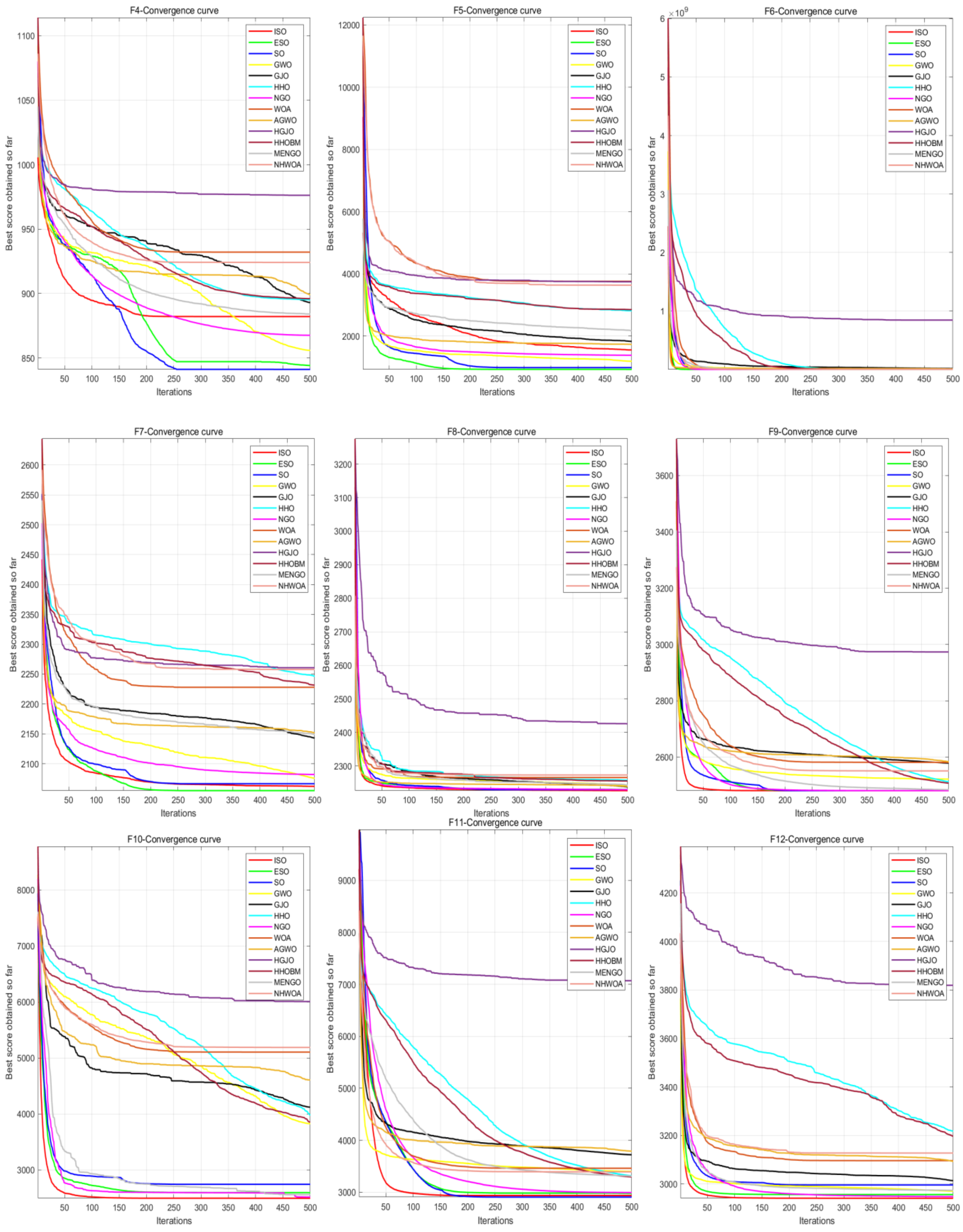


Figure 4. Convergence curves of 13 algorithms on CEC2022 (Dim = 20).

4.3.3. Ablation experiment

Ablation experiment is performed on 23 classical benchmark test functions, and there exist some differences in error between this experiment and the experiment in Subsection 4.3.1, because we mainly investigate the influence of three strategies and their portfolios on SO. Table 7 lists all possible SO variants under three strategies (seven in total), where 1 or 0 indicates whether the strategy is included or not. For example, MSO only has the mirror opposition-based learning mechanism, and NDSO contains the novel evolutionary population dynamics model and differential evolution strategy. Table 8 shows the experimental results, where "−/=/+" denotes that the improved algorithm is lower, equal to or better than SO, respectively. The convergence curves are given in Figure 5.

Table 7. Different SO variants based on three strategies.

Strategy	MSO	NSO	DSO	MNSO	MDSO	NDSO	ISO
MOBL	1	0	0	1	1	0	1
NEPD	0	1	0	1	0	1	1
DES	0	0	1	0	1	1	1

From Table 8, it can be found that ISO has the best optimization capability among the seven SO variants, and MDSO and MNSO rank the second and third, respectively. This indicates that the mirror opposition-based learning mechanism is the most effective factor in improving SO, and the other two strategies can also play a supportive role.

It is worth noting that the NSO and DSO, which contain only one strategy, are ranked lower than SO. This represents that it is not always possible to achieve better results by improving the original algorithm. A strategy may be very effective for a certain problem or function, but for other problems, it may perform poorly. For example, DSO achieves the best optimization results with the first rank on function F18, but it ranks last on function F21.

On the unimodal test functions, the MSO, MNSO, MDSO, and ISO algorithms which contain MOBL achieve higher convergence accuracy with fewer iterations. While the optimization accuracy of DSO and NDSO is reduced, they converge a little faster than SO. The result is a little worse for NSO, which does not converge as fast or accurately as SO. On the multimodal test functions, the situation is not much changed from the previous ones by and large. The only difference is that on function F8, the convergence profiles of various SO variants become easily distinguishable. It can be seen from the figure that the convergence speed of MSO, NSO and MNSO in the early stage is not improved effectively. On the multimodal test functions with fixed dimension, eight algorithms converge more complexly than the previous 13 test functions. On function F14, ISO, MSO, MNSO, and MDSO are slower than SO in the early stages and also all fall into local optimum, but ISO and MNSO converge continuously to get closer to the optimal value in the later iterations. Meanwhile, although the convergence rate of DSO is faster on function F14, it is a bit weak on the late optimization search process and is caught up by other SO variants. On functions F15–F20, ISO and MDSO still hold the lead, followed by DSO and NDSO. MSO and MNSO are more unstable. For example, they perform better than SO on function F15, whereas worse on function F17. NSO is the worst one, which barely beats SO. On functions F21-F23, MNSO takes the first place on function F21, and ISO and MDSO rank first on the other two functions.

