mathematics

MDPI

Article

FSSSA: A Fuzzy Squirrel Search Algorithm Based on Wide-Area Search for Numerical and Engineering Optimization Problems

Lei Chen *, Bingjie Zhao and Yunpeng Ma

School of Information Engineering, Tianjin University of Commerce, Tianjin 300134, China; zhaobingjie@stu.tjcu.edu.cn (B.Z.); mayunpeng@tjcu.edu.cn (Y.M.)
* Correspondence: chenlei@tjcu.edu.cn

Abstract: The Squirrel Search Algorithm (SSA) is widely used due to its simple structure and efficient search ability. However, SSA exhibits relatively slow convergence speed and imbalanced exploration and exploitation. To address these limitations, this paper proposes a fuzzy squirrel search algorithm based on a wide-area search mechanism named FSSSA. The fuzzy inference system and sine cosine mutation are employed to enhance the convergence speed. The wide-area search mechanism is introduced to achieve a better balance between exploration and exploitation, as well as improve the convergence accuracy. To evaluate the effectiveness of the proposed strategies, FSSSA is compared with SSA on 24 diverse benchmark functions, using four evaluation indexes: convergence speed, convergence accuracy, balance and diversity, and non-parametric test. The experimental results demonstrate that FSSSA outperforms SSA in all four indexes. Furthermore, a comparison with eight metaheuristic algorithms is conducted to illustrate the optimization performance of FSSSA. The results indicate that FSSSA exhibits excellent convergence speed and overall performance. Additionally, FSSSA is applied to four engineering problems, and experimental verification confirms that it maintains superior performance in realistic optimization problems, thus demonstrating its practicality.

Keywords: squirrel search algorithm; metaheuristic algorithm; fuzzy inference system; wide-area search mechanism; sine cosine mutation

MSC: 68R12

Citation: Chen, L.; Zhao, B.; Ma, Y. FSSSA: A Fuzzy Squirrel Search Algorithm Based on Wide-Area Search for Numerical and Engineering Optimization Problems. *Mathematics* **2023**, *11*, 3722. https://doi.org/10.3390/math11173722

Academic Editor: Petr Stodola

Received: 29 July 2023
Revised: 23 August 2023
Accepted: 25 August 2023
Published: 29 August 2023

1. Introduction

Optimization problems have existed widely in scientific and engineering fields. Over time, scholars have developed many methods to deal with optimization problems. These methods include the Gradient Descent Optimizer [1,2], Line Search Algorithm [3,4], and Trust Region Algorithm [5,6], among others. However, as problems become increasingly complex, traditional methods face challenges when confronted with optimization problems that involve intricate constraints and complicated calculation processes. To meet such requirements, researchers have introduced metaheuristic algorithms with a simple structure, strong global search capability, robustness, and independence from gradient information. Metaheuristic algorithms have indeed emerged as potent tools for addressing complex optimization problems across diverse fields. These algorithms exhibit superiority over traditional optimization methods, from their efficacy in dealing with intricate constraints, non-linear relationships, and high-dimensional search spaces [7].

In recent times, a large number of metaheuristic algorithms (MA) have been proposed, encompassing both classic metaheuristic algorithms and their improved variants. The classic MA can be classified into seven categories, namely Biology-based (BioA), Math-based (MaA), Physic-based (PhyA), Evolutionary-based (EvoA), Human-social-based (HuSoA), Plant-based (PlA), and Music-based (MuA) algorithms [8]. The classification of MAs are shown in Table 1.

Table 1. Classification of metaheuristic algorithm.

Classification	Algorithm
BioA	Particle Swarm Optimization (PSO) [9], Grey Wolf Optimizer (GWO) [10], FOX-inspired optimization algorithm (FOX) [11]
MaA	Sine Cosine Algorithm (SCA) [12], Circle-Inspired Optimization Algorithm (CIOA) [13]
PhyA	Simulated Annealing Algorithm (SA) [14], Gravitational Search Algorithm (GSA) [15]
EvoA	Genetic Algorithm (GA) [16], Differential Evolution (DE) [17]
HuSoA	Teaching-learning-based Optimization (TLBO) [18], Team Competition and Cooperation Optimization Algorithm (TCCO) [19]
PlA	Invasive Weed Optimization (IWO) [20], Photosynthetic Algorithm (PA) [21]
MuA	Melody Search (MS) [22], Harmony Search (HS) [23]

BioA algorithms draw inspiration from animal group activities in nature. Examples of BioA algorithms include PSO [9], GWO [10] and FOX [11]. MaA algorithms are based on principles and laws in mathematics, such as SCA [12] and CIOA [13]. PhyA algorithms are inspired by physical phenomena in nature, such as the SA [14] and GSA [15]. EvoA algorithms are inspired by the search Algorithms of biological evolutionary mechanisms such as natural selection and genetics; some classical EvoA algorithms include GA [16] and DE [17]. HuSoA algorithms are generally derived from human social phenomena and activities; some classical HuSoA algorithms include, TLBO [18] and TCCO [19]. PlA algorithms are based on intelligent behavior in plants, including IWO [20] and PA [21]. MuA algorithms are inspired by music-related concepts and principles, such as MS [22] and HS [23].

Currently, in the research of metaheuristic algorithms, addressing the issue of local optima while exploring the problem space remains an important research area. A more effective direction in metaheuristic algorithm research is to enhance the internal structure of existing algorithms to tackle various complex optimization problems [24]. In recent years, researchers have proposed various improved variants based on MA to solve complex optimization problems.

Shaukat proposed a modified genetic algorithm (MGA) for optimizing the multi-objective core overloading pattern. In comparison to the classical GA, MGA effectively preserves chromosomes with the best fitness, resulting in a more efficient search for the optimal fuel loading pattern [25]. Lodewijks conducted a comparison of the optimization performance of three state-of-the-art Particle Swarm Optimization (PSO) algorithms for solving the optimization problem of an Airport Baggage Handling Transportation System (BHTS) [26]. The experimental results showed that all three variants of PSO were capable of finding effective and efficient solutions. Among them, the Self-Regulation PSO (SRPSO) algorithm, which exhibited the lowest CPU running time, was selected and adopted. Romeh proposed the Hybrid Vulture Cooperative Multi-Robot Exploration (HVCME) algorithm and applied it to optimize the construction of limited maps in multi-robot exploration [27]. Compared to four other similar algorithms, HVCME demonstrated more effective optimization of limited map construction in unknown indoor environments.

In addition to these algorithms, the Squirrel Search Algorithm [28] has emerged as a BioA algorithm. Inspired by the foraging strategy of squirrels, they utilize parachute-like membranes to slide between trees in search of food. The SSA employs a combination of random search and local search mechanisms, enabling it to effectively search the solution space and converge towards optimal or near-optimal solutions. SSA has demonstrated strong competitiveness when compared to well-known MAs such as PSO, Artificial Bee Colony (ABC), and others. Since its proposal, SSA has been applied to various complex optimization problems such as Production Scheduling [29–31], Image Analysis [32,33], and Biomedicine [34,35]. The versatility and effectiveness of SSA make it a valuable tool for solving optimization problems in diverse domains.

Simultaneously, similar to any other metaheuristic algorithm, may not be suitable for every optimization problem. Researchers might choose to improve SSA to address specific problem characteristics or requirements. SSA also faced challenges such as imbalanced

exploration and exploitation, falling into local optimum, and low convergence accuracy. To address these issues, numerous scholars have studied to improve the performance of SSA. These efforts can be divided into three main categories based on improvement strategies.

Firstly, the adaptive parameter mechanism refers to improving one or more constants in the algorithm to make it adaptive in the position updating process. Zheng added the adaptive strategy of predator existence probability and the optimal selection strategy to enhance the exploitation capabilities of SSA [36]. Wen used the roulette strategy to update the squirrels' position on normal trees, which could increase the population diversity and avoid falling into the local optimum [37]. Second, the search strategy update mechanism refers to improving one or more position update strategies in the algorithm. Wang used the jump search method to improve the winter search strategy and the progressive search method to improve the Levy Flight stage, which make SSA could maintain the population diversity in these two stages [38]. Karaboga used the cloud model to replace the random function of uniform distribution to update the new position of squirrels, which improved the convergence accuracy of SSA [39]. Third, combining with different algorithms refers to the SSA update strategy combining with other algorithms. Sakthivel combined the Pareto dominance principle with SSA to keep the distribution diversity of Pareto optimal solutions in the algorithm's evolution process [40]. Liu used the crossover operator and mutation operator to enhance the squirrel position update stage, which increased the population diversity of SSA and improved the convergence of SSA [41].

The purpose of this paper is to develop a fuzzy squirrel search algorithm based on a wide-area search mechanism (FSSSA). There are three improvement strategies in FSSSA:

1. To accelerate the convergence speed, the adaptive weight w optimized by the fuzzy inference system (FIS) is added to change the step size in three position update strategies.
2. The sine cosine mutation strategy (SCM) enhanced the exploration ability and avoided falling into the local optimum by improving the sliding constant G_c in the position update stage.
3. The wide-area search mechanism (WAS) is used to improve the location update stage of some elite individuals (the first type of location update stage). It can balance the exploration and exploitation and improve the convergence accuracy of the algorithm.

The main contributions of this paper can be summarized as follows:

1. On the basis of SSA, this paper proposes an improved squirrel search algorithm (FSSSA).
2. In the FSSSA, the exploration ability is enhanced by using the FIS and the SCM strategy. The exploitation ability is enhanced by using the WAS strategy.
3. The FSSSA is tested on 24 benchmark functions and four engineering problems. Compared with the other algorithms, FSSSA has a preferable performance.

The rest of this paper is structured as follows. The second section introduces the principle of the basic SSA. The third section introduces three strategies to improve the SSA and calculates the space complexity of the FSSSA. The fourth and fifth sections introduces the validity of FSSSA by benchmark functions and four engineering optimization problems. The sixth part summarizes the research content of this paper and puts forward the prospect.

2. Squirrel Search Algorithm

SSA seeks global optima by sliding between different kinds of trees to find food sources and avoid predators. There are equal numbers of squirrels and trees n in SSA. There are three types of trees in SSA, which are normal trees, oak trees, and pecan trees. The pecan tree is designated as the best solution in the optimization process, while the oak trees represent the next three best solutions. The normal tree represents the remaining normal solutions. Each squirrel independently searches for food and explores existing food through a dynamic foraging strategy during the search process.

2.1. Random Initialization and Fitness Evaluation

Assuming n squirrels engage in sliding search in a d dimensional space. The SSA performs random initialization according to Equation (1).

$$FS_{i,j} = FS_L + U(0,1) \times (FS_U - FS_L)(i = 1,2,\ldots,n)(j = 1,2,\ldots,d) \quad (1)$$

where $FS_{i,j}$ represents the i^{th} position of the squirrel in the j^{th} dimension. The FS_L and FS_U are lower and upper bounds in the search space and $U(0,1)$ is a uniformly distributed random number in the range [0, 1].

The SSA calculates the fitness value corresponding to each squirrel position as Equation (2).

$$f\left(FS_{i,j}\right) = \left(f_1\left(FS_{1,j}\right), f_2\left(FS_{2,j}\right), \cdots, f_n\left(FS_{n,j}\right)\right)(i = 1,2,\ldots,n)(j = 1,2,\ldots,d) \quad (2)$$

When solving optimization problems for minimizing values, the fitness values are arranged from the smallest to the largest. Conversely, when addressing optimization problems for maximizing values, the arrangement is from largest to smallest. The individual with the top-ranked fitness value signifies the squirrel on the pecan tree. Those ranked 2nd to 4th in fitness represent squirrels on oak trees. The remaining fitness values correspond to squirrels on regular trees.

2.2. Update Position

During each iteration, the squirrels had three movement strategies: 1. Squirrels on the oak trees (FS_h^t, the three best solutions) flew to the pecan tree (FS_a^t, the best solution) to store energy for winter. 2. Squirrels on the normal trees (FS_n^t, normal solutions) fly to oak trees to meet their daily energy needs. 3. Some squirrels on the normal trees still head for the pecan tree to store their energy needs for winter. The specific position is updated according to Equations (3)–(5).

Case1 FS_a^t——FS_h^t

$$FS_a^{t+1} = \begin{cases} FS_a^t + d_g \times G_c \times \left(FS_h^t - FS_a^t\right), R_1 \geq P_{dp} \\ \text{Random location} \qquad \text{otherwise} \end{cases} \quad (3)$$

Case2 FS_n^t——FS_a^t

$$FS_n^{t+1} = \begin{cases} FS_n^t + d_g \times G_c \times \left(FS_a^t - FS_n^t\right), R_1 \geq P_{dp} \\ \text{Random location} \qquad \text{otherwise} \end{cases} \quad (4)$$

Case3 FS_n^t——FS_h^t

$$FS_n^{t+1} = \begin{cases} FS_n^t + d_g \times G_c \times \left(FS_h^t - FS_n^t\right), R_1 \geq P_{dp} \\ \text{Random location} \qquad \text{otherwise} \end{cases} \quad (5)$$

where t represents the current iteration, the R_1 stands for a random number in the range of [0, 1] and the G_c is a gliding constant. P_{dp} represents the probability of the existence of predators. The d_g is the random sliding distance constant. When $R_1 \geq P_{dp}$, with no predators in the forest, squirrels glide to find food squirrels have free foraging activities. When $R_1 < P_{dp}$, squirrels will randomly update their position.

2.3. Seasonal Detection Condition

Squirrels' foraging behavior changes when winter comes. In SSA, checking seasonal variation conditions prevents the algorithm from falling into the local optimum. According

to Equations (6) and (7), the seasonal constant (S_c^t) and seasonal detection condition ($S_{\min}$) are calculated to determine whether entering winter.

$$S_c^t = \sqrt{\sum_{k=1}^{d} \left(FS_{a,k}^t - FS_{h,k} \right)^2} \tag{6}$$

where d represents the dimension of the problem. $FS_{a,k}^t$ and $FS_{h,k}^t$, respectively, denote the squirrel on the pecan tree (best solution) and the squirrels on oak trees (three next best solutions).

$$S_{\min} = \frac{10e^{-6}}{(365)^{t/(t_m/2.5)}} \tag{7}$$

where t_m represents the maximum number of iterations.

2.4. Levy Flight

When $S_c < S_{\min}$, the positions of those flying squirrels without food sources (squirrels on normal trees) are updated by Equation (8).

$$FS_{nt}^{new} = FS_L + levy(\mathrm{n}) \times (FS_U - FS_L) \tag{8}$$

Levy Flight allows squirrels to find new locations close to their current sweet spot by Equation (9).

$$levy = 0.01 \times \frac{r_a \times \sigma}{|r_b|^{\frac{1}{\beta}}} \tag{9}$$

where, r_a and r_b are two normally distributed random numbers in the range of [0, 1]. The β is an exponent parameter of the Levy distribution, employed to characterize the distribution's shape. A smaller value of β encourages the algorithm to engage in larger jumps, thereby increasing the likelihood of discovering novel solutions within the search space. Conversely, a larger β value concentrates the step distribution closer to smaller values. The permissible range of "a" lies within [0, 2], and in the context of SSA, it is set to 1.5. The σ is a parameter within the Levy flight model, governing the magnitude of step lengths. It emulates the leap distance of a Levy flight, computed according to Equation (10).

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{1/\beta} \tag{10}$$

where $\Gamma(x) = (x-1)!$.

To provide a clearer representation of the search process in SSA, the pseudocode for Algorithm 1 of SSA is presented below.

It is noteworthy that the stopping criterion used in SSA [28] is the maximum number of iterations. However, from Pseudocode 1 for Algorithm 1, it is evident that during each iteration, SSA performs function evaluations multiple times. This setup might give SSA an advantage over algorithms that perform only one function evaluation per iteration [42]. Therefore, when conducting experimental comparisons involving SSA, it is recommended to avoid conducting comparative experiments with other algorithms based on the same number of iterations. This is to ensure the accuracy of evaluating SSA's optimization performance.

Algorithm 1 Pseudocode for SSA

Begin:
Input the optimization problem information.
Set control parameters population (n), scaling factor (sf), and predators (P_{dp}).
Generate random locations for n flying squirrels using Equation (1)
Evaluate the fitness of each flying squirrel's location.
Sort flying squirrel locations by fitness value.
The best value is defined as the squirrel on the pecan tree, the three of the next best values are the squirrels on the oak trees, and the rest values are the squirrels on the normal trees.
While (the stopping criterion is not satisfied) **do**
For t = 1 to n
Update flying squirrel locations which are on oak trees and moving towards pecan trees using Equation (3)
Update flying squirrel locations which are on normal trees and moving towards oak trees using Equation (4)
Evaluate the fitness of each flying squirrel's location.
Update flying squirrel locations which are on normal trees and moving towards pecan trees using Equation (5)
Evaluate the fitness of each flying squirrel location.
End
Calculate seasonal constant (S_c^t) using Equation (6)
Update the minimum value of seasonal constant ($S_{\min}$) using Equation (7)
If (Seasonal monitoring condition is satisfied)
Randomly relocate flying squirrels on normal trees using Equation (8)
Evaluate the fitness of each flying squirrel's location
end
Reorder the squirrels. The best value is defined as the squirrel on the pecan tree, the three of the next best values are the squirrels on the oak trees, and the rest values are the squirrels on the normal trees.
End
The location of the squirrel on the pecan tree is the final optimal solution
End

3. The Proposed Algorithm (FSSSA)

While SSA does possess strong global search capabilities, the fixed search range and direction in each iteration can result in slow convergence. Additionally, the random search strategy employed by the elite individuals in SSA for exploring the global range can lead to weaker exploitation ability and lower convergence accuracy. Moreover, when dealing with high-dimensional complex optimization problems, SSA may encounter challenges related to falling into local optima. The exploration-exploitation balance becomes crucial in such scenarios to avoid being trapped in suboptimal solutions.

Aiming at the above limitations, this section proposes three strategies, the FIS, the SCM, and the WAS. By employing the FIS, the output of inertia weights w can be obtained to adjust the step size variation during iterations, thereby accelerating the convergence speed of the algorithm. By utilizing the SCM, the sliding constant G_c^{new} can be adjusted to enable the search range and search direction to vary with iterations, thereby enhancing the exploratory capability of the algorithm. Through the use of WAS, the search mechanism of elite individuals can be improved, thereby enhancing their exploitative capability in the vicinity of the optimal solution.

3.1. Introduced Fuzzy Inference System

This paper used FIS to output inertia weights w. It was added into three position updating stages and made the search step size change randomly with the number of iterations to improve the convergence speed of the algorithm.

Under the framework of an adaptive network, the FIS [43] can combine fuzzy inference and control the input of the system. The FIS is divided into five parts: Fuzzification Interface,

Fuzzy Database, Fuzzy Rule Base, Fuzzy Reasoning, and Defuzzification Interface. The FIS is shown in Figure 1.

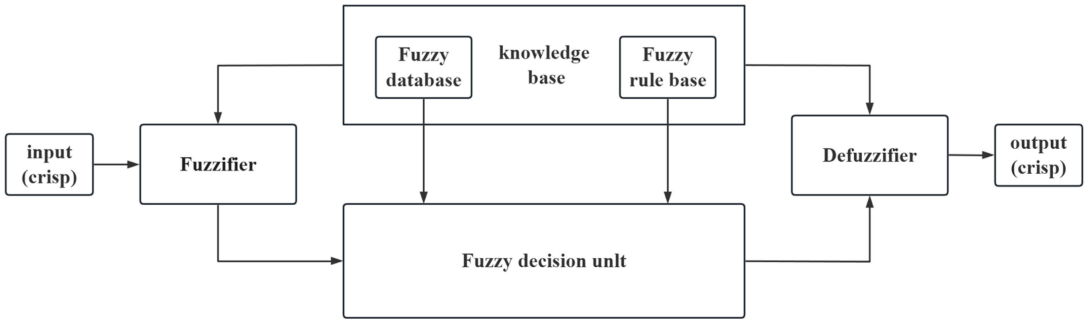

Figure 1. Fuzzy Inference System (FIS).

