International Journal of Computing and Digital Systems
2025, VOL. 17, NO.1, 1-19

http://dx.doi.org/10.12785/ijcds/1571040201

Behavior Analysis of the (DESFO) Algorithm Using Different Transfer Functions and Different Classifiers

Ahmed Sabry Abolaban[1], O. E. Emam [2] and Safaa. M. Azzam [3]

[1,2,3]*Department of Information Systems, Faculty of Computers and Artificial Intelligence, Helwan University, Helwan, Egypt*

Received 18 June 2024, Revised 4 December 2024, Accepted 9 December 2024

Abstract: The process of feature selection (FS) plays a crucial role in optimizing model performance in machine learning. It achieves this by reducing data dimensionality and eliminating irrelevant features. This study uses Transfer functions and Machine learning classifiers to evaluate the proposed DESFO, which is the effectiveness of integrating the Differential Evolution (DE) with Sailfish Optimization (SFO) in FS issues. The DESFO algorithm is tested with both V-shaped and S-shaped transfer functions across various classifiers, such as k-nearest Neighbors (k-NN), Support Vector Machine (SVM), and Random Forest (RF). The study conducted experiments on 14 benchmark datasets from the UCI Machine Learning repository, illustrating that using DESFO with either type of transfer function notably improves classification accuracy and computational efficiency. The findings indicate that V-shaped transfer functions perform well in situations requiring precise feature selection, whereas S-shaped transfer functions show outstanding generalization capabilities. When inspecting the mean accuracy, DESFO-RF employing V-shaped V4 and DESFO-SVM employing S-shaped V4 configurations outperform all other transfer functions in 10 of the 14 benchmarks. Regarding the mean Fitness functions, DESFO-RF employing V-shaped V1 and DESFO-SVM employing S-shaped V3 is superior in 8 of the 14 benchmarks. Moreover, considering the number of selected features, DESFO-RF equipped with an S-shaped V1 stands out in 8 of 14 benchmark datasets.

Keywords: Feature selection, Transfer Functions, V-shaped, S-shaped, Classification, Machine learning, differential evolution, Sailfish, Optimization

1. Introduction

Machine learning (ML) algorithms use features, or measurable aspects of the observed process, for classification. The feature count in ML applications has grown from tens to hundreds over time. Addressing the challenge of irrelevant and redundant variables, Feature Selection (FS) is a technique that helps in understanding data, reducing computational needs, overcoming the curse of dimensionality, and improving prediction accuracy. Supervised learning algorithms utilize data as a matrix [1]. Optimizing high-dimensional datasets often requires reducing these datasets. This can be achieved by finding a similar matrix with fewer features, making it easier to use. Dimensionality reduction simplifies the process by focusing on discoveries with fewer columns. Additionally, selecting appropriate features can lower measurement costs and enhance problem understanding [2], [3].

Due to its numerous benefits, FS is widely used in real-world applications, particularly in classification and regression problems. It has effectively addressed challenges in various domains, such as micro-array analysis, image classification, gene selection [4], facial recognition, and text classification [5]. Feature selection in ML involves using multiple statistical methods such as filters, wrappers, and embedded methods. The filter method is a preprocessing step that selects features based on specific criteria without considering their effect on the algorithm's performance [6], [7]. On the other hand, wrapper methods evaluate feature subsets based on the accuracy of a given predictor and use search strategies to yield nested subsets of variables [8]. Finally, embedded methods perform variable selection during the training process and are specific to certain learning machines, making it impossible to separate the learning and FS steps [9]. Methods for FS, such as wrappers or embedded methods, involve utilizing nonparametric algorithms like decision trees, support vector machines, and neural networks [10].

Over the past few decades, numerous techniques have been introduced for classification. Some of the most commonly utilized methods include K-Nearest-Neighbors (KNN), artificial neural networks (ANNs), support vector

E-mail address: ahmed_sabry_1053@fci.helwan.edu.eg, osamaemamfcih@gmail.com , eng_safaa_azam@yahoo.com

machines (SVMs), and ensembles of classification trees like Random Forest (RF) [11]. ML algorithms are found to be more precise when compared to statistical techniques such as logistic regression or discriminant analysis, particularly when the input datasets have varying statistical distributions or when the feature space is complicated [12]. Recently, with the increased computational power, ML algorithms have gained more attention, improving the quality of pattern recognition systems. Hence, most classification studies report that RF, KNN, and SVM are the top classifiers, achieving high accuracies [9].