Table 8. Comparison results between SO and other 7 improved algorithms on 23 classical benchmark functions.

F(x)		MSO	NSO	DSO	MNSO	MDSO	NDSO	ISO	SO
F1	Ave	**0**	3.46×10^{-46}	5.48×10^{-59}	**0**	**0**	1.01×10^{-58}	**0**	1.05×10^{-96}
	Std	**0**	1.42×10^{-45}	1.84×10^{-58}	**0**	**0**	5.52×10^{-58}	**0**	3.11×10^{-96}
	−/=/+	+	−	−	+	+	−	+	/
	Rank	**1**	8	6	**1**	**1**	7	**1**	5
F2	Ave	**0**	3.56×10^{-19}	1.56×10^{-28}	**0**	**0**	4.18×10^{-31}	**0**	2.41×10^{-45}
	Std	**0**	1.35×10^{-18}	3.45×10^{-28}	**0**	**0**	1.21×10^{-30}	**0**	4.40×10^{-45}
	−/=/+	+	−	−	+	+	−	+	/
	Rank	**1**	8	7	**1**	**1**	6	**1**	5
F3	Ave	**0**	2.40×10^{-30}	2.91×10^{-24}	**0**	**0**	2.51×10^{-28}	**0**	1.32×10^{-57}
	Std	**0**	9.31×10^{-30}	1.56×10^{-23}	**0**	**0**	1.33×10^{-27}	**0**	7.20×10^{-57}
	−/=/+	+	−	−	+	+	−	+	/
	Rank	1	6	8	**1**	**1**	7	**1**	5
F4	Ave	**0**	5.03×10^{-16}	2.95×10^{-18}	**0**	**0**	8.07×10^{-17}	**0**	7.91×10^{-42}
	Std	**0**	1.71×10^{-15}	9.67×10^{-18}	**0**	**0**	4.30×10^{-16}	**0**	1.56×10^{-41}
	−/=/+	+	−	−	+	+	−	+	/
	Rank	**1**	8	6	**1**	**1**	7	**1**	5
F5	Ave	15.9	0.606	26.9	0.752	15.3	0.307	**0.239**	24.7
	Std	12.8	0.956	24.4	3.78	10.7	0.569	**0.542**	20.4
	−/=/+	+	+	−	+	+	+	+	/
	Rank	6	3	8	4	5	2	**1**	7
F6	Ave	0.375	1.56×10^{-2}	9.62×10^{-2}	8.12×10^{-2}	**2.53×10^{-8}**	1.14×10^{-4}	3.22×10^{-8}	0.143
	Std	0.254	2.98×10^{-2}	0.11	0.133	**4.84×10^{-8}**	1.49×10^{-4}	6.33×10^{-8}	0.135
	−/=/+	−	+	+	+	+	+	+	/
	Rank	8	4	6	5	**1**	3	2	7
F7	Ave	3.36×10^{-5}	5.97×10^{-4}	5.84×10^{-4}	2.58×10^{-5}	**2.25×10^{-5}**	7.94×10^{-4}	2.94×10^{-5}	2.08×10^{-4}
	Std	2.84×10^{-5}	4.01×10^{-4}	3.51×10^{-4}	2.36×10^{-5}	**2.31×10^{-5}**	4.46×10^{-4}	3.18×10^{-5}	1.62×10^{-46}
	−/=/+	+	−	−	+	+	−	+	/
	Rank	4	7	6	2	**1**	8	3	5
F8	Ave	**-1.26×10^{4}**	**-1.26×10^{4}**	-1.25×10^{4}	**-1.26×10^{4}**	**-1.26×10^{4}**	**-1.26×10^{4}**	**-1.26×10^{4}**	-1.24×10^{4}
	Std	39.3	2.57	46.7	0.46	65.6	**0.41**	2.24	2.58×10^{2}
	−/=/+	+	+	+	+	+	+	+	/
	Rank	5	4	7	2	6	**1**	3	8
F9	Ave	**0**	**0**	4.13	**0**	**0**	**0**	**0**	4.81
	Std	**0**	**0**	6.79	**0**	**0**	**0**	**0**	9.9
	−/=/+	+	+	+	+	+	+	+	/
	Rank	**1**	**1**	7	**1**	**1**	**1**	**1**	8
F10	Ave	**8.88×10^{-16}**	6.34×10^{-15}	4.56×10^{-15}	**8.88×10^{-16}**	**8.88×10^{-16}**	4.44×10^{-15}	**8.88×10^{-16}**	4.32×10^{-15}
	Std	**0**	1.80×10^{-15}	6.49×10^{-16}	**0**	**0**	**0**	**0**	6.49×10^{-16}
	−/=/+	+	−	−	+	+	−	+	/
	Rank	**1**	8	7	**1**	**1**	6	**1**	5