FIS has been used in many fields. Kumar [44] used the FIS to provide localization results for transnational faults of circuits. Yu [45] used the FIS to optimize the neural network. Therefore, the studies proved that the FIS has a preferable performance and can be used to optimize the metaheuristic algorithms.

Liu used FIS to optimize the PSO according to the model error to make its parameters dynamically self-adaptive [46]. Amador–Angulo used Type-2 FIS to optimize the bee colony optimization and make its parameters dynamically self-adaptive [47]. Many people applied FIS to improve the algorithm and achieved remarkable results.

The FIS designed in this paper, named FSSSA-Type-1, belongs to the category of Type-1 FIS. When dealing with optimization problems, metaheuristic algorithms often encounter uncertainties, such as the form of the objective function and the complexity of the constraint conditions. Type-1 FIS can effectively handle these uncertainties by utilizing fuzzy sets and fuzzy rules for modeling and inference. This approach enhances the adaptability of the algorithm to complex and ambiguous problems. The fuzzy rules defined in this system utilize triangular membership functions. The system diagram of the FSSSA is illustrated in Figure 2. The design of the fuzzy rules output surface as shown in Figure 3.

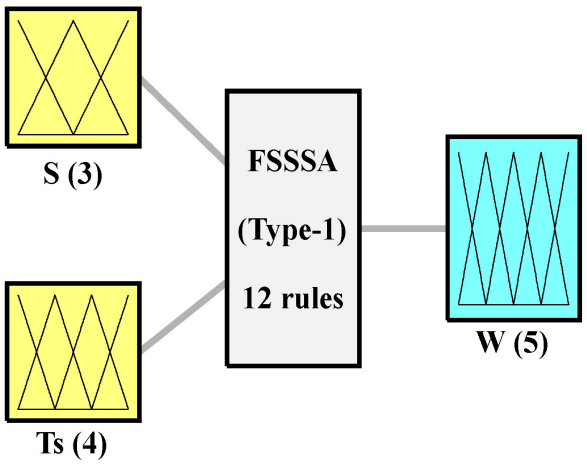

Figure 2. The system block diagram of the FIS.

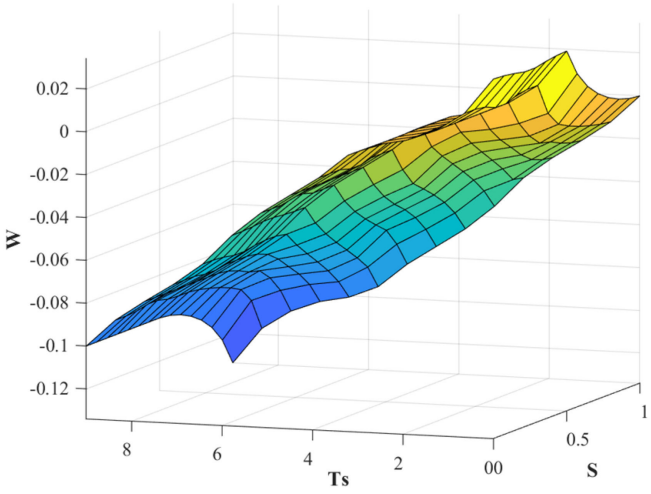

Figure 3. The design of the fuzzy rules output surface.

The inputs of the system are iteration progress (S) and iteration stall time (Ts). Where iteration progress $S = t/t_m$ is the ratio of the current iteration to the number of maximum iterations. t is the current iteration progress and t_m is the maximum number of iterations. The Ts represents the time or number of iterations elapsed when the algorithm is unable to significantly improve the quality of the solution. Its calculation method is depicted in Equation (11).

$$Ts^{t+1} = \begin{cases} \min(Ts^t - 1, 0) \; de \; < f\left(FS_a^t\right) \\ \max(Ts^t + 1, 9) \; de \; > f\left(FS_a^t\right) \end{cases} \tag{11}$$

where $f\left(FS_a^t\right)$ represents the best historical fitness of the t iteration. de represents the optimal historical fitness of the $t+1$ iteration. The single output of the system is inertia weight w.

By observing the output surface in Figure 3, a clear understanding of the distribution of fuzzy outputs obtained by the FSSSA-Type-1 can be achieved. In this process, the inertia weight (w) gradually increases with the increase in iteration stall time, thereby enhancing the algorithm exploration capability in the later stages of iteration. Consequently, this enhances the likelihood of escaping the local optimum and further improves the algorithm performance.

The membership functions of the input and output are shown in Figure 4.

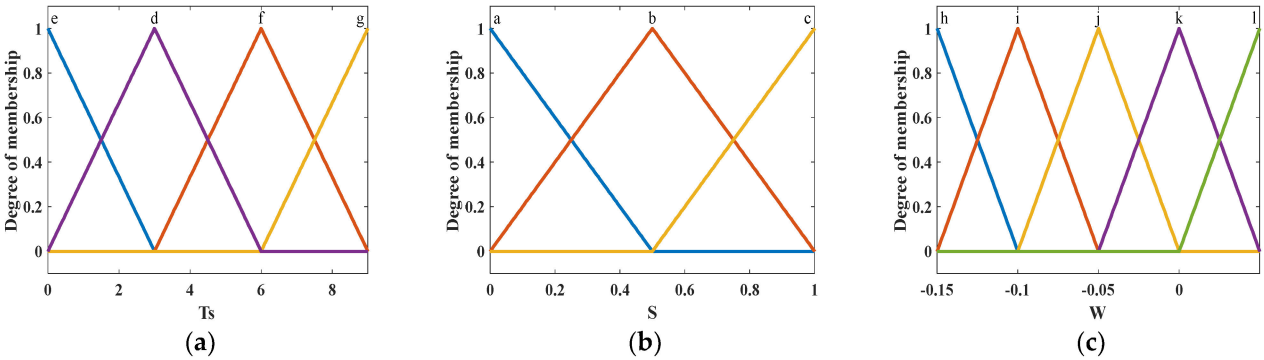

Figure 4. Schematic diagram of membership function: (**a**) iteration progress; (**b**) iteration stall time; (**c**) inertia weight.

The position update strategy of SSA is adjusted by inertia weight w, which increases the exploitation ability and accelerates the convergence speed of the algorithm.

3.2. Introduced Sine Cosine Mutation

In SSA, the search individual only follows the fixed direction. It will lead to slow convergence speed and may fall into the local optimum. The sine and cosine variables in the Sine Cosine Algorithm (SCA) [12] change randomly with iteration. The sine and cosine variables were introduced to improve the gliding constant G_c, so that the search range and direction were changed with iteration.

The investigation stage of the SCA is shown in Equation (12).

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \times \sin(\pi \times r_2) \times \left| r_3 P_i^t - X_i^t \right|, r_4 < 0.5 \\ X_i^t + r_1 \times \cos(\pi \times r_2) \times \left| r_3 P_i^t - X_i^t \right|, r_4 > 0.5 \end{cases} \tag{12}$$

where X_i^t is the position of the current solution in the i dimension at the t iteration and P_i^t is the position of the best point (best solution) in the i dimension. The roles of r_1, r_2, r_3, and r_4 define the moving direction of the next position, the distance moved to the next position, random weights, and random decision parameters. $r_1 = \alpha - \frac{\alpha \times t}{t_m}$ and α is a predetermined constant, where in SCA, its value is set to 3. The r_2 and r_3 are random numbers of [0, 2]. The r_4 is a random number of [0, 1].

According to Equation (12), the direction and the distance of SCA are random. The random combinatorial property of the SCA was used to improve the G_c in Equations (3)–(5). The improved G_c^{new} is no longer a fixed value; it dynamically adapts and changes with each iteration. This method is mathematically formulated as follows.

$$G_c^{new} = \begin{cases} \left(2 - \frac{2 \times t}{t_m}\right) \times \sin(2 \times \pi \times r_2), r_4 < 0.5 \\ \left(2 - \frac{2 \times t}{t_m}\right) \times \cos(2 \times \pi \times r_2), r_4 > 0.5 \end{cases} \tag{13}$$

where r_2 and r_4 are random numbers of [0, 1]. r_2 randomly changes the search step of SSA.

The SSA search direction and range changed accordingly by improving the sliding constant, which increases the convergence speed of SSA and enhances the ability to jump out of the local optimum.

3.3. Introduced Wide-Area Search Mechanism

In SSA, elite individuals randomly jump within the global search range, and the exploitation ability of the algorithm is weak, resulting in low convergence accuracy. This section used the WAS to improve the search strategy of elite individuals.

Many metaheuristic algorithms also add a WAS to improve the algorithm's exploitation ability. Simulated Annealing (SA) [14] added the WAS in the status update phase. It reduced the probability of accepting the new value and increased the exploitation ability. Improved Evolution Algorithm (IEA) [48] used the WAS to improve the evolution operator of Differential Evolution (DE). It increased the population diversity of DE.

The WAS is added to the location update strategy of producers in the Sparrow search algorithm [49] by Equation (14).

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^t \times \exp(\frac{-i}{\alpha \times t_m}), & if\ R_2 < ST \\ X_{i,j}^t + Q \times L, & if\ R_2 > ST \end{cases} \tag{14}$$

where $X_{i,j}^{t+1}$ represents the value of the i^{th} position of the sparrow in the j^{th} dimension during the t^{th} iteration. The t_m denotes the maximum number of iterations. Both α and R_2 are random numbers within the range [0, 1]. The ST represents the safety threshold and. The Q is a random number generated according to a normal distribution. The L is a $1 \times d$ matrix (where d signifies the problem's dimension), with each element being 1.

When $R_2 < ST$, it indicates the absence of predators in the vicinity. In this case, producers (sparrows) narrow their search range with each iteration in a randomized manner, restricting their movement to the neighborhood of the current optimal solution.

Inspired by the Sparrow search algorithm, the position of some elite individuals (FS_a^t——FS_h^t) is updated as Equation (15).

$$FS_a^{t+1} = \begin{cases} G_c^{new} \times \left(FS_h^t - FS_a^t\right) \times \exp(\frac{-i}{\alpha \times t_m}) & R_1 \geq P_{dp} \\ \text{Random location} & \text{otherwise} \end{cases} \tag{15}$$

where both α and R_1 are random numbers within the range [0, 1]. The G_c^{new} is an improved sliding variable according to the Formula (13). The P_{dp} represents the probability of the presence of predators. When $R_1 \geq P_{dp}$, it signifies the absence of predators in the forest, prompting squirrels on oak trees to engage in wide-area search mechanisms to locate pecan tree. When $R_1 < P_{dp}$, it indicates the presence of predators in the forest, and squirrels update their positions randomly to evade the predators.

By incorporating non-deterministic elements based on the acquired information and results during the search process, the efficiency and performance of the search conducted by elite individuals are improved. This approach enables more refined searches within the neighborhood of the optimal value, utilizing the updated positions of elite individuals. It aims to thoroughly explore the entire neighborhood as much as possible, thereby enhancing the exploitation capability and convergence accuracy of SSA.

3.4. SSA with Mixed Strategy

Firstly, the adaptive weight w optimized by the FIS is added to change the step size in three position update strategies. It can improve the exploitation and accelerate the convergence speed of the SSA. Secondly, the SCM is introduced to enhance the sliding constant (G_c) in the position update stage. The search direction and step are adjusted during iteration, which enhances the exploration ability and avoids falling into the local optimum. Finally, the WAS mechanism is introduced to improve the elite individuals. Make the elite individuals jump around the optimal value of the neighborhood. It can balance the exploration and exploitation and improve the convergence accuracy of the algorithm.

The three-position update strategies of the improved SSA are shown in Equations (16)–(18).

Case1 FS_a^t——FS_h^t

$$FS_a^{t+1} = \begin{cases} G_c^{new} \times \left(FS_h^t - FS_a^t\right) \times \exp(\frac{-i}{\alpha \times t_m}) & R_1 \geq P_{dp} \\ \text{Random location} & \text{otherwise} \end{cases} \tag{16}$$

Case2 FS_n^t——FS_a^t

$$FS_n^{t+1} = \begin{cases} w \times FS_n^t + d_g \times G_c^{new} \times \left(FS_a^t - FS_n^t\right) & R_1 \geq P_{dp} \\ \text{Random location} & \text{otherwise} \end{cases} \tag{17}$$

Case3 FS_n^t——FS_h^t

$$FS_n^{t+1} = \begin{cases} w \times FS_n^t + d_g \times G_c^{new} \times \left(FS_h^t - FS_n^t\right) & R_1 \geq P_{dp} \\ \text{Random location} & \text{otherwise} \end{cases} \tag{18}$$

To show the optimization idea of FSSSA more clearly, the pseudocode Algorithm 2 of FSSSA is shown, and the flow chart of FSSSA is shown in Figure 5.

Algorithm 2 Pseudocode for FSSSA

Begin:
Input the optimization problem information.
Set control parameters population (n), scaling factor (sf), and predators (P_{dp}).
Generate random locations for n flying squirrels using Equation (1)
Evaluate the fitness of each flying squirrel's location.
Sort flying squirrel locations by fitness value.
The best value is defined as the squirrel on the pecan tree, the three of the next best values are the squirrels on the oak trees, and the rest values are the squirrels on the normal trees.
while (the stopping criterion is not satisfied) **do**
Calculate iterative stall time (T_s) using Equation (11)
Calculate inertia weight (w) from the fuzzy inference system
Calculate gliding constant (G_c^{new}) using Equation (13)
For t = 1 to n
Update flying squirrel locations which are on oak trees and moving towards pecan trees using Equation (16)
Update flying squirrel locations which are on normal trees and moving towards oak trees using Equation (17)
Evaluate the fitness of each flying squirrel's location and reorder the squirrels.
Update flying squirrel locations which are on normal trees and moving towards pecan trees using Equation (18)
Evaluate the fitness of each flying squirrel location and reorder the squirrels.
end
Calculate seasonal constant (S_c^t) using Equation (6)
Update the minimum value of seasonal constant ($S_{\min}$) using Equation (7)
If (Seasonal monitoring condition is satisfied)
Randomly relocate flying squirrels on normal trees using Equation (8)
Evaluate the fitness of each flying squirrel's location.
end
Reorder the squirrels. The best value is defined as the squirrel on the pecan tree, the three of the next best values are the squirrels on the oak trees, and the rest values are the squirrels on the normal trees.
end
The location of the squirrel on the pecan tree is the final optimal solution
End

3.5. Computational Complexity

The complexity of SSA is composed of population initialization, fitness evaluation, and three strategies for updating location. Remarkably, the complexity of the proposed FSSSA adds the inertia weight designed in FIS, the improved adaptive parameters of SCM, and the improved elite individual location update strategy of WAS. The coefficients involved are the algorithm population N and the number of iterations of the algorithm T. The complexity of population initialization is $O(N)$, the complexity of fitness evaluation is $O(N)$. The complexity of the FIS strategy is $O(N \times T)$, the complexity of the SCM strategy is $O(T)$, and the complexity of the WAS strategy and the other two location update strategies is $O(N \times T)$. Therefore, the complexity of the FSSSA is $O(FSSSA) = 2 \times O(N) + O(N \times T) + O(T) + O(N \times T) = O(N \times T)$. The FSSSA maintains the same computational load as the classical SSA and other classical MAs, such as PSO, DE, and A.

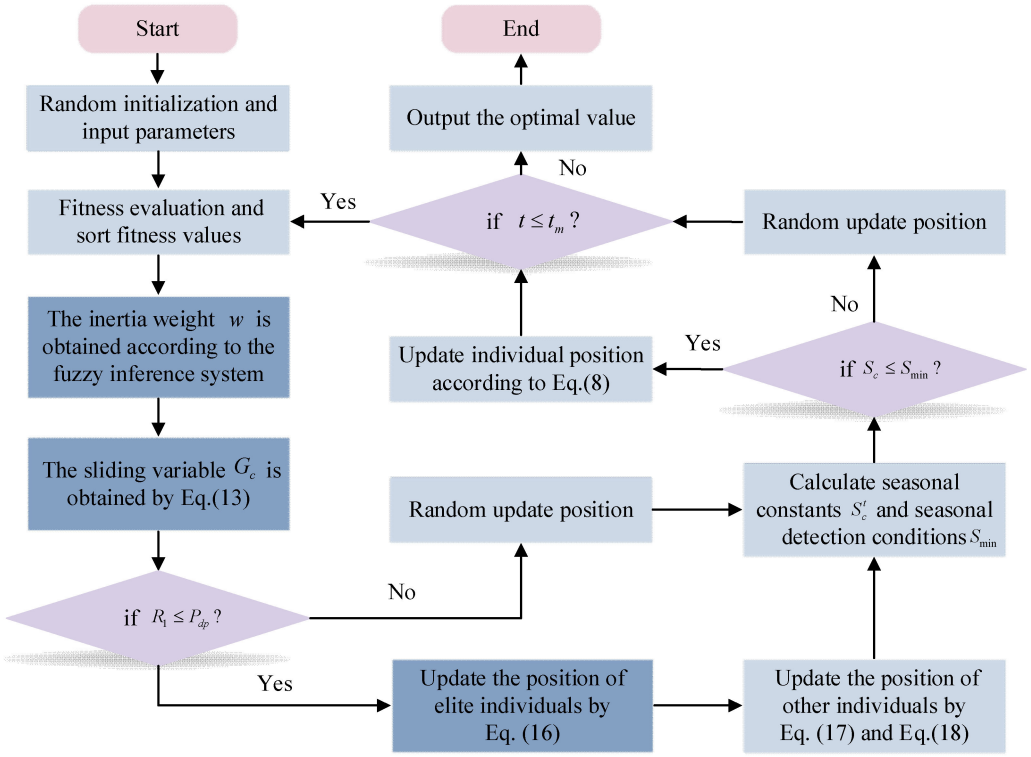

Figure 5. Flow chart of FSSSA.

4. Experimental Studies on Function Optimization Problems

In this section, the effectiveness of FSSSA will be demonstrated clearly and intuitively through classical benchmark optimization problems. Firstly, the parameter tuning of FSSSA will be introduced, analyzing the impact of parameter selection on FSSSA. Secondly, a comparative analysis will be conducted between SSA and the proposed FSSSA on 24 benchmark functions. This analysis will include convergence curves, convergence accuracy, balance, and diversity, aiming to provide a comprehensive evaluation of the optimization capabilities of FSSSA. By assessing these aspects, the effectiveness of the improvement strategy can be validated. The Wilcoxon rank-sum test will be employed to evaluate the significance difference between SSA and FSSSA. Lastly, experimental comparisons will be conducted between FSSSA and other metaheuristic algorithms, including MA and improved variants algorithms, to further evaluate the optimization performance and applicability of FSSSA.

4.1. Benchmark Functions and Parameter Setting

All experiments in this paper are carried out under the environment of III(IXeon(R) CPU AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz, 16 GB RAM, and MATLAB (2020b). To reduce the randomness of the experiment, each experimental result is independently repeated 30 times to take the average.

