symmetry

MDPI

Article

A Tent Lévy Flying Sparrow Search Algorithm for Wrapper-Based Feature Selection: A COVID-19 Case Study

Qinwen Yang [1], Yuelin Gao [2,*] and Yanjie Song [3]

[1] School of Computer Science and Engineering, North Minzu University, Yinchuan 750021, China
[2] Ningxia Key Laboratory of Intelligent Information and Big Data Processing, Yinchuan 750021, China
[3] College of Systems Engineering, National University of Defense Technology, Changsha 410073, China
* Correspondence: gaoyuelin@263.net

Abstract: The "Curse of Dimensionality" induced by the rapid development of information science might have a negative impact when dealing with big datasets, and it also makes the problems of symmetry and asymmetry increasingly prominent. Feature selection (FS) can eliminate irrelevant information in big data and improve accuracy. As a recently proposed algorithm, the Sparrow Search Algorithm (SSA) shows its advantages in the FS tasks because of its superior performance. However, SSA is more subject to the population's poor diversity and falls into a local optimum. Regarding this issue, we propose a variant of the SSA called the Tent Lévy Flying Sparrow Search Algorithm (TFSSA) to select the best subset of features in the wrapper-based method for classification purposes. After the performance results are evaluated on the CEC2020 test suite, TFSSA is used to select the best feature combination to maximize classification accuracy and simultaneously minimize the number of selected features. To evaluate the proposed TFSSA, we have conducted experiments on twenty-one datasets from the UCI repository to compare with nine algorithms in the literature. Nine metrics are used to evaluate and compare these algorithms' performance properly. Furthermore, the method is also used on the coronavirus disease (COVID-19) dataset, and its classification accuracy and the average number of feature selections are 93.47% and 2.1, respectively, reaching the best. The experimental results and comparison in all datasets demonstrate the effectiveness of our new algorithm, TFSSA, compared with other wrapper-based algorithms.

Keywords: sparrow search algorithm; feature selection; COVID-19

Citation: Yang, Q.; Gao, Y.; Song, Y. A Tent Lévy Flying Sparrow Search Algorithm for Wrapper-Based Feature Selection: A COVID-19 Case Study. *Symmetry* **2023**, *15*, 316. https://doi.org/10.3390/sym15020316

Academic Editors: Wenyin Gong and Libao Deng

Received: 28 December 2022
Revised: 17 January 2023
Accepted: 18 January 2023
Published: 22 January 2023

1. Introduction

An iterative series of task sequences, data selection and pretreatment, mining algorithm selection, data mining, pattern evaluation, and knowledge presentation make up knowledge discovery in databases (KDD) [1–3]. The main objective of data preprocessing, as the initial stage in KDD, is to prepare datasets for use by data mining algorithms [4]. However, as information science progresses, the dimensionality of datasets increases dramatically, affecting the performance of clustering and classification approaches [5–7]. High-dimensional datasets also have data redundancy, performance deterioration, and a more extended period to build models [8–10]. These limitations have given rise to more difficulties in data analysis. Feature selection (FS) is frequently used as a preprocessing approach in the data mining process to determine the best subset of features from all available feature sets [11–13]. It eliminates irrelevant and redundant features, simplifies clustering and classification, enhances accuracy, and the problem of symmetry and asymmetry is also solved to a certain extent [14–16].

While some feature selection methods can solve the problem exactly for linear models only with promising results, exact methods can only handle hundreds or thousands of features at best. Another shortcoming of most feature selection methods is that they arbitrarily seek to identify only one solution to the problem. However, in practice, there

are often multiple predictive or even information-equivalent solutions. Especially in fields where there is inherent redundancy in the underlying problem, as in molecular biology, there are often multiple solutions [17,18].

There are many ways to solve the FS problem, which can generally be divided into the following three categories: filters, wrappers, and embedded methods [19,20]. For the filtering method, the features in a given feature set are first sorted according to a series of criteria. Then the features with a higher ranking are formed into feature subsets [21]. Although the obtained feature subset is not necessarily the optimal subset, its calculation efficiency is very high, so this method is often used for high-dimensional FS problems. Representative filtering methods include minimum redundant F-score criteria [22], maximum correlation (mRMR) [23], Gini index [24], and correlation coefficient [25]. Wrapper approaches often use a predetermined learning process that is evaluated using a subset of features [26]. In most circumstances, wrapper methods outperform filter approaches that are not dependent on any learning mechanism. The embedded methods are embedded in the learning algorithm, and a subset of features can be obtained when the training process of the classification algorithm is completed [27]. The embedded method can solve the problem of excessive redundancy in the results of the filter methods based on feature sorting. It can also solve the problem of excessive time complexity of the wrapper methods, which is a compromise between the filter and wrapper [28–30].

Various methods for discovering optimal feature subsets have arisen in the wake of the wrapper-based method, including heuristic search, complete search, greedy search, and random search, to mention a few [31–34]. However, most of these methods suffer from local optima and expensive computational costs due to the use of greedy search methods [35,36]. Over the past three decades, Evolutionary Algorithms (EA) have been very reliable in solving various optimization problems, such as image processing [37], intrusion detection [38], path planning [39], particle filtering [40], production scheduling [22], support vector machines [41], home healthcare [42], wireless sensors [43], and neural network models [44].

Due to its capabilities in seeking competitive solutions employing tactics that perform well in exploration, EA has recently gained much attention in tackling FS challenges [45–47]. These approaches include Genetic Algorithm (GA) [48], Particle Swarm Optimization (PSO) [49], and White Shark Optimizer (WSO) [50]. A comprehensive review of nature-inspired FS techniques can be found in [22], and a detailed analysis of EA to FS can be found in [51]. Here are some examples.

Based on GA, the K-Nearest-Neighbors (K-NN) approach for diagnosing patient diseases was proposed in [52]. It used a hybrid genetic algorithm to perform efficient feature selection. The K-NN algorithm is utilized to diagnose lung cancer after an experimental technique was employed to determine the ideal value of K. In [53], by minimizing the numerous objectives of the FS, a non-dominated sorting genetic algorithm (NSGA) is employed to solve the multi-objective optimization problem (MOP). Recently, Xue et al. [54] proposed a multi-objective binary genetic algorithm called MOBGA-AOS with five crossover operators.

Based on PSO, Song et al. [55] proposes a K-NN and mutual information-based bare-bones PSO (BBPSO) feature selection algorithm. The adaptive flip mutation operator intends to assist particles in breaking out from the optimal local solution. In [56], a high-dimensional FS problem is solved using a multi-stage hybrid FS algorithm (HFS-C-P) based on PSO. Recently, Li et al. [57] proposed an improved Sticky Binary PSO (ISBPSO) algorithm for FS.

Grey Wolf Optimizer (GWO) and Sparrow Search Algorithm (SSA) are other EAs that also investigate solving the FS problem. Jangir et al. [58] proposed that non-dominated sorting GWO (NSGWO) is used to perform FS to improve the categorization of cervical lesions by reducing the number of textural features while increasing the classification accuracy. Recently, Sathiyabhama et al. [59] proposed an FS framework based on GWO and a rough set method called GWORS for finding salient features from extracted mammograms.

Based on SSA, Chen et al. [60] proposed a spark-based improved SSA (SPISSA) used to search feature subsets on intrusion detection datasets.

In addition, applying FS technology based on EA in detecting COVID-19 patients is also becoming more extensive. da Silva et al. [61] combined single models such as Stereo Regression, Quantile Random Forest, K-NN, Bayesian Regression Neural Network, and Support Vector Regression with Variational Mode Decomposition (VMD) to create a hybrid model to forecast COVID-19 cases in Brazil. The VMD-based model proved to be one of the most effective strategies for forecasting COVID-19 cases five days in advance. Dey et al. [62] presented a hybrid model. To begin, they extracted several characteristics from the COVID-19-affected lungs. The Manta-Ray Foraging-based Golden Ratio Optimizer (MRFGRO), a hybrid meta-heuristic FS technique, is then presented to pick the most critical subset of characteristics. Although the findings show that the proposed strategy is quite effective, the model was only tested on the CT scan dataset. Shaban et al. [63] proposes Distance-Biased Naive Bayes (DBNB), a new approach for detecting COVID-19-infected patients. DBNB picks the most informative characteristics for identifying COVID-19 patients through a novel FS technique called Advanced PSO (APSO). APSO is a hybrid strategy that uses filter and wrapper approaches to offer accurate but crucial classification features.

In conclusion, many FS approaches employ EA to avoid increasing computing complexity in the high-dimensional dataset. These algorithms use primitive mechanisms and operations to solve an optimization problem and iteratively search for the optimal solution. Nonetheless, the No Free Lunch (NFL) [64] theorem states that existing procedures can constantly be improved. Moreover, there is currently insufficient research in the literature to solve the FS problem using the SSA, motivating us to suggest a variant of SSA for FS in Section 3.

The SSA is a new and well-organized EA that can be used in different areas for solving optimization problems, such as brain tumor detection [65], parameter identification [66], configuration network [67], and fault diagnosis of wind turbines [68]. However, SSA still has the problem of being easy to fall into the local optimum, and so far, the application of the SSA for FS is very scarce [69]. Motivated by the above analysis, we propose a Tent Lévy Flying Sparrow Search Algorithm (TFSSA) in this paper to increase the capability of SSA in confronting FS challenges, where the main contributions are summarized below.

- A TFSSA is proposed for feature selection problems, and it is utilized to solve a COVID-19 case study.
- An improved Tent chaos strategy, Lévy flights (LFs) mechanism, and Self-adaptive hyper-parameters are integrated into TFSSA to improve SSA's exploratory behavior and perform well in the CEC2020 benchmark function.
- A comprehensive comparison of TFSSA and nine different algorithms for feature selection problems is undertaken in nine aspects.
- The proposed TFSSA's improved searching capabilities are tested on 21 well-known feature selection datasets with excellent results.

The remainder of this paper is organized as follows: Section 2 presents the background of the SSA. The proposed TFSSA is described in Section 3. Section 4 presents the proposed TFSSA algorithms for FS, while the experimental results with discussions are reported in Section 5. Section 6 demonstrates the adoption of the proposed TFSSA in a COVID-19 application. Finally, we conclude this paper in Section 7.

2. SSA

2.1. Background of SSA

The literary representation of various animal, insect, and bacterial populations in nature provides a fascinating field of study for diverse scientific researchers. By simulating the foraging and reproduction behaviors of animal, insect, or bacterial communities, researchers draw inspiration from it to various abstract swarm intelligence and evolutionary behaviors into quantifiable vital indicators, which, in turn, form mathematical models that can be used to address various realities questions. The introduction of many meta-heuristic

algorithms has greatly enriched optimization techniques and provided new tools for exploring the concepts and mechanisms of the biological world from another perspective. Based on the above, In 2020, Xue Jiankai proposed the SSA to enhance optimization technology and decode the complexity involved in the process [70].

2.2. Advantages of SSA from Other EA

SSA differs from other EA with several advantages. First, SSA does not update rules according to simulated social creatures' step size but sets up rules according to its algorithm mechanism. It can handle various optimization issues with only four proprietary parameters to change. Second, SSA's mathematical model makes it suited for resolving a range of engineering optimization issues, particularly those that involve high dimensions. Third, SSA's resilience and simplicity allow it to identify global solutions to complicated optimization problems with high convergence rates. Fourth, SSA has gradually become a strong competitor with a broad interest in developing low-cost and robust solutions to actual optimization issues.

2.3. Rule Design

The classical SSA is primarily motivated by a sparrow population's foraging behavior. It is a search algorithm with high optimization and efficiency capabilities [70–73]. For simplicity, the biology of sparrow populations during foraging is idealized and normalized as the following behaviors.

(1) Producers (leaders) have access to plentiful food sources and are responsible for ensuring that all scroungers (followers) have access to foraging sites.
(2) Some sparrows will be chosen as patrollers (guards). When patrollers come across a predator, they will sound an alarm. When the safety threshold is exceeded, the producer must direct the scroungers (followers) to other safe regions.
(3) Sparrows that can discover a better food source earn more energy and are promoted to producers (leaders). At the same time, hungry scroungers (followers) are more likely to fly elsewhere to forage to gain more energy, and the producer-to-forager ratio remains steady.
(4) Scroungers (followers) hunt for food after the finest producers (leaders). Simultaneously, certain predators may observe producers (leaders) and steal food.
(5) When threatened, sparrows near the flock's edge moved swiftly to a safe region, while sparrows in the center of the flock moved randomly to approach other sparrows in the safe area.

2.4. Algorithm Design

The algorithm design of SSA is summarized in the following main steps.

Step 1: Parameter initialization, which includes the number of sparrows (N), the number of producers (PN), the number of patrollers ($N - PN$), the number of guards (GN), GN is a subset of N, the safety threshold (ST), the warning value (R_2), and the maximum iterations (T_max). The following matrix can be used to depict the initial position of sparrows:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1j} & \cdots & x_{1D} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2j} & \cdots & x_{2D} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3j} & \cdots & x_{3D} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & x_{i3} & \cdots & x_{ij} & \cdots & x_{iD} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & x_{N3} & \cdots & x_{Nj} & \cdots & x_{ND} \end{bmatrix}, \quad (1)$$

where X is the initial location of the sparrow population, N is the number of sparrow populations, D is the variable dimension of the problem to be optimized, $i = 1, 2, \ldots, N$, and $j = 1, 2, \ldots, D$. The fitness value (F_X) of the sparrow population is represented by vectors as follows:

$$F_X = \begin{bmatrix} f([x_{11} & x_{12} & x_{13} & \ldots & x_{1j} & \ldots & x_{1D}]) \\ f([x_{21} & x_{22} & x_{23} & \ldots & x_{2j} & \ldots & x_{2D}]) \\ f([x_{31} & x_{32} & x_{33} & \ldots & x_{3j} & \ldots & x_{3D}]) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ f([x_{i1} & x_{i2} & x_{i3} & \ldots & x_{ij} & \ldots & x_{iD}]) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ f([x_{N1} & x_{N2} & x_{N3} & \ldots & x_{Nj} & \ldots & x_{ND}]) \end{bmatrix}, \quad (2)$$

where the value of f represents the fitness value of the sparrow individual, $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, D$.

Step 2: According to the design of (1) and (2) in Section 2.3, the producer usually has a better fitness value, and it has a higher priority to capture food than other individuals in the search process. Producers are responsible for finding food for the entire population and providing foraging directions for others. As a result, producers have access to a broader search space than scavengers. The producer position of the sparrow population is updated in each iteration as follows:

$$X_i^{t+1} = \begin{cases} X_i^t \cdot \exp(-\frac{i}{\lambda \cdot T\_max}), & \text{if } R_2 < ST \\ X_i^t + L \cdot Q, & \text{if } R_2 \geq ST \end{cases}, \quad (3)$$

where $i = 1, 2, \ldots, PN$; T_max is the maximum number of iterations; t indicates the current iteration; X_i^t represents the value of the ith sparrow at iteration t; $ST \in [0.5, 1]$ represents the safety threshold, and $R_2 \in [0, 1]$ represents the warning value; $\lambda \in (0, 1]$ is a random number; L shows a vector of $1 \cdot D$; Q is a random number; $Q \sim N(0, 1)$. When $R_2 < ST$, it means that there are no predators around, and it is safe. The producer turns on the wide-area search mode, and the entire population is in a safe foraging state. $R_2 \geq ST$, which means the patrollers find the danger and raise the alarm to warn the companions that there are predators around, and all the sparrows fly to other safe areas to avoid the danger.

Step 3: According to rules (3) and (4) in Section 2.3, some followers keep a closer eye on leaders (producers). When the followers spot the producers who have located the food, they will promptly leave their current place to collect the food. The scrounger's position update formula is as follows:

$$X_i^{t+1} = \begin{cases} Q \cdot \exp\left(\frac{X_{\text{worst}} - X_i^t}{i^2}\right), & \text{if } i > N/2 \\ X_P^{t+1} + \left|X_i - X_P^{t+1}\right| \cdot L \cdot (A^T)^2 \cdot A, & \text{otherwise} \end{cases}, \quad (4)$$

where $i = PN + 1, PN + 2, \ldots, N$, A represents a vector of $1 \cdot D$, where each element is randomly assigned ± 1. X_{worst} represents the current global worst position. X_P is the current optimal position of the producer. When $i > N/2$, the ith scavenger with a lower fitness value did not get food and was in a state of starvation and had to fly to other places to find food at this time.

Step 4: We hypothesized that the patrollers made up 10–20% of the population in our simulations, where GN is the number of guards (patrollers). These sparrows' initial placements are generated at random. The mathematical model is expressed as follows according to rule (5) in Section 2.3:

$$X_i^{t+1} = \begin{cases} X_{best}^t + \beta \cdot \left| X_i^t - X_{best}^t \right|, & \text{if } f_i > f_g \\ X_i^t + K \cdot \left(\frac{|X_i^t - X_{worst}^t|}{(f_i - f_w) + \varepsilon} \right), & \text{if } f_i = f_g \end{cases}, \quad (5)$$

where X_{best} represents the current global optimal position; β is a random number that obeys the standard Gaussian distribution; ε is the smallest constant to avoid division by zero errors; $K \in [-1, 1]$ is a random number; the current fitness value of the sparrow is f_i; and f_g and f_w represent the current global best fitness value and worst fitness value, respectively.

For simplicity, $f_i > f_g$ represents the sparrow at the edge of the group. X_{best} represents the center of the population location around which it is safe. $f_i = f_g$ indicates that sparrows in the middle of the population need to approach other sparrows because they know the danger. K is the step size control coefficient, which indicates the direction in which the sparrow moves.

Step 5: Calculate, compare, and update the current position of the sparrow population, and sort and update the fitness values.

Step 6: Repeat Steps 2 through 5 until the maximum number of iterations (T_max) has been reached, at which point the best position (X_{best}) and best solution (f_{best}) will be output.

Algorithm 1 demonstrates the algorithmic structure of the classic SSA.

Algorithm 1 SSA

Input:
The number of sparrows(N)
The number of producers(PN)
The number of guards(GN)
The warning value(R_2)
The maximum iterations(T_max)
Output:
The best position(X_{best})
The best solution(f_g)

$t \leftarrow 0$;
while ($t < T\_max$) **do**
  Calculate and update the F_X, f_g, f_w and R_2;
  for each leaders $i \in [1, PN]$ **do**
    The location of leaders(producers) is updated using Equation (3);
  end for
  for each followers $i \in [PN + 1, N]$ **do**
    The location of followers(scroungers) is updated using Equation (4);
  end for
  for each patrollers $i \in [1, GN]$ **do**
    The location of patrollers is updated using Equation (5);
  end for
  Find the current new location X_i^{t+1}; // If the new location is better than before, update it.
  Rank the F_X;
  $t \leftarrow t + 1$;
end while
return X_{best}, f_g.

3. The Proposed Algorithm

This section delineates the TFSSA. As mentioned in Section 2, although the SSA has the advantages of faster convergence and more vital optimization-seeking abilities, the original SSA, like other traditional EAs, is more subject to the population's poor diversity and falls into a local optimum. The placements of the sparrows in the solution space are randomly distributed, and a random walk method [66] is used when no nearby sparrows surround the current individual. This mode delays the convergence trend and reduces the

convergence accuracy for a limited number of iterations. Our proposed algorithm aspires to improve SSA's complete optimization performance and address these shortcomings.

3.1. Initialized Population

Initialization is a severe step in the meta-heuristic algorithm and affects convergence speed and solution accuracy. The primary motivation of the most advanced initialization methods is to cover the search space as evenly as possible based on generating a small initial population. However, these methods are affected by the dimension disaster, high computational cost, and sensitivity to parameters, which ultimately reduce the convergence speed of the algorithm [73,74].