Numerous advanced literature reviews have emphasized the difficulties faced in FS across diverse ML domains. As per Zhigljavsky [13], FS is likely a combinatorial optimization problem that falls under the NP-complete category because of the exponential increase in potential solutions with adding more dataset features. This makes it challenging to find feature subsets that are close to optimal. FS has been classified as an NP-hard problem that exhibits an exponential increase in computational time with complexity [14]. As a result, considerable attention has been given to using metaheuristic (MH) algorithms, which effectively optimize diverse scenarios [15].

There are four main types of algorithms used in MH: SI algorithms, EA, PhA, and Human-based algorithms [16]. SI algorithms are based on animal behaviors and swarms, such as particle swarm optimization (PSO) and ant colony optimization (ACO), and are commonly used in vehicle routing and FS applications. The Artificial Bee Colony (ABC) algorithm is another type of SI algorithm [17]. Evolutionary algorithms (EA) are designed to replicate the natural process of evolution, including mutation and selection. Examples of such algorithms include Genetic Algorithm (GA) and Differential Evolution (DE). Physical laws inspire some algorithms in the field of PhA. These algorithms include BBBC, GSA, and MVO. Algorithms based on human behaviors, like TBLA, SELOA, sine cosine algorithm, and volleyball Premier League Algorithm, are designed to optimize various processes [18]. MH algorithms follow a similar pattern that involves two key phases: exploration and exploitation. These algorithms randomly generate operators to explore the problem space during the exploration phase. In the exploitation phase, Try to identify the optimal solution by harmonizing these phases. This approach helps prevent getting stuck in local optima traps.

When dealing with the FS tasks, it becomes crucial to incorporate Transfer Functions (TFs). According to various studies, TFs are highly recommended for several reasons. To begin with, TFs are not bound to any particular algorithm and don't influence an algorithm's search behavior. Furthermore, the algorithm's computational complexity remains unaffected since the TF is computed for each solution in every iteration. Finally, using a TF can enhance both exploration and exploitation [19], [20], [21].

A. Motivation

The DE algorithm was proposed by Storn et al. [22] as a stochastic search approach that operates on populations. It is a practical and straightforward approach for globally optimizing continuous search, and it has also been utilized in diverse fields. The Sailfish Optimizer (SFO) was created and introduced by Shadravan et al. [23]. It depends on the behavior of a flock of sailfish hunting a flock of sardines. It imitates their hunting strategy by attacking the flock of sardines and retrograding after capturing their prey. This algorithm operates on the principle of population and aims to optimize performance. It has become well-liked in optimization because of its efficiency and robustness. Consequently, this paper introduces a hybrid technique named DESFO algorithm with eight TFs and three ML classifiers, combining both FS algorithms.

To apply the DESFO algorithm to solve an FS problem, a mapping function must change continuous values obtained by the DESFO algorithm into binary values of either 0 or 1. These binary values represent the decision variables [24]. TFs [21] determine how quickly the decision variables (DV) value ranges from 0 to 1 and vice versa.

In ML, k-nearest Neighbor (k-NN), Random Forest (RF), and Support Vector Machine (SVM) are commonly used ML classification algorithms. This study uses DESFO as an optimizer to identify the considerable appropriate attributes. As fitness evaluators, it employs k-NN, RF, and SVM. Using these evaluators, a new wrapper FS method is composed [24].

Metaheuristics have been quite successful in FS. However, many existing methods have focused solely on the k-NN classifier while neglecting the SVM in several instances. The Random Forest (RF) technique has received minimal focus, even though SVM and RF generally produce superior results compared to k-NN in various classification tasks [25].

B. Contribution

This paper proposes applying the DESFO algorithm, Transfer Functions (TFs), and three ML classifiers. It introduces innovative contributions, which are summarized as follows:

1) The DESFO algorithm is described by incorporating and replicating DE and SFO.
2) The method utilizes eight TFs, specifically the V-shaped and S-shaped to change the position values into binary values.
3) The DESFO algorithm is used for wrapper FS.
4) The performance of DESFO with TFs and classifiers is assessed using measures such as the mean of fitness, accuracy, and the mean count of features chosen.
5) The proposed DESFO algorithm's performance is evaluated using k-NN, RF, and SVM machine learning classifiers with the mentioned metrics.

C. Structure

Section 2 discusses the latest findings and critiques in art and literature. Section 3 outlines the prior research conducted. Section 4 describes the methods behind the newly proposed DESFO algorithm, including using the eight TFs and other related procedures. Section 5 showcases the outcomes of experiments, comparing the performance of the multi-variants of the DESFO algorithm with TFs and ML classifiers. Finally, Section 6 encapsulates the study's conclusions.