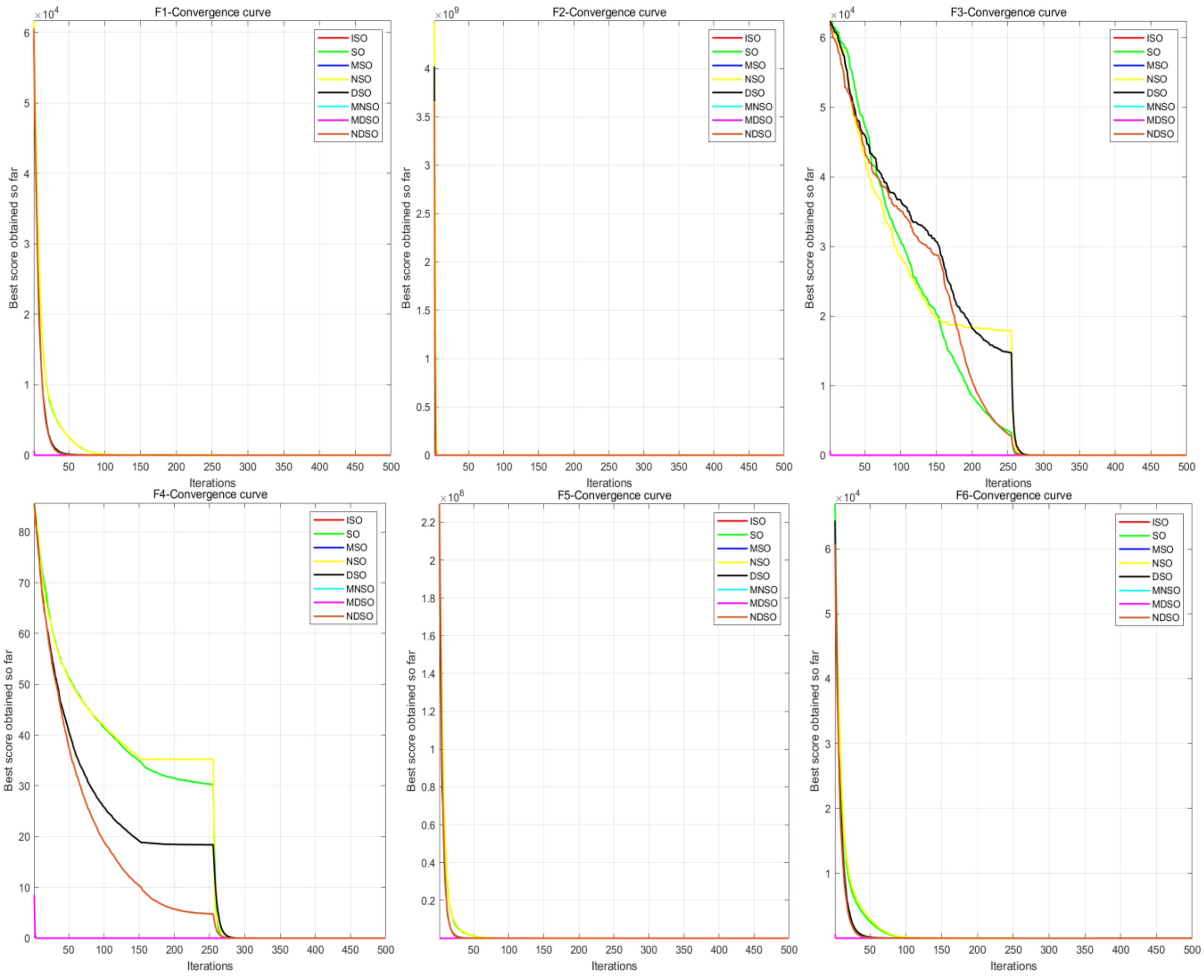
Continued on next page

F(x)		MSO	NSO	DSO	MNSO	MDSO	NDSO	ISO	SO
F11	Ave	**0**	**0**	2.43×10^{-3}	**0**	**0**	2.08×10^{-3}	**0**	2.40×10^{-2}
	Std	**0**	**0**	4.66×10^{-3}	**0**	**0**	4.18×10^{-3}	**0**	3.51×10^{-2}
	−/=/+	+	+	+	+	+	+	+	/
	Rank	**1**	**1**	8	**1**	**1**	6	**1**	7
F12	Ave	3.63×10^{-4}	1.92×10^{-4}	7.23×10^{-3}	1.43×10^{-4}	**6.87×10^{-10}**	1.69×10^{-6}	6.46×10^{-9}	9.08×10^{-2}
	Std	5.41×10^{-4}	2.94×10^{-4}	6.98×10^{-3}	1.59×10^{-4}	**1.09×10^{-9}**	2.37×10^{-6}	2.81×10^{-8}	0.22
	−/=/+	+	+	+	+	+	+	+	/
	Rank	6	5	7	4	**1**	3	2	8
F13	Ave	0.121	4.80×10^{-3}	7.85×10^{-3}	3.20×10^{-3}	3.67×10^{-4}	6.73×10^{-6}	**1.71×10^{-8}**	0.211
	Std	0.127	1.35×10^{-2}	9.36×10^{-3}	1.25×10^{-2}	2.01×10^{-3}	8.16×10^{-6}	**2.56×10^{-8}**	0.197
	−/=/+	+	+	+	+	+	+	+	/
	Rank	7	5	6	4	3	2	**1**	8
F14	Ave	1.13	**0.998**	**0.998**	**0.998**	1.23	**0.998**	**0.998**	1.07
	Std	0.503	1.63×10^{-7}	1.45×10^{-3}	4.08×10^{-6}	0.958	**5.83×10^{-17}**	1.47×10^{-11}	0.252
	−/=/+	−	+	+	+	+	+	+	/
	Rank	7	3	5	4	8	**1**	2	6
F15	Ave	**3.07×10^{-4}**	4.72×10^{-4}	7.15×10^{-4}	**3.07×10^{-4}**	**3.07×10^{-4}**	3.69×10^{-4}	**3.07×10^{-4}**	4.61×10^{-4}
	Std	5.33×10^{-12}	8.56×10^{-5}	1.82×10^{-4}	1.75×10^{-12}	3.94×10^{-19}	2.32×10^{-4}	**3.80×10^{-19}**	1.81×10^{-4}
	−/=/+	+	−	−	+	+	+	+	/
	Rank	4	7	8	3	2	5	**1**	6
F16	Ave	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**	**–1.03**
	Std	**5.90×10^{-16}**	1.00×10^{-15}	6.71×10^{-16}	6.05×10^{-16}	6.78×10^{-16}	6.78×10^{-16}	6.78×10^{-16}	**5.90×10^{-16}**
	−/=/+	=	−	−	−	−	−	−	/
	Rank	**1**	8	3	7	4	4	4	**1**
F17	Ave	**0.398**	**0.398**	**0.398**	**0.398**	**0.398**	**0.398**	**0.398**	**0.398**
	Std	**0**	1.98×10^{-6}	**0**	**0**	**0**	**0**	**0**	**0**
	−/=/+	=	−	=	=	=	=	=	/
	Rank	**1**	8	**1**	**1**	**1**	**1**	**1**	**1**
F18	Ave	**3**	**3**	**3**	**3**	**3**	**3.00**	**3**	**3**
	Std	2.11×10^{-15}	2.59×10^{-5}	**1.22×10^{-15}**	2.88×10^{-15}	1.33×10^{-15}	2.04×10^{-15}	2.06×10^{-15}	1.84×10^{-15}
	−/=/+	−	−	+	−	+	−	−	/
	Rank	6	7	**1**	8	2	4	5	3
F19	Ave	**–3.86**	**–3.86**	**–3.86**	**–3.86**	**–3.86**	**–3.86**	**–3.86**	**–3.86**
	Std	2.61×10^{-15}	1.41×10^{-15}	2.71×10^{-15}	2.68×10^{-15}	2.71×10^{-15}	2.71×10^{-15}	2.71×10^{-15}	**2.49×10^{-15}**
	−/=/+	−	−	−	−	−	−	−	/
	Rank	2	8	4	3	4	4	4	**1**
F20	Ave	**–3.32**	**–3.32**	**–3.32**	**–3.32**	**–3.32**	**–3.32**	**–3.32**	**–3.32**
	Std	1.42×10^{-15}	2.50×10^{-4}	2.29×10^{-15}	1.36×10^{-15}	1.36×10^{-15}	1.36×10^{-15}	**1.34×10^{-15}**	1.82×10^{-15}
	−/=/+	+	−	−	+	+	+	+	/
	Rank	5	8	7	2	2	2	**1**	6
F21	Ave	**–10.2**	–10.1	–10.0	**–10.2**	**–10.2**	**–10.2**	**–10.2**	–10.1
	Std	1.59×10^{-2}	2.56×10^{-2}	0.393	**5.71×10^{-15}**	7.12×10^{-15}	1.05×10^{-4}	7.17×10^{-15}	0.105
	−/=/+	+	+	−	+	+	+	+	/
	Rank	5	6	8	**1**	2	4	3	7