The efficiency of EA is greatly affected by chaotic mapping, which has the advantages of uniform ergodicity, sensitivity to initial values, and fast search speed. Using the randomicity, ergodicity, and regularity of chaotic variables to solve optimization problems can make the algorithm jump out of local optimization, maintain population diversity, and improve the global search ability to a certain extent. However, different chaotic maps significantly impact the chaotic optimization process. Various scholars have introduced chaos mapping and chaos search into EA, trying to improve the problem of falling into local optimums in the latter period and improve the convergence speed and accuracy of the algorithm. The chaotic map used more in the literature is the Logistic map. Still, its value probability is high in the two ranges [0, 0.1] and [0.9, 1], and the optimization speed is affected by the uneven Logistic traversal, so the algorithm's efficiency will be significantly reduced. Many papers have pointed out that the Tent map has better ergodic uniformity and faster convergence speed than the logistic map and have further proved that the Tent map can be used as a chaotic sequence to generate optimization algorithms through strict mathematical reasoning. The Tent mapping expression is shown in Equation (6).

$$x_{i+1} = \begin{cases} \frac{x_i}{a}, & 0 \le x \le 1/2 \\ \frac{1-x_i}{1-a}, & 1/2 < x \le 1 \end{cases}. \tag{6}$$

Equation (6) after Bernoulli shift transformation is as follows:

$$x_{i+1} = (2x_i) \bmod 1, \tag{7}$$

where *mod* is the modulo function. Tent mapping has the advantages of randomness, consistency, and orderliness, and it is a standard method for scholars to find the optimal solution [72,75]. On the other hand, a chaotic Tent map has flaws, such as a short period and unstable period points [76]. Therefore, the Tent chaos map is enhanced by the ψ, as given in Equation (8), to avoid slipping into a tiny period or an unstable periodic point [74].

$$x_{i+1} = \begin{cases} \frac{x_i}{a} + \psi & 0 \le x \le a \\ \frac{1-x_i}{1-a} + \psi & a < x \le 1 \end{cases}, \tag{8}$$

where $a = 0.7$ in the current experiments, $\psi = rand(0,1) \times 1/N$, and N represents the population of sparrows. Equation (8) after Bernoulli shift transformation is as follows:

$$x_{i+1} = (2x_i) \bmod 1 + \psi, \tag{9}$$

where *mod* is the modulo function, $\psi = rand(0,1) \times 1/N$, and N represents the population of sparrows.

Therefore, in TFSSA, Equations (1) and (9) in the traditional SSA are replaced by Equation (8) to increase the sparrow population diversity. At this time, the improved Tent chaotic sequence is introduced based on the original SSA to initialize the sparrow population. Although the algorithm not only retains the randomness of the initial individuals but also improves the population diversity at the initial stage, it still cannot guarantee that the population diversity will still have the same degree later. However, in subsequent experiments, it was found that the population's diversity is not well guaranteed, and the

scavengers constantly hop around the producers, making the algorithm fall into local optimization to a large extent in the algorithm's later stages. In this case, we consider introducing the LF mechanism to improve the algorithm's performance further.

3.2. LF Mechanism

"LFs" are named after the French mathematician Paul·Lévy (1886–1971), who first proposed the concept in 1937. LFs try to strengthen the optimization process with diversity and universality, which helps the algorithm find the search location effectively and avoid local minima. Therefore, LFs embed in the SSA mechanism to improve the overall optimization efficiency. The foraging activities of most animals are also characteristic of LFs, for example, the routes of plankton, termites, bumblebees, birds, and primates. LFs seem to be a common law for creatures to survive in a resource-scarce environment when foraging has similar patterns. The trajectory of human beings when traveling and shopping also belongs to LFs.

It can be seen from the SSA rule design (3) that when the producer's food does not have enough temptation, hungry scavengers may fly to other places to look for food. However, according to the SSA rule design (4), scavengers mainly search for food from the producer and go elsewhere to look for food. Generally, they only search for food within a relatively close range of the producer. Therefore, most sparrows may only move around areas with poor solution quality. On the other hand, for each iteration, the individual sparrow will move indiscriminately to the sparrow (producer) whose food is better than his own. This situation increases the algorithm's complexity and leads to low convergence accuracy and a higher possibility of falling into the optimal local solution. Random numbers obeying the Lévy distribution have the characteristics of short-distance walking and long-distance jumping, which will significantly improve the disadvantage of hungry sparrows (scavengers) that only search for food within a relatively close range of producers.

In summary, this part combines the LF strategy and the inertia weight factor into the classic SSA to improve its ability to expand the search scope and avoid local optimization. In this way, TFSSA can locate the optimal global solution more effectively. Equations (10)–(13) describe this mechanism. Equation (10) can be used to express the Lévy distribution [77].

$$Lévy(\alpha) \sim \mu = e^{-1-\alpha}, 0 < \alpha \leq 2, \tag{10}$$

where α is a stability index, and $\alpha = 1.5$, the μ is a Gaussian distribution. The inertia-weighting factor σ is expressed by Equation (11).

$$\sigma = 1 - t/T\_max, \tag{11}$$

then, the sparrow's position, x_{iD}^{t}, is mutated by the random roulette strategy in Equation (12). If rand $> \sigma$,

$$x_{iD}^{t'} = x_{iD}^{t} + L(\alpha) \cdot \left(x_{iD}^{t} - x_{best}^{t}\right), \tag{12}$$

else the $x_{iD}^{best'}$ is also changed by Equation (13).

$$x_{iD}^{best'} = x_{iD}^{best} \cdot (1 + L(\alpha)), \tag{13}$$

where $L(\alpha)$ is a number chosen randomly from the Lévy distribution. This part mainly combines the LF strategy with classic SSA and uses LF characteristics to improve its ability to expand the search scope and avoid local optimization. LFs can optimize the diversity of search agents, enabling the algorithm to explore search locations and avoid local minima effectively. The combination of LFs and the SSA algorithm improves the population diversity to a certain extent and enhances the robustness and global optimization capability of the SSA algorithm. However, in many experiments, it is found that the occasional long-distance jump of LFs has no significant impact on the final performance of the algorithm, as expected. Because of its poor performance in the CEC2020 benchmark function, we are considering further improving the algorithm from the manufacturer's location formula. We use super adaptive parameters in the next section to update the producer location and improve the global search capability.

3.3. Self-Adaptive Hyper-Parameters

In the rule design of SSA in Section 2.3, SSA mainly divides the sparrow population into producers (leaders) and scavengers (followers). Producers need more search space to find food sources, while scavengers mainly follow producers to find food. Therefore, the global search capability of the original SSA is highly dependent on the search scope of the producer.

In Equation (3), $R_2 < ST$ means that there are no predators at present, and the producer (leader) opens the wide-area search mode. In this mode, the location update of producers (leaders) is mainly affected by $\exp(-\frac{i}{\alpha \cdot T\_max})$. When α in Equation (3) gets a large random value, the value of $\exp(-\frac{i}{\alpha \cdot T\_max})$ will gradually decrease from (0, 1) to (0, 0.4) as i becomes larger. To sum up, we expand the search range of producers by adaptive control factors. The adaptive control factors are shown below in Equation (14).

$$w = w_0 \times c^t, \tag{14}$$

where t is the current iteration number; $w_0 = 1$ is the initial weight; c is the adaptive factor of w, which can be modified depending on the actual problem; and w is the initial weight. According to the subsequent sensitivity analysis, the performance of the TFSSA is relatively stable and achieves its best classification accuracy in most cases of datasets when c is 0.9. Therefore, in our current research, c is set to 0.9 to maintain w at a low value, enhancing the global search capability and broadening the producers' search scope. The original producer position is updated from Equation (3) to Equation (15).

$$X_i^{t+1} = \begin{cases} X_i^t \cdot \exp(-\frac{i}{w \cdot \alpha \cdot T\_max}), & \text{if } R_2 < ST \\ X_i^t + Q \cdot L, & \text{if } R_2 \geq ST \end{cases}. \tag{15}$$

In addition, to detect and warn companions to avoid predators in time during the foraging process, one-tenth to one-fifth of sparrows are selected as guards, also called patrollers. When the patroller realizes the danger and issues an alarm, the entire population will immediately conduct anti-predation behaviors, thereby improving the entire population's predation ability and risk-prevention capabilities. In other words, the presence of patrollers can help sparrow populations achieve better SSA solutions. When the number of patrollers is large, it is beneficial to improve the global optimization ability of sparrows. However, as the number of patrollers is reduced, it aids in accelerating SSA convergence. Therefore, this paper proposes an adaptive update formula for the number of patrollers, aiming to improve the algorithm's performance by adjusting the number of patrollers, as shown by Equation (16), the formula can non-linearly reduce during the iteration.

$$GN = GN_{\max} - Round\left[(GN_{\max} - GN_{\min}) \times \frac{t}{T\_max}\right], \tag{16}$$

where GN represents the number of patrollers; $GN_{\max}$ represents the maximum number of patrollers; $GN_{\min}$ represents the minimum number of patrollers; the $Round$ function is used to round values; t is the current iteration; and $max\_iteration$ is the maximum number of iterations. The GN in the original SSA is set to be chosen at random from 10–20% of the sparrow population. Equation (16) in this paper replaces the random selection behavior of the number of patrollers in the original SSA and better balances the algorithm convergence speed and global optimization ability. When all the sparrows find the optimal solution, this paper mutates the optimal sparrow individuals again to improve the global convergence accuracy further.

3.4. Optimal Individual Mutation by ψ-Tent Chaos

The original SSA is prone to fall into local extrema in the later iterations. To solve this problem, the optimal individual position is perturbed in each iteration, and only one individual is randomly mutated in each iteration. That is, when the sparrow finds the optimal solution, the enhanced Tent chaos is used to mutate the optimal sparrow individual, which further improves the global convergence accuracy and optimizes the shortcomings

of the original algorithm in global search and local search [78]. Therefore, in TFSSA, the optimal sparrow individuals are changed by Equations (17) and (18).

$$r = \frac{e^{2\cdot(1-k/\,T\_max)} - e^{-2\cdot(1-k/\,T\_max)}}{e^{2\cdot(1-k/\,T\_max)} + e^{-2\cdot(1-k/\,T\_max)}}, \tag{17}$$

if $rand < r$, then the optimal sparrow position $x_{iD}^{best'}$ is updated by Equation (18).

$$x_{iD}^{best'} = x_{iD}^{best} \cdot \left(1 + \ \psi\text{-}Tent\left(x_{iD}^{best}\right)\right), \tag{18}$$

where $\psi\text{-}Tent\left(x_{iD}^{best}\right)$ can be calculated by Equation (8). The overall flow of TFSSA is in Algorithm 2.

Algorithm 2 TFSSA

Input:
The number of sparrows(N)
The number of producers(PN)
The number of guards(GN)
The safety threshold(ST)
The warning value(R_2)
The maximum iterations(T_max)
Output:
The best position so far(X_{best})
The best solution so far(f_g)

Initialize a flock of sparrows' location X // Pretreatment by Equations (8) and (9).
$t \leftarrow 0$;
while ($t < T\_max$) **do**
  Rank the fitness vaule F_X using Equation (2);
  Find the f_g and f_w;
  Update the $R_2 \leftarrow$ a random value in [0, 1], and calculate the σ using Equation (17).
  for each leaders $i \in [1, PN]$ **do**
    The location of leaders(producers) is updated using Equation (15); // The original producer position is updated from Equation (3) to Equation (15).
  end for
  for each followers $i \in [PN + 1, N]$ **do**
    The location of followers(scroungers) is updated using Equation (4);
  end for
  for each patrollers $i \in [1, GN]$ **do**
    The location of patrollers is updated using Equation (5); // The GN is updated using Equation (16).
  end for
  Update X_{best} and f_g.
  for $m \in [1, N]$ **do**
    if ($rand > \sigma$) **then**
      The X_{best} is updated using Equation (12). // σ indicates the inertia weighting factor.
    else
      The X_{best} is mutated using Equation (13).
    end if
  end for
  Update X_{best} and f_g.
  Calculate the r using Equation (17).
  if ($rand < r$) **then**
    $X_{best} \leftarrow x_{iD}^{best'}$; // The X_{best} is mutated using Equation (18).
  end if
  Rearrange all of the population's F_X in ascending order.
  $X_{best} \leftarrow x_{best}^{t+1}$; // Update X_{best}.

$f_g \leftarrow f(X_{best})$; // Update f_g.
$t \leftarrow t + 1$;
end while
return X_{best}, f_g.

3.5. Computational Complexity Analysis

This subsection uses the well-known Big-O notation to present the proposed TFSSA's time and computational space complexity. Although the proposed TFSSA and SSA have the same time complexity of $O(N)$ and space complexity of $O(D \times N)$, TFSSA performs better than SSA in the sequence experiments.

3.5.1. Time Complexity Analysis

The time complexity depends on the size of the sparrow population (N), the dimension of the problem (D), the maximum number of iterations (T_max), the number of producers (PN) and scroungers (N-PN) along with the number of patrollers (GN). The time complexity of stage (1) in SSA is $O(D \times N)$, the time complexity of stages (2) and (3) is $O(D \times N)$, and the time complexity of stage (4) is $O((PN+ N\text{-}PN + GN) \times T\_max \times D)$, which is $O(N \times T\_max \times D)$; hence the total time complexity of SSA is $O(N)$. The proposed TFSSA mainly includes the stages shown in Figure 1. The computational cost of TFSSA is primarily different from that of stage (4) in comparison to SSA. TFSSA has a computational complexity of $O(T\_max \times D \times 3N)$ when it comes to the sparrow's location updating phase. To summarize, the proposed TFSSA and classical SSA have a time complexity of $O(N)$.

Figure 1. Main steps of TFSSA.

3.5.2. Space Complexity Analysis

The space complexity of TFSSA relative to the amount of memory space depends on the number of sparrows and the dimensions of the problem. This determines the total amount of memory space required for the input values that the proposed TFSSA uses for execution. Therefore, without considering the auxiliary space, the space complexity of TFSSA and SSA is $O(D \times N)$.

4. TFSSA Applied for FS

In this section, we introduce the application of TFSSA in classification tasks. In the novel algorithm, We start by discretizing the initial position of each sparrow in the group generated by the chaotic initialization of ψ-Tent to each dimension. Then, we set the fitness function utilized in TFSSA to evaluate individual sparrow placements. Finally, the process iterates until the stop criterion is met and the optimal feature subset's feature space is obtained. In the following, we detail the application in FS of the proposed methods.

4.1. Initialization

The initialization stage is the first step of the EA, in which, according to Equations (8) and (9), a sparrow individual of N is generated through the chaotic initialization of ψ-Tent. In this study, we try to identify the significant 1 value and reject the other feature 0 value. Before starting the fitness evaluation process, according to Equations (8) and (9) and Figure 2, we first discretize the initial position of each sparrow in the group to the position on each dimension; that is, 0 (not selected) or 1 (selected), and convert it to a random binary value (between 0 and 1).

F_1	F_2	F_3	F_4	···	F_{D-2}	F_{D-1}	F_D
1	1	0	0	···	1	0	1

Figure 2. Solution representation.

4.2. Fitness Evaluation

In this part, the TFSSA is exploited in FS for classification problems. The different feature combinations for a feature vector of size η would be 2^{η}, which is a massive space of features to be searched thoroughly. As a result, TFSSA is utilized to choose the optimal feature subset's feature space. Equation (19) shows the fitness function utilized in TFSSA to evaluate individual sparrow placements.

$$Fitness \ = \lambda E_R(D) + \mu \frac{|S|}{|T|}, \tag{19}$$

where $E_R(D)$ is the error rate for the classifier of condition attribute set, $|S|/|T|$ denotes the ratio of chosen features to total features, $\lambda \in [0, 1]$ and $\mu = 1 - \lambda$.

K-Nearest Neighbors (K-NN) [79] is a popular classification method that may be used to evaluate fitness functions as a simple candidate classifier. The smallest distance between the query instance and the training examples determines the K-NN classifier. A crucial characteristic of wrapper techniques in FS is the use of the classifier as a guide to the FS activity. The following three primary items can be used to classify wrapper-based feature selection: (1) Method of classification. (2) Criteria for evaluating features. (3) Search method. As demonstrated in Equation (19), TFSSA is employed as a search strategy that may adaptively explore the feature space to maximize the feature evaluation criterion. A sparrow's location in the search space reflects one feature combination or solution since each dimension represents a different feature combination or solution.

4.3. Termination

In each iteration, the position of sparrows (producers, scavengers, patrollers) is updated (refer to Algorithm 2), and the continuous value of the position vector is recorded after each iteration for future use in the continuous position update of the entire constant iteration. Next, the process iterates until the stop criterion is met, that is, the maximum number of function evaluations in this study.

5. Experimental

In this section, we introduce the evaluation of TFSSA in benchmark functions and multi-perspective analysis. Then, we discuss the performance of the proposed algorithm in FS.

5.1. Evaluation of TFSSA

The CEC2020 benchmark suite is selected to evaluate the effectiveness and superiority of the proposed algorithm, TFSSA, and compare it with seven other algorithms, including the Artificial Bee Colony Algorithm (ABC), PSO, Competitive Swarm Optimizer (CSO), DE, SSA, Optimal Foraging Algorithm (OFA), and Success History-based Adaptive Differential Evolution (SHADE). The reasons for choosing the CEC2020 benchmark suite are discussed in Section 5.1.1.

5.1.1. Benchmark Functions

CEC benchmarks are the most widely used benchmark problems and have been used by many research scientists to test their algorithms. The most popular single-objective optimization test function set includes CEC2005 [80], CEC2008 [81], CEC2010 [82], CEC2013 [83], CEC2014 [84], CEC2017 [85], CEC2020 [86], and the Single-Objective optimization problem (SOP) [87]. The single-objective optimization algorithm is the basis for building more complex methods, such as multi-objective, super multi-objective, multi-modal multi-

objective, niche, and constrained optimization algorithms. Therefore, it is crucial to improve single-objective optimization algorithms because they will also impact other areas. These algorithm improvements, to some extent, depending on the feedback of experiments conducted using single objective benchmark functions, which are basic components of more complex tasks. With the improvement in algorithms, researchers must develop more challenging functions to adapt to new problems. The interaction between methods and problems promoted progress, and CEC2020, developed by researchers further, promoted this symbiotic relationship.

The improved methods and problems sometimes need to update the traditional test standards, and the traditional test standards (such as SOP) can not guarantee enough persuasiveness when facing new, improved algorithms. Therefore, this paper uses the classical test function set CEC2020 to test the comprehensive performance of the proposed algorithm TFSSA. CEC2020 includes one unimodal function (CEC2020_F1), three basic functions (CEC2020_F2–CEC2020_F4), three mixed functions (CEC2020_F5–CEC2020_ F7), and three synthesis functions (CEC2020_F8–CEC2020_F10), as shown in Table 1. The MATLAB and C code for the CEC2020 test suite is available online (https://github.com/yyy24601/2020-Bound-Constrained-Opt-Benchmark, accessed on 20 January 2023).

Table 1. CEC2020 test suite.

	No.	Functions	$F_i^* = F_i(x^*)$
Unimodal Function	1	CEC 2017 [85] F1	100
Basic Functions	2	CEC 2014 [84] F11	1100
	3	CEC 2017 [85] F7	700
	4	CEC 2017 [85] F19	1900
Hybrid Functions	5	CEC 2014 [84] F17	1700
	6	CEC 2017 [85] F16	1600
	7	CEC 2014 [84] F21	2100
Composition Functions	8	CEC 2017 [85] F22	2200
	9	CEC 2017 [85] F24	2400
	10	CEC 2017 [85] F25	2500
Search Range = $[-100, 100]^D$			

5.1.2. Parameter Setting

The experiments are implemented with MATLAB (version 9.11.01769968 (R2021b)) running on a 64-bit Windows with Intel (R) Xeon (R) E-2224 CPU 3.40GHz CPU, NVIDIA Quadro P1000, and 16.0 GB RAM.