4.1.1. Parameter Setting

The algorithm parameters used in the experimental test are the White Shark Optimizer (WSO) algorithm [50], Runge Kutta Optimization (RUN) algorithm [51], Weighted Mean of Vectors (INFO) algorithm [52], PSO [9], Group Teaching Optimization Algorithm with Information Sharing (ISGTOA) [53], Seagull Optimization Algorithm (SOA) [54], Grey Wolf optimizer (GWO) [10] and Ensemble Sinusoidal Differential Covariance Matrix Adaptation with Euclidean Neighborhood (LSHADE-cnEpSin, one of the winners of CEC 2017 competition) [55]. The algorithm parameters above are consistent with those in the original paper of the algorithm. Specific settings are shown in Table 2.

Table 2. Algorithm parameter setting.

Algorithm	Pop	Parameters
SSA	50	$P_{dp} = 0.1$; $R_1, R_2, R_3 \in [0,1]$; ρ= 1.204; $V = 5.25$; $S = 0.0154$; $C_L = 0.7$; $C_D = 0.6$; $h_g = 8$; $sf = 18$; $G_C = 1.9$
SSSA	50	$P_{dp} = 0.1$; $R_1, R_2, R_3 \in [0,1]$; ρ= 1.204; $V = 5.25$; $S = 0.0154$; $C_L = 0.7$; $C_D = 0.6$; $h_g = 8$; $sf = 18$
FSSA	50	$P_{dp} = 0.1$; $R_1, R_2, R_3 \in [0,1]$; ρ= 1.204; $V = 5.25$; $S = 0.0154$; $C_L = 0.7$; $C_D = 0.6$; $h_g = 8$; $sf = 18$; $G_C = 1.9$
FSSSA	50	$P_{dp} = 0.1$; $R_1, R_2, R_3 \in [0,1]$; ρ= 1.204; $V = 5.25$; $S = 0.0154$; $C_L = 0.7$; $C_D = 0.6$; $h_g = 8$; $sf = 18$
WSO	50	$p_{\min} = 0.5$; $p_{\max} = 1.5$; $\tau = 4.11$; $f_{\min} = 0.07$; $f_{\max} = 0.75$; $a_0 = 6.25$; $a_1 = 100$; $a_2 = 0.0005$
RUN	50	$r_1 = 1/-1$; $\varphi \in [0,1]$; $\beta \in [0,1]$; $c = 5 \times rand$; $v = 2 \times rand$; $g \in [0,2]$; $r_2 = 1/-1/0$
INFO	50	$a_1 \neq a_2 \neq a_3 \in [1,50]$; $r \in [0.1, 0.5]$; $\mu = 0.05 \times randn$; $\varphi \in [0,1]$
PSO	50	$c_1 = c_2 = 1.141$
ISGTOA	50	$\lambda \in [0,1]$; $a = b = c = d = rand$
SOA	50	$r_1 = r_2 = rand$; $k \in [0, 2\pi]$; $f_c = 2$; $b = 1$
GWO	50	$r_1 = r_2 = rand$; $a = 2 - 2t/t_{\max}$; $A = 2a \times r_1 - a$; $C = 2 \times r_2$
LSHADE-cnEpSin	-	$c_1 = c_2 = 1.141$

4.1.2. Benchmark Functions

This paper conducts experiments on 24 classic benchmark test functions [56,57]. According to the character of functions, they can be described as unimodal, multimodal, separable, and non-separable functions. The function set in this paper is shown in Table 3, including 11 unimodal and 13 multimodal functions, 12 separable and 12 non-separable functions, among which F12 and F17-F19 are four shifted functions. The shifted function refers to the operation of shifting the graph of a function in space, wherein the function's image is horizontally or vertically shifted along the coordinate axes. In these functions, the best position is moved or rotated to other locations primarily to avoid situations where certain algorithms would copy one parameter to another parameter to generate neighboring solutions.

Table 3. Benchmark functions.

No	Functions	Function Name	Range	Dim	Character	F_{min}
F1	$f(x) = \sum_{i=1}^{n} x_i^2$	Sphere	[−100, 100]	30/50	US	0
F2	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	Schwefel 2.22	[−10, 10]	30/50	UN	0
F3	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	Schwefel 1.2	[−100, 100]	30/50	UN	0
F4	$f(x) = \max_i \{ \lvert x_i \rvert, 1 \leq i \leq n \}$	Schwefel 2.21	[−100, 100]	30/50	UN	0
F5	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	Rastrigin	[−5.12, 5.12]	30/50	MS	0
F6	$f(x) = \sum_{i=1}^{n} \lvert x_i + 0.5 \rvert^{i+1}$	Step	[−500, 500]	30/50	US	0
F7	$f(x) = \sum_{i=1}^{n} i x_i^4$	Quartic	[−1.28, 1.28]	30/50	US	0
F8	$f(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0,1]$	Quartic WN	[−1.28, 1.28]	30/50	US	0

Table 3. *Cont.*

No	Functions	Function Name	Range	Dim	Character	F_{min}
F9	$f(x) = 0.1\left\{ \begin{array}{l} \sin^2(3\pi x_1) + \sum_{i=1}^{\dim}(x_i-1)^2 \\ \left[1+\sin^2(3\pi x_{i+1}+1)\right] \\ +(x_{\dim}-1)^2\left[1+\sin^2(2\pi x_{dim})\right] \end{array} \right\} + \sum_{i=1}^{\dim} Ufun(x_i,5,100,4)$	Penalized2	$[-50, 50]$	30/50	MN	0
F10	$f(x) = \sum_{i=1}^{n}\left(10^6\right)^{(i-1)/(n-1)} x_i^2$	Elliptic	$[-100, 100]$	30/50	UN	0
F11	$f(x) = \sum_{i=1}^{n}\left(x_i^2 - 10(\cos 2\pi x_i) + 10\right)$	Rastrigin	$[-5.12, 5.12]$	30/50	MS	0
F12	$f(x) = \sum_{i=1}^{n} z_i^2, \vec{z} = \vec{x} - \vec{o}$	Shifted sphere	$[-100, 100]$	30/50	US	0
F13	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{\dim}d\sum_{i=1}^{\dim}x_i^2}\right) - \exp\left(\frac{1}{\dim}\sum_{i=1}^{\dim}\cos(2\pi x_i)\right) + 20 + e$	Ackley	$[-32, 32]$	30/50	MN	0
F14	$f(x) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Griewank	$[-600, 600]$	30/50	MN	0
F15	$f(x) = \sum_{i=1}^{n} i x_i^2$	SumSquares	$[-10, 10]$	30/50	US	0
F16	$f(x) = \sum_{i=1}^{n}\lvert x_i\rvert^{(i+1)}$	SumPower	$[-10, 10]$	30/50	MS	0
F17	$f(x) = \sum_{i=1}^{n}\left(z_i^2 - 10\cos(2\pi z_i) + 10\right), \vec{z} = \vec{x} - \vec{o}$	Shifted rastrigin	$[-5.12, 5.12]$	30/50	MS	0
F18	$f(x) = \sum_{i=1}^{n}\frac{1}{4000}z_i^2 - \prod_{i=1}^{n}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1, \vec{z} = \vec{x} - \vec{o}$	Shifted griewank	$[-600, 600]$	30/50	MN	0
F19	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}z_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi z_i)\right) + e, \vec{z} = \vec{x} - \vec{o}$	Shifted ackley	$[-32, 32]$	30/50	MN	0
F20	$f(x) = \sum_{i=1}^{n} x_i^6\left(2 + \sin\frac{1}{x_i}\right)$	Csendes	$[-1, 1]$	30/50	MS	0
F21	$f(x) = \sum_{i=1}^{n-1}\left(0.5 + \frac{\sin^2\sqrt{100x_i^2+x_{i+1}^2} - 0.5}{1+0.001\left(x_i^2-2x_ix_{i+1}+x_{i+1}^2\right)^2}\right)$	Pathological	$[-100, 100]$	30/50	MN	0
F22	$f(x) = \sum_{i=1}^{n}\left[y_i^2 - 10\cos(2\pi y_i) + 10\right], y_i = \left\{ \begin{array}{l} x_i, \lvert x_i\rvert < \frac{1}{2} \\ \frac{round(2x_i)}{2}, \lvert x_i\rvert \geq \frac{1}{2} \end{array} \right.$	Non-Continuo-usrastrigin	$[-5.12, 5.12]$	30/50	MS	0
F23	$f(x) = \left(\sum_{i=1}^{n}\lvert x_i\rvert\right)\exp\left(-\sum_{i=1}^{n}\sin(x_i^2)\right)$	Xin-She Yang Second	$[-2\pi, 2\pi]$	30/50	MN	0
F24	$f(x) = \sum_{i=1}^{n-1}\left(x_i^2\right)^{(x_{i+1}^2+1)} + \left(x_{i+1}^2\right)^{(x_i^2+1)}$	Brown	$[-1, 4]$	30/50	UM	0

The character of functions is detailed in the Character column of Table 1. U, M, S, and N are used to indicate unimodal, multimodal, separable, and non-separable functions [58].

It should be noted that when the dimension increases, the search space and the corresponding difficulty will be increased. It is more challenging to solve high-dimensional problems than low-dimensional ones [59]. Therefore, dimensions 30 and 50 are used for experimental tests in this paper.

4.2. FSSSA Parameter Tuning

The performance of the algorithm is influenced by various factors, including population size, number of iterations, and parameters. In the FSSSA, the scaling factor sf plays a crucial role in balancing the exploration and exploitation phases [28]. Therefore, selecting an appropriate sf value is vital for achieving optimal performance with FSSSA. Based on relevant literature and previous experiments, a value sf in the range of 16 to 37 can achieve the desired accuracy level without compromising algorithm stability. In order to provide a clearer understanding of the impact of sf on FSSSA's performance across benchmark functions, manual adjustments were made using functions F8 and F9 to observe experimental results more effectively.

Table 4 records the average and standard deviation of parameter settings as $sf = \{10, 15, 18, 20, 30, 40\}$ over 30 runs with 20,000 function evaluations (FEs), as well as the convergence curve shown in Figure 6 and the box plot illustrated in Figure 7.

Table 4. The effect of sf on the performance of FSSSA on benchmark functions.

Function	Item	*sf* = 10	*sf* = 15	*sf* = 18	*sf* = 20	*sf*=30	*sf* = 40
F8	mean	1.00×10^{-4}	7.81×10^{-5}	$\mathbf{2.25 \times 10^{-5}}$	8.36×10^{-5}	1.00×10^{-4}	2.34×10^{-5}
	std	9.18×10^{-5}	7.84×10^{-5}	$\mathbf{1.99 \times 10^{-5}}$	6.57×10^{-5}	9.18×10^{-5}	1.74×10^{-5}
F9	mean	8.98×10^{-1}	7.66×10^{-1}	$\mathbf{6.19 \times 10^{-1}}$	7.43×10^{-1}	8.98×10^{-1}	8.71×10^{-1}
	std	2.91×10^{-1}	2.13×10^{-1}	$\mathbf{1.43 \times 10^{-1}}$	2.21×10^{-1}	2.91×10^{-1}	2.31×10^{-1}

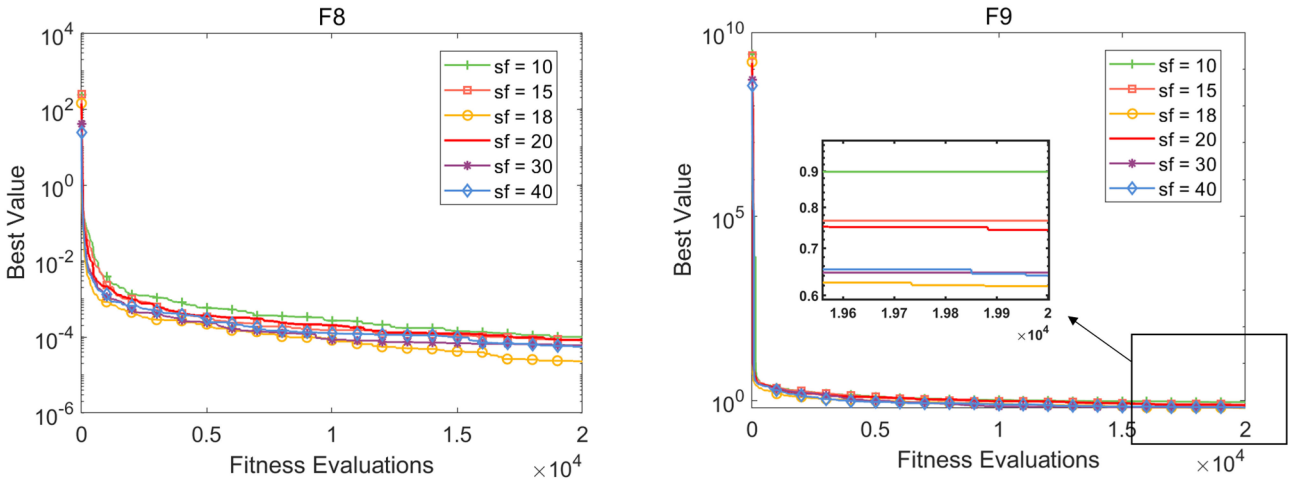

Figure 6. Convergence curves with different scaling factor (sf) Settings.

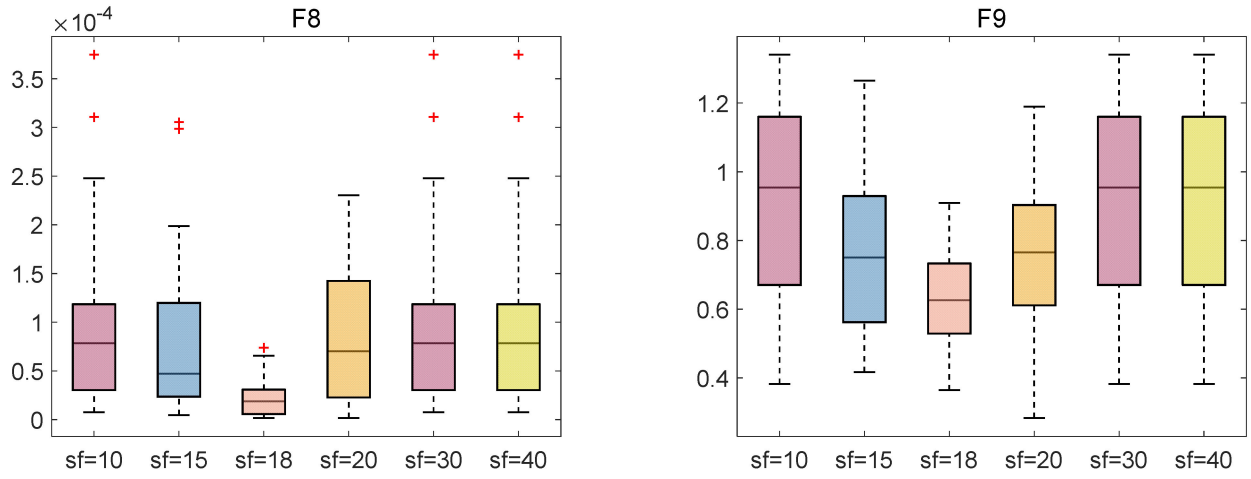

Figure 7. Box diagram with different scaling factor (sf) Settings.

By examining Figure 6 and Table 4, it becomes evident that when $sf = 18$, FSSSA achieves the lowest average and standard deviation on F8 and F9. This indicates that with $sf = 18$, FSSSA exhibits the best optimization performance and the highest stability across the benchmark functions.

From the nature of the box plot [60], it can be seen in F9 that the $sf = 18$ has the smallest quartile range of box plot, the $sf = 15$ and the $sf = 20$ have wide interquartile ranges of box plot. Consequently, the optimal solution is more stable when $sf = 18$. In F8, while $sf = 10$, $sf = 18$, $sf = 30$, and $sf = 40$ have a similar quartile range of box plots, the median of $sf = 18$ is smaller than the others. It proves that the optimal solution of $sf = 18$ has the most advantages.

In this paper, the value of the scaling factor sf selected is 18, which improves the accuracy and stability of the FSSSA.

The specific settings for the other parameters of FSSSA have been detailed in Table 2. Most of these parameters are fixed and have minimal impact on the algorithm's performance across benchmark functions. Although this study did not employ a parameter tuning mechanism, such as CRS-Tuning [61], F-Race [62], or REVAC [63], a systematic process of experimental adjustment combined with a deep understanding of algorithm characteristics, as well as drawing from previous literature on SSA research and empirical outcomes, led to the identification of a well-considered parameter configuration. This approach aimed to attain optimal algorithm performance for specific problems, ensuring the rigor and replicability of the experiments conducted in this study.

4.3. Compared with SSA

In this section, convergence accuracy under the same iteration, evaluation times under the same accuracy, balance and diversity analysis, and the nonparametric statistical tests experiments are used to verify the effectiveness of the FSSSA. Table 5 shows that four types of SSA are formed by combining policies. FIS is represented as the fuzzy inference system strategy, SCM is represented as the sine cosine mutation strategy, WAS is represented as the wide-area search mechanism strategy. A 0 means that the strategy is not used, and 1 means that the strategy is used.

Table 5. The combined form of the three strategies.

	FIS	**SCM**	**WAS**
SSA	0	0	0
FSSA	1	1	0
SSSA	0	0	1
FSSSA	1	1	1

4.3.1. Convergence Accuracy under Fixed Number of Iterations

This section evaluates the convergence accuracy of SSA and improved SSA under fixed 20,000 FEs on the benchmark function in Table 3. The parameter Settings of the algorithm are shown in Table 2. The benchmark function selects 30 dimensions. The experimental results are the mean value and standard deviation after 30 independent runs. The ordinate is the logarithm base 10 of fitness value, and the best results of each test function are shown in bold in the table.

Figure 8 shows the convergence curves of SSA, FSSA, SSSA, and FSSSA in partial functions. Table 6 records the mean value, standard deviation, and optimal value of each algorithm run 30 times.