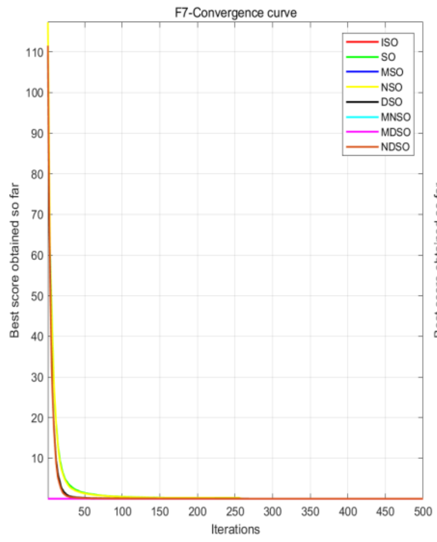
Continued on next page

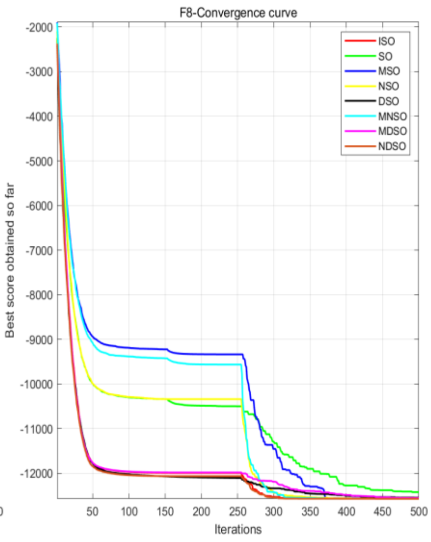
F(x)		MSO	NSO	DSO	MNSO	MDSO	NDSO	ISO	SO
F22	Ave	**–10.4**	**–10.4**	–10.3	**–10.4**	**–10.4**	**–10.4**	**–10.4**	–10.3
	Std	**0**	2.63×10^{-2}	0.14	4.66×10^{-16}	1.28×10^{-15}	1.29×10^{-5}	1.36×10^{-15}	0.496
	−/=/+	+	+	+	+	+	+	+	/
	Rank	**1**	6	7	2	3	5	4	8
F23	Ave	**–10.5**	**–10.5**	**–10.5**	**–10.5**	**–10.5**	**–10.5**	**–10.5**	–10.3
	Std	1.58×10^{-15}	1.50×10^{-2}	3.04×10^{-2}	1.58×10^{-15}	1.81×10^{-15}	$\mathbf{5.71\times10^{-16}}$	1.81×10^{-15}	0.522
	−/=/+	+	+	+	+	+	+	+	/
	Rank	2	6	7	2	4	**1**	4	8
−/=/+		4/2/17	12/0/11	13/1/9	3/1/19	3/1/19	9/1/13	3/1/19	/
Average rank		3.35	5.87	6.09	2.61	2.43	3.91	**2.09**	5.65
Total rank		4	7	8	3	2	5	**1**	6

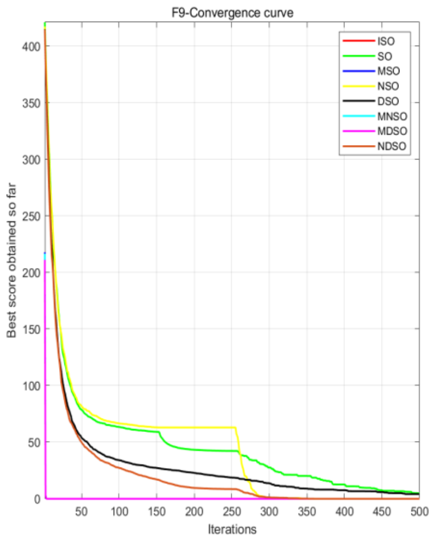
From the convergence curves in Figure 5, we notice that the convergence processes of ISO and MDSO have no significant difference on most of the classical test set, and they both possess excellent optimization capabilities. Only on functions F5, F14, and F21, MDSO performs worse than ISO. Other SO variants have a mediocre convergence effect, and may even converge worse than SO on some functions. Therefore, we conclude that ISO exhibits better convergence behavior.