ABC [88], PSO [89], CSO [90], DE [91], SSA [70], OFA [92], and SHADE [93] are used as benchmark algorithms for comparison. The number of trials for releasing a food resource of ABC is 20, the inertia weight of PSO is 0.4, the social factor of CSO is 0.1, the conjugate constant of DE is 0.9, the mutation factor of DE is 0.5, the historical memory size of SHADE is 100, the chaos disturbance factor parameter is 0.7, and the LF parameter is 1.5. The population in all algorithms is 100, and the number of runs is 30. Each algorithm repeats the experiment 30 times independently to obtain statistical results. The maximum number of function evaluations is 10,000. The solution schemes are tested on the CEC2020 function at 10D, 15D, and 20D.

5.1.3. Statistical Test

The significance level is used to compare whether the two algorithms significantly differ in performance. We use the Wilcoxon rank sum test with $\alpha = 0.05$ [94]. The original assumption is that the performance of TFSSA and the comparison algorithm is independent.

When rejecting the original hypothesis, this paper uses three symbols to indicate whether there is a significant difference in the performance between TFSSA and the comparison algorithm.

(1) +: TFSSA performs significantly better than the comparison algorithm.

(2) =: The performance of TFSSA is not significantly related to the performance of the comparison algorithm.

(3) −: TFSSA's performance is not significantly better than the comparison algorithm.

5.1.4. Solution Accuracy Analysis

This section displays the average value (Mean), standard deviation (Std), and Wilcoxon rank sum test results produced by various algorithms on CEC2020 for each test function. The best results from all experiments are highlighted in bold. The following is a complete description and analysis of the experimental results:

F1 in Tables 2–4 shows the optimization results of the unimodal function obtained by different algorithms. Under the circumstances of 10D, 15D, and 20D, the submitted TFSSA for a single-peak function has an average weight and standard difference capital compared to other methods.

F2–F4 in Tables 2–4 display the basic function improvement results obtained by the differential arithmetic method. Under the circumstances of 10D and 20D, TFSSA has the highest average CSO award among 10D, F4 has the highest average, and F2 and F3 have the highest functional average. For the standard difference direction, F2 and F3 shows the best results in 10D, and F3 shows the best results in 15D. Due to this and other arithmetic ratios, the proposed TFSSA has improved performance on basic functions. The number of events increased as a result of the above findings, and the supplied TFSSA displayed the ideal performance.

F5–F7 in Tables 2–4 show the results obtained by different algorithms for the optimization of mixed functions. It can be seen that in the case of 10D, TFSSA obtained the two best averages of F6 and F7; in the case of 15D, TFSSA obtained the best average of F6; in the case of 20D, TFSSA also obtained the best average of F6. SHADE obtains the best mean of F5 in 10D; CSO obtains the best mean of F5 and F7 in both 15D and 20D. This indicates that the stability of TFSSA slightly increases with the increase in the dimension of the solution decline. In particular, TFSSA has obvious advantages on the F6 function and achieves the best results in all dimensions compared to the comparison algorithm.

F8–F10 in Tables 2–4 show the results of the synthesis function optimization obtained by different algorithms. It can be seen that in the case of 10D, TFSSA obtained the best mean of F8; in the case of 15D, TFSSA obtained the best mean of F8 and F10; in the case of 20D, TFSSA also obtained the best mean of F9. In the case of 10D, the best means of F9 and F10 are obtained by SSA and DE; in the case of 15D, the best means of F8 and F9 are obtained by DE; in the case of 20D, the best means of F9 and F10 are obtained by PSO and SHADE.

Table 2. Mean, standard deviations, and Wilcoxon rank sum test results of different algorithms on CEC2020 at 10D.

	ABC Mean (Std)	**PSO Mean (Std)**	**CSO Mean (Std)**	**DE Mean (Std)**	**SSA Mean (Std)**	**OFA Mean (Std)**	**SHADE Mean (Std)**	**TFSSA Mean (Std)**
CEC2020_F1	2.7683×10^4 (7.29×10^4) +	3.6043×10^3 (3.38×10^3) +	1.9567×10^3 (9.27×10^2) =	3.9673×10^4 (2.09×10^4) -	4.2342×10^3 (4.82×10^3) =	2.7980×10^5 (1.54×10^5) +	4.7054×10^3 (4.24×10^3)+	5.7683×10^2 (2.31×10^2)
CEC2020_F2	1.4226×10^3 (1.63×10^2) +	1.4462×10^3 (1.54×10^2) -	1.1631×10^3 (6.93×10^1) +	1.3938×10^3 (8.74×10^1) +	1.3095×10^3 (1.19×10^2) +	1.4722×10^3 (1.58×10^2) +	1.2221×10^3 (5.64×10^1) +	1.1508×10^3 (6.11×10^0)
CEC2020_F3	7.1526×10^2 (3.11×10^0) =	7.1286×10^2 (5.03×10^0) +	7.0711×10^2 (1.44×10^0) +	7.1502×10^2 (2.83×10^0) +	7.1267×10^2 (5.42×10^0) +	7.2197×10^2 (5.19×10^0) =	7.1070×10^2 (1.63×10^0) +	7.0620×10^2 (8.06×10^{-1})
CEC2020_F4	1.9009×10^3 (2.90×10^{-1}) +	1.9008×10^3 (7.54×10^{-1}) +	1.9003×10^3 (9.82×10^{-2}) +	1.9008×10^3 (2.76×10^{-1}) +	1.9005×10^3 (2.74×10^{-1}) +	1.9028×10^3 (1.01×10^0) -	1.9006×10^3 (1.17×10^{-1}) -	1.9003×10^3 (1.60×10^{-1})
CEC2020_F5	1.7150×10^3 (1.76×10^1) =	8.9157×10^3 (6.58×10^3) -	1.7254×10^3 (2.73×10^1) +	1.7314×10^3 (1.28×10^1) =	1.7552×10^3 (7.78×10^1) =	1.7356×10^3 (1.49×10^1) =	1.7065×10^3 (2.65×10^0) +	1.7595×10^3 (6.06×10^1)
CEC2020_F6	1.6044×10^3 (4.07×10^0) +	1.6336×10^3 (4.65×10^1) -	1.6037×10^3 (6.16×10^0) -	1.6033×10^3 (1.43×10^0) -	1.6077×10^3 (1.40×10^1) +	1.6110×10^3 (1.10×10^1) -	1.6011×10^3 (2.08×10^{-1}) -	1.6001×10^3 (1.46×10^{-1})
CEC2020_F7	2.1003×10^3 (3.11×10^{-1}) +	2.1070×10^3 (1.23×10^1) -	2.1007×10^3 (3.31×10^{-1}) -	2.1008×10^3 (1.46×10^{-1}) +	2.1036×10^3 (1.02×10^1) +	2.1024×10^3 (1.39×10^0) =	2.1001×10^3 (2.87×10^{-2}) +	2.1000×10^3 (1.22×10^{-2})
CEC2020_F8	2.2175×10^3 (3.40×10^1) =	2.2490×10^3 (4.61×10^1) =	2.2326×10^3 (4.56×10^1) +	2.2200×10^3 (3.11×10^0) =	2.2456×10^3 (4.84×10^1) =	2.2586×10^3 (3.28×10^1) +	2.2396×10^3 (5.28×10^1) =	2.2016×10^3 (1.22×10^0)
CEC2020_F9	2.5879×10^3 (7.23×10^1) +	2.5694×10^3 (1.13×10^2) +	2.5163×10^3 (5.33×10^1) +	2.5227×10^3 (6.61×10^0) +	2.5095×10^3 (2.47×10^1) =	2.5232×10^3 (6.69×10^0) +	2.5112×10^3 (4.74×10^1) =	2.5485×10^3 (8.11×10^1)
CEC2020_F10	2.8474×10^3 (2.18×10^{-2}) =	2.8515×10^3 (1.08×10^1) +	2.8474×10^3 (5.47×10^{-3}) +	2.7394×10^3 (6.38×10^1) +	2.8433×10^3 (1.48×10^1) =	2.8518×10^3 (2.56×10^0) -	2.8474×10^3 (7.79×10^{-2}) +	2.8475×10^3 (9.47×10^{-2})
+/-/=	6/0/4	5/4/1	7/2/1	6/2/2	5/0/5	4/3/3	6/2/2	

Table 3. Mean, standard deviations, and Wilcoxon rank sum test results of different algorithms on CEC2020 at 15D.

	ABC Mean (Std)	**PSO Mean (Std)**	**CSO Mean (Std)**	**DE Mean (Std)**	**SSA Mean (Std)**	**OFA Mean (Std)**	**SHADE Mean (Std)**	**TFSSA Mean (Std)**
CEC2020_F1	4.4019×10^{6} (2.51×10^{6}) +	2.9111×10^{8} (2.59×10^{8}) +	4.0135×10^{5} (8.96×10^{5}) +	7.9228×10^{7} (3.10×10^{7}) +	4.8618×10^{7} (5.11×10^{7}) +	2.1070×10^{8} (9.23×10^{7}) -	5.8744×10^{5} (2.12×10^{5}) -	1.1879×10^{5} (9.47×10^{4})
CEC2020_F2	2.9428×10^{3} (2.08×10^{2}) +	2.1909×10^{3} (4.08×10^{2}) -	1.5976×10^{3} (2.67×10^{2}) =	2.6368×10^{3} (1.67×10^{2}) +	1.6696×10^{3} (2.04×10^{2}) -	2.7231×10^{3} (1.67×10^{2}) +	2.2149×10^{3} (1.80×10^{2}) +	1.4524×10^{3} (2.29×10^{2})
CEC2020_F3	7.6011×10^{2} (7.84×10^{0}) -	7.4882×10^{2} (1.57×10^{1}) +	7.2060×10^{2} (4.90×10^{0}) +	7.7524×10^{2} (9.53×10^{0}) -	7.5645×10^{2} (1.44×10^{1}) +	7.8067×10^{2} (9.54×10^{0}) +	7.4497×10^{2} (5.67×10^{0}) +	7.1830×10^{2} (3.48×10^{0})
CEC2020_F4	1.9048×10^{3} (6.95×10^{-1}) -	2.3811×10^{3} (1.25×10^{3}) +	1.9013×10^{3} (5.85×10^{-1}) =	1.9063×10^{3} (9.44×10^{-1}) -	1.9290×10^{3} (1.10×10^{2}) -	1.9394×10^{3} (2.93×10^{1}) +	1.9032×10^{3} (4.50×10^{-1}) +	1.9016×10^{3} (5.06×10^{-1})
CEC2020_F5	2.7081×10^{5} (2.35×10^{5}) =	1.9609×10^{5} (2.33×10^{5}) =	2.1201×10^{3} (1.57×10^{2}) +	3.1328×10^{3} (3.25×10^{2}) +	3.2132×10^{5} (4.63×10^{5}) +	5.6867×10^{4} (3.80×10^{4}) =	2.5813×10^{3} (2.04×10^{2}) +	3.0341×10^{5} (5.41×10^{5})
CEC2020_F6	1.7540×10^{3} (8.24×10^{1}) +	1.9243×10^{3} (1.23×10^{2}) -	1.6886×10^{3} (7.38×10^{1}) -	1.8042×10^{3} (5.76×10^{1}) +	1.7443×10^{3} (1.02×10^{2}) -	1.8783×10^{3} (6.44×10^{1}) -	1.6480×10^{3} (3.32×10^{1}) -	1.6370×10^{3} (5.21×10^{1})
CEC2020_F7	2.7754×10^{4} (4.28×10^{4}) =	1.1125×10^{4} (8.70×10^{3}) =	2.3156×10^{3} (1.34×10^{2}) +	2.5866×10^{3} (1.85×10^{2}) +	1.8706×10^{4} (2.32×10^{4}) +	1.3730×10^{4} (8.65×10^{3}) =	2.3187×10^{3} (8.84×10^{1}) +	4.8971×10^{4} (1.05×10^{5})
CEC2020_F8	2.3084×10^{3} (1.33×10^{1}) +	2.3514×10^{3} (3.41×10^{1}) +	2.3118×10^{3} (2.18×10^{0}) +	2.3251×10^{3} (1.75×10^{1}) -	2.3326×10^{3} (4.36×10^{1}) +	2.3514×10^{3} (1.35×10^{1}) +	2.3101×10^{3} (4.05×10^{-2}) -	2.3101×10^{3} (3.76×10^{-2})
CEC2020_F9	2.7803×10^{3} (9.44×10^{0}) +	2.7573×10^{3} (8.56×10^{1}) -	2.7211×10^{3} (6.03×10^{1}) =	2.6728×10^{3} (4.05×10^{1}) +	2.7205×10^{3} (9.73×10^{1}) +	2.7540×10^{3} (6.15×10^{1}) =	2.7454×10^{3} (5.71×10^{1}) -	2.7188×10^{3} (8.71×10^{1})
CEC2020_F10	2.9523×10^{3} (2.17×10^{1}) +	2.9583×10^{3} (3.36×10^{1}) +	2.9215×10^{3} (2.21×10^{1}) +	2.9531×10^{3} (7.78×10^{0}) -	2.9476×10^{3} (3.08×10^{1}) +	2.9844×10^{3} (1.96×10^{1}) +	2.9298×10^{3} (2.21×10^{1}) +	2.9160×10^{3} (1.31×10^{0})
+/-/=	6/2/2	5/3/2	6/1/3	6/4/0	7/3/0	5/2/3	6/4/0	

Table 4. Mean, standard deviations, and Wilcoxon rank sum test results of different algorithms on CEC2020 at 20D.

	ABC Mean (Std)	**PSO Mean (Std)**	**CSO Mean (Std)**	**DE Mean (Std)**	**SSA Mean (Std)**	**OFA Mean (Std)**	**SHADE Mean (Std)**	**TFSSA Mean (Std)**
CEC2020_F1	8.2541×10^{8} (2.51×10^{8}) +	6.1650×10^{9} (2.74×10^{9}) +	2.5690×10^{9} (1.70×10^{9}) +	2.4253×10^{9} (6.48×10^{8}) +	1.8891×10^{9} (1.08×10^{9}) -	3.2793×10^{9} (9.17×10^{8}) +	8.1623×10^{6} (2.73×10^{6}) +	3.2993×10^{6} (1.14×10^{6})
CEC2020_F2	5.7675×10^{3} (3.20×10^{2}) +	3.9708×10^{3} (4.21×10^{2}) -	3.7089×10^{3} (4.74×10^{2}) -	5.5763×10^{3} (1.98×10^{2}) =	2.7448×10^{3} (3.21×10^{2}) +	5.6418×10^{3} (2.89×10^{2}) +	4.4956×10^{3} (2.84×10^{2}) -	1.5610×10^{3} (2.27×10^{2})
CEC2020_F3	9.5314×10^{2} (2.25×10^{1}) -	9.2851×10^{2} (3.56×10^{1}) +	8.1250×10^{2} (1.74×10^{1}) -	9.6884×10^{2} (1.69×10^{1}) +	9.5392×10^{2} (7.45×10^{1}) +	9.7358×10^{2} (2.29×10^{1}) -	8.2764×10^{2} (7.51×10^{0}) +	7.4445×10^{2} (8.55×10^{0})
CEC2020_F4	2.0108×10^{3} (5.50×10^{1}) -	1.8210×10^{4} (3.72×10^{4}) -	4.9532×10^{3} (7.09×10^{3}) -	2.4562×10^{3} (4.35×10^{2}) -	3.8546×10^{3} (3.45×10^{3}) -	3.3717×10^{3} (9.53×10^{2}) +	1.9107×10^{3} (1.03×10^{0}) -	1.9041×10^{3} (1.07×10^{0})
CEC2020_F5	5.8713×10^{6} (3.06×10^{6}) +	1.9239×10^{6} (1.82×10^{6}) =	2.6514×10^{4} (2.36×10^{4}) +	2.8432×10^{5} (8.82×10^{4}) +	1.6705×10^{6} (1.85×10^{6}) =	1.5941×10^{6} (8.94×10^{5}) =	4.6806×10^{4} (2.20×10^{4}) +	1.1683×10^{6} (1.04×10^{6})
CEC2020_F6	2.1056×10^{3} (1.34×10^{2}) -	2.4993×10^{3} (2.56×10^{2}) +	2.0628×10^{3} (1.78×10^{2}) -	2.4975×10^{3} (1.14×10^{2}) -	1.9359×10^{3} (1.10×10^{2}) -	2.6072×10^{3} (2.13×10^{2}) -	1.8690×10^{3} (6.26×10^{1}) -	1.6400×10^{3} (5.22×10^{1})
CEC2020_F7	9.5712×10^{5} (7.41×10^{5}) =	6.8662×10^{5} (1.23×10^{6}) =	8.0781×10^{3} (7.77×10^{3}) +	3.1638×10^{4} (1.46×10^{4}) +	7.7198×10^{5} (7.98×10^{5}) =	4.9021×10^{5} (2.77×10^{5}) =	8.3634×10^{3} (2.10×10^{3}) +	8.0619×10^{5} (9.61×10^{5})
CEC2020_F8	2.5147×10^{3} (4.22×10^{1}) -	3.7116×10^{3} (9.12×10^{2}) -	2.6301×10^{3} (1.80×10^{2}) -	3.5211×10^{3} (3.88×10^{2}) +	3.1946×10^{3} (9.62×10^{2}) -	3.1260×10^{3} (2.32×10^{2}) -	2.3229×10^{3} (1.01×10^{1}) -	2.3127×10^{3} (5.46×10^{-1})
CEC2020_F9	2.9444×10^{3} (1.10×10^{1}) +	2.8361×10^{3} (9.71×10^{0}) +	2.9084×10^{3} (2.37×10^{1}) +	2.9937×10^{3} (2.12×10^{1}) =	2.9290×10^{3} (3.21×10^{1}) +	3.0541×10^{3} (3.02×10^{1}) +	2.9137×10^{3} (9.92×10^{0}) +	3.1352×10^{3} (1.02×10^{2})
CEC2020_F10	3.0418×10^{3} (4.50×10^{1}) +	3.3440×10^{3} (1.50×10^{2}) +	3.0693×10^{3} (5.39×10^{1}) -	3.1881×10^{3} (7.56×10^{1}) -	3.0981×10^{3} (7.90×10^{1}) +	3.2489×10^{3} (8.23×10^{1}) +	2.9166×10^{3} (1.40×10^{0}) +	2.9607×10^{3} (3.37×10^{1})
+/-/=	5/4/1	5/3/2	4/6/0	5/3/2	4/4/2	5/3/2	6/4/0	

In conclusion, TFSSA obtained seven, five, and six optimal averages and zero, two, and one suboptimal average among the ten functions in all dimensions, respectively, indicating that dimensional changes with less impact constrain the algorithm's accuracy in finding solutions. In the case of 10D, SHADE obtains an optimal mean and three optimal standard deviations, SSA and DE each get an optimal mean, and CSO and DE each obtain an optimal standard deviation. Under the condition of 15D, the performance of the proposed TFSSA is challenged by CSO. CSO obtains three optimal means and one optimal standard deviation. At the same time, ABC and DE also achieved an optimal mean and standard deviation, and SHADE achieved three optimal means and the best standard deviation. In the 20D case, PSO and CSO obtain one and two best mean values, and PSO, DE, and SHADE get one, one, and four best standard deviations, respectively. According to NFL theory, it is almost impossible for one algorithm to solve all optimization problems efficiently. Therefore, the proposed TFSSA algorithm cannot obtain the best results on all classical test functions. However, compared with other algorithms, the best results it gets are still ideal, which verifies the superiority of the TFSSA algorithm to a certain extent. It can be seen from the results that the proposed TFSSA has good performance on CEC2020 at 10D, 15D, and 20D.

5.1.5. Algorithm Stability Analysis

From the results of the Wilcoxon test in Tables 2–4, it is observed that TFSSA significantly outperforms ABC, PSO, CSO, DE, and OFA on more than half of the functions of SHADE performance. Compared with the performance of DE and SHADE, at 15D, the performance of TFSSA is significantly improved on six functions, but the performance on four functions is reduced substantially. In other words, TFSSA performs much better than DE and SHADE at 15D. Compared with the performance of CSO, in the case of 10D, the performance of TFSSA is significantly improved on seven functions. Still, the performance of the two functions is reduced dramatically, and at 15D, the performance of TFSSA is considerably lower than that of CSO. Significantly improved performance on six functions but significantly reduced performance on two functions, with 20D TFSSA greatly enhanced performance on four parts but decreased performance considerably on six. It also shows that the stability of TFSSA at 10D is higher than that of 15D and higher than that of 20D to a certain extent.