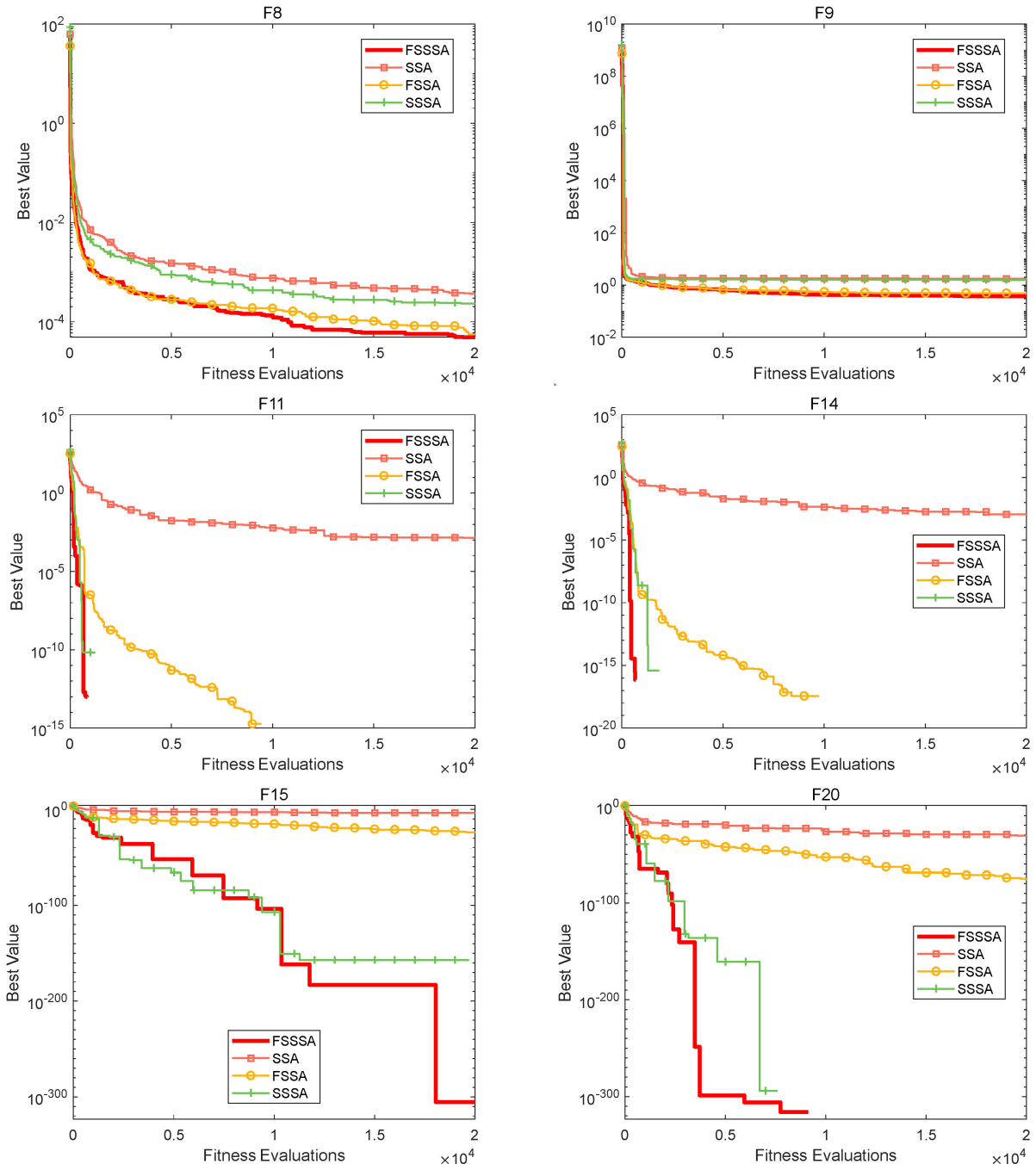

Figure 8. Convergence curve with 20,000 (30 dimensions).

Table 6. Convergence accuracy for 20,000 EFs (30 dimensions).

Function	Item	SSA	FSSA	SSSA	FSSSA
F1	mean	3.63×10^{-5}	4.20×10^{-26}	1.17×10^{-158}	$\mathbf{6.17 \times 10^{-316}}$
	std	1.86×10^{-4}	1.08×10^{-25}	6.43×10^{-158}	**0**
	rank	4	3	2	**1**
F2	mean	3.55×10^{-3}	4.21×10^{-13}	3.68×10^{-154}	$\mathbf{7.81 \times 10^{-216}}$
	std	1.04×10^{-2}	4.54×10^{-13}	2.02×10^{-153}	**0**
	rank	4	3	2	**1**

Table 6. *Cont.*

Function	Item	SSA	FSSA	SSSA	FSSSA
F3	mean	2.63×10^{-1}	2.00×10^{-24}	0	**0**
	std	1.41×10^{1}	5.92×10^{-24}	0	**0**
	rank	4	3	1	**1**
F4	mean	2.14×10^{-4}	4.17×10^{-14}	2.71×10^{-114}	$\mathbf{3.42 \times 10^{-166}}$
	std	2.98×10^{-4}	5.41×10^{-14}	1.48×10^{-113}	**0**
	rank	4	3	2	**1**
F5	mean	4.13×10^{-4}	0	0	**0**
	std	5.53×10^{-4}	0	0	**0**
	rank	4	1	1	**1**
F6	mean	0	0	0	**0**
	std	0	0	0	**0**
	rank	1	1	1	**1**
F7	mean	1.58×10^{-17}	1.97×10^{-50}	0	**0**
	std	5.49×10^{-17}	4.76×10^{-50}	0	**0**
	rank	4	3	1	**1**
F8	mean	3.63×10^{-4}	5.61×10^{-5}	2.34×10^{-4}	$\mathbf{4.87 \times 10^{-5}}$
	std	2.62×10^{-4}	6.17×10^{-6}	2.23×10^{-4}	$\mathbf{5.33 \times 10^{-7}}$
	rank	4	2	3	**1**
F9	mean	1.77	4.93×10^{-1}	1.67	$\mathbf{3.74 \times 10^{-1}}$
	std	4.01×10^{-1}	1.62×10^{-1}	4.01×10^{-1}	$\mathbf{8.21 \times 10^{-2}}$
	rank	4	2	3	**1**
F10	mean	7.81×10^{1}	6.33×10^{-22}	9.76×10^{-211}	**0**
	std	2.19×10^{2}	1.14×10^{-21}	0	**0**
	rank	4	3	2	**1**
F11	mean	1.42×10^{-3}	0	0	**0**
	std	4.02×10^{-3}	0	0	**0**
	rank	4	1	1	**1**
F12	mean	1.99×10^{-4}	4.19×10^{-26}	1.09×10^{-278}	**0**
	std	9.20×10^{-4}	8.85×10^{-26}	0	**0**
	rank	4	3	2	**1**
F13	mean	2.49×10^{-2}	8.61×10^{-14}	0	**0**
	std	3.23×10^{-2}	1.33×10^{-13}	0	**0**
	rank	4	3	1	**1**
F14	mean	1.17×10^{-3}	0	0	**0**
	std	6.03×10^{-3}	0	0	**0**
	rank	4	1	1	**1**
F15	mean	2.66×10^{-5}	1.13×10^{-24}	5.97×10^{-203}	$\mathbf{4.84 \times 10^{-308}}$
	std	1.18×10^{-3}	3.04×10^{-24}	0	**0**
	rank	4	3	2	**1**
F16	mean	2.00×10^{-8}	4.45×10^{-28}	1.27×10^{-278}	**0**
	std	8.65×10^{-8}	1.13×10^{-27}	0	**0**
	rank	4	3	2	**1**
F17	mean	4.39×10^{-5}	0	0	**0**
	std	7.96×10^{-5}	0	0	**0**
	rank	4	1	1	**1**
F18	mean	2.48×10^{-4}	0	0	**0**
	std	9.45×10^{-4}	0	0	**0**
	rank	4	1	1	**1**

Table 6. *Cont.*

Function	Item	SSA	FSSA	SSSA	FSSSA
F19	mean	3.80×10^{-2}	7.17×10^{-14}	8.88×10^{-183}	$\mathbf{8.88 \times 10^{-16}}$
	std	6.00×10^{-2}	9.31×10^{-14}	0	**0**
	rank	4	3	1	**1**
F20	mean	5.10×10^{-29}	5.15×10^{-78}	0	**0**
	std	2.39×10^{-28}	1.96×10^{-77}	0	**0**
	rank	4	3	1	**1**
F21	mean	9.71×10^{-6}	0	0	**0**
	std	1.64×10^{-5}	0	0	**0**
	rank	4	1	1	**1**
F22	mean	8.50×10^{-4}	0	0	**0**
	std	1.79×10^{-3}	0	0	**0**
	rank	4	1	1	**1**
F23	mean	3.51×10^{-12}	3.14×10^{-13}	**0**	7.06×10^{-320}
	std	1.86×10^{-15}	3.69×10^{-13}	**0**	0
	rank	4	3	**1**	2
F24	mean	3.79×10^{-6}	6.02×10^{-26}	8.83×10^{-183}	$\mathbf{9.88 \times 10^{-324}}$
	std	7.94×10^{-6}	1.90×10^{-25}	1.48×10^{-113}	**0**
	rank	4	3	2	**1**

AverageRank

	Rank	+/−/=	AVR
FSSSA	1	~	1.041
SSA	4	23/0/1	3.875
FSSA	3	16/0/8	2.25
SSSA	2	10/1/13	1.375

In Table 6, the symbols "+", "−", and "=", respectively, indicate that the average convergence accuracy of FSSSA over 30 runs (mean) is better than, worse than, or equal to the average convergence accuracy of the comparison algorithms (mean). Average Value Rank (AVR) represents the average ranking value of each algorithm in 24 functions, and "Rank" indicates the final rank. The average ranking shows that the AVR of FSSSA is 1.041, which is the lowest among all algorithms, indicating the best optimization performance of FSSSA. The data in the "+/−/=" indicates that FSSSA outperforms SSA in 23 benchmark functions, outperforms FSSA in 16 benchmark functions, and outperforms SSSA in 10 benchmark functions.

Combining Figure 8 and Table 6, except for F6, FSSA, SSSA, and FSSSA outperform SSA in the remaining 23 benchmark functions, indicating that the three variants of SSA proposed in this paper exhibit significant improvements. SSSA demonstrates higher convergence accuracy than FSSA in 14 functions. Most of these functions are unimodal functions or separable multimode functions, and the local optimums of multimode functions are relatively more and not far away. This puts a high requirement on the exploitation of the algorithms. The WAS strategy can effectively improve the exploitation of the SSA, so the WAS strategy has better optimization performance when solving the above functions. The convergence accuracy of FSSA at F8 and F9 is better than SSSA. The slope of the F8 function is small and the distance between the local optimal values of F9 is far. These two kinds of functions require more exploration ability of algorithms. The FSSA containing SCM and FIS strategy is better than other algorithms in these kinds of functions. It is proved that the SCM and the FIS strategies can improve the exploration of the algorithm.

In addition, The SSSA can find the optimal value on most functions. Therefore, the algorithm improved by the WAS strategy has better convergence accuracy. The standard deviation of the FSSA is minor, so the algorithm improved by FIS and SCM strategy has a more stable optimization effect. The FSSSA notably outperforms other algorithms on the 24 functions.

Therefore, the optimization performance of the FSSSA combined with the three strategies can adapt to more types of functions and the optimization performance is more stable.

4.3.2. The Number of Functional Evaluations with Fixed Target Accuracy

In order to intuitively demonstrate the effectiveness of the improvement strategy, this study tested the number of FEs required for SSA, FSSA, SSSA, and FSSSA on 24 benchmark functions under a fixed accuracy. The fixed accuracy was set to 1.00×10^{-100}, and the dimensions of the benchmark functions were set to 50. The experimental results were compared and analyzed based on the average, maximum, and minimum values obtained from 30 repetitions. To facilitate comparison and observation, the experimental data were rounded to integers. The stopping criteria for this experiment were set to reach either the maximum FEs or achieve the target accuracy. The maximum number of FEs was set to 20,000. The experimental data are presented in Table 7.

Table 7. Experimental results with fixed convergence accuracy (50 dimensions).

Function	Limitation	Item	SSA	FSSA	SSSA	FSSSA
F1	1.00×10^{-100}	max	20,000	20,000	17,400	**10,815**
		min	20,000	20,000	400	**200**
		mean	20,000	20,000	3913	**3622**
F2	1.00×10^{-100}	max	20,000	20,000	20,000	20,000
		min	20,000	20,000	200	**100**
		mean	20,000	20,000	7370	**6630**
F3	1.00×10^{-100}	max	20,000	20,000	20,000	**13,796**
		min	20,000	20,000	200	**100**
		mean	20,000	20,000	6881	**4043**
F4	1.00×10^{-100}	max	20,000	20,000	20,000	20,000
		min	20,000	20,000	900	**500**
		mean	20,000	20,000	7774	**7584**
F5	1.00×10^{-100}	max	20,000	20,000	2600	**1814**
		min	20,000	1729	100	**100**
		mean	20,000	10,788	764	**496**
F6	1.00×10^{-100}	max	800	300	300	**200**
		min	100	100	100	**100**
		mean	253	167	180	**127**
F7	1.00×10^{-100}	max	20,000	20,000	15,333	**6875**
		min	20,000	20,000	100	**100**
		mean	20,000	20,000	2798	**2014**
F8	1.00×10^{-100}	max	20,000	20,000	20,000	20,000
		min	20,000	20,000	20,000	20,000
		mean	20,000	20,000	20,000	20,000
F9	1.00×10^{-100}	max	20,000	20,000	20,000	20,000
		min	20,000	20,000	20,000	20,000
		mean	20,000	20,000	20,000	20,000
F10	1.00×10^{-100}	max	20,000	20,000	20,000	**14,501**
		min	20,000	20,000	100	**100**
		mean	20,000	20,000	4787	**4355**
F11	1.00×10^{-100}	max	20,000	16,667	2100	**1713**
		min	20,000	1188	100	**100**
		mean	20,000	9425	680	**537**
F12	1.00×10^{-100}	max	20,000	20,000	20,000	**14,308**
		min	20,000	20,000	100	**100**
		mean	20,000	20,000	4397	**4382**

Table 7. *Cont.*

Function	Limitation	Item	SSA	FSSA	SSSA	FSSSA
F13	1.00×10^{-100}	max	20,000	20,000	**419**	3861
		min	20,000	20,000	**100**	100
		mean	20,000	20,000	**309**	988
F14	1.00×10^{-100}	max	20,000	14,613	3800	**1941**
		min	20,000	834	100	**100**
		mean	20,000	7532	843	**590**
F15	1.00×10^{-100}	max	20,000	20,000	16,800	**16,159**
		min	20,000	20,000	100	**100**
		mean	20,000	20,000	5858	**5053**
F16	1.00×10^{-100}	max	20,000	20,000	20,000	**11,413**
		min	20,000	20,000	200	**100**
		mean	20,000	20,000	4684	**4527**
F17	1.00×10^{-100}	max	20,000	17,836	1300	**1980**
		min	20,000	2555	200	**100**
		mean	20,000	10,450	850	**601**
F18	1.00×10^{-100}	max	20,000	13,683	2900	**2401**
		min	20,000	1545	100	**100**
		mean	20,000	7524	800	**599**
F19	1.00×10^{-100}	max	20,000	20,000	20,000	20,000
		min	20,000	20,000	20,000	20,000
		mean	20,000	20,000	20,000	20,000
F20	1.00×10^{-100}	max	20,000	20,000	42,317	**41,071**
		min	20,000	20,000	100	**100**
		mean	20,000	20,000	1477	**1115**
F21	1.00×10^{-100}	max	20,000	19,994	1904	**519**
		min	20,000	1031	100	**100**
		mean	20,000	13,669	575	**270**
F22	1.00×10^{-100}	max	20,000	18,429	2401	**3395**
		min	20,000	1982	100	**100**
		mean	20,000	10,792	824	**751**
F23	1.00×10^{-100}	max	20,000	20,000	**511**	3915
		min	20,000	20,000	201	**100**
		mean	20,000	20,000	**296**	1507
F24	1.00×10^{-100}	max	20,000	20,000	15,900	**14,239**
		min	20,000	20,000	200	**200**
		mean	20,000	20,000	4524	**4073**

As shown in Table 7, SSA struggled to achieve the target accuracy within 20,000 evaluations. FSSA achieved the target accuracy on 8 benchmark functions, while SSSA and FSSSA achieved the target accuracy on 21 benchmark functions. Except for F13 and F23, FSSSA consistently achieved the target accuracy with the fewest FEs. Although FSSSA performed relatively weaker compared to SSSA on F13 and F23, overall, FSSSA demonstrated strong stability.

In conclusion, among the 24 benchmark functions under a fixed accuracy, FSSSA exhibited the best optimization performance. Thus, FSSSA effectively improved the performance of SSA.

4.3.3. Nonparametric Statistical Tests with SSA

In this section, non-parametric statistical tests are used to examine the performance differences among the four algorithms listed in Table 5 [64]. This study adopts the Wilcoxon signed-rank test at a 5% significance level for statistical analysis. The *p*-values computed from the Wilcoxon signed-rank test are adjusted using the Bonferroni–Holm correction [65]. The computed and adjusted *p*-values are presented in Table 8. In the Wilcoxon signed-rank

test, if the Corrected p-value is less than 0.05, it indicates that the improved algorithm shows significant differences compared to SSA. If the p-value is greater than 0.05, it means that the improved algorithm is not significantly different from SSA.

Table 8. p-value results for SSA and improved SSA.

Comparison	R+	R−	p-Value	Corrected p-Value	<0.05?
FSSA vs. SSA	297	3	2.70×10^{-5}	2.70×10^{-5}	yes
SSSA vs. SSA	297	3	2.70×10^{-5}	8.10×10^{-5}	yes
FSSSA vs. SSA	297	3	2.70×10^{-5}	5.40×10^{-5}	yes

Based on the results in Table 8, it can be observed that the proposed FSSA, SSSA, and FSSSA algorithms exhibit significant differences when compared to the original SSA algorithm.

4.3.4. Balance and Diversity Analysis

Striking a balance between exploration and exploitation is one of the key factors in designing new algorithms or enhancing existing ones. Exploration involves traversing the entire search space to discover promising regions, known as global search capability. The exploitation phase involves refining the search by utilizing the promising regions already discovered to find the optimal solution, known as local search capability. When an appropriate equilibrium is achieved between exploration and exploitation, algorithms tend to exhibit favorable convergence behavior [66,67].

In this study, the population diversity measurement method proposed by Hussain et al. [68] was adopted to assess the algorithm's balancing capability. This method assesses the algorithm's balancing capability by calculating the average variation of distances within the population across different dimensions. If the average value decreases gradually during iterations, it is considered as the exploitation phase. Conversely, if the average value increases gradually, it is considered as the exploration phase. If the dimension diversity decreases while the average value remains unchanged, it indicates that the algorithm has converged.

Despite its simplicity and intuitiveness, the diversity measurement method is limited to evaluating the entire population and cannot directly express the exploration or exploitation status of individual solutions within the population [69]. In this study, this measurement method was employed to evaluate the balancing and diversity performance of FSSSA and SSA across 24 test functions, providing clear and substantial evidence in support of the effectiveness of FSSSA's improvement strategy.

However, in practical problem-solving scenarios, different problems may require adjusting the trade-off between exploration and exploitation according to specific circumstances. Therefore, in practical applications, it may be necessary to employ more sophisticated methods and metrics to determine when to prioritize exploration or exploitation, in order to better optimize the algorithm's performance and discover superior solutions [69].

The balance and diversity of FSSSA and SSA are tested on 24 test functions, as shown in Figure 9. Figure 9a is the balance analysis diagram of FSSSA, Figure 9b is the balance analysis diagram of SSA, and Figure 9c is the diversity analysis diagram of FSSSA and SSA. In Figure 9a,b, the x-axis is the number of iterations, and the y-axis is the percentage. There are two curves in Figure 9a,b, the red is the exploitation curve, and the blue is the exploration curve, respectively representing the proportion of the exploitation and exploration in a certain iteration. In Figure 9c, the x-axis is the number of iterations, and the y-axis is the population diversity. The red and blue curves represent the diversity of FSSSA and SSA.