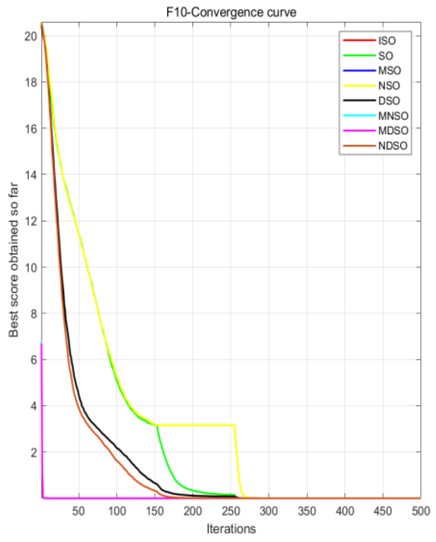


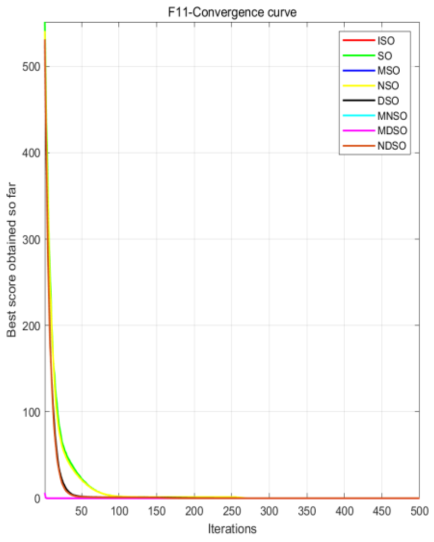
Continued on next page

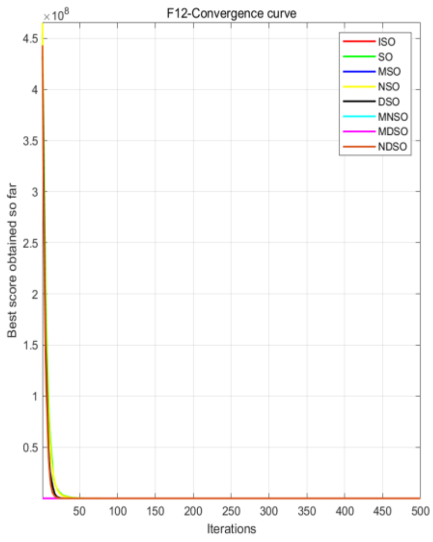


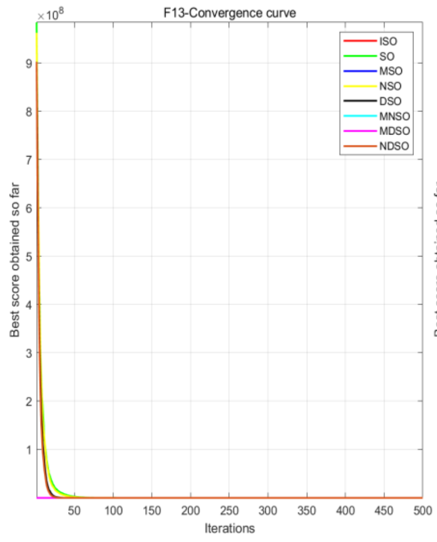


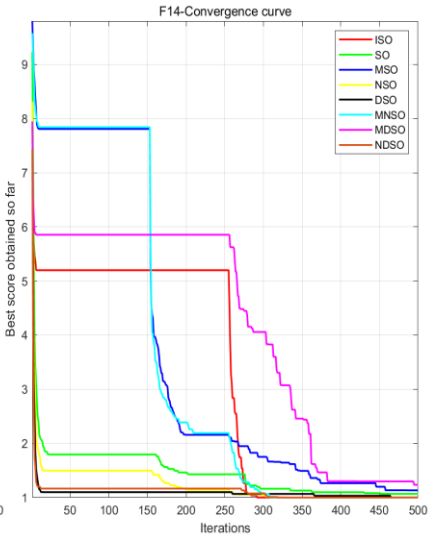


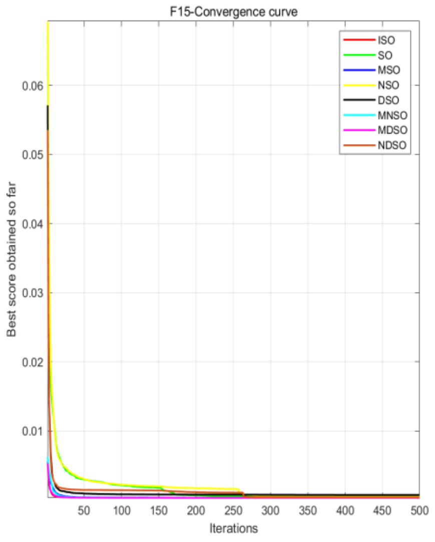












Continued on next page

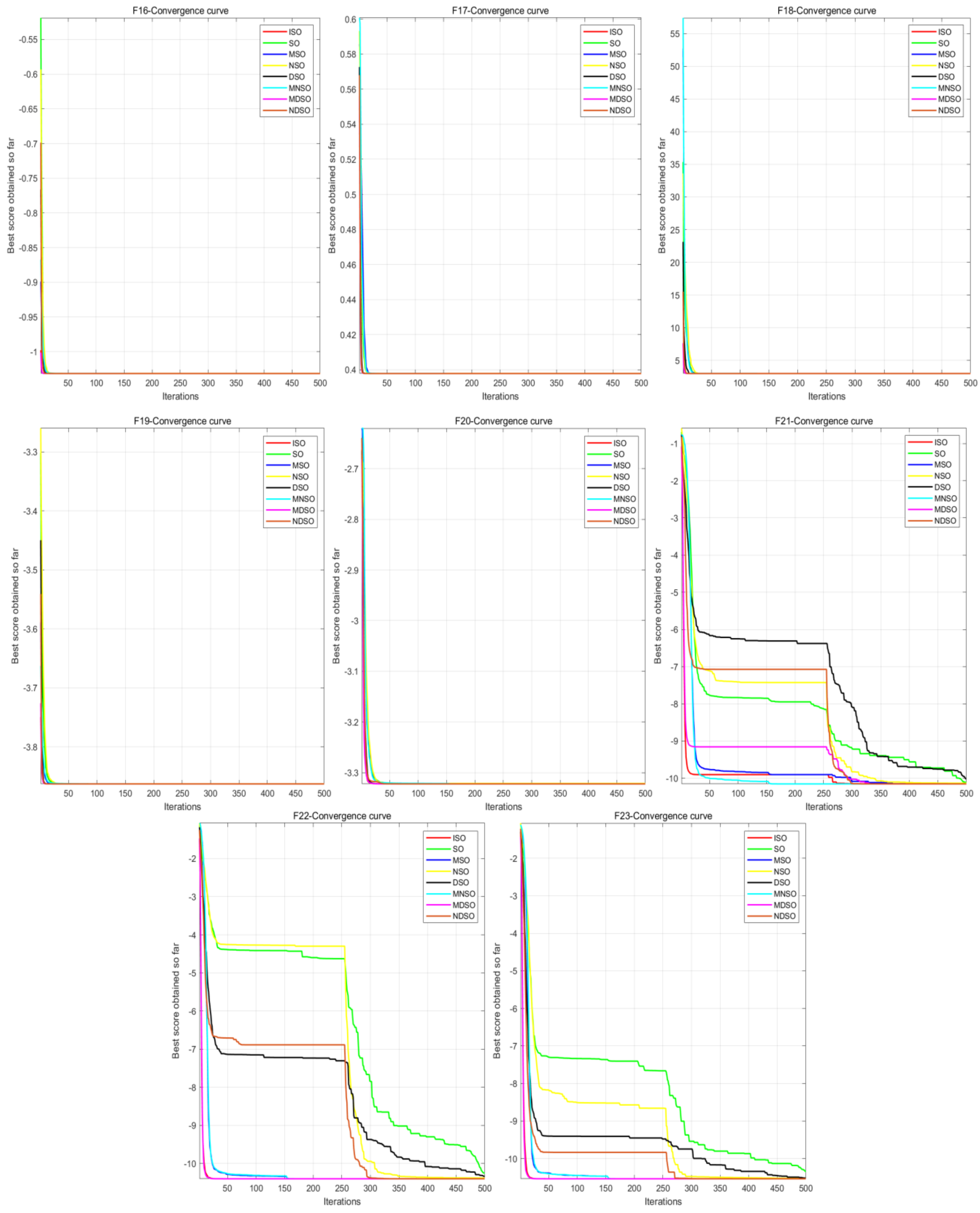


Figure 5. Convergence curves of 8 algorithms on 23 classical benchmark functions.

4.3.4. SVM parameter optimization experiment

SVM integrates several standard machine learning techniques to overcome the problems of local minima, dimensional catastrophe and so on, and it provides better generalization capabilities than traditional neural network learning algorithms. However, the problem of parameter selection that affects its performance has not been solved.

The optimal selection of the SVM parameters is a key step that directly affects the performance of SVM, which has become a major research field in the area of kernel methods. For instance, Kalita et al. [45] proposed a dynamic framework based on moth flame optimization (MFO) and knowledge-based-search (KBS) to optimize the penalty factor C and Gaussian kernel parameter σ. Their experiments showed that KBS helps in controlling the exponential growth of time complexity when MFO is used to optimize these two parameters. Huang et al. [46] introduced an improved black widow optimization (IBWO) algorithm and constructed the IBWO-SVM to select better parameters. From these two articles, it can be seen that optimizing model parameters by swarm intelligence algorithms can effectively improve the prediction accuracy. Literature [47–49] also utilized this type of method and achieved good results.

The parameter setting of the SVM classification experiment is shown in Table 9. The classification accuracy in the test set is used as the measure criterion. In order to reduce the random error of the experiment results, we conduct 10 rounds of experiments and the average values are shown in Table 10.

Table 9. Parameter setting of SVM classification experiment.

Parameters	Value or Range
Population size (N)	50
Maximum number of iterations (T)	500
Dimension (Dim)	2
Penalty factor (C)	[0.01,100]
Gaussian kernel width (σ)	[0.01,100]

Table 10. Classification results of SVM after parameter optimization by swarm intelligence algorithms.

Name	GWO-SVM	GJO-SVM	HHO-SVM	NGO-SVM	WOA-SVM	SO-SVM	ISO-SVM
Iris	95.47%	95.47%	**96.00%**	95.20%	92.00%	95.47%	**96.00%**
Parkinsons	81.33%	81.12%	81.02%	80.92%	75.51%	81.33%	**81.53%**
Fire	81.53%	81.27%	80.47%	79.67%	34.00%	81.40%	**81.60%**
Heart	85.95%	86.42%	85.74%	85.88%	75.00%	86.08%	**86.49%**
Ionosphere	94.26%	94.32%	94.32%	94.32%	64.77%	94.15%	**94.38%**

Analyzing the data in Table 9, it can be seen that the classification accuracy based on ISO is higher than SO in all five datasets. Compared with other algorithms, ISO has the same classification result as HHO on the Iris dataset, and for the remaining four datasets, ISO achieves the highest accuracy. It can be concluded that ISO is more effective and stronger than SO, and when applied to the optimization of SVM parameters, it can achieve significant results.