5.1.6. Convergence Rate Analysis

This subsection presents the convergence rates obtained by different algorithms when solving the CEC2020 test function. The convergence speed of obtaining the optimal global solution is an important indicator for checking the performance of EA. Figures 3–5 show TFSSA and comparison algorithms at 10D, 15D, and 20D on CEC2020, respectively, and the convergence plot obtained during the process of solving the test function. Among them, the abscissa represents the number of function evaluations, and the ordinate is the minimum value obtained each time the algorithm runs independently.

It can be seen that in the 10D case, on F1, F4, F6, F7, and F8, the convergence speed of TFSSA is significantly faster than most of the other comparison algorithms, among which F6 and F7 are more significant; in the 15D case, TFSSA converges considerably quicker than most of the different comparison algorithms on F1, F4, F5, F7, and F8; in the 20D case, TFSSA converges significantly faster than most of the other comparison algorithms on F1, F4, F5, F6, and F7, especially in the early stages of evolution of these classical test functions, showing faster convergence rates. In addition, the advantages of TFSSA on other parts are not obvious. Except for F2 and F3 at 10D, 15D, and 20D, F9 at 10D, 15D, and 20D, and F8 at 20D, the convergence speed of TFSSA is slightly worse than other algorithms, but the accuracy is still the best. The experiment shows that the TFSSA algorithm's exploration ability in the later stage is relatively strong, which means that the proposed TFSSA algorithm can maintain relatively high population diversity and avoid premature convergence.

Overall, TFSSA showed the best convergence speed for most tested functions throughout the optimization process. Therefore, it can be concluded that the proposed TFSSA has a relatively good exploration ability on most of the test functions.

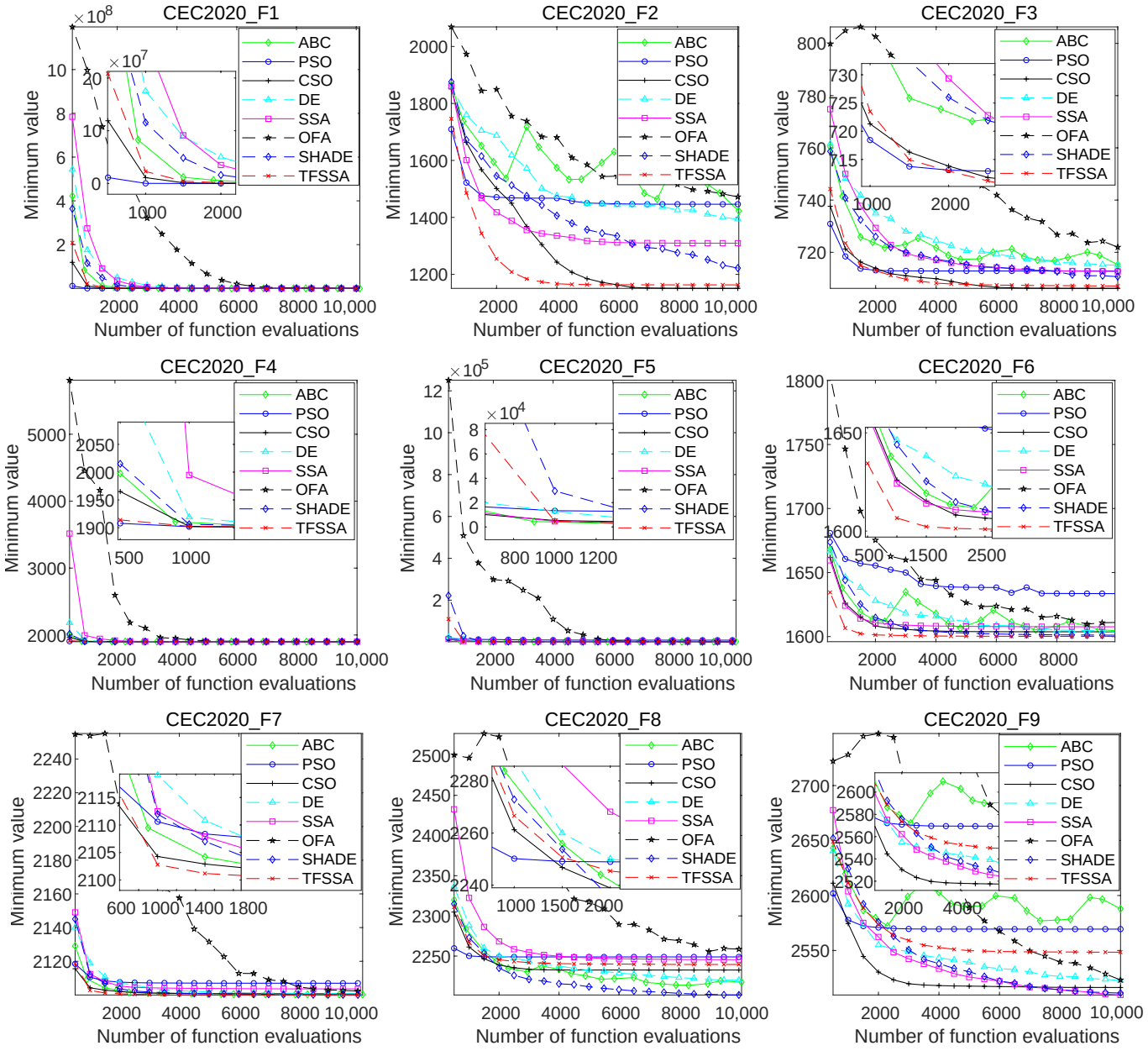

Figure 3. Convergence curves of different algorithms on CEC2020 at 10D, F1–F9.

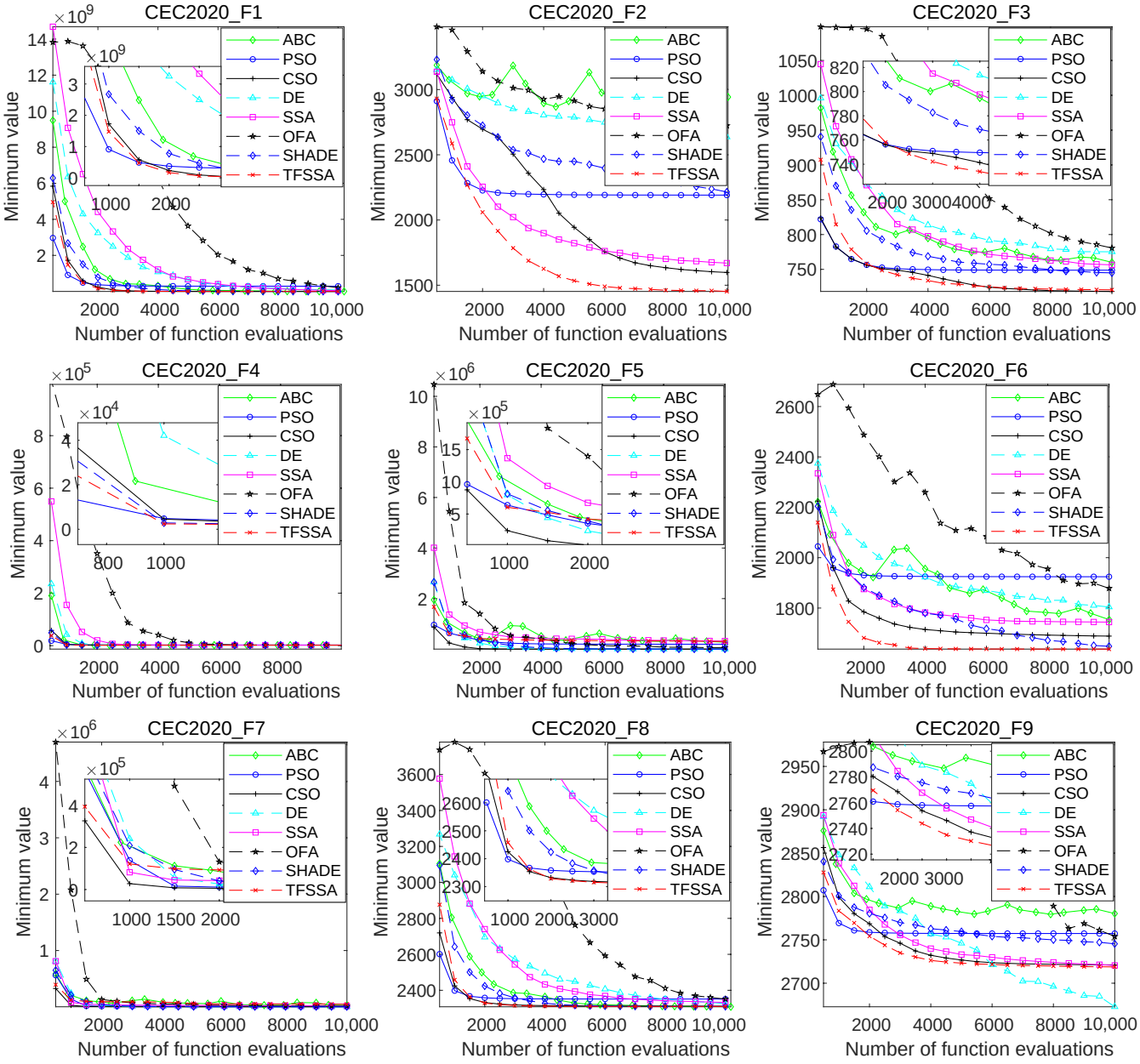

Figure 4. Convergence curves of different algorithms on CEC2020 at 15D, F1–F9.

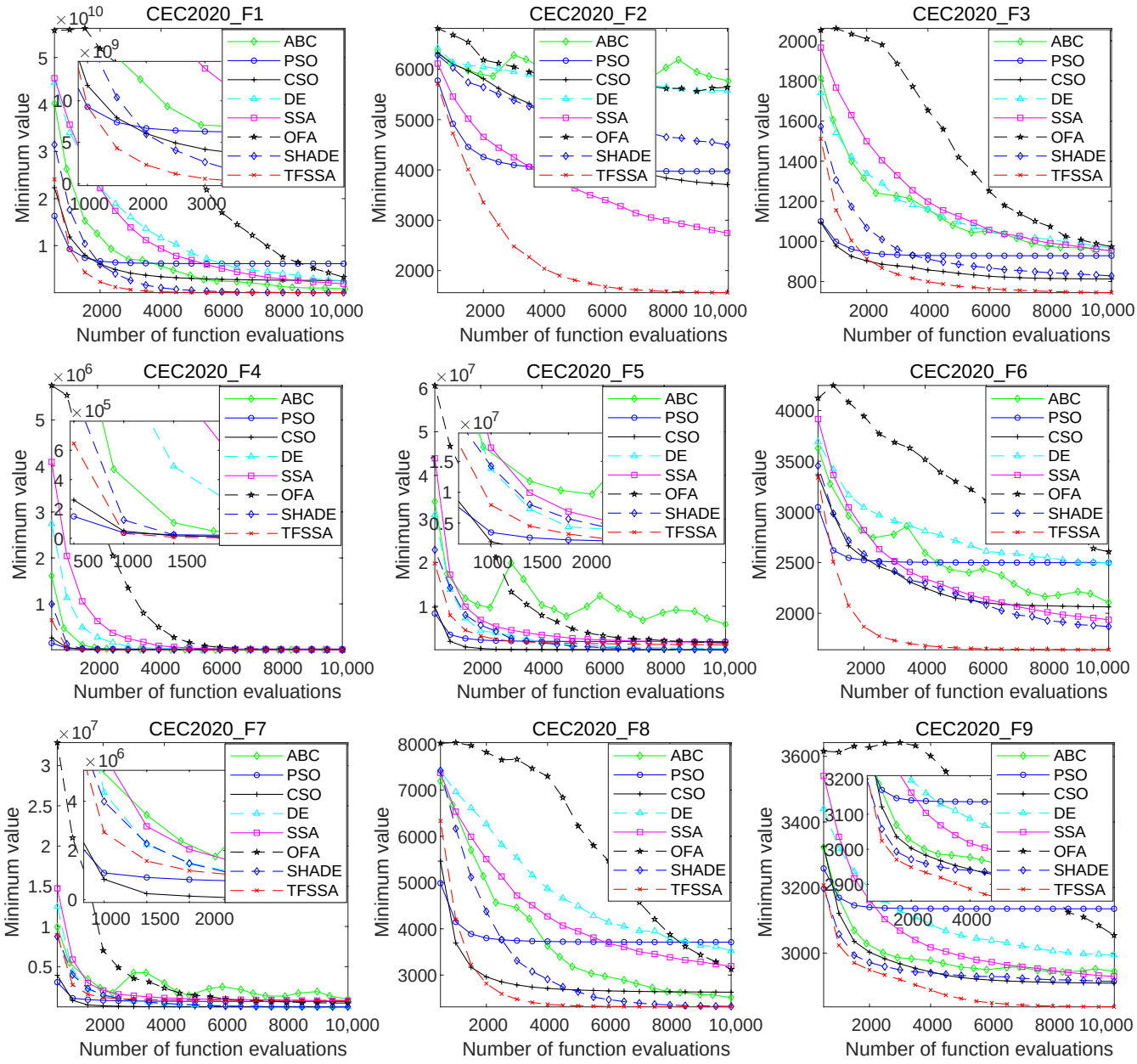

Figure 5. Convergence curves of different algorithms on CEC2020 at 20D, F1–F9.

5.1.7. Sensitivity Analysis

This section investigates the sensitivity of TFSSA to (1) parameter a, (2) parameter α, and (3) parameter c. This analysis helps determine which parameters are more robust and sensitive to various input values and which parameters have a greater impact on the accuracy of TFSSA. This study conducts a complete TFSSA design with some functions selected from the CEC2020 test suite. These functions include: CEC2020_F1 (20D), CEC2020_F2 (20D), CEC2020_F3 (20D), CEC2020_F1 (10D), CEC2020_F2 (10D), and CEC2020_F3 (10D). This experiment uses the same fitness function settings as before to ensure fairness. TFSSA's sensitivity analysis results for the test function for all dimensions considered were investigated based on the average fitness value of 30 independent runs.

(1) Control parameter a: In the initialization of TFSSA, the control parameter a is involved. To check the sensitivity of TFSSA to a, different values of this parameter were simulated based on keeping other parameters unchanged, which are 0.75, 0.7, 0.65, and 0.6. The influence of different parameter values on the mean value of TFSSA is shown in Table 5.

(2) Control parameter α: In the LF mechanism, control parameter α is involved. To check the sensitivity of TFSSA to α, different values of this parameter were simulated based on keeping other parameters unchanged, which are 1.4, 1.5, 1.6, and 1.7. The influence of different parameter values on the mean value of TFSSA is shown in Table 6.

(3) Control parameter c: In the adaptive hyperparameter, c is the adaptation factor of w, ensuring that w stays as a small value. To check the sensitivity of TFSSA to c, different values of this parameter were simulated based on keeping other parameters unchanged, which are 0.8, 0.85, 0.9, and 0.95. The influence of different parameter values on the mean value of TFSSA is shown in Table 7.

Overall, as can be seen from the results in Tables 5–7, TFSSA has a relatively good sensitivity to parameters a, α, and c, providing relatively reasonable results. TFSSA produced the best results when a, α, and c were 0.7, 1.5 and 0.9, respectively. It is of great help to study the sensitivity of the above control parameters to the performance of TFSSA. Furthermore, these parameters must be fine-tuned to help TFSSA obtain the best global solution.

Table 5. The mean value of TFSSA under different values for parameter a.

a	CEC2020 Functions					
	F1(20D)	**F2(20D)**	**F3(20D)**	**F1(10D)**	**F2(10D)**	**F3(10D)**
0.75	6.1904×10^5	1.5283×10^3	7.3278×10^2	1.1600×10^5	1.3817×10^3	7.2266×10^2
0.7	2.1390×10^2	1.1939×10^3	7.0635×10^2	1.4375×10^3	1.3653×10^3	7.2227×10^2
0.65	4.2859×10^4	1.4243×10^3	7.1528×10^2	1.2341×10^5	1.3561×10^3	7.2343×10^2
0.6	1.6447×10^5	1.3368×10^3	7.2416×10^2	1.1986×10^5	1.3808×10^3	7.2283×10^2

Table 6. The mean value of TFSSA under different values for parameter α.

α	CEC2020 Functions					
	F1(20D)	**F2(20D)**	**F3(20D)**	**F1(10D)**	**F2(10D)**	**F3(10D)**
1.4	3.1000×10^2	1.5988×10^3	7.3278×10^2	1.6447×10^5	1.3496×10^3	7.2484×10^2
1.5	2.7088×10^6	1.5737×10^3	7.3094×10^2	8.2635×10^4	1.3368×10^3	7.2301×10^2
1.6	3.1951×10^6	1.5540×10^3	7.3159×10^2	1.4045×10^5	1.3567×10^3	7.2416×10^2
1.7	3.2479×10^6	1.5684×10^3	7.3130×10^2	1.2841×10^5	1.3618×10^3	7.2345×10^2

Table 7. The mean value of TFSSA under different values for parameter c.

c	CEC2020 Functions					
	F1(20D)	**F2(20D)**	**F3(20D)**	**F1(10D)**	**F2(10D)**	**F3(10D)**
0.8	3.0500×10^2	1.5829×10^3	7.4364×10^2	1.1963×10^5	1.3653×10^3	7.2374×10^2
0.85	3.1280×10^3	1.5726×10^3	7.3054×10^2	1.6447×10^5	1.3688×10^3	7.2416×10^2
0.9	2.1200×10^2	1.2726×10^3	7.2054×10^2	1.1696×10^5	1.3368×10^3	7.2203×10^2
0.95	4.9988×10^3	1.1999×10^3	7.0672×10^2	1.2519×10^5	1.3780×10^3	7.2311×10^2

5.1.8. Runtime Analysis

Tables 8–10 show the running time of TFSSA and the comparison algorithm at 10D, 15D, and 20D. It can be seen that the running time of TFSSA on all test functions is slightly longer than most comparison algorithms. The main reasons for the above phenomenon are as follows:

1. When mutating the optimal individual, TFSSA compares the calculated r with the random value *rand* and mutates the optimal individual. This stage is more expensive than the original SSA.

2. The optimal individual must reorder the fitness function values after passing through the ψ-Tent chaotic mutation. Sorting is time-consuming, so this stage is also one of the main reasons for the increase in running time.

The running time of TFSSA in the study is slightly higher than most of these comparison algorithms. Still, in the end, considering the performance improvement, these additional running times are negligible to a certain extent.

Table 8. Running times of different algorithms on CEC2020 at 10D.

ABC	PSO	CSO	DE	SSA	OFA	SHADE	TFSSA
0.117	0.123	0.190	0.132	0.194	0.103	0.178	0.141
0.136	0.144	0.210	0.148	0.217	0.105	0.213	0.164
0.130	0.136	0.199	0.141	0.196	0.103	0.202	0.146
0.121	0.130	0.195	0.131	0.202	0.100	0.198	0.136
0.172	0.126	0.209	0.144	0.215	0.120	0.213	0.153
0.148	0.130	0.231	0.152	0.200	0.119	0.196	0.160
0.142	0.154	0.231	0.195	0.228	0.109	0.209	0.196
0.202	0.168	0.256	0.167	0.295	0.151	0.238	0.167
0.260	0.176	0.391	0.221	0.312	0.166	0.281	0.215
0.260	0.188	0.265	0.192	0.257	0.146	0.238	0.215

Table 9. Running times of different algorithms on CEC2020 at 15D.