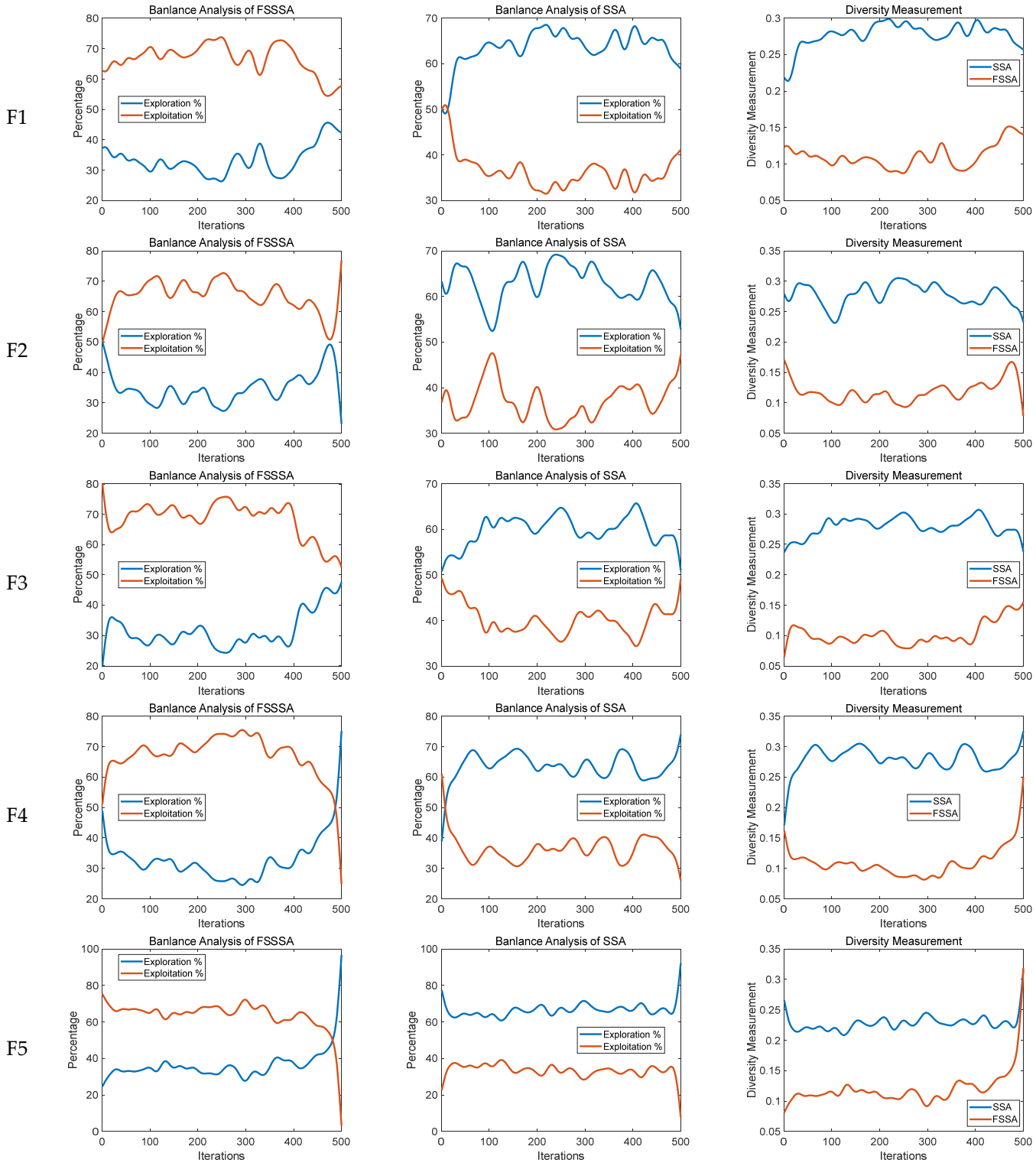

Figure 9. *Cont.*

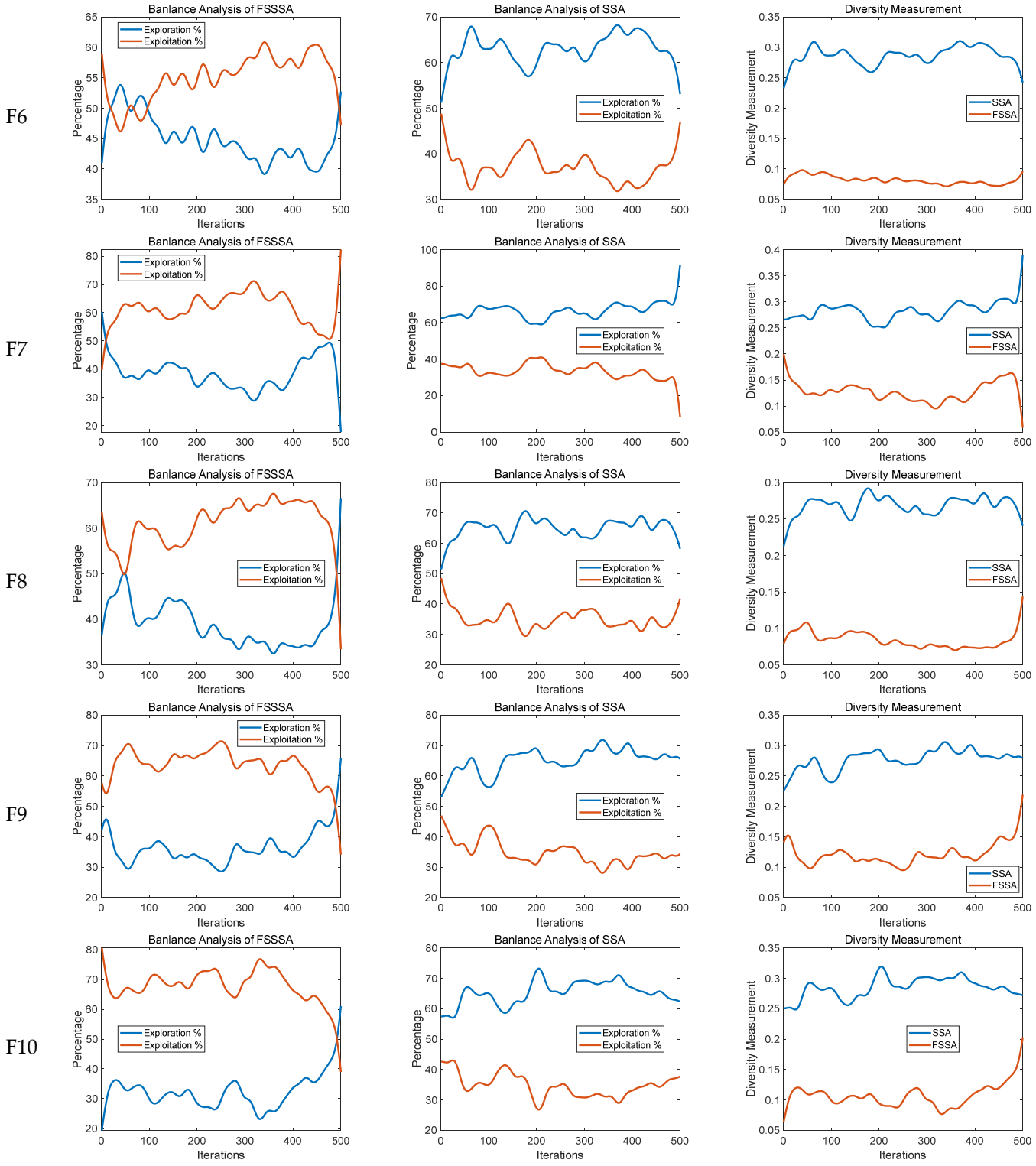

Figure 9. *Cont.*

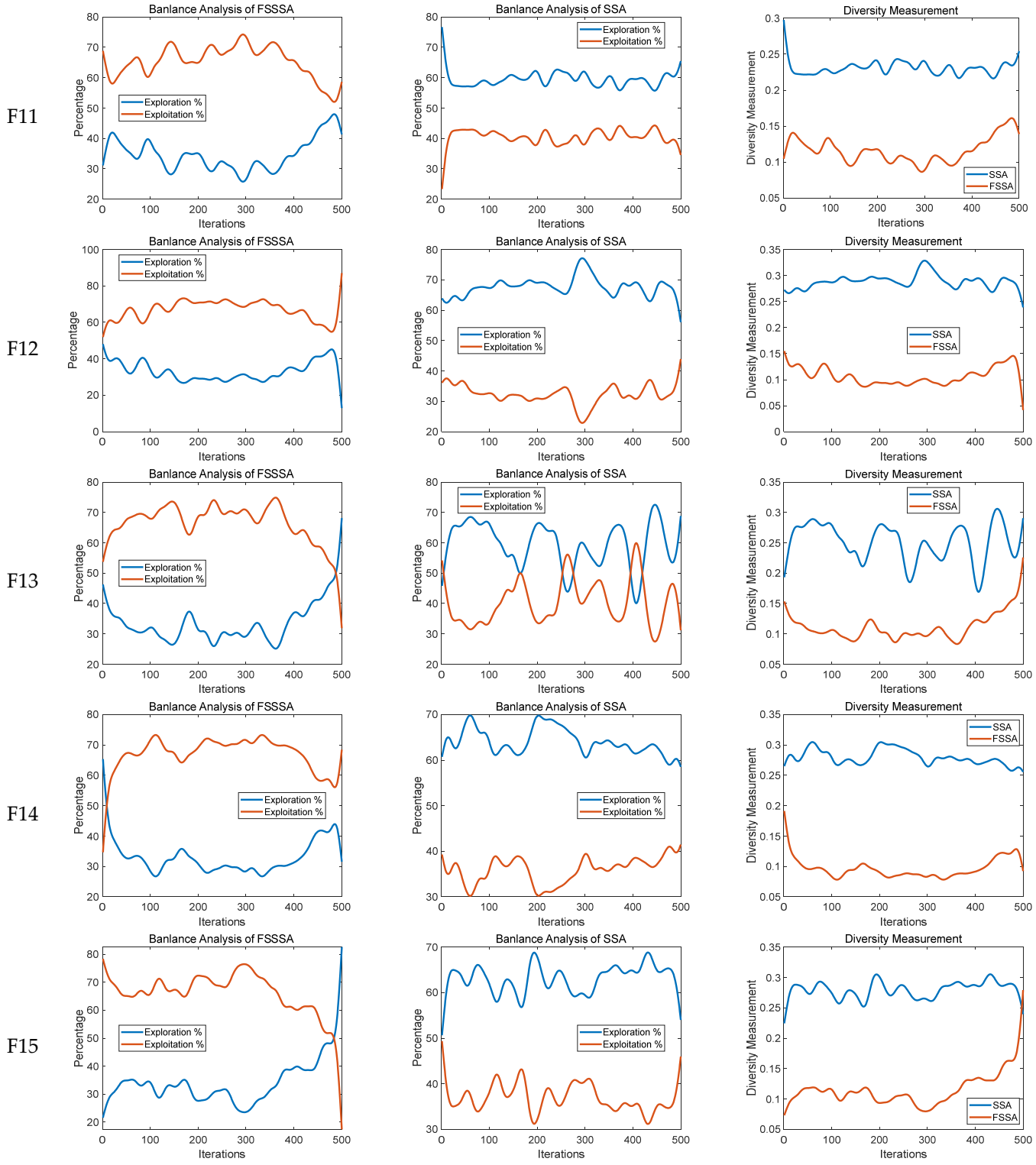

Figure 9. *Cont.*

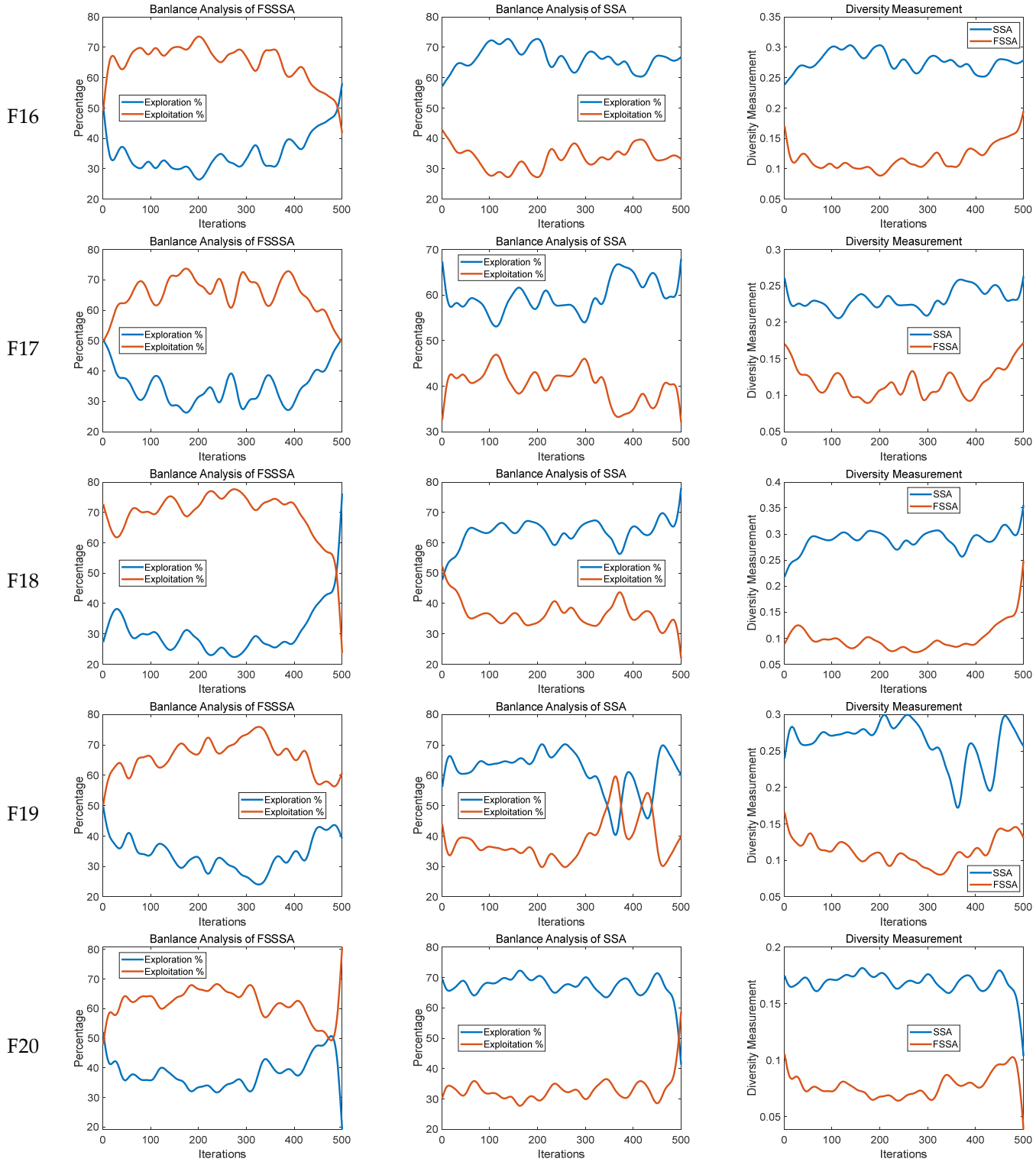

Figure 9. *Cont.*

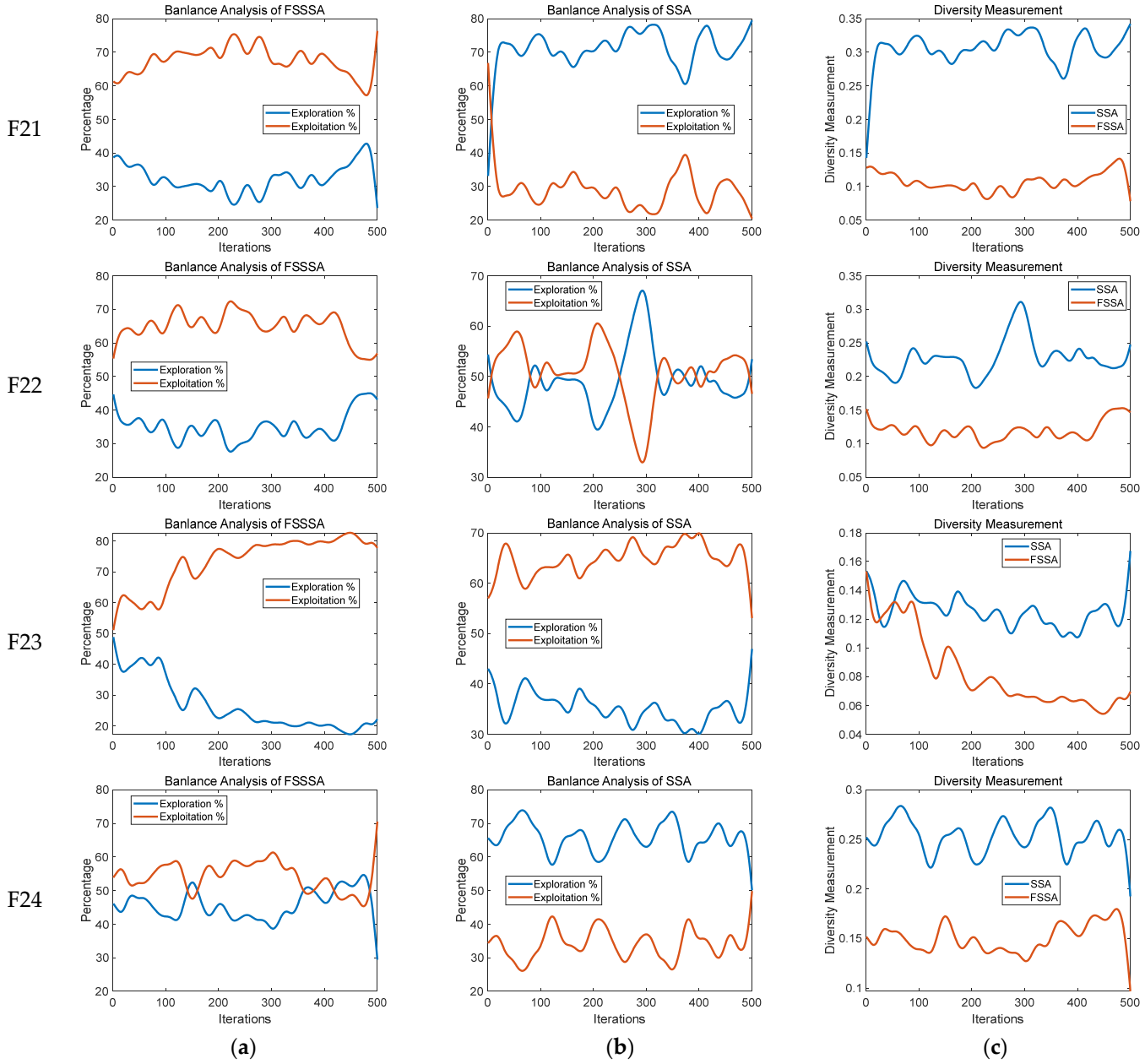

Figure 9. (**a**) the balance analysis of FSSSA; (**b**) The balance analysis of SSA; (**c**) The diversity analysis of FSSSA and SSA.

It can be seen from the balance analysis diagram of FSSSA and SSA that, except for F22 and F23, the exploration of the SSA is larger than the exploitation. Therefore, the local search ability is weak, resulting in low convergence accuracy. However, in the search process of the FSSSA, the exploitation stage is larger than the exploration stage, which provides excellent local search ability for FSSSA. Except for F23, the proportion of the exploration stage also increases steadily in the late iteration stage to prevent FSSSA from falling into the local optimum.

In addition, according to the diversity analysis diagram of FSSSA and SSA, the population diversity of SSA is higher due to the more stages of random location update and strong global search ability. Although the population diversity of the FSSSA is low, most of the function iterations show an upward trend in the late stage. It ensures the population diversity of the FSSSA in the late-stage search.

The balance and diversity analysis of SSA and FSSSA show that the proposed FSSSA can effectively balance the exploration and exploitation of the algorithm and performs perfectly.