2. Related Work

Sorour et al. [11] introduced an improved Binary Crayfish Optimization Algorithm (BCOA) to enhance FS in supervised classification tasks. The study aimed to address the challenge of selecting the most relevant features from large datasets to improve the performance of machine learning classifiers. By enhancing the BCOA, the authors optimized FS, which reduced the number of features while maintaining or improving classification accuracy. The proposed algorithm was tested on various benchmark datasets, demonstrating its effectiveness in lowering computational costs and improving the efficiency of classifiers. Key advantages of this approach include its ability to handle high-dimensional data and improve model performance by selecting the most significant features. However, the algorithm's complexity and the computational effort required for tuning hyperparameters were noted as potential limitations. Overall, the study contributes to FS by offering an innovative optimization algorithm for supervised classification.

Abdelkader et al. [14] proposed an efficient data mining technique to assess the satisfaction levels of higher education students with online learning during the COVID-19 pandemic. The study aimed to analyze large-scale online learning data and extract meaningful patterns to understand better the factors influencing student satisfaction. The authors utilized various data mining algorithms to evaluate student feedback and performance data, identifying key variables that impacted satisfaction, such as course design, technology infrastructure, and instructor interaction. The proposed technique was tested on real-world datasets, demonstrating its effectiveness in accurately predicting satisfaction levels and providing actionable insights for improving online education. The advantages of the study include its ability to process large amounts of data efficiently and its practical implications for enhancing the quality of online learning. However, one limitation noted was the potential difficulty in generalizing the results to all educational contexts due to technological access and resource differences. This study provides a valuable tool for academic institutions to improve student experiences in online learning environments.

Hussien et al. [5] developed an improved Binary Meerkat Optimization (BMO) algorithm to enhance FS for supervised learning classification tasks. The study introduced modifications to the original BMO algorithm to improve its performance in selecting relevant features from high-dimensional datasets, which is critical for increasing the accuracy and efficiency of machine learning classifiers. The authors evaluated their proposed approach on various datasets and demonstrated that the improved BMO algorithm effectively reduced the number of features while maintaining or boosting classification accuracy. Key advantages of this method include its ability to enhance computational efficiency by reducing the feature set and improving classifier performance. However, the study also noted potential drawbacks, such as the complexity of the algorithm and the computational resources required for larger datasets. Overall, the work contributes to FS by providing an innovative and effective optimization technique.

El-Mageed et al. [7] Utilized a refined Nuclear Reaction Optimization (NRO) algorithm for targeted gene selection in cancer classification within large-scale datasets. This research aimed to address the challenges posed by the vast number of gene features in cancer datasets, which can hinder the performance of machine learning classifiers. By enhancing the NRO algorithm, the authors optimized the selection of relevant genes, thereby reducing the dimensionality of the data while preserving classification accuracy. The method was applied to several gene expression datasets, and the results indicated that the improved NRO algorithm effectively enhanced cancer classification performance compared to traditional techniques. The advantages of the study include its ability to handle high-dimensional data efficiently and its improvement in classification outcomes. However, similar to other meta-heuristic approaches, the complexity of tuning the algorithm and the computational cost involved in real-world applications were noted as potential drawbacks. This work provides a promising approach to gene selection in the context of cancer classification.

Chen et al. [26] introduced an enhanced version of the Dragonfly Algorithm. The approach improves the Dragonfly Algorithm by incorporating a BDA-DDO that adjusts step size adaptively and introduces a novel differential operator for quicker convergence. It includes a directed variant for targeted searches and an adaptive method to diversify populations, tested successfully on 14 UCI datasets.

Chantar et al. [27] introduced an advanced binary Grey Wolf Optimizer (GWO) incorporated into a wrapper FS strategy for addressing issues in classifying Arabic texts. This modified binary GWO is employed as a wrapper-based FS method. The efficacy of this approach was evaluated across different learning models, including decision trees, K-nearest neighbors, Naive Bayes, and SVM classifiers. Three public Arabic datasets were used for evaluation to determine the performance of different BGWO-based wrapper approaches.

Gafar et al. [4]focused on enhancing gene selection for cancer classification within high-dimensional datasets by utilizing an improved African Vultures Algorithm (AVA). The research aimed to tackle the challenge of selecting relevant genes in cancer classification tasks, particularly with complex, high-dimensional datasets. The authors introduced an improved version of the AVA aimed at boosting the performance of ML classifiers through the selection of the most informative genes.. The methodology was tested on various gene expression datasets, showing that the enhanced AVA effectively reduced data dimensionality while maintaining high classification accuracy. Key advantages of this approach include its efficiency in processing large datasets and improving classification results. However, the complexity of the algorithm and the need for hyperparameter tuning were identified as potential drawbacks. Overall, the study presents an innovative solution to gene selection challenges in cancer classification, offering significant improvements in data handling and model performance.