ABC	PSO	CSO	DE	SSA	OFA	SHADE	TFSSA
0.27314	0.17447	0.21609	0.18038	0.25067	0.13861	0.19847	0.19285
0.17174	0.14590	0.21438	0.23227	0.24845	0.12954	0.20751	0.19680
0.14075	0.14449	0.18067	0.17106	0.24346	0.12337	0.18830	0.19825
0.13625	0.13080	0.19684	0.15475	0.22128	0.12704	0.18636	0.17833
0.18806	0.15771	0.22111	0.19518	0.25376	0.12731	0.21287	0.20431
0.16081	0.16362	0.22160	0.18305	0.25280	0.11954	0.21252	0.19229
0.18113	0.16160	0.22597	0.18884	0.25475	0.13430	0.21125	0.20429
0.18804	0.17925	0.28799	0.19967	0.27466	0.15683	0.22740	0.23170
0.20090	0.22087	0.27973	0.26221	0.29791	0.17547	0.26660	0.25203
0.21197	0.21208	0.33228	0.21209	0.30600	0.18752	0.27462	0.23527

Table 10. Running times of different algorithms on CEC2020 at 20D.

ABC	PSO	CSO	DE	SSA	OFA	SHADE	TFSSA
0.28840	0.19697	0.19697	0.17951	0.27007	0.12649	0.20115	0.22174
0.18208	0.22409	0.22409	0.28066	0.29550	0.15021	0.22463	0.22936
0.14329	0.19587	0.19587	0.19650	0.30428	0.12872	0.21098	0.22780
0.15119	0.20386	0.20386	0.19689	0.25890	0.12878	0.21079	0.20586
0.17717	0.22385	0.22385	0.20411	0.27980	0.14777	0.22679	0.23241
0.17056	0.20903	0.20903	0.19523	0.27751	0.14084	0.21278	0.22460
0.19126	0.22421	0.22421	0.20025	0.29574	0.16213	0.22949	0.21372
0.22325	0.27305	0.27305	0.24980	0.30859	0.19916	0.25927	0.25078
0.22689	0.32813	0.32813	0.29938	0.32780	0.22705	0.30924	0.28994
0.24966	0.31390	0.31390	0.25017	0.34546	0.18456	0.29214	0.26953

5.2. Performance of Proposed Model

5.2.1. Description of Data

The utility and strength of our suggested strategy will be thoroughly investigated by selecting features from well-known datasets. Twenty-one datasets are from the UCI machine learning repository [95] and can be accessed online (https://www.openml.org/search, accessed on 20 January 2023). Table 11 gives a summary of the datasets used. The number of features (#Feat), samples (#SMP), classes (#CL), and the area to which each dataset belongs are all provided for each dataset.

Table 11. Dataset descriptions.

No.	Dataset	#Feat	#SMP	#CL	Area
1	BreastCO	9	699	2	Medical
2	BreastCWD	30	569	2	Medical
3	Clean-1	166	476	2	Physical
4	Clean-2	166	6598	2	Physical
5	CongressVR	16	435	2	Social
6	Exactly-1	13	1000	2	Biology
7	Exactly-2	13	1000	2	Biology
8	StatlogH	13	270	5	Life
9	IonosphereVS	34	351	2	Physical
10	KrvskpEW	36	3196	2	Game
11	Lymphography	18	148	4	Medical
12	M-of-n	13	1000	2	Biology
13	Penglung	325	73	2	Biology
14	Semeion	265	1593	2	Computer
15	SonarMR	60	208	2	Physical
16	Spectheart	22	267	2	Life
17	3T Endgame	9	958	2	Game
18	Vote	16	300	2	Life
19	WaveformV2	40	5000	3	Physical
20	Wine	13	178	3	Physical
21	Zoology	16	101	7	Life

5.2.2. Parameter Configuration

Several top-of-the-line and most recent FS techniques are contrasted with the suggested approach, which is summarized as follows:

- Genetic Algorithm (GA) [96].
- Dragonfly Algorithm (DA) [97].
- Ant Lion Optimizer (ALO) [98].
- Sparrow Search Algorithm (SSA) [70].
- Sine Cosine Algorithm (SCA) [99].
- Particle Swarm Optimizer (PSO) [89].
- binary Butterfly Optimization Algorithm (bBOA) [100].
- Brain Storm Optimizer (BSO) [101].
- Improved Sparrow Search Algorithm (ISSA) [102].
- Grey Wolf Optimizer (GWO) [103].

Each algorithm runs 20 times independently with a random seed. For all subsequent tests, the maximum number of repetitions is set at 100. In the population, there are seven search agents. For our evaluations, we test our approach with a 10-fold cross-validation. Table 12 shows the global and algorithm-specific parameter settings. To ensure a fair comparison of the algorithms, the parameters of the algorithms are gathered from the literature. The main purpose of this research is to evaluate the performance of numerous FS methods compared to the proposed methodology. The K-NN classifier is a popular wrapper approach for FS. When K = 5, the method produces superior results.

Table 12. Experiment parameter configuration.

Parameter Description	Value (s)
a parameter in Tent chaos	0.7
α parameter in Lévy flights	1.5
λ parameter in *Fitness*	0.99
μ parameter in *Fitness*	0.01
Count of runs (M)	20

Table 12. *Cont.*

Parameter Description	Value (s)
The amount of search agents	7
The amount of T_max	100
Problem Dimensions	No. of features in each datasets
K for cross-validation	10
Search field	$\{0, 1\}$
GA crossover ratio	0.9
GA mutation ratio	0.1
Selection strategy in GA	Roulette wheel
A factors in WOA	$[0, 2]$
Acceleration factors in PSO	$[0, 2]$
Inertia index(w) in PSO	$[0.9, 0.6]$
A factors in GWO	$\{0, 2\}$
Mutation rate r in ALO	$[0, 0.9]$
Parameter(a) in bBOA	0.1
Parameter(c) in bBOA	$[0.01, 0.25]$
The amount of clusters in BSO	5

5.2.3. Evaluation Criteria

For each experiment, we randomly split each dataset into three unequal parts at random: training, testing, and validation datasets, with a ratio of 6:2:2. The dataset partition process was repeated ten times in each 10-fold cross-validation, and the average performance of accuracy for these ten results is compared for all methods. The following assurances are captured from the validation data for each run:

- *Classification average accuracy (AvgPerf)* is a metric that indicates how accurate the classifier is given the provided feature set. Equation (20) can be used to receive the classification average accuracy.

$$AvgPerf = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{j=1}^{M}\text{Match}(C_i, L_i), \tag{20}$$

  where M denotes the amount of times the optimizer is run to pick the feature subset, N denotes the number of points in the test set, C_i denotes the output label of the classifier for data point i, and L_i denotes the data point i's reference class label. If the two input labels are identical, the *Match* function returns 1 if they are. Otherwise, it returns 0.

- *Statistical Best* is the optimistic fitness value (the minimum value) obtained after each feature selection method runs M times, as shown in Equation (21).

$$Best = \min_{i=1}^{M} g_*^i, \tag{21}$$

  where g_*^i indicates the best result determined after i times of operation.

- *Statistical Worst* is the pessimistic result, which can be expressed as shown in Equation (22).

$$Worst = \max_{i=1}^{M} g_*^i. \tag{22}$$

- *Statistical Mean* is the average value of the solution obtained by running under the condition of M times, as shown in Equation (23).

$$Mean = \frac{1}{M}\sum_{i=1}^{M} g_*^i. \tag{23}$$

- *Statistical Std* is a representation of the variation in the obtained minimum (best) solutions for M different runs of a stochastic optimizer. Std is a stability and robustness metric for optimizers; if Std is small, the optimizer always converges to the same solution; on the contrary, the optimizer produces numerous random outcomes, as shown in Equation (24).

$$Std = \sqrt{\frac{1}{M-1}\sum(g_*^i - Mean)^2}. \tag{24}$$

- *Selection average size (AVGSelectionSZ)* represents the average amount of features selected, as shown in Equation (25).

$$AVGSelectionSZ = \frac{1}{M}\sum_{i=1}^{M}\frac{\text{size}(g_*^i)}{Di}, \tag{25}$$

  where Di is the dimension of each dataset, and size(x) is the amount of on values for the vector x.

- *Wilcoxon rank sum test* is a nonparametric statistical test designed to see if the results of a proposed new technique are statistically different from those of other comparative techniques. The rank sum test produces a p-value parameter that compares the significance level of the two methods. The p-value is less than 0.05, which indicates that the two methods are significantly different [104,105].

5.2.4. Comparison of TFSSA and Other FS Methods

In this section, the performance of the best strategy, TFSSA, is compared to that of nine approaches (including the BSO, ALO, PSO, GWO, GA, bBOA, DA, SSA, and ISSA) that have been widely used to address the FS problem in the literature. Some performance indicators used to evaluate the algorithm's performance include classification average accuracy, selected average feature number, selected average feature rate, statistical best fitness, statistical worst fitness, statistical mean fitness, statistical Std, calculation time, and Wilcoxon rank-sum test.

In Table 13, the classification average accuracy achieved by each algorithm is compared. TFSSA is preferred over other algorithms in most datasets except Exactly-1 and SonarMR. Furthermore, Figure 6 shows the overall average classification accuracy selected by different algorithms on all datasets. We can see that the proposed algorithm ranks first with a classification accuracy of 0.9011. This result confirms that the proposed TFSSA can effectively explore the solution search space and find the optimal feature subset with the highest classification accuracy.

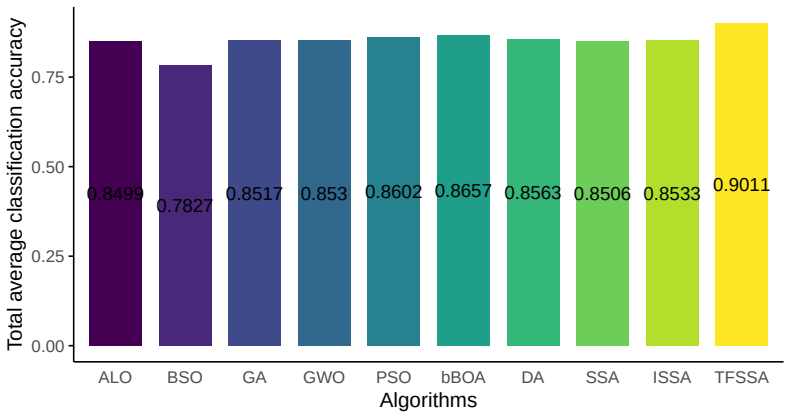

Figure 6. The average classification accuracy selected by the algorithms.

Table 13. Comparison of the classification accuracy of each algorithm.

No.	Datasets	ALO	BSO	GA	GWO	PSO	bBOA	DA	SSA	ISSA	TFSSA
1	BreastCO	0.9591	0.9200	0.9597	0.9603	0.9609	0.9286	0.9626	0.9600	0.9611	**0.9668**
2	BreastCWD	0.9392	0.9020	0.9488	0.9375	0.9385	0.9396	0.9385	0.9347	0.9396	**0.9718**
3	Clean-1	0.8465	0.8261	0.8697	0.8580	0.8549	0.8562	0.8541	0.8431	0.8585	**0.8923**
4	Clean-2	0.9496	0.9391	0.9423	0.9463	0.9465	0.9480	0.9487	0.9462	0.9510	**0.9667**
5	CongressVR	0.9370	0.8547	0.9413	0.9327	0.9235	0.9280	0.9318	0.9321	0.9349	**0.9521**
6	Exactly-1	0.7061	0.6021	0.7306	0.7249	0.7471	**0.8531**	0.7481	0.7091	0.7197	0.8524
7	Exactly-2	0.6980	0.6345	0.6940	0.6929	0.6959	0.6527	0.7007	0.6985	0.6977	**0.7472**
8	StatlogH	0.7773	0.6948	0.7867	0.7768	0.7788	0.7583	0.7773	0.7595	0.7842	**0.8127**
9	IonosphereVS	0.8595	0.8538	0.8938	0.8682	0.8485	0.8639	0.8708	0.8890	0.8826	**0.9042**
10	KrvskpEW	0.9006	0.7603	0.9215	0.9143	0.9200	0.8580	0.9269	0.8929	0.8980	**0.9360**
11	Lymphography	0.7863	0.6931	0.8164	0.7629	0.7906	0.8613	0.7793	0.7736	0.7880	**0.8667**
12	M-of-n	0.8184	0.7033	0.7988	0.8272	0.8425	0.8689	0.8293	0.8361	0.8549	**0.9020**
13	Penglung	0.8072	0.7676	0.6721	0.8341	0.8140	0.8482	0.8268	0.8331	0.7951	**0.8745**
14	Semeion	0.9584	0.9461	0.9557	0.9471	0.9476	0.9480	0.9521	0.9449	0.9504	**0.9729**
15	SonarMR	0.8487	0.7936	**0.8750**	0.8622	0.8667	0.8614	0.8506	0.8449	0.8506	0.8634
16	Spectheart	0.7881	0.7507	0.8097	0.7846	0.7841	0.7643	0.8000	0.7826	0.7871	**0.8443**
17	3T Endgame	0.7587	0.6601	0.7609	0.7537	0.8622	0.8667	0.7564	0.7557	0.7546	**0.8983**
18	Vote	0.9258	0.8413	0.9333	0.9196	0.9258	0.9618	0.9227	0.9196	0.9200	**0.9695**
19	WaveformV2	0.7066	0.6150	0.6921	0.7096	0.7192	0.7827	0.7154	0.7091	0.7044	**0.7929**
20	Wine	0.9543	0.8652	0.9536	0.9476	0.9521	0.9474	0.9551	0.9506	0.9566	**0.9843**
21	Zoology	0.9216	0.8131	0.9294	**0.9525**	0.9451	0.8827	0.9359	0.9476	0.9307	**0.9525**
	AVG.	0.8499	0.7827	0.8517	0.8530	0.8602	0.8657	0.8563	0.8506	0.8533	0.9011

The **bolded** values represent the best outcomes.

Table 14 compares the average number and ratio of features selected by different algorithms. Both tables show that TFSSA outperforms the other algorithms in the 13 datasets. Although the number of features chosen by TFSSA is not optimal in the other eight datasets, it is not significantly different from other outperformed methods. Figure 7 shows the population's average number of features and proportions chosen by the algorithm. The experiment shows that the average number of features and ratios selected by TFSSA in all datasets ranks first, with 30.97 and 0.468, respectively. Although the advantage is not apparent, it can prove that TFSSA outperforms other algorithms in most datasets to ensure high classification accuracy. In analyzing algorithm performance, we want to pay more attention to the classification average accuracy and the average number of features.

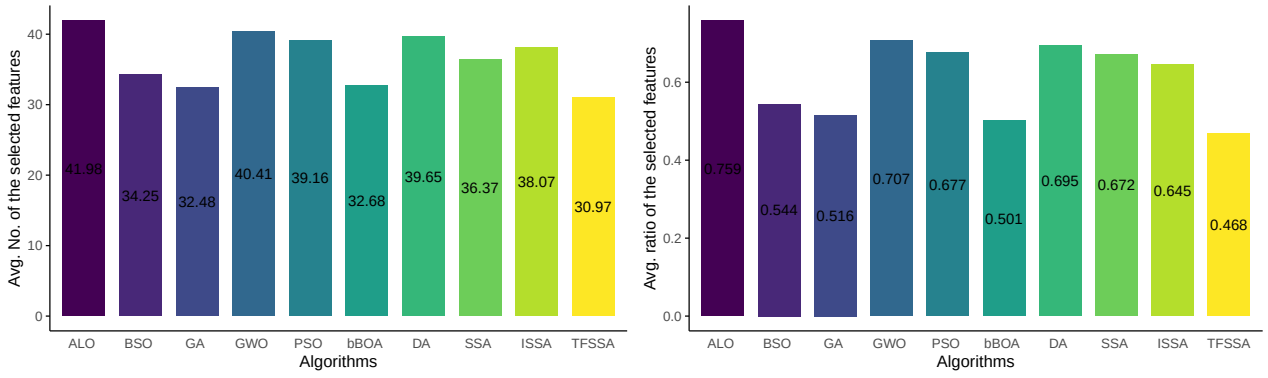

Figure 7. Comparison among algorithms' total average number of features and the selected feature ratio.

Table 14. Comparison of the selected average No. of features (AVG.NOF.) and the selected feature ratio (AVG_Ri) of each algorithm.

No.	Dataset	ALO	BSO	GA	GWO	PSO	bBOA	DA	SSA	ISSA	TFSSA
		AVG.NOF. (AVG_Ri.)	**AVG.NOF. (AVG_Ri.)**	**AVG.NOF. (AVG_Ri.)**	**AVG.NOF.** (AVG_Ri.)	**AVG.NOF.** (AVG_Ri.)	**AVG.NOF. (AVG_Ri.)**	**AVG.NOF. (AVG_Ri.)**	**AVG.NOF. (AVG_Ri.)**	**AVG.NOF. (AVG_Ri.)**	**AVG.NOF. (AVG_Ri.)**
1	BreastCO	7.00 (0.778)	6.40 (0.711)	6.10 (0.678)	6.90 (0.767)	5.70 (0.633)	5.60 (0.622)	6.27 (0.697)	7.20 (0.800)	5.70 (0.633)	**4.40 (0.489)**
2	BreastCWD	24.27 (0.809)	13.73 (0.458)	12.20 (0.407)	19.00 (0.633)	18.33 (0.611)	16.80 (0.560)	20.00 (0.667)	18.27 (0.609)	20.47 (0.682)	**8.40 (0.280)**
3	Clean-1	132.00 (0.795)	98.73 (0.595)	98.90 (0.596)	109.60 (0.660)	104.93 (0.632)	91.80 (0.553)	109.67 (0.661)	94.87 (0.572)	**90.20 (0.543)**	90.27 (0.544)
4	Clean-2	95.00 (0.572)	101.00 (0.608)	94.10 (0.567)	106.00 (0.639)	109.40 (0.659)	92.40 (0.557)	100.40 (0.605)	90.40 (0.545)	92.40 (0.557)	**90.28 (0.544)**
5	CongressVR	9.87 (0.617)	7.53 (0.471)	7.10 (0.444)	9.80 (0.613)	10.80 (0.675)	6.40 (0.400)	10.87 (0.679)	8.40 (0.525)	9.00 (0.563)	**6.15 (0.384)**
6	Exactly-1	12.87 (0.990)	7.73 (0.595)	8.10 (0.623)	12.07 (0.928)	9.00 (0.692)	7.60 (0.585)	10.53 (0.810)	12.80 (0.985)	10.47 (0.805)	**6.48 (0.498)**
7	Exactly-2	8.40 (0.646)	6.27 (0.482)	7.10 (0.546)	7.53 (0.579)	9.40 (0.723)	4.80 (0.369)	8.67 (0.667)	6.27 (0.482)	9.00 (0.692)	**4.62 (0.355)**
8	StatlogH	10.40 (0.800)	6.60 (0.508)	6.60 (0.508)	8.80 (0.677)	9.07 (0.698)	5.80 (0.446)	9.60 (0.738)	7.47 (0.575)	8.47 (0.652)	**4.86 (0.374)**
9	IonosphereVS	20.13 (0.592)	15.93 (0.469)	**13.50 (0.397)**	17.33 (0.510)	19.20 (0.565)	16.20 (0.476)	18.00 (0.529)	19.67 (0.579)	19.07 (0.561)	17.14 (0.504)
10	KrvskpEW	35.80 (0.994)	17.80 (0.494)	18.00 (0.500)	31.60 (0.878)	25.60 (0.711)	17.60 (0.489)	28.60 (0.794)	29.40 (0.817)	20.80 (0.578)	**16.91 (0.470)**
11	Lymphography	13.33 (0.741)	9.47 (0.526)	8.90 (0.494)	11.80 (0.656)	11.73 (0.652)	**8.40 (0.467)**	12.53 (0.696)	12.20 (0.678)	8.87 (0.493)	9.17 (0.509)
12	M-of-n	11.27 (0.867)	6.90 (0.531)	7.68 (0.591)	11.27 (0.867)	10.87 (0.836)	6.80 (0.523)	12.13 (0.933)	12.33 (0.948)	10.67 (0.821)	**6.30 (0.485)**
13	Penglung	172.07 (0.529)	160.60 (0.494)	**153.00 (0.471)**	162.80 (0.501)	183.33 (0.564)	172.00 (0.529)	175.20 (0.539)	162.33 (0.499)	182.67 (0.562)	161.42 (0.497)
14	Semeion	187.80 (0.709)	162.00 (0.611)	149.40 (0.564)	203.60 (0.768)	171.60 (0.648)	143.20 (0.540)	193.00 (0.728)	161.80 (0.611)	194.40 (0.734)	**142.38 (0.537)**
15	SonarMR	48.00 (0.800)	30.60 (0.510)	30.30 (0.505)	41.60 (0.693)	37.60 (0.627)	32.80 (0.547)	29.40 (0.490)	34.13 (0.569)	37.13 (0.619)	**22.36 (0.373)**
16	Spectheart	13.87 (0.630)	10.87 (0.494)	**7.00 (0.318)**	13.20 (0.600)	12.07 (0.549)	10.80 (0.491)	14.67 (0.667)	11.33 (0.515)	9.60 (0.436)	8.60 (0.391)
17	3T Endgame	8.80 (0.978)	5.88 (0.653)	5.80 (0.644)	7.53 (0.837)	6.73 (0.748)	5.60 (0.622)	7.20 (0.800)	8.07 (0.897)	7.47 (0.830)	**5.29 (0.588)**
18	Vote	8.40 (0.525)	7.87 (0.492)	5.80 (0.363)	8.47 (0.529)	9.33 (0.583)	**5.20 (0.325)**	8.87 (0.554)	8.53 (0.533)	9.60 (0.600)	8.67 (0.542)
19	WaveformV2	39.60 (0.990)	29.00 (0.725)	30.40 (0.760)	36.60 (0.915)	35.80 (0.895)	25.00 (0.625)	36.00 (0.900)	37.20 (0.930)	34.40 (0.860)	**24.56 (0.614)**
20	Wine	11.07 (0.852)	6.67 (0.513)	6.73 (0.518)	10.73 (0.825)	10.07 (0.775)	**6.20 (0.477)**	9.53 (0.733)	9.07 (0.698)	9.40 (0.723)	6.34 (0.488)
21	Zoology	11.67 (0.729)	7.67 (0.479)	5.35 (0.334)	12.40 (0.775)	11.80 (0.738)	**5.20 (0.325)**	11.47 (0.717)	11.93 (0.746)	9.60 (0.600)	5.78 (0.361)
	AVG.	41.98 (0.759)	34.25 (0.544)	32.48 (0.516)	40.41 (0.707)	39.16 (0.677)	32.68 (0.501)	39.65 (0.695)	36.37 (0.672)	38.07 (0.645)	30.97 (0.468)

The **bolded** values represent the best outcomes.