4.4. Further discussions

In the function comparison experiments with the above 12 algorithms, we can see that ISO has better optimization performance and convergence speed. However, it has some weaknesses compared to other evolutionary algorithms. First, ISO may not be as effective as some evolutionary algorithms in global search, resulting in the search result to hover around a local optimum. Second, due to the fact that the performance of ISO is more sensitive to parameter settings, different problems may require different parameter configurations, adding a certain degree of complexity.

When dealing with nonlinear optimization problems, the advantage of ISO lies in its flexibility and adaptability (i.e., robustness), allowing it to adapt to complex changes. Non-convex optimization problems usually involve complex objective functions that may contain multiple local minimums and complex structures [50]. However, the three improvement strategies of ISO can greatly improve the algorithm's ability to jump out of local optima. In practice, noisy is a common problem; it contains uncertain variables, both multivariate and multi-objective. The key challenge for ISO in optimization problems involving uncertain variables is how to deal with these uncertainties. As it stands now, ISO has no clear superiority. We have two ideas about the solution to this problem. The first is the decomposition mechanism proposed by Deng et al. [51]. This mechanism can resolve such uncertain variables separately to further improve the optimization performance. The second combines the ISO with deep learning to create a new hybrid model. Literature [52–54] describes the advantages of approaches based on deep learning. ISO has strong potential in multivariate and multi-objective optimization problems. This is due to the fact that in the function test experiment of CEC2022, ISO ranks second among all the 13 algorithms with an average ranking of 2.5 when the dimension is set to 10, whereas when the dimension is increased to 20, ISO takes the first place with an average ranking of 1.83. Overall, the ISO has certain advantages in dealing with complex optimization problems, although it also faces many challenges.

5. Conclusions

In this article, ISO is presented with three improvement strategies, and combined with SVM for the first time to provide an effective solution to the SVM parameter optimization problem. From the experimental results, it can be seen that the proposed ISO is effective and the ISO-optimized SVM parameters improves the classification accuracy by 0.2–0.5% over the SO-optimized SVM parameters. However, the proposed method still has the potential to upgrade. For example, although ISO and SO are consistent in time complexity, ISO may take more time in practice. This requires attention to whether mechanisms with high time overhead in the algorithm can be replaced with simple and efficient strategies. Overall, in future research, we will focus mainly on the improvement of ISO and apply it to solve more complex problems in a wider range of fields.