4.4. FSSSA with Advanced Metaheuristic Algorithms

To further verify the optimization performance of the FSSSA, eight metaheuristic algorithms are selected for experimental testing on the benchmark function in Table 3. FSSSA compared with White Shark Optimizer, Runge Kutta Optimization algorithm, Weighted Mean of Vectors algorithm, Equilibrium Optimizer, Group Teaching Optimization Algorithm with Information Sharing, Gull Optimization Algorithm, Grey Wolf Optimizer and Ensemble Sinusoidal Differential Covariance Matrix Adaptation with Euclidean Neighborhood (LSHADE-cnEpSin, one of the winners of CEC 2017 competition). The population number of all algorithms is set as 50, the population of the LSHADE-cnEpSin is not set because the population of it changes with the update process. The number of FEs is 20,000. The other parameters are set as shown in Table 2, and the dimension of the benchmark function is 50.

4.4.1. Comparative Analysis of Convergence Accuracy

The experimental data are shown in Table 9, and the comparison of convergence curves of 50 dimensions is shown in Figure 10.

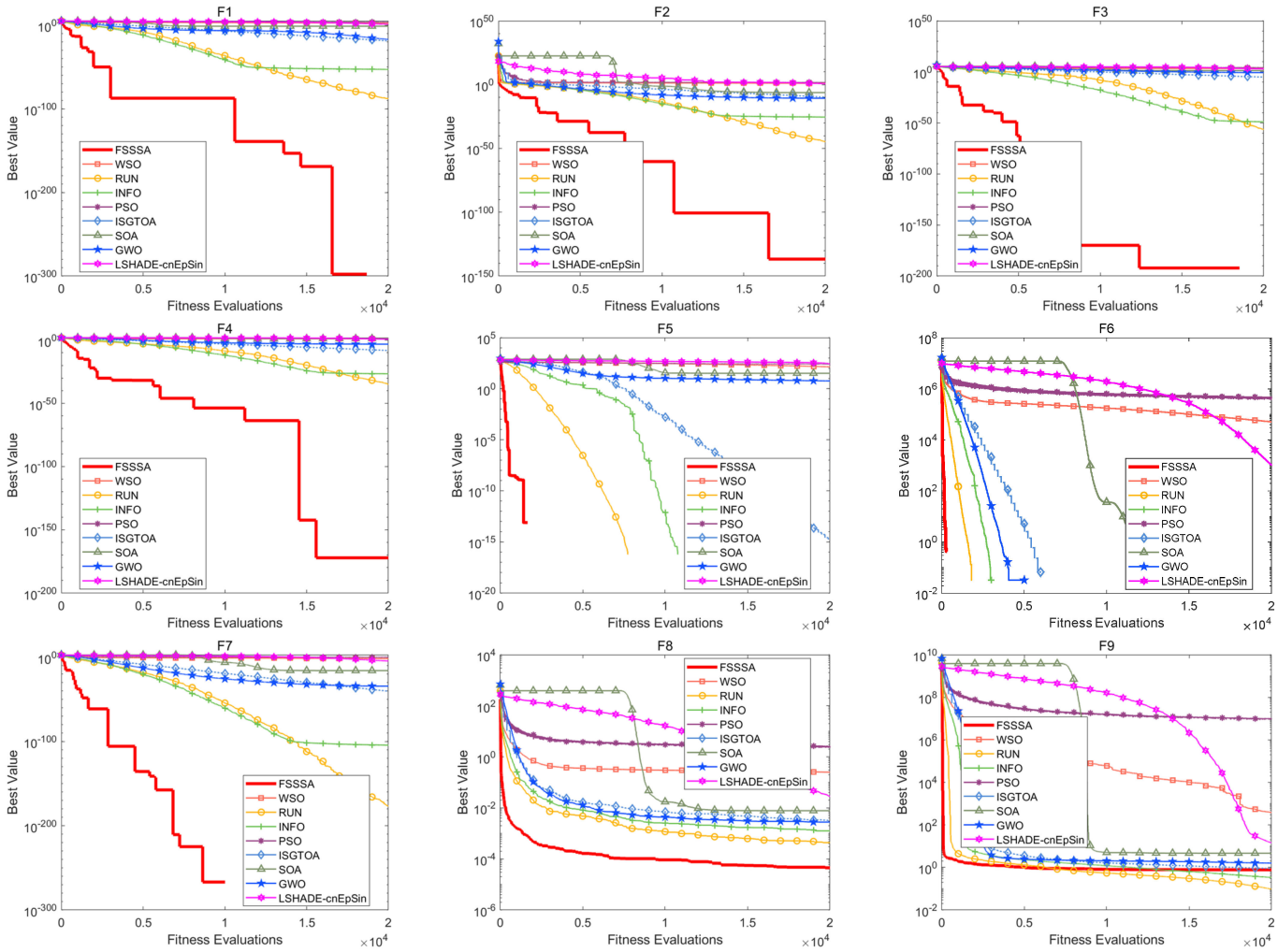

Figure 10. *Cont.*

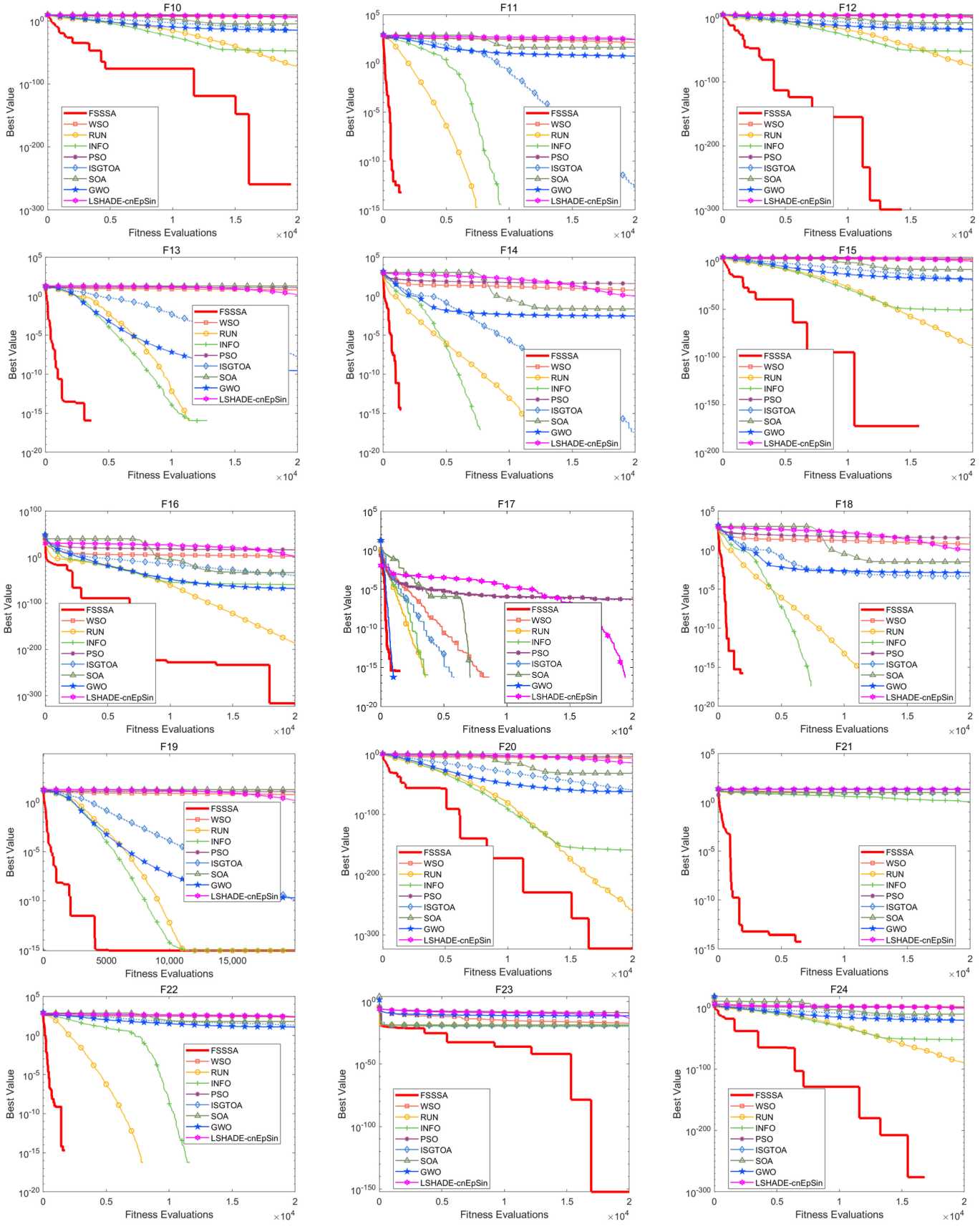

Figure 10. Convergence curve with 20,000 FEs (50 dimensions).

Table 9. Convergence accuracy under 20,000 EFs (50 dimensions).

Fun	Item	WSO	RUN	INFO	PSO	ISGTOA	SOA	GWO	LSHADE-cnEpSin	FSSSA
F1	mean	5.55×10^{2}	5.92×10^{-77}	5.40×10^{-53}	4.71×10^{3}	1.09×10^{-19}	5.69×10^{-8}	2.33×10^{-18}	1.18×10^{1}	**0**
	std	2.40×10^{2}	2.63×10^{-76}	3.76×10^{-53}	1.52×10^{3}	5.42×10^{-20}	1.29×10^{-7}	2.08×10^{-18}	4.23	**0**
	rank	8	2	3	9	4	6	5	7	**1**
F2	mean	1.09×10^{1}	2.38×10^{-45}	3.95×10^{-26}	5.97×10^{1}	1.73×10^{-10}	6.09×10^{-7}	2.27×10^{-11}	5.02	$\mathbf{1.96 \times 10^{-137}}$
	std	2.18	1.09×10^{-44}	1.41×10^{-26}	3.05×10^{1}	7.15×10^{-11}	4.51×10^{-7}	9.11×10^{-12}	1.44	$\mathbf{1.07 \times 10^{-136}}$
	rank	8	2	3	9	5	6	4	7	**1**
F3	mean	2.03×10^{3}	4.26×10^{-57}	1.17×10^{-49}	1.89×10^{4}	7.61×10^{-6}	2.01×10^{1}	2.19×10^{-1}	8.95×10^{2}	**0**
	std	6.06×10^{2}	1.53×10^{-56}	1.28×10^{-49}	6.38×10^{3}	1.44×10^{-5}	3.73×10^{1}	4.53×10^{-1}	2.46×10^{2}	**0**
	rank	8	2	3	9	4	6	5	7	**1**
F4	mean	1.25×10^{1}	1.19×10^{-35}	2.55×10^{-27}	2.91×10^{1}	6.53×10^{-9}	1.75×10^{1}	6.51×10^{-4}	6.12	**0**
	std	1.67	2.74×10^{-35}	1.58×10^{-27}	3.49	2.79×10^{-9}	2.31×10^{1}	5.97×10^{-4}	1.26	**0**
	rank	7	2	3	9	4	8	5	6	**1**
F5	mean	1.32×10^{2}	0	0	2.65×10^{2}	6.51×10^{-16}	3.31×10^{1}	7.85	2.72×10^{2}	$\mathbf{5.64 \times 10^{-173}}$
	std	4.38×10^{1}	0	0	3.57×10^{1}	3.25×10^{-15}	2.48×10^{1}	8.71	1.95×10^{1}	**0**
	rank	7	1	1	8	4	6	5	9	**1**
F6	mean	5.79×10^{4}	0	0	4.50×10^{5}	0	3.33×10^{-2}	0	1.01×10^{3}	**0**
	std	1.50×10^{4}	0	0	2.14×10^{5}	0	1.83×10^{-1}	0	3.37×10^{2}	**0**
	rank	8	1	1	9	1	6	1	7	**1**
F7	mean	1.84×10^{-2}	2.11×10^{-177}	3.26×10^{-105}	7.93×10^{-1}	2.66×10^{-41}	1.19×10^{-16}	3.95×10^{-35}	3.17×10^{-5}	**0**
	std	1.38×10^{-2}	0	4.76×10^{-105}	4.81×10^{-1}	3.21×10^{-41}	5.78×10^{-16}	6.97×10^{-35}	2.39×10^{-5}	**0**
	rank	8	2	3	9	4	6	5	7	**1**
F8	mean	2.49×10^{-1}	4.34×10^{-4}	1.25×10^{-3}	2.53	3.12×10^{-3}	7.75×10^{-3}	2.77×10^{-3}	2.87×10^{-2}	$\mathbf{4.49 \times 10^{-5}}$
	std	1.24×10^{-1}	3.12×10^{-4}	1.04×10^{-3}	7.59×10^{-1}	1.60×10^{-3}	4.93×10^{-3}	1.36×10^{-3}	1.11×10^{-2}	$\mathbf{5.85 \times 10^{-5}}$
	rank	8	2	3	9	5	6	4	7	**1**
F9	mean	3.82×10^{2}	$\mathbf{9.44 \times 10^{-2}}$	3.37×10^{-1}	1.01×10^{7}	8.86×10^{-1}	4.66	1.60	1.43×10^{1}	7.77×10^{-1}
	std	7.69×10^{2}	$\mathbf{7.28 \times 10^{-2}}$	2.55×10^{-1}	7.60×10^{6}	3.83×10^{-1}	5.66×10^{-1}	3.55×10^{-1}	5.39	2.39×10^{-1}
	rank	8	1	2	9	4	6	5	7	3
F10	mean	2.50×10^{6}	6.13×10^{-73}	2.81×10^{-48}	7.92×10^{7}	1.06×10^{-15}	1.98×10^{-5}	2.25×10^{-15}	2.01×10^{5}	**0**
	std	1.55×10^{6}	3.07×10^{-72}	2.67×10^{-48}	2.92×10^{7}	7.11×10^{-16}	4.49×10^{-5}	1.86×10^{-15}	6.97×10^{4}	**0**
	rank	8	2	3	9	4	6	5	7	**1**

Table 9. *Cont.*

Fun	Item	WSO	RUN	INFO	PSO	ISGTOA	SOA	GWO	LSHADE-cnEpSin	FSSSA
F11	mean	1.44×10^{2}	0	0	2.54×10^{2}	1.02×10^{-13}	3.04×10^{1}	7.43	2.68×10^{2}	**0**
	std	5.95×10^{1}	0	0	5.80×10^{1}	4.61×10^{-13}	2.15×10^{1}	5.01	2.26×10^{1}	**0**
	rank	7	1	1	8	4	6	5	9	**1**
F12	mean	5.54×10^{2}	3.22×10^{-76}	5.79×10^{-53}	4.80×10^{3}	1.86×10^{-19}	2.55×10^{-8}	2.22×10^{-18}	1.16×10^{1}	**0**
	std	1.76×10^{2}	1.61×10^{-75}	5.66×10^{-53}	1.40×10^{3}	2.17×10^{-19}	2.74×10^{-8}	2.69×10^{-18}	3.19	**0**
	rank	8	2	3	9	4	6	5	7	**1**
F13	mean	6.30	0	0	1.27×10^{1}	1.74×10^{-8}	2.00×10^{1}	2.91×10^{-10}	1.54	**0**
	std	6.47×10^{-1}	0	0	1.11	9.46×10^{-8}	7.87×10^{-4}	1.46×10^{-10}	2.64×10^{-1}	**0**
	rank	7	1	1	8	5	9	4	6	**1**
F14	mean	6.17	0	0	3.68×10^{1}	0	3.00×10^{-2}	1.67×10^{-3}	1.11	**0**
	std	1.94	0	0	1.36×10^{1}	0	5.80×10^{-2}	5.22×10^{-3}	4.54×10^{-2}	**0**
	rank	8	1	1	9	1	6	5	7	**1**
F15	mean	1.10×10^{2}	1.49×10^{-89}	1.15×10^{-51}	9.21×10^{2}	3.18×10^{-20}	4.71×10^{-9}	5.57×10^{-19}	2.90	**0**
	std	4.30×10^{1}	4.90×10^{-89}	1.39×10^{-51}	3.31×10^{2}	2.95×10^{-20}	6.36×10^{-9}	5.23×10^{-19}	1.26	**0**
	rank	8	2	3	9	4	6	5	7	**1**
F16	mean	4.44×10^{1}	1.92×10^{-187}	3.64×10^{-60}	9.50×10^{15}	1.34×10^{-40}	3.90×10^{-34}	1.89×10^{-68}	2.07×10^{2}	**0**
	std	1.16×10^{2}	0	8.31×10^{-60}	4.56×10^{16}	2.20×10^{-40}	1.37×10^{-33}	9.79×10^{-68}	4.93×10^{2}	**0**
	rank	7	2	4	9	5	6	3	8	**1**
F17	mean	0	0	0	5.89×10^{-7}	0	0	0	0	**0**
	std	0	0	0	1.36×10^{-6}	0	0	0	0	**0**
	rank	1	1	1	9	1	1	1	1	**1**
F18	mean	5.83	0	0	4.07×10^{1}	6.66×10^{-10}	3.07×10^{-2}	3.68×10^{-3}	1.10	**0**
	std	1.55	0	0	1.67×10^{1}	3.65×10^{-9}	3.68×10^{-2}	8.36×10^{-3}	3.17×10^{-2}	**0**
	rank	8	1	1	9	4	6	5	7	**1**
F19	mean	5.99	8.88×10^{-16}	8.88×10^{-16}	1.26×10^{1}	6.55×10^{-1}	2.00×10^{1}	2.78×10^{-10}	1.48	$\mathbf{8.88 \times 10^{-16}}$
	std	7.18×10^{-1}	0	0	1.34	3.59	9.10×10^{-4}	1.35×10^{-10}	2.37×10^{-1}	**0**
	rank	7	1	1	8	5	9	4	6	**1**
F20	mean	6.63×10^{-7}	5.25×10^{-280}	5.15×10^{-160}	4.60×10^{-5}	3.93×10^{-60}	1.58×10^{-34}	2.28×10^{-61}	6.17×10^{-15}	**0**
	std	9.73×10^{-7}	0	1.21×10^{-159}	5.01×10^{-5}	1.05×10^{-59}	8.49×10^{-34}	1.18×10^{-60}	1.38×10^{-14}	**0**
	rank	8	2	3	9	5	6	4	7	**1**

Table 9. *Cont.*

Fun	Item	WSO	RUN	INFO	PSO	ISGTOA	SOA	GWO	LSHADE-cnEpSin	FSSSA
F21	mean	7.75	2.32×10^{2}	1.58×10^{1}	2.05×10^{2}	9.76	9.51	2.04×10^{1}	1.97×10^{1}	**0**
	std	2.85	3.58×10^{3}	2.45×10^{2}	5.57	1.04	9.52×10^{-1}	5.20×10^{-1}	3.85×10^{-1}	**0**
	rank	4	3	2	9	6	5	8	7	**1**
F22	mean	2.20×10^{2}	0	0	2.73×10^{2}	1.39×10^{1}	5.44×10^{1}	1.27×10^{1}	2.06×10^{2}	**0**
	std	6.10×10^{1}	0	0	3.75×10^{1}	3.53×10^{1}	4.64×10^{1}	6.99	2.35×10^{1}	**0**
	rank	8	1	1	9	5	6	4	7	**1**
F23	mean	3.57×10^{-18}	1.36×10^{-19}	1.51×10^{-20}	8.33×10^{-10}	5.43×10^{-20}	1.70×10^{-19}	3.63×10^{-12}	1.33×10^{-14}	$\mathbf{5.57 \times 10^{-153}}$
	std	1.34×10^{-17}	2.62×10^{-20}	8.87×10^{-21}	2.76×10^{-9}	9.49×10^{-21}	4.14×10^{-21}	9.16×10^{-12}	2.28×10^{-14}	$\mathbf{5.57 \times 10^{-152}}$
	rank	6	4	2	9	3	5	8	7	**1**
F24	mean	3.76×10^{1}	6.81×10^{-90}	7.45×10^{-53}	4.36	1.06×10^{-21}	2.70×10^{-11}	5.13×10^{-21}	1.44×10^{-1}	**0**
	std	1.66×10^{1}	2.92×10^{-89}	6.86×10^{-53}	1.47	1.01×10^{-21}	7.62×10^{-11}	4.71×10^{-21}	4.57×10^{-2}	**0**
	rank	9	2	3	8	4	6	5	7	**1**

Average Rank

Algorithm	Rank	+/−/=	AVR
FSSSA	1	~	1.083
WSO	8	23/0/1	7.250
RUN	2	14/1/9	1.708
INFO	3	14/1/9	2.167
PSO	9	24/0/0	8.792
ISGTOA	4	21/0/3	3.958
SOA	6	23/0/1	6.042
GWO	5	22/0/2	4.583
LSHADE-cnEpSin	7	23/0/1	6.833

From the convergence curve in Figure 10, it can be observed that, except for F9, FSSSA outperforms other algorithms in the remaining functions. In F9, although FSSSA exhibits weaker optimization performance compared to RUN and ISGTOA, it demonstrates faster convergence speed in 24 functions compared to the other 8 algorithms.