As a result, the number of selected attributes affected by the classification accuracy value is often slightly less than the fitness value. Tables 15–18 present the statistical measures (best, worst, mean, and Std) obtained by different runs of the algorithm on each dataset. TFSSA has a lower fitness value than other algorithms, checking the results. Among them, the average fitness value of TFSSA maintains a leading edge in 17 datasets, and bBOA outperforms different algorithms in 5 datasets. The overall average fitness of TFSSA ranked first, with a value of 0.098. The best fitness value of TFSSA maintains the lead in all datasets except Exactly-1, and its overall best fitness value is 0.076, ranking first. The worst fitness value of TFSSA outperforms other algorithms in 17 datasets, bBOA outperforms different algorithms in 4 datasets, and GA outperforms other algorithms in dataset Clean-1. Table 18 shows that the standard deviation of TFSSA outperforms different algorithms in 21 datasets and Figure 8 compares the total average standard deviation for mean fitness values among algorithms, while the standard deviation of GA outperforms different algorithms in 8 datasets. The standard deviation of bBOA outperforms other algorithms in 2 datasets. Outperforming different algorithms, the standard deviation of GWO outperforms other algorithms in the dataset StatlogH.

Table 15. Comparison of the average fitness measure of each algorithm.

No.	Dataset	ALO	BSO	GA	GWO	PSO	bBOA	DA	SSA	ISSA	TFSSA
1	BreastCO	0.048	0.084	0.046	0.047	0.045	0.040	0.041	0.046	0.048	**0.032**
2	BreastCWD	0.068	0.102	0.055	0.068	0.068	**0.042**	0.068	0.067	0.071	0.045
3	Clean-1	0.160	0.177	0.134	0.147	0.150	0.113	0.151	0.147	0.160	**0.108**
4	Clean-2	0.055	0.065	0.062	0.060	0.060	0.051	0.057	0.054	0.058	**0.041**
5	CongressVR	0.069	0.149	0.063	0.073	0.082	0.045	0.074	0.042	0.072	**0.035**
6	Exactly-1	0.301	0.400	0.270	0.282	0.257	**0.040**	0.257	0.286	0.298	0.229
7	Exactly-2	0.305	0.367	0.308	0.310	0.308	0.260	0.303	0.306	0.303	**0.240**
8	StatlogH	0.228	0.307	0.216	0.228	0.226	**0.180**	0.228	0.220	0.244	0.185
9	IonosphereVS	0.145	0.149	0.109	0.136	0.124	0.096	0.133	0.122	0.116	**0.081**
10	KrvskpEW	0.108	0.242	0.083	0.094	0.086	0.054	0.080	0.140	0.116	**0.044**
11	Lymphography	0.219	0.309	0.187	0.241	0.214	0.189	0.225	0.216	0.231	**0.109**
12	M-of-n	0.188	0.299	0.205	0.180	0.164	0.027	0.178	0.152	0.172	**0.024**
13	Penglung	0.196	0.235	0.129	0.169	0.190	0.118	0.177	0.209	0.170	**0.106**
14	Semeion	0.045	0.049	0.039	0.050	0.059	0.036	0.055	0.057	0.052	**0.021**
15	SonarMR	0.158	0.209	0.128	0.143	0.138	0.086	0.155	0.154	0.159	**0.079**
16	Spectheart	0.216	0.252	0.192	0.219	0.219	0.160	0.205	0.217	0.220	**0.120**
17	3T Endgame	0.249	0.342	0.243	0.252	0.253	**0.205**	0.249	0.251	0.251	0.219
18	Vote	0.079	0.162	0.070	0.085	0.079	0.044	0.082	0.085	0.085	**0.037**
19	WaveformV2	0.300	0.386	0.319	0.297	0.287	0.265	0.291	0.301	0.298	**0.254**
20	Wine	0.054	0.139	0.051	0.060	0.055	**0.023**	0.052	0.050	0.056	**0.023**
21	Zoology	0.085	0.190	0.073	0.055	0.062	0.034	0.071	0.075	0.059	**0.021**
	AVG.	0.156	0.220	0.142	0.152	0.149	0.100	0.149	0.152	0.154	0.098

The **bolded** values represent the best outcomes.

Table 16. Comparison of the best fitness measure of each algorithm.

No.	Dataset	ALO	BS0	GA	GWO	PSO	bBOA	DA	SSA	ISSA	TFSSA
1	BreastCO	0.038	0.046	0.040	0.038	0.039	0.024	0.031	0.038	0.038	**0.022**
2	BreastCWD	0.066	0.065	0.048	0.048	0.049	0.032	0.051	0.052	0.059	**0.029**
3	Clean-1	0.118	0.130	0.122	0.117	0.100	0.088	0.118	0.100	0.122	**0.074**
4	Clean-2	0.049	0.058	0.062	0.056	0.058	0.037	0.050	0.052	0.054	**0.033**
5	CongressVR	0.044	0.076	0.054	0.042	0.048	0.030	0.035	0.045	0.041	**0.026**
6	Exactly-1	0.267	0.328	0.015	0.173	0.138	**0.005**	0.155	0.089	0.229	0.224
7	Exactly-2	0.252	0.296	0.295	0.279	0.275	0.225	0.238	0.237	0.270	**0.221**
8	StatlogH	0.172	0.206	0.202	0.189	0.178	0.138	0.159	0.163	0.194	**0.134**
9	IonosphereVS	0.111	0.101	0.099	0.088	0.081	0.060	0.104	0.092	0.078	**0.056**
10	KruskpLW	0.093	0.133	0.063	0.090	0.052	0.036	0.062	0.084	0.111	**0.032**
11	Lymphography	0.165	0.220	0.168	0.193	0.179	0.183	0.166	0.168	0.169	**0.064**
12	M-of-n	0.160	0.170	0.140	0.128	0.064	0.005	0.157	0.101	0.035	**0.003**
13	Penglung	0.085	0.085	0.137	0.085	0.086	0.033	0.035	0.062	0.112	**0.029**
14	Semeion	0.041	0.046	0.033	0.044	0.042	0.029	0.040	0.047	0.045	**0.020**
15	SonarMR	0.128	0.139	0.109	0.090	0.091	0.072	0.113	0.081	0.129	**0.069**
16	Spectheart	0.144	0.198	0.170	0.149	0.166	0.122	0.142	0.159	0.173	**0.118**
17	3T Endgame	0.213	0.252	0.232	0.223	0.204	0.195	0.217	0.219	0.213	**0.183**
18	Vote	0.043	0.065	0.061	0.051	0.039	0.016	0.051	0.060	0.050	**0.012**
19	WaveformLW	0.294	0.338	0.312	0.283	0.271	0.254	0.278	0.291	0.291	**0.250**
20	Wine	0.029	0.061	0.038	0.019	0.028	0.005	0.031	0.028	0.016	**0.003**
21	Zoology	0.026	0.025	0.061	0.007	0.008	0.002	0.007	0.009	0.026	**0.002**
	AVG.	0.121	0.145	0.117	0.114	0.105	0.076	0.107	0.104	0.117	0.076

The **bolded** values represent the best outcomes.

Table 17. Comparison of the worst fitness measure of each algorithm.

No.	Dataset	ALO	BSO	GA	GWO	PSO	bBOA	DA	SSA	ISSA	TFSSA
1	BreastCO	0.059	0.196	0.051	0.054	0.06	0.041	0.059	0.056	0.058	**0.036**
2	BreastCWD	0.083	0.144	0.063	0.085	0.078	0.049	0.09	0.095	0.088	**0.049**
3	Clean-1	0.193	0.208	**0.143**	0.187	0.186	**0.138**	0.178	0.214	0.200	0.153
4	Clean-2	0.060	0.073	0.071	0.063	0.061	0.068	0.059	0.057	0.064	**0.043**
5	CongressVR	0.110	0.267	0.083	0.110	0.149	0.058	0.107	0.096	0.120	**0.053**
6	Exactly-1	0.343	0.448	0.378	0.344	0.384	**0.115**	0.319	0.375	0.335	0.285
7	Exactly-2	0.355	0.517	0.331	0.333	0.335	0.291	0.330	0.337	0.363	**0.287**
8	StatlogH	0.289	0.378	0.261	0.256	0.288	0.195	0.284	0.277	0.299	**0.191**
9	IonosphereVS	0.168	0.195	0.134	0.179	0.163	0.118	0.157	0.155	0.157	**0.114**
10	KruskpLW	0.118	0.344	0.150	0.096	0.164	0.064	0.097	0.176	0.121	**0.060**
11	Lymphography	0.251	0.378	0.220	0.299	0.276	0.194	0.303	0.261	0.299	**0.146**
12	M-of-n	0.224	0.391	0.288	0.235	0.287	**0.110**	0.210	0.236	0.212	0.206
13	Penglung	0.300	0.460	0.190	0.246	0.328	0.169	0.326	0.379	0.273	**0.153**
14	Semeion	0.049	0.056	0.043	0.064	0.072	0.049	0.070	0.077	0.065	**0.025**
15	SonarMR	0.216	0.253	0.156	0.218	0.187	0.109	0.217	0.198	0.214	**0.103**
16	Spectheart	0.271	0.322	0.218	0.265	0.265	0.209	0.252	0.271	0.262	**0.201**
17	3T Endgame	0.275	0.436	0.255	0.309	0.331	**0.216**	0.293	0.307	0.293	0.225
18	Vote	0.113	0.256	0.088	0.169	0.119	0.057	0.124	0.138	0.118	**0.056**
19	WaveformV2	0.304	0.434	0.319	0.316	0.303	0.265	0.299	0.313	0.305	**0.259**
20	Wine	0.075	0.303	0.082	0.142	0.075	0.028	0.086	0.077	0.076	**0.026**
21	Zoology	0.158	0.430	0.101	0.203	0.182	0.048	0.125	0.181	0.107	**0.039**
	AVG.	0.191	0.309	0.173	0.199	0.204	0.123	0.190	0.204	0.192	0.129

The **bolded** values represent the best outcomes.

Table 18. Comparison of the standard deviation fitness measure of each algorithm.

No.	Dataset	ALO	BSO	GA	GWO	PSO	bBOA	DA	SSA	ISSA	TFSSA
1	BreastCO	0.008	0.023	0.005	0.011	0.008	0.006	0.011	0.013	0.009	**0.005**
2	BreastCWD	0.007	0.044	0.003	0.006	0.006	0.003	0.010	0.005	0.005	**0.003**
3	Clean-1	0.017	0.048	**0.008**	0.019	0.026	0.010	0.021	0.016	0.020	0.018
4	Clean-2	0.023	0.036	0.136	0.051	0.067	0.012	0.038	0.066	0.025	**0.010**
5	CongressVR	0.030	0.066	0.014	0.013	0.019	0.020	0.022	0.023	0.025	**0.011**
6	Exactly-1	0.031	0.051	0.020	0.023	0.029	**0.011**	0.028	0.035	0.028	0.020
7	Exactly-2	0.019	0.023	**0.012**	0.022	0.023	0.059	0.015	0.019	0.021	0.054
8	StatlogH	0.009	0.094	0.033	**0.002**	0.045	0.008	0.014	0.037	0.005	0.011
9	IonosphereVS	0.026	0.049	0.016	0.030	0.028	0.014	0.041	0.032	0.042	**0.010**
10	KrvskpEW	0.020	0.074	0.054	0.030	0.058	0.033	0.018	0.035	0.049	**0.031**
11	Lymphography	0.025	0.037	**0.013**	0.040	0.029	0.018	0.023	0.030	0.021	0.018
12	M-of-n	0.035	0.036	0.016	0.031	0.027	0.035	0.030	0.031	0.025	**0.015**
13	Penglung	0.020	0.053	**0.006**	0.026	0.034	0.007	0.023	0.025	0.021	0.018
14	Semeion	0.019	0.057	0.009	0.029	0.019	0.010	0.019	0.024	0.020	**0.008**
15	SonarMR	0.004	0.088	0.003	0.012	0.012	0.001	0.009	0.009	0.006	**0.001**
16	Spectheart	0.012	0.067	0.011	0.033	0.012	**0.010**	0.013	0.014	0.015	0.012
17	3T Endgame	0.035	0.128	**0.015**	0.047	0.050	0.044	0.037	0.047	0.028	0.041
18	Vote	0.021	0.026	**0.009**	0.022	0.028	0.016	0.018	0.029	0.022	0.015
19	WaveformV2	0.004	0.005	0.004	0.002	0.001	0.001	0.003	0.002	0.004	**0.001**
20	Wine	0.072	0.102	**0.018**	0.046	0.077	0.056	0.077	0.093	0.052	**0.018**
21	Zoology	0.007	0.002	**0.001**	0.005	0.002	0.004	0.003	0.006	0.004	0.002

The **bolded** values represent the best outcomes.

The average execution time of each method in the experiment is shown in Table 19. Because almost all optimization algorithms employ the same amount of iterations, the computation time can be used to compare algorithm performance. We receive the following observations from Table 19. The ten EAs have intimate performances regarding the time consumption for all 21 datasets. As we all know, an EA-based feature selection technique requires a classifier to evaluate an individual. The time it takes the classifier to assess a set of features and samples is usually proportional to the number of features and samples. Therefore, the fitness function is the most time-consuming part of EA-based feature selection algorithms for datasets with many features or/and models, such as WaveformV2, Clean-2, and Semeion. The 10 EA-based algorithms used in the trials all had the same maximum number of evaluations as their termination conditions, which resulted in identical time consumption. Among them, TFSSA has the best computing time on seven

datasets. In comparison, GWO performs better on five datasets, and GA performs better than other optimizers on six datasets, DA, SSA, and ISSA each have a better performance on one dataset.

Table 19. Comparison of the running time of each algorithm.

No.	Dataset	ALO	BSO	GA	GWO	PSO	bBOA	DA	SSA	ISSA	TFSSA
1	BreastCO	4.90	3.02	2.34	3.45	2.39	2.48	2.36	2.32	2.45	**2.27**
2	BreastCWD	2.87	2.85	2.37	3.61	2.43	2.82	2.41	2.36	2.35	**2.32**
3	Clean-1	5.31	3.58	4.07	**3.39**	3.66	4.05	3.61	3.50	3.54	3.84
4	Clean-2	310.83	223.69	279.32	**158.67**	227.16	171.84	223.70	173.07	182.94	159.32
5	CongressVR	2.88	3.33	2.81	3.32	2.64	3.03	2.59	2.61	2.72	**2.59**
6	Exactly-1	3.92	4.58	**3.85**	4.04	4.53	4.06	4.63	5.18	4.65	4.96
7	Exactly-2	4.22	4.62	**4.15**	4.52	4.82	4.60	4.88	4.20	4.22	4.20
8	StatlogH	2.69	2.96	**2.46**	3.09	2.49	2.78	2.52	2.62	2.47	2.50
9	IonosphereVS	3.14	3.10	2.58	3.25	2.64	2.96	2.60	2.54	2.57	**2.47**
10	KrvskpEW	18.16	11.56	10.42	**9.53**	17.16	13.84	15.89	13.89	13.03	12.07
11	Lymphography	2.68	2.94	2.46	2.98	3.04	2.82	**2.38**	2.87	2.91	2.69
12	M-of-n	4.08	4.07	3.53	**3.39**	4.56	4.04	3.74	4.26	4.14	4.19
13	Penglung	7.65	3.10	2.51	2.49	2.56	4.13	2.55	2.50	2.50	**2.45**
14	Semeion	28.41	14.33	**13.10**	31.67	24.51	19.92	24.06	21.82	19.21	15.45
15	SonarMR	3.30	2.93	**2.39**	2.97	2.62	2.92	2.72	2.75	2.59	2.45
16	Spectheart	2.88	2.96	2.45	3.00	2.40	2.80	2.38	2.40	**2.38**	2.29
17	3T Endgame	4.36	3.99	**3.28**	4.38	4.49	3.91	4.38	4.25	4.10	4.45
18	Vote	2.89	3.26	2.62	3.25	2.57	2.82	2.60	2.53	2.62	**2.47**
19	WaveformV2	40.51	25.03	23.48	**20.63**	27.09	35.56	43.72	34.14	36.64	21.26
20	Wine	2.68	2.92	2.46	3.13	2.45	2.68	2.43	**2.43**	2.47	2.52
21	Zoology	2.79	4.85	2.33	3.25	2.24	2.66	2.30	2.21	2.19	**2.15**

In addition, Table 20 shows the Wilcoxon rank-sum test p-values at the 5% significance level for the Wilcoxon rank-sum test. A p-value of less than 0.05 implies that the null hypothesis of no meaningful difference at the 5% level is rejected. The p-values in Table 20 confirm that the results of TFSSA are significantly different from those of classical and top-of-the-line algorithms on most datasets. Specifically, in 12 datasets, the performance is outstanding, including BreastCWD, Clean-2, Exactly-1, Exactly-2, StatlogH, Lymphography, M-of-n, SonarMR, Spectheart, 3T Endgame, Vote, and Wine.

Table 20. p-values of the Wilcoxon test of TFSSA vs. others.