Use of AI tools declaration

The author declares he has not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This paper is supported in part by the Natural Science Foundation of Jiangxi Province of China (No. 20242BAB26024) and Program of Academic Degree and Graduate Education and Teaching

Reform in Jiangxi Province of China (No. JXYJG-2022-172).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. C. Cortes, V. N. Vapnik, Support-vector networks, *Mach. Learn.*, **20** (1995), 273–297. https://doi.org/10.1007/BF00994018
2. M. V, D. T, M. Kalaiyarasi, Classification of newspaper article classification by employing support vector machine in comparison with perceptron to improve accuracy, in *2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, (2023), 1–6. https://doi.org/10.1109/ICONSTEM56934.2023.10142309
3. Q. Wang, R. Peng, J. Wang, Y. Xie, Y. Zhou, Research on text classification method of LDA-SVM based on PSO optimization, in *2019 Chinese Automation Congress (CAC)*, (2019), 1974–1978. https://doi.org/10.1109/CAC48633.2019.8996952
4. P. Preethi, H. R. Mamatha, Region-based convolutional neural network for segmenting text in epigraphical images, *Artif. Intell. Appl.*, **1** (2022), 119–127, https://doi.org/10.47852/bonviewAIA2202293
5. A. P. Baldovino, F. N. Vergonio, J. P. Tomas, Child attention detection through facial expression recognition using SVM algorithm, in *Proceedings of the 2019 International Conference on Information Technology and Computer Communications*, (2019), 52–58. https://doi.org/10.1145/3355402.3355411
6. H. Zhou, G. Yu, Research on pedestrian detection technology based on the SVM classifier trained by HOG and LTP features, *Future Gener. Comp. Syst.*, **125** (2021), 604–615. https://doi.org/10.1016/j.future.2021.06.016
7. Y. Xu, Y. Li, C. K. Ahn, X. Chen, Seamless indoor pedestrian tracking by fusing INS and UWB measurements via LS-SVM assisted UFIR filter, *Neurocomputing*, **388** (2020), 301–308. https://doi.org/10.1016/j.neucom.2019.12.121
8. J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT press, Cambridge, 1992.
9. R. Storn, K. Price, Differential evolution–A simple and efficient adaptive scheme for global optimization over continuous spaces, *J. Global Optim.*, **23** (1997), 341–359. https://doi.org/10.1023/A:1008202821328
10. Z. Hu, X. Xu, Q. Su, H. Zhu, J. Guo, Grey prediction evolution algorithm for global optimization, *Appl. Math. Model.*, **79** (2020), 145–160. https://doi.org/10.1016/j.apm.2019.10.026
11. I. Segovia-Domínguez, R. Herrera-Guzmán, J. P. Serrano-Rubio, A. Hernández-Aguirre, Geometric probabilistic evolutionary algorithm, *Expert Syst. Appl.*, **144** (2020), 113080. https://doi.org/10.1016/j.eswa.2019.113080
12. D. Bertsimas, J. Tsitsiklis, Simulated Annealing, *Stat. Sci.*, **8** (1993), 10–15.
13. S. Mirjalili, SCA: A sine cosine algorithm for solving optimization problems, *Knowl.-Based Syst.*, **96** (2016), 120–133. https://doi.org/10.1016/j.knosys.2015.12.022
14. E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A gravitational search algorithm, *Inform. Sciences*, **179** (2009), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

15. W. Zhao, L. Wang, Z. Zhang, Atom search optimization and its application to solve a hydrogeologic parameter estimation problem, *Knowl.-Based Syst.*, **163** (2019), 283–304. https://doi.org/10.1016/j.knosys.2018.08.030
16. Anita, A. Yadav, AEFA: Artificial electric field algorithm for global optimization, *Swarm Evol. Comput.*, **48** (2019), 93–108. https://doi.org/10.1016/j.swevo.2019.03.013
17. M. Cheng, M. N. Sholeh, Optical microscope algorithm: A new metaheuristic inspired by microscope magnification for solving engineering optimization problems, *Knowl.-Based Syst.*, **279** (2023), 110939. https://doi.org/10.1016/j.knosys.2023.110939
18. R. V. Rao, V. J. Savsani, D. P. Vakharia, Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems, *Comput. Aided Design*, **43** (2011), 303–315. https://doi.org/10.1016/j.cad.2010.12.015
19. M. Kumar, A. J. Kulkarni, S. C. Satapathy, Socio evolution & learning optimization algorithm: A socio-inspired optimization methodology, *Future Gener. Comput. Syst.*, **81** (2018), 252–272. https://doi.org/10.1016/j.future.2017.10.052
20. P. Zhang, L. Wang, Z. Fei, L. Wei, M. Fei, M. I. Menhas, A novel human learning optimization algorithm with Bayesian inference learning, *Knowl.-Based Syst.*, **271** (2023), 110564. https://doi.org/10.1016/j.knosys.2023.110564
21. B. Das, V. Mukherjee, D. Das, Student psychology based optimization algorithm: A new population based optimization algorithm for solving optimization problems, *Adv. Eng. Softw.*, **146** (2020), 102804. https://doi.org/10.1016/j.advengsoft.2020.102804
22. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Softw.*, **69** (2014), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
23. M. S. Kiran, TSA: Tree-seed algorithm for continuous optimization, *Expert Syst. Appl.*, **42** (2015), 6686–6698. https://doi.org/10.1016/j.eswa.2015.04.055
24. S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Softw.*, **95** (2016), 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
25. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872. https://doi.org/10.1016/j.future.2019.02.028
26. N. Chopra, M. M. Ansari, Golden jackal optimization: A novel nature-inspired optimizer for engineering applications, *Expert Syst. Appl.*, **198** (2022), 116924. https://doi.org/10.1016/j.eswa.2022.116924
27. M. Dehghani, Š. Hubálovský, P. Trojovský, Northern goshawk optimization: A new swarm-based algorithm for solving optimization problems, *IEEE Access*, **9** (2021), 162059–162080. https://doi.org/10.1109/ACCESS.2021.3133286
28. C. Ma, H. Huang, Q. Fan, J. Wei, Y. Du, W. Gao, Grey wolf optimizer based on Aquila exploration method, *Expert Syst. Appl.*, **205** (2022), 117629. https://doi.org/10.1016/j.eswa.2022.117629
29. H. Kang, R. Liu, Y. Yao, F. Yu, Improved Harris hawks optimization for non-convex function optimization and design optimization problems, *Math. Comput. Simulat.*, **204** (2023), 619–639. https://doi.org/10.1016/j.matcom.2022.09.010
30. T. Lou, Z. Yue, Y. Jiao, Z. He, A hybrid strategy-based GJO algorithm for robot path planning, *Expert Syst. Appl.*, **238** (2024), 121975. https://doi.org/10.1016/j.eswa.2023.121975
31. K. Li, H. Huang, S. Fu, C. Ma, Q. Fan, Y. Zhu, A multi-strategy enhanced northern goshawk optimization algorithm for global optimization and engineering design problems, *Comput. Method. Appl. Mech. Eng.*, **415** (2023), 116199. https://doi.org/10.1016/j.cma.2023.116199