In Table 9, the symbols "+", "−", and "=", respectively, indicate that the average convergence accuracy of FSSSA over 30 runs (mean) is better than, worse than, or equal to the average convergence accuracy of the comparison algorithms (mean). AVR represents the average ranking value of each algorithm in 24 functions, and "Rank" indicates the final rank. Considering the data comparison in Table 9, it can be deduced that FSSSA is capable of finding the theoretical optimal values in 18 benchmark functions. Apart from F9, FSSSA's convergence accuracy is higher than other algorithms in 23 benchmark functions. Regarding F9, RUN achieves the highest convergence accuracy, while FSSSA exhibits the fastest convergence speed.

The data in the "+/−/=" rows indicates that the average convergence accuracy of FSSSA over 30 runs (mean) outperforms SOA and LSHADE-cnEpSin in 23 benchmark functions, outperforms RUN and INFO in 14 benchmark functions, outperforms PSO in 24 benchmark functions, outperforms ISGTOA in 21 benchmark functions, and outperforms GWO in 22 benchmark functions. FSSSA's AVR is 1.083, ranking first among all algorithms.

In conclusion, the FSSSA notably outperforms the other eight metaheuristic algorithms.

4.4.2. Nonparametric Statistical Tests with Other Algorithms

In order to verify the effectiveness of the experiment in the previous section, the nonparametric Wilcoxon signed-rank test and the Friedman test [64] were used to compare the FSSSA with eight algorithms. In the Wilcoxon signed-rank test, The p-values are adjusted using the Bonferroni–Holm correction [65]. The computed and corrected p-values are shown in Table 10. If the Corrected p-value is less than 0.05, it indicates that FSSSA is significantly different. If the p-value is greater than 0.05, it means that FSSSA is not significantly different compared to the other algorithms.

Table 10. p-value results for FSSSA and other metaheuristic algorithms.

Comparison	R+	R−	p-Value	Corrected p-Value	<0.05?
FSSSA vs. WSO	291	9	2.70×10^{-5}	5.4×10^{-5}	yes
FSSSA vs. RUN	229	71	8.99×10^{-3}	7.19×10^{-2}	no
FSSSA vs. INFO	229	71	8.99×10^{-3}	6.29×10^{-2}	no
FSSSA vs. PSO	300	0	1.80×10^{-5}	1.80×10^{-5}	yes
FSSSA vs. ISGTOA	282	28	6.00×10^{-5}	3.60×10^{-4}	yes
FSSSA vs. SOA	291	9	2.70×10^{-5}	8.10×10^{-5}	yes
FSSSA vs. GWO	288	12	4.00×10^{-5}	1.60×10^{-4}	yes
FSSSA vs. LSHADE-cnEpSin	291	9	2.70×10^{-5}	5.40×10^{-5}	yes

From Table 10, it is evident that FSSSA exhibits significant differences when compared with the six metaheuristic algorithms. In comparison with the RUN and INFO algorithms, the corrected p-values are greater than 0.05. Although not statistically significant, FSSSA's optimization performance is superior to both the RUN and INFO algorithms in 14 functions.

The Friedman test allows for multiple comparisons among several algorithms by calculating ranks based on observed results. The experimental results are shown in Table 11. In the Friedman test, to compare the significance differences between FSSSA and other algorithms, we calculate the Critical Difference (CD) using the Bonferroni–Dunn test [70]. The Bonferroni–Dunn test is more suitable for comparing a particular algorithm with the remaining k-1 algorithms. If the average Rank Difference (RD) between FSSSA and other algorithms is greater than the CD, then it indicates that FSSSA statistically outperforms those algorithms. If it is less than the CD, then it indicates that there is no statistically significant difference between them.

Table 11. Friedman test for nine algorithms.

	FSSSA	**WSO**	**RUN**	**INFO**	**PSO**	**ISGTOA**	**SOA**	**GWO**	**LSHADE-cnEpSin**
Mean rank	1.63	7.40	2.25	2.71	8.79	4.25	6.19	4.81	6.98
Final rank	1	8	2	3	9	4	7	6	8
p-value	2.2129×10^{-32}								
Statistic	19.2375								
Critical Difference (CD)	2.1535								

Comparison	**Rank Difference (RD)**	**<2.1535?**
FSSSA-WSO	5.77	yes
FSSSA-RUN	0.62	no
FSSSA-INFO	1.08	no
FSSSA-PSO	7.16	yes
FSSSA-ISGTOA	2.62	yes
FSSSA-SOA	4.56	yes
FSSSA-GWO	3.18	yes
FSSSA-LSHADE-cnEpSin	5.35	yes

From Table 11, it is evident that FSSSA shows significant differences compared to 8 algorithms, and there is no significant difference when compared to RUN and INFO. However, FSSSA ranks first in terms of average ranks. Considering the average ranks and values of each algorithm, the overall analysis indicates that FSSSA's optimization capability surpasses the other 8 algorithms.

It can be observed that in the comparative experiments with SSA, FSSSA demonstrates excellent performance in terms of convergence speed, convergence accuracy, balance, diversity, and non-parametric statistical tests, thereby confirming the effectiveness of the improvement strategies. In the comparative experiments with eight other metaheuristic algorithms, FSSSA achieves the first rank in terms of convergence speed, convergence accuracy, and non-parametric statistical tests, thus demonstrating the outstanding optimization performance of FSSSA.

5. Application to Engineering Optimization Problems

Engineering problems are common challenges and demands in practical applications, often characterized by diversity and complexity, spanning across various fields and contexts. Selecting engineering problems as test cases allows for a better assessment of the algorithm's practicality and adaptability, providing valuable solutions for real-world applications. At the same time, engineering problems are also widely used to verify the performance of optimization algorithms [12,28]. Therefore, this study chose four engineering design problems [71], namely Speed Reducer (SR), Cantilever Beam (CB), Optimal Design of the I-shaped Beam (ODIB), and Piston Lever (PL). These problems originate from different application domains, and each problem has distinct design requirements and optimization objectives. Specific optimization problems are covered in each section.

The FSSSA algorithm was compared with the classic SSA and Biogeography-based Optimization (BBO) [72]. To ensure the fairness of the experiment, the population number was set to 50, the FEs were set to 20,000, and the other parameters were consistent with the original paper. Each algorithm was independently run 30 times and recorded separately after taking its average value to reduce the randomness of the experiment.

5.1. Speed Reducer (SR)

The SR is an essential part of the gearbox in the mechanical system, as shown in Figure 11 [71], and is widely used [73]. This optimization problem has 11 constraints and 7 variables. The mathematical expression to describe this problem is shown in Equation (19).

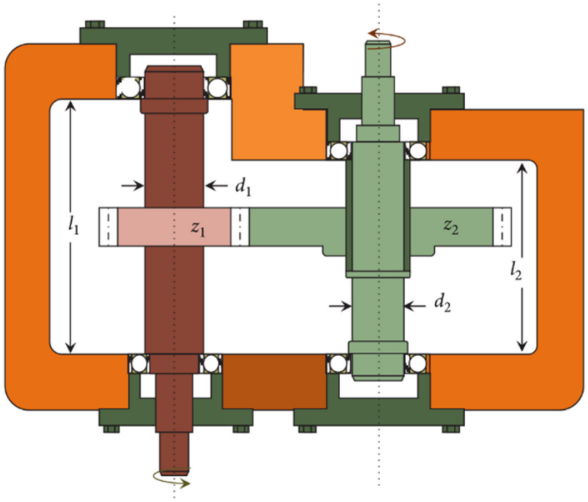

Figure 11. Schematic diagram of speed reducer [71].

Minimize:

$$f(X) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ -1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

Subject to:

$$g_1(X) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0,$$
$$g_2(X) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0,$$
$$g_3(X) = \frac{1.93x_4^3}{x_2x_6^4x_3} - 1 \leq 0,$$
$$g_4(X) = \frac{1.93x_5^3}{x_2x_7^4x_3} - 1 \leq 0,$$
$$g_5(X) = \frac{\sqrt{(745x_4/x_2x_3)^2 + 16.9\times10^6}}{110x_6^3} - 1 \leq 0,$$
$$g_6(X) = \frac{\sqrt{(745x_5/x_2x_3)^2 + 157.5\times10^6}}{85x_7^3} - 1 \leq 0,$$
$$g_7(X) = \frac{x_2x_3}{40} - 1 \leq 0,$$
$$g_8(X) = \frac{5x_2}{x_1} - 1 \leq 0,$$
$$g_9(X) = \frac{x_1}{12x_2} - 1 \leq 0,$$
$$g_{10}(X) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0,$$
$$g_{11}(X) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0,$$

Variable range:

$$\begin{gathered} 2.6 \leq x_1 \leq 3.6, \\ 0.7 \leq x_2 \leq 0.8, \\ x_3 \in \{17, 18, 19, \ldots, 28\}, \\ 7.3 \leq x_4, x_5 \leq 8.3, \\ 2.9 \leq x_6 \leq 3.9, \\ 5 \leq x_7 \leq 5.5. \end{gathered} \tag{19}$$

where x_1 (b in Figure 11) represents the width of the surface, x_2 (m in Figure 11) represents the tooth mold, x_3 (z in Figure 11) represents the number of teeth in the pinion, x_4 (l_1 in Figure 11) represents the length of the first shaft between bearings, x_5 (l_2 in Figure 11)

represents the length of the second shaft between bearings, x_6 (d_1 in Figure 11) represents the diameter of the second axis, and x_7 (d_2 in Figure 11) represents the diameter of the second axis.

The experimental results are shown in Table 12, and the convergence curve is shown in Figure 15a. The results show that the FSSSA algorithm is superior to other algorithms in terms of final accuracy.

Table 12. Speed reducer experimental data.

Algorithm	Best	Worst	Mean	Std
BBO	3.26×10^3	3.73×10^3	3.49×10^3	1.31×10^2
SSA	3.22×10^3	3.86×10^3	3.46×10^3	1.39×10^2
FSSSA	3.09×10^3	3.15×10^3	3.20×10^3	5.79×10^1

5.2. Cantilever Beam (CB)

The CB is a weight optimization problem of a cantilever beam with a square cross-section, which is generally an example of structural engineering design. The structure diagram is shown in Figure 12 [71], and the mathematical expression to describe this kind of problem is shown in Equation (20).

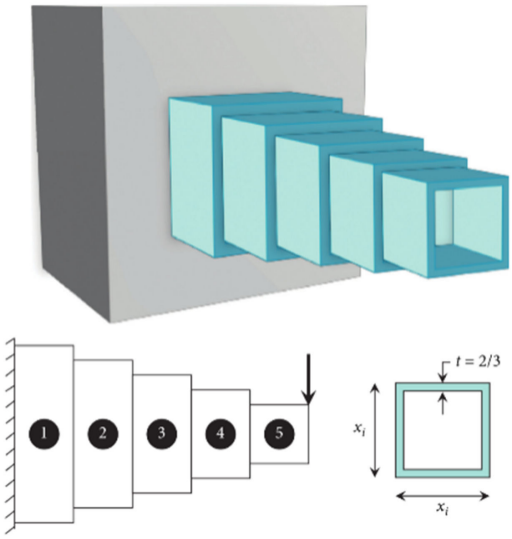

Figure 12. Schematic diagram of cantilever beam [71].

Minimize:

$$f(X) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5),$$

Subject to:

$$g(X) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0,$$

Variable range:

$$0.01 \leq x_i \leq 100, \quad i = 1, \ldots, 5. \tag{20}$$

where x_1 to x_5, respectively represents the width (or height) of five hollow square blocks with constant thickness, which are called decision variables, their thickness t remains unchanged (here $t = 2/3$).

The experimental results are shown in Table 13, and the convergence curve is shown in Figure 15b. The results show that the FSSSA algorithm is superior to other algorithms in terms of final accuracy.

Table 13. Cantilever beam experimental data.

Algorithm	Best	Worst	Mean	Std
BBO	2.00	3.78	2.65	3.82×10^{-1}
SSA	1.56	1.74	1.63	5.38×10^{-2}
FSSSA	1.44	1.56	1.56	3.47×10^{-2}

5.3. Optimal Design of the I-Shaped Beam

The ODIB is a vertical disturbance optimization problem of the I-beam. The primary purpose is to minimize the vertical deflection of the I-beam under the constraints of cross-sectional area and stress under preset loads. The structure diagram is shown in Figure 13 [71], and the mathematical expression to describe this problem is shown in Equation (21).

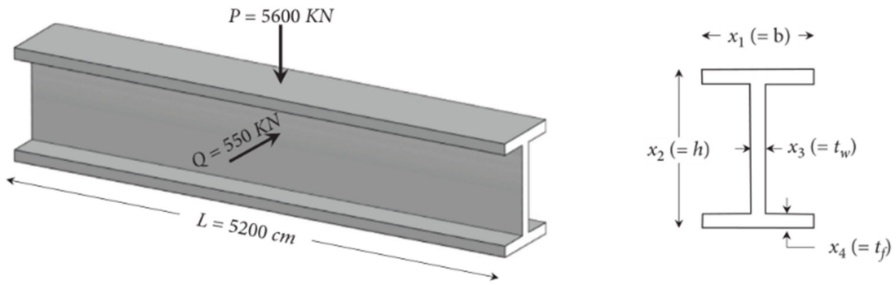

Figure 13. Schematic diagram of optimal design of the I-shaped beam [71].

Minimize:

$$f(X) = \frac{5000}{x_3(x_2 - 2x_4)^3/12 + (x_1 x_4^3/6) + 2bx_4(x_2 - x_4/2)^2},$$

Subject to:

$$g_1(X) = 2x_1x_3 + x_3(x_2 - 2x_4) \leq 300,$$
$$g_2(X) = \frac{18x_2 \times 10^4}{x_3(x_2-2x_4)^3+2x_1x_3\left(4x_4^2+3x_2(x_2-2x_4)\right)} + \frac{15x_1 \times 10^3}{(x_2-2x_4)x_3^2+2x_3x_1^3} \leq 56,$$

Variable range:

$$\begin{aligned} 10 \leq x_1 \leq 50, \\ 10 \leq x_2 \leq 80, \\ 0.9 \leq x_3 \leq 5, \\ 0.9 \leq x_4 \leq 5. \end{aligned} \tag{21}$$

where, x_1 represents flange width (b in Figure 13), x_2 section height (h in Figure 13), x_3 web thickness (t_w in Figure 13), x_4 flange thickness (t_f in Figure 13), $f(x)$ is vertical disturbance of I-beam, L and E are beam length and elastic modulus, which are 5200 and 523.104, respectively.

The experimental results are shown in Table 14, and the convergence curve is shown in Figure 15c. The results show that the FSSSA algorithm outperforms other algorithms regarding final accuracy.

Table 14. Optimal design of the I-shaped beam experimental data.

Algorithm	Best	Worst	Mean	Std
BBO	1.42×10^{-2}	1.57×10^{-1}	3.91×10^{-2}	3.72×10^{-2}
SSA	1.38×10^{-2}	4.21×10^{-1}	1.31×10^{-1}	1.17×10^{-1}
FSSSA	1.31×10^{-2}	1.31×10^{-1}	3.10×10^{-2}	4.46×10^{-3}

5.4. Piston Lever (PL)

The PL is an optimization problem for positioning several piston components, with the primary goal of minimizing fuel consumption by increasing the piston rod from 0° to 45°. The structure diagram is shown in Figure 14 [71], and the mathematical expression to describe this kind of problem is shown in Equation (22).

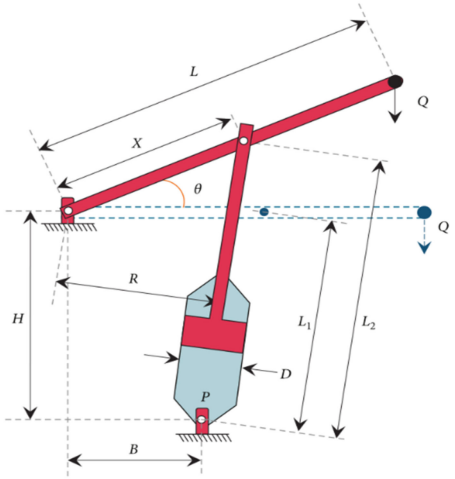

Figure 14. Schematic diagram of piston lever [71].

Minimize:

$$f(X) = \frac{1}{4}\pi x_3^2(L_2 - L_1),$$

Subject to:

$$\begin{aligned} g_1(X) &= QL\cos\theta - R \times F \leq 0, \\ g_2(X) &= Q(L - x_4) - M_{\max} \leq 0, \\ g_3(X) &= 1.2(L_2 - L_1) - L_1 \leq 0, \\ g_4(X) &= \tfrac{x_3}{2} - x_2 \leq 0, \end{aligned}$$

where:

$$\begin{aligned} R &= \tfrac{|-x_4(x_4\sin\theta + x_1) + x_1(x_2 - x_4\cos\theta)|}{\sqrt{(x_4 - x_2)^2 + x_1^2}}, \\ F &= \tfrac{\pi P x_3^2}{4}, \\ L_1 &= \sqrt{(x_4 - x_2)^2 + x_1^2}, \\ L_2 &= \sqrt{(x_4\sin\theta + x_1)^2 + (x_2 - x_4\cos\theta)^2}, \\ \theta &= 45^\circ, \\ Q &= 10{,}000 \text{ lbs}, \\ L &= 240 \text{ in}, \\ M_{\max} &= 1.8 \times 10^6 \text{lbs in}, \\ P &= 1500 \text{ psi}, \end{aligned}$$

Variable range:

$$\begin{aligned} 0.05 \leq x_1, x_2, x_4 \leq 500, \\ 0.05 \leq x_3 \leq 120. \end{aligned} \tag{22}$$

where x_1, x_2, x_3, and x_4, respectively, represent the positioning of four optimized piston components, corresponding to H, B, D, and X, respectively, in Figure 14.

The experimental results are shown in Table 15, and the convergence curve is shown in Figure 15d. The results show that the FSSSA algorithm is superior to other algorithms in terms of final accuracy.

Table 15. Piston lever experimental data.