	Dataset	ALO	BSO	GA	GWO	PSO	bBOA	DA	SSA	ISSA	TFSSA
1	BreastCO	1.09×10^{-2}	5.06×10^{-3}	5.03×10^{-3}	6.91×10^{-3}	1.09×10^{-2}	1.14×10^{-1}	6.87×10^{-3}	5.06×10^{-3}	7.63×10^{-3}	6.11×10^{-4}
2	BreastCWD	6.48×10^{-4}	6.23×10^{-4}	6.32×10^{-4}	6.49×10^{-4}	6.43×10^{-4}	6.58×10^{-4}	7.12×10^{-4}	6.58×10^{-4}	6.45×10^{-4}	7.62×10^{-3}
3	Clean-1	6.58×10^{-4}	9.85×10^{-4}	2.15×10^{-3}	8.01×10^{-4}	2.15×10^{-3}	8.03×10^{-4}	1.79×10^{-3}	6.53×10^{-4}	4.51×10^{-3}	7.21×10^{-1}
4	Clean-2	1.03×10^{-2}	6.23×10^{-4}	6.23×10^{-4}	6.23×10^{-4}	6.23×10^{-4}	2.07×10^{-3}	6.23×10^{-4}	6.23×10^{-4}	6.23×10^{-4}	6.11×10^{-4}
5	CongressVR	9.85×10^{-4}	6.58×10^{-4}	1.19×10^{-3}	1.21×10^{-3}	6.58×10^{-4}	4.51×10^{-3}	2.16×10^{-3}	1.47×10^{-3}	4.51×10^{-3}	3.44×10^{-1}
6	Exactly-1	6.52×10^{-4}	6.58×10^{-4}	6.42×10^{-4}	6.48×10^{-4}	6.58×10^{-4}	6.48×10^{-5}	6.58×10^{-4}	6.52×10^{-4}	6.58×10^{-4}	5.39×10^{-3}
7	Exactly-2	9.87×10^{-4}	6.58×10^{-4}	6.47×10^{-4}	6.58×10^{-4}	8.05×10^{-4}	1.21×10^{-3}	8.05×10^{-4}	8.05×10^{-4}	8.98×10^{-3}	8.86×10^{-3}
8	StatlogH	9.87×10^{-4}	6.53×10^{-4}	6.47×10^{-4}	9.79×10^{-4}	6.53×10^{-4}	8.05×10^{-4}	6.41×10^{-3}	6.53×10^{-4}	7.59×10^{-3}	8.86×10^{-3}
9	IonosphereVS	3.09×10^{-2}	2.31×10^{-2}	8.20×10^{-1}	7.83×10^{-2}	3.34×10^{-1}	6.09×10^{-2}	3.07×10^{-1}	5.32×10^{-1}	4.60×10^{-1}	5.23×10^{-2}
10	KrvskpEW	4.35×10^{-2}	4.31×10^{-2}	7.96×10^{-2}	4.31×10^{-2}	7.96×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.19×10^{-2}
11	Lymphography	6.58×10^{-4}	6.58×10^{-4}	6.47×10^{-4}	6.58×10^{-4}	6.58×10^{-4}	6.58×10^{-4}	6.52×10^{-4}	6.58×10^{-4}	6.58×10^{-4}	6.11×10^{-4}
12	M-of-n	6.58×10^{-4}	6.58×10^{-4}	6.53×10^{-4}	6.53×10^{-4}	6.58×10^{-4}	6.58×10^{-4}	6.58×10^{-4}	6.53×10^{-4}	6.58×10^{-4}	6.43×10^{-4}
13	Penglung	1.71×10^{-2}	3.14×10^{-3}	1.40×10^{-1}	8.83×10^{-2}	3.09×10^{-2}	7.83×10^{-2}	3.56×10^{-2}	2.68×10^{-2}	4.95×10^{-1}	3.02×10^{-1}
14	semeion	1.38×10^{-1}	4.31×10^{-2}	7.96×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	5.31×10^{-2}
15	SonarMR	6.58×10^{-4}	6.58×10^{-4}	6.53×10^{-4}	6.58×10^{-4}	1.79×10^{-3}	6.58×10^{-4}	6.58×10^{-4}	6.53×10^{-4}	6.58×10^{-4}	6.43×10^{-4}
16	Spectheart	1.25×10^{-2}	6.58×10^{-4}	4.48×10^{-3}	6.58×10^{-4}	6.58×10^{-4}	5.37×10^{-3}	1.79×10^{-3}	6.58×10^{-4}	1.46×10^{-2}	4.59×10^{-2}
17	3T Endgame	6.50×10^{-4}	6.58×10^{-4}	6.53×10^{-4}	6.58×10^{-4}	8.03×10^{-4}	6.58×10^{-4}	6.58×10^{-4}	6.53×10^{-4}	8.05×10^{-4}	1.46×10^{-2}
18	Vote	9.85×10^{-4}	6.53×10^{-4}	6.47×10^{-4}	9.87×10^{-4}	6.53×10^{-4}	6.58×10^{-4}	6.58×10^{-4}	6.58×10^{-4}	6.53×10^{-4}	2.30×10^{-2}
19	WaveformV2	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	4.31×10^{-2}	2.23×10^{-1}	5.24×10^{-2}
20	Wine	6.48×10^{-4}	6.53×10^{-4}	6.50×10^{-4}	3.77×10^{-3}	8.03×10^{-4}	6.52×10^{-4}	8.01×10^{-4}	6.48×10^{-4}	1.19×10^{-3}	4.19×10^{-3}
21	Zoology	3.56×10^{-2}	3.14×10^{-3}	4.67×10^{-2}	6.09×10^{-1}	3.07×10^{-1}	1.40×10^{-1}	6.91×10^{-2}	3.63×10^{-1}	9.55×10^{-1}	2.60×10^{-1}

The $p \geq 0.05$ are underlined.

Overall, the results in Tables 13–18 show that TFSSA can balance exploration and exploitation in the optimization search process. This experiment employed four large datasets: Clean-2 (No. 4), krvskpEW (No. 10), Penglung (No. 13), and Semeion (No. 14). The results indicate that TFSSA outperforms other algorithms in both small and large datasets. TFSSA outperforms all algorithms in classification average accuracy, selected average feature number, chose average feature rate, measures of fitness (best, worst, mean, and Std), and the Wilcoxon rank-sum test.

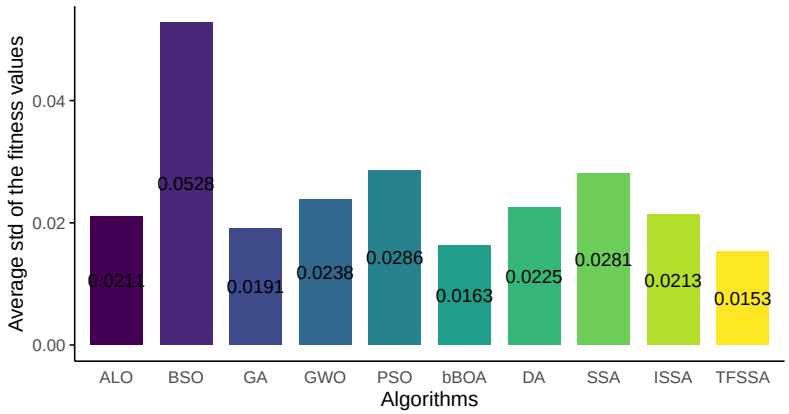

Figure 8. Comparison of total average standard deviation for mean fitness values among algorithms.

We can conclude from all of these experiments that employing the improved Tent chaos, LF strategy, and self-adaptive hyper-parameters improves the robustness and performance of the proposed algorithm. This method solves FS difficulties by combining global search algorithms (suitable for exploration) and local search algorithms (suitable for development). Establishing a balance between exploration and production in the FS problem is critical to avoiding many local solutions and discovering an accurate approximation of the optimal solution. This is the primary reason behind TFSSA's improved performance compared to the comparative algorithm used in this study. TFSSA has the fewest features and the highest accuracy among the ten approaches. However, compared to the other methods utilized in this study, TFSSA takes more calculation time. Another drawback of the suggested random wrapper-based FS strategy is the imprecision with which the optimization results can be repeated. The algorithm's subset of features selected for different applications has been noted, which may mislead users when determining which subset to evaluate.

6. Real-World Dataset Instances

COVID-19 is an infectious disease caused by SARS-CoV-2, which has led to an epidemic that has continued to this day and has become one of the epidemics with the most significant number of deaths in human history [106]. The first known patient with the disease was diagnosed in Wuhan, Hubei Province, China, at the end of 2019 (although the disease is likely to have infected humans before). Since then, the disease has been detected worldwide and is still spreading. At the same time, humanity hopes to defeat the virus through various technologies, so it has once again started a protracted war against the virus. According to research, Artificial Intelligence (AI) has become a weapon with great potential to fight SARS-CoV-2 [107].

This section employs the proposed TFSSA for 2019 Coronavirus Disease patient health prediction, as shown in Figure 9. The dataset of COVID-19 patients (https://github.com/yyy24601/Covid-19-Patient-Health-Analytics, accessed on 20 January 2023) was gathered completely from [108]. Tables 21 and 22 give a summary of the real-world datasets used. This study aimed to predict illness and health based on a given variable. First, the 15 attributes are then translated into numerical numbers. Then, dividing the data into two groups: the training set and the test set, with a ratio of 8:2.

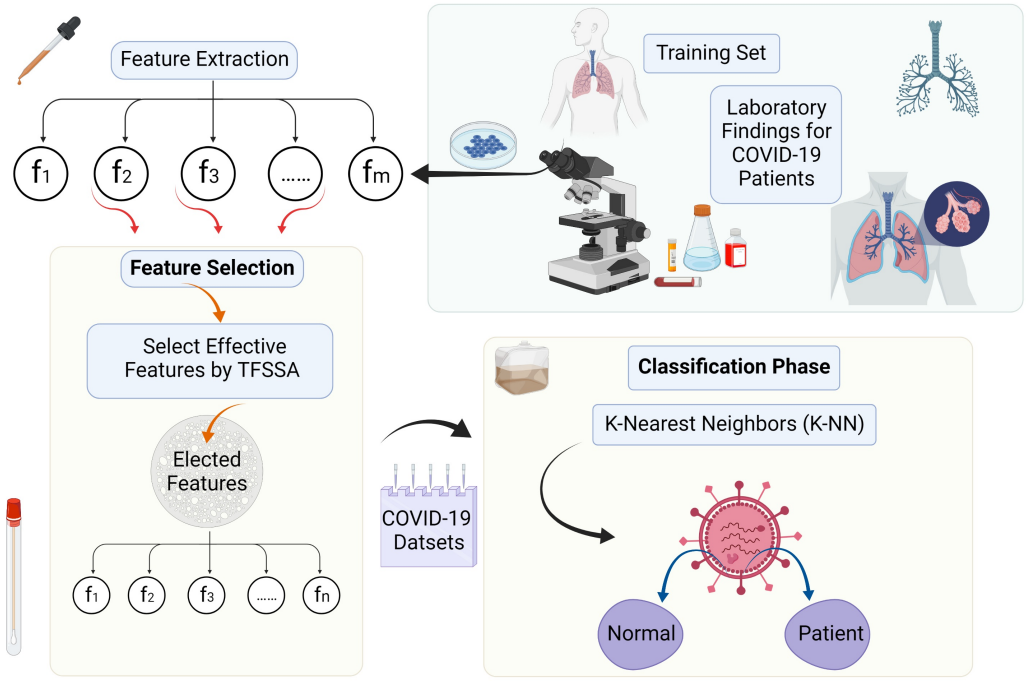

Figure 9. The proposed TFSSA classification strategy for COVID-19.

Table 21. COVID-19 dataset description.

Dateset	No. Features	No. Instances	Area
COVID-19	15	1085	Medical

Table 22. The description of the 2019 Coronavirus Disease dataset.

No.	Features	Feature Description
1	code(id)	Patients' identification numbers
2	location	The place where patients are situated
3	nationality	The country from which the patients come
4	gender	The patients' gender
5	age	How old patients are
6	sym_on	When people first show symptoms
7	hosp_vis	The date patients visit hospital
8	vis_wuhan	Whether or not the patients visited Wuhan, CN
9	from_wuhan	Whether or not the patients from Wuhan, CN
10	symptom_1	One of the symptoms encountered by patients
11	symptom_2	One of the symptoms encountered by patients
12	symptom_3	One of the symptoms encountered by patients
13	symptom_4	One of the symptoms encountered by patients
14	symptom_5	One of the symptoms encountered by patients
15	symptom_6	One of the symptoms encountered by patients

As can be seen from Figure 10, TFSSA achieves the highest average classification accuracy of 93.47% and the lowest average feature selection number of 2.1. On the other hand, the results reveal that for TFSSA inpatient health prediction, around three features were sufficient. According to the results, the most popular features were id, age, and nationality. The list of features selected by all FS algorithms is shown in Table 23, where the selected features are the main features selected by all FS algorithms in all experiments, and the features not shown in the table are the features that are eliminated. Furthermore, the data suggest that the TFSSA algorithm has never chosen symptom_4, symptom_5, or symptom_6. Further, to validate TFSSA's classification performance, we try to remove symptoms 4, 5, and 6, and the difference is minor compared to previous experimental findings. As a result, these features cannot appropriately detect the data pattern in the

patient health prediction process. The performance of TFSSA is observed after eliminating these characteristics, and the classification accuracy is barely affected. To continue studying the performance of TFSSA, we remove the original feature (ID) from the dataset. The experiment revealed that the classification average accuracy is about 91.3%. The researchers said that in the future, more abundant, detailed, and comprehensive clinical features should be collected to more accurately predict the health status of patients.

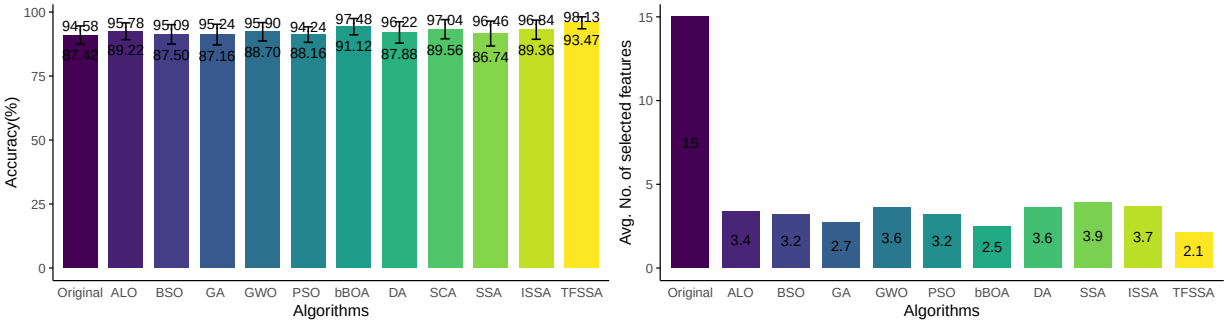

Figure 10. Accuracy rating and feature size of TFSSA on the COVID-19 dataset.

Table 23. The list of features selected by all FS algorithms.

Algorithm	id	Age	Nationality	sym_on	from_wuhan
ALO	✓	✓	✓		
BSO	✓	✓	✓		
GA	✓	✓	✓		
GWO	✓	✓	✓		
PSO	✓	✓	✓		
bBOA		✓	✓		
DA	✓	✓	✓	✓	
SSA	✓	✓	✓		✓
ISSA	✓	✓	✓		✓
TFSSA		✓	✓		

7. Conclusions

In this paper, we propose a TFSSA that mainly combines a Tent chaotic map, LF, and self-adaptive hyper-parameters to solve the optimization problems. First, we test the performance of TFSSA using the scientific standard test function—the CEC2020 benchmark function and compare it with seven methods in multiple aspects. Second, TFSSA combines a K-NN classifier to solve the FS problem in wrapper-based mode. Twenty-one datasets from the UC Irvine Machine Learning Repository are utilized to validate the proposed method's performance. In addition, the method is also applied to the diagnosis and prediction of COVID-19. Nine criteria are reported to evaluate each technique: classification average accuracy, average selection size, average selection rate, measures of fitness (best, worst, mean, and Std), computation time, and rank-sum test. Comparing TFSSA with five top-of-the-line methods (BSO, ALO, PSO, GWO, and GA) and the four latest high-performance methods (bBOA, DA, SSA, and ISSA), the experimental results show TFSSA achieves the goal of lowering the number of features and boosting the model's accuracy by removing as many irrelevant and redundant features as possible. Therefore, TFSSA can find the best feature subset and obtain high accuracy when applied to various FS tasks. During the experiments, we also found that multiple sophisticated initialization processes can be employed in TFSSA to improve the speed. How to strengthen multiple advanced initialization procedures will be our future work.

Author Contributions: Conceptualization, Q.Y. and Y.G.; methodology, Q.Y. and Y.G.; software, Q.Y.; validation, Q.Y., Y.G. and Y.S.; formal analysis, Y.G. and Y.S.; investigation, Y.G.; resources, Y.G.; data curation, Q.Y. and Y.G.; writing—original draft preparation, Q.Y., Y.G. and Y.S.; writing—review and editing, Q.Y., Y.G. and Y.S.; visualization, Q.Y., Y.G. and Y.S.; supervision, Y.G. and Y.S.; project

administration, Y.G.; funding acquisition, Y.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Natural Science Foundation of Key Project in Ningxia, China (No. 2022AAC02043), the National Natural Science Foundation of China (No. 11961001, No. 61561001), the Construction Project of First-class Subjects in Ningxia Higher Education, China (No. NXYLXK2017B09), the Major Proprietary Funded Project of North Minzu University, China (No. ZDZX201901), and Basic discipline research projects supported by Nanjing Securities (NJZQJCXK202201).

Data Availability Statement: Datasets related to this article can be found at (https://archive.ics.uci.edu/ml/datasets.php, accessed on 20 January 2023), (https://github.com/yyy24601/TFSSA, accessed on 20 January 2023) and (https://github.com/yyy24601/COVID-19, accessed on 20 January 2023).

Acknowledgments: We acknowledge the valuable comments from the anonymous reviewers. We would also like to thank the Editors for their generous comments and support during the review process.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Too, J.; Mirjalili, S. A hyper learning binary dragonfly algorithm for feature selection: A COVID-19 case study. *Knowl.-Based Syst.* **2021**, *212*, 106553. [CrossRef]
2. Frawley, W.J.; Piatetsky-Shapiro, G.; Matheus, C.J. Knowledge discovery in databases: An overview. *AI Mag.* **1992**, *13*, 57.
3. Cios, K.J.; Pedrycz, W.; Swiniarski, R.W. Data mining and knowledge discovery. In *Data Mining Methods for Knowledge Discovery*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 1–26.
4. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [CrossRef]
5. García, S.; Ramírez-Gallego, S.; Luengo, J.; Benítez, J.M.; Herrera, F. Big data preprocessing: Methods and prospects. *Big Data Anal.* **2016**, *1*, 9. [CrossRef]
6. Alasadi, S.A.; Bhaya, W.S. Review of data preprocessing techniques in data mining. *J. Eng. Appl. Sci.* **2017**, *12*, 4102–4107.
7. Mishra, P.; Biancolillo, A.; Roger, J.M.; Marini, F.; Rutledge, D.N. New data preprocessing trends based on ensemble of multiple preprocessing techniques. *TrAC Trends Anal. Chem.* **2020**, *132*, 116045. [CrossRef]
8. Kamiran, F.; Calders, T. Data preprocessing techniques for classification without discrimination. *Knowl. Inf. Syst.* **2012**, *33*, 1–33. [CrossRef]
9. Luengo, J.; García-Gil, D.; Ramírez-Gallego, S.; García, S.; Herrera, F. *Big Data Preprocessing*; Springer: Cham, Switzerland, 2020.
10. Shen, C.; Zhang, K. Two-stage improved Grey Wolf optimization algorithm for feature selection on high-dimensional classification. *Complex Intell. Syst.* **2021**, *8*, 2769–2789. [CrossRef]
11. Fu, W.; Wang, K.; Tan, J.; Zhang, K. A composite framework coupling multiple feature selection, compound prediction models and novel hybrid swarm optimizer-based synchronization optimization strategy for multi-step ahead short-term wind speed forecasting. *Energy Convers. Manag.* **2020**, *205*, 112461. [CrossRef]
12. Di Mauro, M.; Galatro, G.; Fortino, G.; Liotta, A. Supervised feature selection techniques in network intrusion detection: A critical review. *Eng. Appl. Artif. Intell.* **2021**, *101*, 104216. [CrossRef]
13. Kashef, S.; Nezamabadi-pour, H.; Nikpour, B. Multilabel feature selection: A comprehensive review and guiding experiments. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1240. [CrossRef]
14. Zheng, Q.; Yang, M.; Tian, X.; Jiang, N.; Wang, D. A full stage data augmentation method in deep convolutional neural network for natural image classification. *Discrete Dyn. Nat. Soc.* **2020**, *2020*, 4706576. [CrossRef]
15. Lee, C.Y.; Hung, C.H. Feature ranking and differential evolution for feature selection in brushless DC motor fault diagnosis. *Symmetry* **2021**, *13*, 1291. [CrossRef]
16. Li, J.; Gao, Y.; Wang, K.; Sun, Y. A dual opposition-based learning for differential evolution with protective mechanism for engineering optimization problems. *Appl. Soft Comput.* **2021**, *113*, 107942. [CrossRef]
17. Tsamardinos, I.; Charonyktakis, P.; Papoutsoglou, G.; Borboudakis, G.; Lakiotaki, K.; Zenklusen, J.C.; Juhl, H.; Chatzaki, E.; Lagani, V. Just Add Data: Automated predictive modeling for knowledge discovery and feature selection. *NPJ Precis. Oncol.* **2022**, *6*, 38. [CrossRef]
18. Song, Y.; Wei, L.; Yang, Q.; Wu, J.; Xing, L.; Chen, Y. RL-GA: A reinforcement learning-based genetic algorithm for electromagnetic detection satellite scheduling problem. *Swarm Evol. Comput.* **2023**, *77*, 101236. [CrossRef]
19. Guyon, I.; Elisseeff, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
20. Zhang, J.; Lin, Y.; Jiang, M.; Li, S.; Tang, Y.; Tan, K.C. Multi-label Feature Selection via Global Relevance and Redundancy Optimization. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Yokohama, Japan, 7–15 January 2020; pp. 2512–2518.