32. X. Lin, X. Yu, W. Li, A heuristic whale optimization algorithm with niching strategy for global multi-dimensional engineering optimization, *Comput. Ind. Eng.*, **171** (2022), 108361. https://doi.org/10.1016/j.cie.2022.108361
33. F. A. Hashim, A. G. Hussien, Snake optimizer: A novel meta-heuristic optimization algorithm, *Knowl.-Based Syst.*, **242** (2022), 108320. https://doi.org/10.1016/j.knosys.2022.108320
34. H. Li, G. Xu, B. Chen, S. Huang, Y. Xia, S. Chai, Dual-mutation mechanism-driven snake optimizer for scheduling multiple budget constrained workflows in the cloud, *Appl. Soft Comput.*, **149** (2023), 110966. https://doi.org/10.1016/j.asoc.2023.110966
35. C. Wang, S. Jiao, Y. Li, Q. Zhang, Capacity optimization of a hybrid energy storage system considering wind-solar reliability evaluation based on a novel multi-strategy snake optimization algorithm, *Expert Syst. Appl.*, **231** (2023), 120602. https://doi.org/10.1016/j.eswa.2023.120602
36. R. A. Khurma, D. Albashish, M. Braik, A. Alzaqebah, A. Qasem, O. Adwan, An augmented snake optimizer for diseases and COVID-19 diagnosis, *Biomed. Signal Proces.*, **84** (2023), 104718. https://doi.org/10.1016/j.bspc.2023.104718
37. C. Yan, N. Razmjooy, Optimal lung cancer detection based on CNN optimized and improved snake optimization algorithm, *Biomed. Signal Proces.*, **86** (2023), 105319. https://doi.org/10.1016/j.bspc.2023.105319
38. E. H. Houssein, N. Abdalkarim, K. Hussain, E. Mohamed, Accurate multilevel thresholding image segmentation via oppositional snake optimization algorithm: Real cases with liver disease, *Comput. Biol. Med.*, **169** (2024), 107922. https://doi.org/10.1016/j.compbiomed.2024.107922
39. M. S. Braik, A. I. Hammouri, M. A. Awadallah, M. A. Al-Betar, O. A. Alzubi, Improved versions of snake optimizer for feature selection in medical diagnosis: a real case COVID-19, *Soft Comput.*, **27** (2023), 17833–17865. https://doi.org/10.1007/s00500-023-09062-3
40. K. K. Mohammed, S. Mekhilef, Improved snake optimizer algorithm-based GMPPT with a fast response to the load variations under different weather conditions for PV systems, *IEEE T. Ind. Electron.*, **71** (2024), 7147–7157. https://doi.org/10.1109/TIE.2023.3301526
41. Y. Li, B. Tang, S. Jiao, Q. Su, Snake optimization-based variable-step multiscale single threshold slope entropy for complexity analysis of signals, *IEEE T. Instrum. Meas.*, **72** (2023), 1–13. https://doi.org/10.1109/TIM.2023.3317908
42. G. Hu, R. Yang, M. Abbas, G. Wei, BEESO: Multi-strategy boosted snake-inspired optimizer for engineering applications, *J. Bionic Eng.*, **20** (2023), 1791–1827. https://doi.org/10.1007/s42235-022-00330-w
43. L. Yao, P. Yuan, C. Tsai, T. Zhang, Y. Lu, S. Ding, ESO: An enhanced snake optimizer for real-world engineering problems, *Expert Syst. Appl.*, **230** (2023), 120594. https://doi.org/10.1016/j.eswa.2023.120594
44. H. R. Tizhoosh, Opposition-based learning: A new scheme for machine intelligence, in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, (2005), 695–701. https://doi.org/10.1109/CIMCA.2005.1631345
45. D. J. Kalita, V. P. Singh, V. Kumar, A dynamic framework for tuning SVM hyper parameters based on moth-flame optimization and knowledge-based-search, *Expert Syst. Appl.*, **168** (2021), 114139. https://doi.org/10.1016/j.eswa.2020.114139
46. Q. Huang, C. Wang, Y. Ye, L. Wang, N. Xie, Recognition of EEG based on improved black widow algorithm optimized SVM, *Biomed. Signal Proces.*, **81** (2023), 104454. https://doi.org/10.1016/j.bspc.2022.104454

47. Y. Qiu, J. Zhou, Short-term rockburst prediction in underground project: insights from an explainable and interpretable ensemble learning model, *Acta Geotech.*, **18** (2023), 6655–6685. https://doi.org/10.1007/s11440-023-01988-0
48. Y. Qiu, J. Zhou, Short-Term rockburst damage assessment in burst-prone mines: An explainable XGBOOST hybrid model with SCSO algorithm, *Rock Mech. Rock Eng.*, **56** (2023), 8745–8770. https://doi.org/10.1007/s00603-023-03522-w
49. S. Cheng, J. Gao, H. Qi, Determination of the pile drivability using random forest optimized by particle swarm optimization and bayesian optimizer, *Comput. Model. Eng.*, **141** (2024), 871–892. https://doi.org/10.32604/cmes.2024.052830
50. Y. Song, L. Han, B. Zhang, W. Deng, A dual-time dual-population multi-objective evolutionary algorithm with application to the portfolio optimization problem, *Eng. Appl. Artif. Intel.*, **133** (2024), 108638. https://doi.org/10.1016/j.engappai.2024.108638
51. W Deng, X Cai, D Wu, Y Song, H Chen, X Ran, et al., MOQEA/D: multi-objective QEA with decomposition mechanism and excellent global search and its application, *IEEE T. Intell. Transp. Syst.*, (2024). https://doi.org/10.1109/TITS.2024.3373510
52. K. Bhosle, V. Musande, Evaluation of deep learning CNN model for recognition of devanagari digit, in *Artificial intelligence and applications*, (2023), 114–118. https://doi.org/10.47852/bonviewAIA3202441
53. T. O. Akande, O. O. Alabi, S. A. Ajagbe, A deep learning-based CAE approach for simulating 3D vehicle wheels under real-world conditions, in *Artificial Intelligence and Applications*, (2024). https://doi.org/10.47852/bonviewAIA42021882
54. Q. Sun, J. Chen, L. Zhou, S. Ding, S. Han, A study on ice resistance prediction based on deep learning data generation method, *Ocean Eng.*, **301** (2024), 117467. https://doi.org/10.1016/j.oceaneng.2024.117467

AIMS Press

©2024 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)