Algorithm	Best	Worst	Mean	Std
BBO	4.33×10^2	4.85×10^4	7.56×10^3	1.00×10^4
SSA	4.52×10^2	1.22×10^5	2.61×10^4	3.28×10^4
FSSSA	4.90×10^1	2.67×10^3	4.59×10^2	5.72×10^2

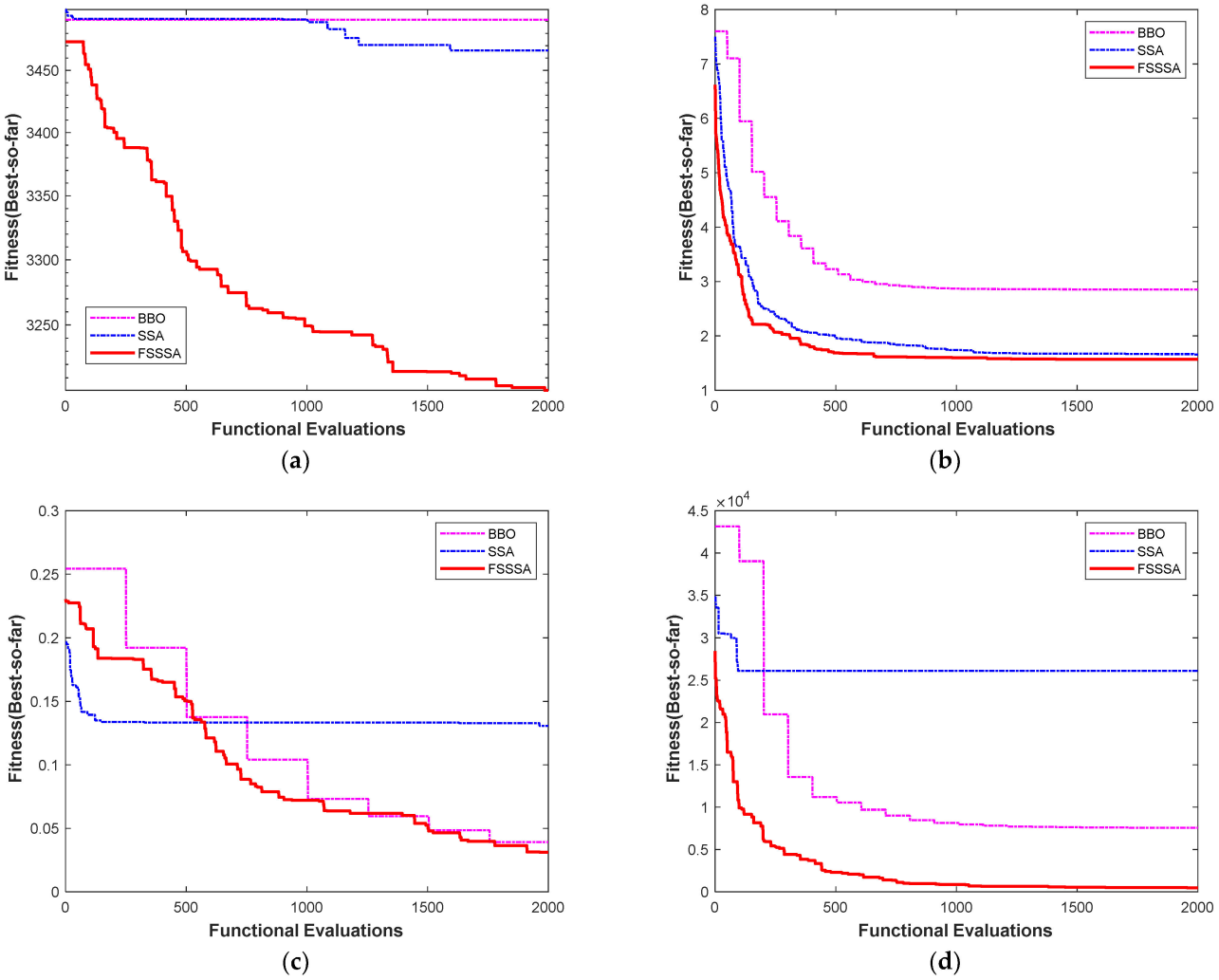

Figure 15. Convergence curve of engineering problems: (**a**) Speed reducer; (**b**) Cantilever beam; (**c**) Optimal design of the I-shaped beam; (**d**) Piston lever.

In summary, FSSSA performs well in the optimization of four engineering problems, except for some stability issues observed in the CB problem. When compared to SSA and BBO, FSSSA has shown the ability to find superior design solutions, significantly improving system performance and efficiency while satisfying the given constraints.

Overall, the performance advantages of FSSSA make it a reliable choice for solving engineering optimization problems. Its overall effectiveness and potential for applications make FSSSA a valuable tool for seeking better design solutions.

6. Conclusions

This paper proposes an improved squirrel search algorithm (FSSSA) to solve the problems of the slow convergence speed and unbalanced search stages of the SSA. Using the fuzzy inference system, sine cosine mutation, and the wide-area search mechanism to enhance the performance of SSA. By comparing the improved SSA with four evaluation index experiments on 24 benchmark functions, the effectiveness of the FSSSA is proved. From the convergence accuracy test and evaluation times tests, FSSSA performs excellently in the convergence speed. According to the balance and diversity analysis, FSSSA can better balance the exploration and exploitation ability and improve the convergence accuracy. By comparing the results in convergence accuracy and non-parametric statistical experiments, it can be analyzed that FSSSA maintains the top rank with other algorithms. In addition, FSSSA is applied to four kinds of engineering problems, and the experimental results show that FSSSA is more competitive in dealing with real complex problems.

This study will be helpful to further research on SSA. To meet the requirement of the complex problems, it can simplify the actual steps of FSSSA, and reduce the running time and computational complexity of the algorithm. In addition, FSSSA can be used to solve the multi-objective and feature selection.

Author Contributions: Methodology, L.C.; software, B.Z.; validation, L.C., B.Z. and Y.M.; formal analysis, B.Z.; investigation, L.C.; resources, L.C. and Y.M.; data curation, B.Z.; writing—original draft preparation, B.Z.; writing—review and editing, B.Z. and L.C.; visualization, B.Z.; supervision, L.C.; project administration, Y.M.; funding acquisition, L.C. and Y.M. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the National Natural Science Foundation of China (No.61535008), the National Natural Science Foundation of China (No.62203332), and the Natural Science Foundation of Tianjin (No.20JCQNJC00430).

Conflicts of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Yuan, G.; Li, T.; Hu, W. A conjugate gradient algorithm for large-scale nonlinear equations and image restoration problems. *Appl. Numer. Math.* **2020**, *147*, 129–141. [CrossRef]
2. Salomon, R. Evolutionary algorithms and gradient search: Similarities and differences. *IEEE Trans. Evol. Comput.* **1998**, *2*, 45–55. [CrossRef]
3. Yuan, G.; Wei, Z.; Yang, Y. The global convergence of the Polak–Ribière–Polyak conjugate gradient algorithm under inexact line search for nonconvex functions. *J. Comput. Appl. Math.* **2019**, *362*, 262–275. [CrossRef]
4. Ghalavand, N.; Khorram, E.; Morovati, V. An adaptive nonmonotone line search for multiobjective optimization problems. *Comput. Oper. Res.* **2021**, *136*, 105506. [CrossRef]
5. Yuan, Y. Recent advances in trust region algorithms. *Math Program.* **2015**, *151*, 249–281. [CrossRef]
6. Powell, M.J.D. On the convergence of a wide range of trust region methods for unconstrained optimization. *IMA J. Numer. Anal.* **2010**, *30*, 289–301. [CrossRef]
7. Stanczak, J. Biologically inspired methods for control of evolutionary algorithms. *Control. Cybern.* **2003**, *32*, 411–433.
8. Akyol, S.; Alatas, B. Plant intelligence based metaheuristic optimization algorithms. *Artif. Intell. Rev.* **2017**, *47*, 417–462. [CrossRef]
9. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the MHS'95 the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
10. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
11. Mohammed, H.; Rashid, T. FOX: A FOX-inspired optimization algorithm. *Appl. Intell.* **2023**, *53*, 1030–1050. [CrossRef]
12. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
13. de Souza, O.A.P.; Miguel, L.F.F. CIOA: Circle-Inspired Optimization Algorithm, an algorithm for engineering optimization. *Softwarex* **2022**, *19*, 101192. [CrossRef]
14. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
15. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
16. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
17. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2010**, *15*, 4–31. [CrossRef]

18. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching-learning-based Optimization: A novel method for constrained mechanical design optimization problems. *Comput. Aided Des.* **2011**, *43*, 303–315. [CrossRef]
19. Wu, T.; Wu, X.; Chen, J.; Chen, X.; Homayoon Ashrafzadeh, A. A novel metaheuristic algorithm: The team competition and cooperation optimization algorithm. *Comput. Mater. Contin.* **2022**, *73*, 2879–2896. [CrossRef]
20. Mehrabian, A.R.; Lucas, C. A novel numerical optimization algorithm inspired from weed colonization. *Ecol. Inform.* **2006**, *1*, 355–366. [CrossRef]
21. Alatas, B. Photosynthetic algorithm approaches for bioinformatics. *Expert Syst. Appl.* **2011**, *38*, 10541–10546. [CrossRef]
22. Ashrafi, S.M.; Dariane, A.B. Performance evaluation of an improved harmony search algorithm for numerical optimization: Melody search (MS). *Eng. Appl. Artif. Intell.* **2013**, *26*, 1301–1321. [CrossRef]
23. Lee, K.S.; Geem, W.G. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 3902–3933. [CrossRef]
24. Dokeroglu, T.; Sevinc, E.; Kucukyilmaz, T.; Cosar, A. A survey on new generation metaheuristic algorithms. *Comput. Ind. Eng.* **2019**, *137*, 106040. [CrossRef]
25. Shaukat, N.; Ahmad, A.; Mohsin, B.; Khan, R.; Khan, S.U.D.; Khan, S.U.D. Multiobjective Core Reloading Pattern Optimization of PARR-1 Using Modified Genetic Algorithm Coupled with Monte Carlo Methods. *Sci. Technol. Nucl. Install.* **2021**, *2021*, 1802492. [CrossRef]
26. Lodewijks, G.; Cao, Y.; Zhao, N.; Zhang, H. Reducing CO_2 Emissions of an Airport Baggage Handling Transport System Using a Particle Swarm Optimization Algorithm. *IEEE Access* **2021**, *9*, 121894–121905. [CrossRef]
27. Romeh, A.E.; Mirjalili, S.; Gul, F. Hybrid Vulture-Coordinated Multi-Robot Exploration: A Novel Algorithm for Optimization of Multi-Robot Exploration. *Mathematics* **2023**, *11*, 2474. [CrossRef]
28. Jain, M.; Singh, V.; Rani, A. A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm Evol. Comput.* **2019**, *44*, 148–175. [CrossRef]
29. Sanaj, M.S.; Joe Prathap, P.M. Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere. *Eng. Sci. Technol. Int. J.* **2020**, *23*, 891–902. [CrossRef]
30. Basu, M. Squirrel search algorithm for multi-region combined heat and power economic dispatch incorporating renewable energy sources. *Energy* **2019**, *182*, 296–305. [CrossRef]
31. Zade, B.M.H.; Mansouri, N.; Javidi, M.M. SAEA: A security-aware and energy-aware task scheduling strategy by Parallel Squirrel Search Algorithm in cloud environment. *Expert Syst. Appl.* **2021**, *176*, 114915. [CrossRef]
32. Ghosh, M.; Sen, S.; Sarkar, R.; Maulik, U. Quantum squirrel inspired algorithm for gene selection in methylation and expression data of prostate cancer. *Appl. Soft Comput.* **2021**, *105*, 107221. [CrossRef]
33. Zhang, Y.; Wei, C.; Zhao, J.; Qiang, Y.; Wu, W.; Hao, Z. Adaptive mutation quantum-inspired squirrel search algorithm for global optimization problems. *Alex. Eng. J.* **2022**, *61*, 7441–7476. [CrossRef]
34. Shi, M.; Wang, C.; Li, X.; Li, M.; Wang, L.; Xie, N. EEG signal classification based on SVM with improved squirrel search algorithm. *Biomed. Tech.* **2020**, *66*, 137–152. [CrossRef] [PubMed]
35. Cenitta, D.; Arjunan, R.V.; Prema, K.V. Ischemic Heart Disease Prediction Using Optimized Squirrel Search Feature Selection Algorithm. *IEEE Access* **2022**, *10*, 122995–123006. [CrossRef]
36. Zheng, T.; Luo, W. An Improved Squirrel Search Algorithm for Optimization. *Complexity* **2019**, *2019*, 6291968. [CrossRef]
37. Wen, Q.; Huo, L. A dimensional learning squirrel search algorithm based on roulette strategy. In Proceedings of the 2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML), Hangzhou, China, 25–27 March 2022; pp. 32–38.
38. Wang, Y.; Du, T. An improved squirrel search algorithm for global function optimization. *Algorithms* **2019**, *12*, 80. [CrossRef]
39. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
40. Sakthivel, V.P.; Suman, M.; Sathya, P.D. Combined economic and emission power dispatch problems through multi-objective squirrel search algorithm. *Appl. Soft Comput.* **2021**, *100*, 106950. [CrossRef]
41. Liu, Z.; Zhang, F.; Wang, X.; Zhao, Q.; Zhang, C.; Liu, T.; Zhang, B. A discrete squirrel search optimization based algorithm for Bi-objective TSP. *Wirel. Netw.* **2021**. [CrossRef]
42. Ravber, M.; Liu, S.H.; Mernik, M.; Črepinšek, M. Maximum number of generations as a stopping criterion considered harmful. *Appl. Soft Comput.* **2022**, *128*, 109478. [CrossRef]
43. Jang, J.S.R. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Trans. Syst. Man Cybern.* **1993**, *23*, 665–685. [CrossRef]
44. Kumar, A.N.; Sanjay, C.; Chakravarthy, M. Fuzzy inference system-based solution to locate the cross-country faults in parallel transmission line. *Int. J. Electr. Eng. Educ.* **2021**, *58*, 83–96. [CrossRef]
45. Yu, H.; Lu, J.; Zhang, G. Topology Learning-Based Fuzzy Random Neural Networks for Streaming Data Regression. *IEEE Trans. Fuzzy Syst.* **2022**, *30*, 412–425. [CrossRef]
46. Liu, D.; Xiao, Z.; Li, H.; Liu, D.; Hu, X.; Malik, O.P. Accurate Parameter Estimation of a Hydro-Turbine Regulation System Using Adaptive Fuzzy Particle Swarm Optimization. *Energies* **2019**, *12*, 3903. [CrossRef]
47. Amador-Angulo, L.; Castillo, O. A new fuzzy bee colony optimization with dynamic adaptation of parameters using interval type-2 fuzzy logic for tuning fuzzy controllers. *Soft Comput.* **2018**, *22*, 571–594. [CrossRef]
48. Li, G.; Liu, M. The summary of differential evolution algorithm and its improvements. In Proceedings of the 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Chengdu, China, 20–22 August 2010; pp. V3-153–V3-156.

49. Xue, J.; Shen, B. A novel swarm intelligence optimization approach: Sparrow search algorithm. *Syst. Sci. Control. Eng.* **2020**, *8*, 22–34. [CrossRef]
50. Braik, M.; Hammouri, A.; Atwan, J.; Al-Betar, M.A.; Awadallah, M.A. White Shark Optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. *Knowl. Based Syst.* **2022**, *243*, 108457. [CrossRef]
51. Ahmadianfar, I.; Heidari, A.A.; Gandomi, A.H.; Chu, X.; Chen, H. RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method. *Expert Syst. Appl.* **2021**, *181*, 115079. [CrossRef]
52. Ahmadianfar, I.; Heidari, A.A.; Noshadian, S.; Chen, H.; Gandomi, A.H. INFO: An efficient optimization algorithm based on weighted mean of vectors. *Expert Syst. Appl.* **2022**, *195*, 116516. [CrossRef]
53. Zhang, Y.; Chi, A. Group teaching optimization algorithm with information sharing for numerical optimization and engineering optimization. *J. Intell. Manuf.* **2021**, *34*, 1547–1571. [CrossRef]
54. Dhiman, G.; Kumar, V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. *Knowl. Based Syst.* **2019**, *165*, 169–196. [CrossRef]
55. Noor, H.A.; Mostafa, Z.A.; Ponnuthurai, N.S. Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation, Donostia-San Sebastián, Spain, 7 July 2017; pp. 372–379.
56. Yuan, B.; Gallagher, M. Experimental results for the special session on real-parameter optimization at CEC 2005: A simple, continuous EDA. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; pp. 1792–1799.
57. Song, X.; Zhao, M.; Yan, Q.; Xing, S. A high-efficiency adaptive artificial bee colony algorithm using two strategies for continuous optimization. *Swarm Evol. Comput.* **2019**, *50*, 100549. [CrossRef]
58. Li, X.; Wu, H.; Yang, Q.; Tan, S.; Xue, P.; Yang, X. A multistrategy hybrid adaptive whale optimization algorithm. *J. Comput. Des. Eng.* **2022**, *9*, 1952–1973. [CrossRef]
59. Ortiz-Boyer, D.; Hervas-Martinez, C.; Garcia-Pedrajas, N. CIXL2: A crossover operator for evolutionary algorithms based on population features. *J. Artif. Intell. Res.* **2005**, *24*, 1–48. [CrossRef]
60. Lem, S.; Onghena, P.; Verschaffel, L.; Dooren, W.V. The heuristic interpretation of box plots. *Learn. Instr.* **2013**, *26*, 22–35. [CrossRef]
61. Veček, N.; Mernik, M.; Filipič, B.; Črepinšek, M. Parameter tuning with Chess Rating System (CRS-Tuning) for meta-heuristic algorithms. *Inf. Sci.* **2016**, *372*, 446–469. [CrossRef]
62. Klazar, R.; Engelbrecht, A.P. Parameter optimization by means of statistical quality guides in F-Race. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 2547–2552.
63. Smit, S.K.; Eiben, A.E. Beating the 'world champion' evolutionary algorithm via REVAC tuning. In Proceedings of the IEEE Congress on Evolutionary Computation, Barcelona, Spain, 27 September 2010; pp. 1–8.
64. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
65. Lesack, K.; Naugler, C. An open-source software program for performing Bonferroni and related corrections for multiple comparisons. *J. Pathol. Inform.* **2011**, *2*, 52. [CrossRef]
66. Alba, E.; Dorronsoro, B. The exploration/exploitation trade-off in dynamic cellular genetic algorithms. *IEEE Trans. Evol. Comput.* **2005**, *9*, 126–142. [CrossRef]
67. Lynn, N.; Suganthan, P.N. Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. *Swarm Evol. Comput.* **2015**, *24*, 11–24. [CrossRef]
68. Hussain, K.; Salleh, M.N.M.; Cheng, S.; Shi, Y. On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Comput. Appl.* **2019**, *31*, 7665–7683. [CrossRef]
69. Jerebic, J.; Mernik, M.; Liu, S.H.; Ravber, M.; Baketaric, M.; Mernik, L.; Crepinsek, M. A novel direct measure of exploration and exploitation based on attraction basins. *Expert Syst. Appl.* **2021**, *167*, 114353. [CrossRef]
70. Demšar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn Res.* **2006**, *7*, 1–30.
71. Bayzidi, H.; Talatahari, S.; Saraee, M.; Lamarche, C.P. Social Network Search for Solving Engineering Optimization Problems. *Comput. Intell. Neurosci.* **2021**, *2021*, 8548639. [CrossRef]
72. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
73. Sattar, D.; Salim, R. A smart metaheuristic algorithm for solving engineering problems. *Eng. Comput.* **2020**, *37*, 2389–2417. [CrossRef]