21. Xue, B.; Zhang, M.; Browne, W.N. Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. *Appl. Soft Comput.* **2014**, *18*, 261–276.
22. Diao, R.; Shen, Q. Nature inspired feature selection meta-heuristics. *Artif. Intell. Rev.* **2015**, *44*, 311–340.
23. Peng, H.; Long, F.; Ding, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1226–1238. [CrossRef]
24. Park, C.H.; Kim, S.B. Sequential random k-nearest neighbor feature selection for high-dimensional data. *Expert Syst. Appl.* **2015**, *42*, 2336–2342. [CrossRef]
25. Oh, I.S.; Lee, J.S.; Moon, B.R. Hybrid genetic algorithms for feature selection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 1424–1437. [PubMed]
26. Du, G.; Zhang, J.; Luo, Z.; Ma, F.; Ma, L.; Li, S. Joint imbalanced classification and feature selection for hospital readmissions. *Knowl.-Based Syst.* **2020**, *200*, 106020. [CrossRef]
27. Zhao, M.; Jha, A.; Liu, Q.; Millis, B.A.; Mahadevan-Jansen, A.; Lu, L.; Landman, B.A.; Tyska, M.J.; Huo, Y. Faster Mean-shift: GPU-accelerated clustering for cosine embedding-based cell segmentation and tracking. *Med. Image Anal.* **2021**, *71*, 102048. [CrossRef] [PubMed]
28. Zhao, M.; Chang, C.H.; Xie, W.; Xie, Z.; Hu, J. Cloud shape classification system based on multi-channel cnn and improved fdm. *IEEE Access* **2020**, *8*, 44111–44124. [CrossRef]
29. Zimbardo, G.; Malara, F.; Perri, S. Energetic particle superdiffusion in solar system plasmas: Which fractional transport equation? *Symmetry* **2021**, *13*, 2368. [CrossRef]
30. Bi, Y.; Xue, B.; Mesejo, P.; Cagnoni, S.; Zhang, M. A Survey on Evolutionary Computation for Computer Vision and Image Analysis: Past, Present, and Future Trends. *arXiv* **2022**, arXiv:2209.06399.
31. Xu, J.; Sun, Y.; Qu, K.; Meng, X.; Hou, Q. Online group streaming feature selection using entropy-based uncertainty measures for fuzzy neighborhood rough sets. *Complex Intell. Syst.* **2022**, *8*, 5309–5328. [CrossRef]
32. Chen, L.Q.; Wang, C.; Song, S.L. Software defect prediction based on nested-stacking and heterogeneous feature selection. *Complex Intell. Syst.* **2022**, *8*, 3333–3348. [CrossRef]
33. Xu, J.; Yuan, M.; Ma, Y. Feature selection using self-information and entropy-based uncertainty measure for fuzzy neighborhood rough set. *Complex Intell. Syst.* **2021**, *8*, 287–305. [CrossRef]
34. Jain, R.; Joseph, T.; Saxena, A.; Gupta, D.; Khanna, A.; Sagar, K.; Ahlawat, A.K. Feature selection algorithm for usability engineering: A nature inspired approach. *Complex Intell. Syst.* **2021**, 1–11. [CrossRef]
35. Jin, B.; Cruz, L.; Gonçalves, N. Deep facial diagnosis: Deep transfer learning from face recognition to facial diagnosis. *IEEE Access* **2020**, *8*, 123649–123661. [CrossRef]
36. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [CrossRef]
37. Djemame, S.; Batouche, M.; Oulhadj, H.; Siarry, P. Solving reverse emergence with quantum PSO application to image processing. *Soft Comput.* **2019**, *23*, 6921–6935. [CrossRef]
38. Hosseini, S.; Zade, B.M.H. New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN. *Comput. Netw.* **2020**, *173*, 107168. [CrossRef]
39. Wu, H.; Gao, Y.; Wang, W.; Zhang, Z. A hybrid ant colony algorithm based on multiple strategies for the vehicle routing problem with time windows. *Complex Intell. Syst.* **2021**, 1–18. [CrossRef]
40. Moghaddasi, S.S.; Faraji, N. A hybrid algorithm based on particle filter and genetic algorithm for target tracking. *Expert Syst. Appl.* **2020**, *147*, 113188. [CrossRef]
41. Hamdi, T.; Ali, J.B.; Di Costanzo, V.; Fnaiech, F.; Moreau, E.; Ginoux, J.M. Accurate prediction of continuous blood glucose based on support vector regression and differential evolution algorithm. *Biocybern. Biomed. Eng.* **2018**, *38*, 362–372. [CrossRef]
42. Euchi, J.; Masmoudi, M.; Siarry, P. Home health care routing and scheduling problems: A literature review. *4OR* **2022**, *20*, 351–389. [CrossRef]
43. Harizan, S.; Kuila, P. Evolutionary algorithms for coverage and connectivity problems in wireless sensor networks: A study. In *Design Frameworks for Wireless Networks*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 257–280.
44. Mirjalili, S. Evolutionary algorithms and neural networks. In *Studies in Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2019; Volume 780.
45. Kamath, U.; Compton, J.; Islamaj-Doğan, R.; De Jong, K.A.; Shehu, A. An evolutionary algorithm approach for feature generation from sequence data and its application to DNA splice site prediction. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2012**, *9*, 1387–1398. [CrossRef]
46. Abd-Alsabour, N. A review on evolutionary feature selection. In Proceedings of the 2014 European Modelling Symposium, Pisa, Italy, 21–23 October 2014; pp. 20–26.
47. Jadhav, S.; He, H.; Jenkins, K. Information gain directed genetic algorithm wrapper feature selection for credit rating. *Appl. Soft Comput.* **2018**, *69*, 541–553. [CrossRef]
48. Ghamisi, P.; Benediktsson, J.A. Feature selection based on hybridization of genetic algorithm and particle swarm optimization. *IEEE Geosci. Remote Sens. Lett.* **2014**, *12*, 309–313. [CrossRef]
49. Wang, X.; Yang, J.; Teng, X.; Xia, W.; Jensen, R. Feature selection based on rough sets and particle swarm optimization. *Pattern Recognit. Lett.* **2007**, *28*, 459–471. [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. Xue, B.; Zhang, M.; Browne, W.N.; Yao, X. A survey on evolutionary computation approaches to feature selection. *IEEE Trans. Evol. Comput.* **2015**, *20*, 606–626. [CrossRef]
52. Maleki, N.; Zeinali, Y.; Niaki, S.T.A. A k-NN method for lung cancer prognosis with the use of a genetic algorithm for feature selection. *Expert Syst. Appl.* **2021**, *164*, 113981. [CrossRef]
53. Zhou, Y.; Zhang, W.; Kang, J.; Zhang, X.; Wang, X. A problem-specific non-dominated sorting genetic algorithm for supervised feature selection. *Inf. Sci.* **2021**, *547*, 841–859. [CrossRef]
54. Xue, Y.; Zhu, H.; Liang, J.; Słowik, A. Adaptive crossover operator based multi-objective binary genetic algorithm for feature selection in classification. *Knowl.-Based Syst.* **2021**, *227*, 107218. [CrossRef]
55. Song, X.f.; Zhang, Y.; Gong, D.w.; Sun, X.y. Feature selection using bare-bones particle swarm optimization with mutual information. *Pattern Recognit.* **2021**, *112*, 107804. [CrossRef]
56. Song, X.F.; Zhang, Y.; Gong, D.W.; Gao, X.Z. A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data. *IEEE Trans. Cybern.* **2021**, *52*, 9573–9586. [CrossRef]
57. Li, A.D.; Xue, B.; Zhang, M. Improved binary particle swarm optimization for feature selection with new initialization and search space reduction strategies. *Appl. Soft Comput.* **2021**, *106*, 107302. [CrossRef]
58. Jangir, P.; Jangir, N. A new non-dominated sorting grey wolf optimizer (NS-GWO) algorithm: Development and application to solve engineering designs and economic constrained emission dispatch problem with integration of wind power. *Eng. Appl. Artif. Intell.* **2018**, *72*, 449–467. [CrossRef]
59. Sathiyabhama, B.; Kumar, S.U.; Jayanthi, J.; Sathiya, T.; Ilavarasi, A.; Yuvarajan, V.; Gopikrishna, K. A novel feature selection framework based on grey wolf optimizer for mammogram image analysis. *Neural Comput. Appl.* **2021**, *33*, 14583–14602. [CrossRef]
60. Chen, H.; Ma, X.; Huang, S. A Feature Selection Method for Intrusion Detection Based on Parallel Sparrow Search Algorithm. In Proceedings of the 2021 16th International Conference on Computer Science & Education (ICCSE), Lancaster, UK, 17–21 August 2021; pp. 685–690.
61. Da Silva, R.G.; Ribeiro, M.H.D.M.; Mariani, V.C.; dos Santos Coelho, L. Forecasting Brazilian and American COVID-19 cases based on artificial intelligence coupled with climatic exogenous variables. *Chaos Solitons Fractals* **2020**, *139*, 110027. [CrossRef] [PubMed]
62. Dey, A.; Chattopadhyay, S.; Singh, P.K.; Ahmadian, A.; Ferrara, M.; Senu, N.; Sarkar, R. MRFGRO: A hybrid meta-heuristic feature selection method for screening COVID-19 using deep features. *Sci. Rep.* **2021**, *11*, 24065. [CrossRef]
63. Shaban, W.M.; Rabie, A.H.; Saleh, A.I.; Abo-Elsoud, M. Accurate detection of COVID-19 patients based on distance biased Naïve Bayes (DBNB) classification strategy. *Pattern Recognit.* **2021**, *119*, 108110. [CrossRef]
64. Adam, S.P.; Alexandropoulos, S.A.N.; Pardalos, P.M.; Vrahatis, M.N. No free lunch theorem: A review. In *Approximation and Optimization*; Springer: Berlin, Germany, 2019; pp. 57–82. [CrossRef]
65. Liu, T.; Yuan, Z.; Wu, L.; Badami, B. An optimal brain tumor detection by convolutional neural network and enhanced sparrow search algorithm. *Proc. Inst. Mech. Eng. Part H J. Eng. Med.* **2021**, *235*, 459–469. [CrossRef]
66. Zhu, Y.; Yousefi, N. Optimal parameter identification of PEMFC stacks using Adaptive Sparrow Search Algorithm. *Int. J. Hydrogen Energy* **2021**, *46*, 9541–9552. [CrossRef]
67. Zhang, C.; Ding, S. A stochastic configuration network based on chaotic sparrow search algorithm. *Knowl.-Based Syst.* **2021**, *220*, 106924. [CrossRef]
68. Tuerxun, W.; Chang, X.; Hongyu, G.; Zhijie, J.; Huajian, Z. Fault diagnosis of wind turbines based on a support vector machine optimized by the sparrow search algorithm. *IEEE Access* **2021**, *9*, 69307–69315. [CrossRef]
69. Gad, A.G.; Sallam, K.M.; Chakrabortty, R.K.; Ryan, M.J.; Abohany, A.A. An improved binary sparrow search algorithm for feature selection in data classification. *Neural Comput. Appl.* **2022**, *34*, 15705–15752. [CrossRef]
70. Xue, J.; Shen, B. A novel swarm intelligence optimization approach: Sparrow search algorithm. *Syst. Sci. Control Eng.* **2020**, *8*, 22–34. [CrossRef]
71. Wu, R.; Huang, H.; Wei, J.; Ma, C.; Zhu, Y.; Chen, Y.; Fan, Q. An improved sparrow search algorithm based on quantum computations and multi-strategy enhancement. *Expert Syst. Appl.* **2023**, *215*, 119421. [CrossRef]
72. Ma, J.; Hao, Z.; Sun, W. Enhancing sparrow search algorithm via multi-strategies for continuous optimization problems. *Inf. Process. Manag.* **2022**, *59*, 102854. [CrossRef]
73. Wang, P.; Zhang, Y.; Yang, H. Research on economic optimization of microgrid cluster based on chaos sparrow search algorithm. *Comput. Intell. Neurosci.* **2021**, *2021*, 5556780. [CrossRef]
74. Zhang, N.; Zhao, Z.; Bao, X.; Qian, J.; Wu, B. Gravitational search algorithm based on improved Tent chaos. *Control Decis.* **2020**, *35*, 893–900.
75. Kuang, F.; Xu, W.; Jin, Z. Artificial bee colony algorithm based on self-adaptive Tent chaos search. *Control Theory Appl.* **2014**, *31*, 1502–1509.
76. Shan, L.; Qiang, H.; Li, J.; Wang, Z. Chaotic optimization algorithm based on Tent map. *Control Decis.* **2005**, *20*, 179–182.
77. Yang, X.S. Firefly algorithm, Levy flights and global optimization. In *Research and Development in Intelligent Systems XXVI*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 209–218.

78. Cao, W.; Tan, Y.; Huang, M.; Luo, Y. Adaptive bacterial foraging optimization based on roulette strategy. In Proceedings of the International Conference on Swarm Intelligence, Barcelona, Spain, 26–28 October 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 299–311.
79. Altman, N.S. An introduction to kernel and nearest-neighbor nonparametric regression. *Am. Stat.* **1992**, *46*, 175–185.
80. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.P.; Auger, A.; Tiwari, S. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL Rep.* **2005**, *2005005*, 2005.
81. Tang, K.; Yáo, X.; Suganthan, P.N.; MacNish, C.; Chen, Y.P.; Chen, C.M.; Yang, Z. *Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization*; Nature Inspired Computation and Applications Laboratory, USTC: Beijing, China, 2007; Volume 24, pp. 1–18.
82. Mallipeddi, R.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real-Parameter Optimization*; Nanyang Technological University: Singapore, 2010; Volume 24.
83. Liang, J.J.; Qu, B.Y.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*; Technical Report; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China; Nanyang Technological University: Singapore, 2013; Volume 635, p. 490.
84. Liang, J.; Qu, B.; Suganthan, P.; Chen, Q. *Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-Based Real-Parameter Single Objective Optimization*; Technical Report 201411A; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China; Nanyang Technological University: Singapore, 2014; Volume 29, pp. 625–640.
85. Wu, G.; Mallipeddi, R.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization*; Technical Report; National University of Defense Technology: Changsha, China; Kyungpook National University: Daegu, Republic of Korea; Nanyang Technological University: Singapore, 2017.
86. Mohamed, A.W.; Hadi, A.A.; Mohamed, A.K.; Awad, N.H. Evaluating the performance of adaptive GainingSharing knowledge based algorithm on CEC 2020 benchmark problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.
87. Yao, X.; Liu, Y.; Lin, G. Evolutionary programming made faster. *IEEE Trans. Evol. Comput.* **1999**, *3*, 82–102.
88. Karaboga, D.; Akay, B. A comparative study of artificial bee colony algorithm. *Appl. Math. Comput.* **2009**, *214*, 108–132. [CrossRef]
89. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
90. Cheng, R.; Jin, Y. A competitive swarm optimizer for large scale optimization. *IEEE Trans. Cybern.* **2014**, *45*, 191–204. [CrossRef] [PubMed]
91. Liu, J.; Lampinen, J. A fuzzy adaptive differential evolution algorithm. *Soft Comput.* **2005**, *9*, 448–462. [CrossRef]
92. Zhu, G.Y.; Zhang, W.B. Optimal foraging algorithm for global optimization. *Appl. Soft Comput.* **2017**, *51*, 294–313. [CrossRef]
93. Viktorin, A.; Pluhacek, M.; Senkerik, R. Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on CEC2014 benchmark set. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 4797–4803.
94. Li, J.; Gao, Y.; Zhang, H.; Yang, Q. Self-adaptive opposition-based differential evolution with subpopulation strategy for numerical and engineering optimization problems. *Complex Intell. Syst.* **2022**, *8*, 2051–2089. [CrossRef]
95. Asuncion, A.; Newman, D. *UCI Machine Learning Repository*; Irvine University of California: Irvine, CA, USA, 2007.
96. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
97. Mirjalili, S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput. Appl.* **2016**, *27*, 1053–1073. [CrossRef]
98. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
99. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
100. Arora, S.; Anand, P. Binary butterfly optimization approaches for feature selection. *Expert Syst. Appl.* **2019**, *116*, 147–160. [CrossRef]
101. Shi, Y. Brain storm optimization algorithm. In Proceedings of the International Conference in Swarm Intelligence, Chongqing, China, 12–15 June 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 303–309.
102. Yuan, J.; Zhao, Z.; Liu, Y.; He, B.; Wang, L.; Xie, B.; Gao, Y. DMPPT control of photovoltaic microgrid based on improved sparrow search algorithm. *IEEE Access* **2021**, *9*, 16623–16629. [CrossRef]
103. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
104. Wilcoxon, F. Individual comparisons by ranking methods. In *Breakthroughs in Statistics*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 196–202.
105. 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]
106. Sayed, A.M.; Khattab, A.R.; AboulMagd, A.M.; Hassan, H.M.; Rateb, M.E.; Zaid, H.; Abdelmohsen, U.R. Nature as a treasure trove of potential anti-SARS-CoV drug leads: A structural/mechanistic rationale. *RSC Adv.* **2020**, *10*, 19790–19802. [CrossRef]

107. Chen, X.; Tang, Y.; Mo, Y.; Li, S.; Lin, D.; Yang, Z.; Yang, Z.; Sun, H.; Qiu, J.; Liao, Y.; et al. A diagnostic model for coronavirus disease 2019 (COVID-19) based on radiological semantic and clinical features: A multi-center study. *Eur. Radiol.* **2020**, *30*, 4893–4902. [CrossRef]
108. Iwendi, C.; Bashir, A.K.; Peshkar, A.; Sujatha, R.; Chatterjee, J.M.; Pasupuleti, S.; Mishra, R.; Pillai, S.; Jo, O. COVID-19 patient health prediction using boosted random forest algorithm. *Front. Public Health* **2020**, *8*, 357. [CrossRef]