Mosa et al. [3] focused on improving the credit card fraud detection (CCFD) technique by leveraging the combination of (MH) optimization techniques and (ML) algorithms. The authors employed several meta-heuristic algorithms, such as genetic algorithms and particle swarm optimization, to enhance the performance of traditional machine learning models. The study used a publicly available credit card fraud dataset from Kaggle, containing over 280,000 transactions, including both fraudulent and non-fraudulent instances. The results demonstrated that the proposed approach outperformed other baseline models regarding accuracy, precision, recall, and F1 score, making it highly efficient in detecting fraudulent transactions. The study's advantages included high performance in identifying fraud with relatively low computational cost and effectively handling the imbalanced dataset. However, one disadvantage was the complexity of tuning the meta-heuristic algorithms, which could be time-consuming and challenging to optimize for real-world deployment. The study contributes to fraud detection by proposing an innovative and effective hybrid approach.

Fatahi et al. [28] introduced an improved version of the Binary Quantum-based Avian Navigation Optimizer Algorithm (IBQANA). IBQANA updates the Binary Quantum-based Avian Navigation Optimizer for better Feature Subset Selection in medical data, fixing past inefficiencies. It introduces the Hybrid Binary Operator (HBO) for accurate binary transitions of continuous values and the Distance-based Binary Search Strategy (DBSS) for improved search efficiency and faster convergence by blending exploration and exploitation with a variable probability approach to dodge local optima.

3. Preliminary Work

A. DESFO Algorithm

The No Free Lunch Theorem (NFL) [29] indicates that an algorithm capable of optimally solving all optimization issues does not exist. An algorithm's capability for FS varies with the dataset, indicating a need for better metaheuristic approaches to tackle FS challenges efficiently. The DESFO algorithm introduced by Azzam et al. [30], which combines Differential Evolution (DE) and Swarm Fish Optimization (SFO), aims to improve FS and classification accuracy, addressing an unmet need in current research.

1) DE Algorithm

Storn et al. [22] presented the DE algorithm, recognized for its effectiveness among Evolutionary Algorithms and is notable for quick convergence and simplicity. Utilizing just three parameters: Population size (NP), Crossover rate (Cr), and Scaling Factor (F), DE effectively solves a wide array of optimization problems. It starts with an initial solution set, generating new solutions by modifying existing ones based on the weighted differences between pairs of other solutions, also known as mutant solutions. The DE algorithm has proven effective and embraced for addressing optimization problems in diverse domains [31].

Mutation: In every iteration (t), Differential Evolution (DE) uses a mutation operator to create a new donor vector or mutant vector for each solution. This operator specifies three potential solutions randomly, emphasizing that a mutant or donor vector is made by multiplying the difference between two vectors by a scale factor, followed by adding this scaled difference to a third solution [22]:

$$V_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}). \tag{1}$$

Three unique integers $r1$, $r2$, and $r3$ are chosen randomly, each lying within the range of 1 to NP, and NP is an integer and positive, which is four or more. Moreover, these integers' values differ from the current index, denoted as i. After that, the differential amplification $(x_{r2,G} - x_{r3,G})$ is boosted by a static F factor , varying between [0, 2].

Crossover: Behind mutation, a new trial vector child is created from the solution using a crossover search operator. The exponential and binomial operators are commonly employed as direct crossover search methods. This applies to every decision variable denoted by j. If rand$(j) \leq C_r$, as shown in Equation 2:

$$u_{i,j,G} = \begin{cases} u_{i,j,G} & \text{if } \mathrm{rand}(j) \leq C_r \text{ or } j = j_{rand}. \\ x_{i,j,G} & \text{otherwise.} \end{cases} \tag{2}$$

Where rand(j), denoted as the "$j_t h$ evaluation," is randomly chosen in the range of [0, 1]. This process guarantees a minimum of one trial vector for the design variable (DV). The C_r is the crossover rate, critical for determining the variable count, and is sourced from the vector of the donor. It is ensured that $V_{i,G+1}$ generates minimally one parameter to $u_{i,j,G}$.

Choosing: A choosing operator is used to identify the best solution by comparing the child's and parent's objective function values. If the child exhibits a decreased value for objective function, it is retained for future iterations. Otherwise, the parent vector remains in the current generation, as shown in Equation 3:

$$x_{i,G+1} = \begin{cases} u_{i,G} & \text{if } f(u_{i,G}) \leq f(x_{i,G}), \\ x_{i,G} & \text{otherwise.} \end{cases} \tag{3}$$

To decide whether to include it in the new generation (G + 1), $x_{i,G+1}$, which is the trial vector, is compared to the target vector $x_{i,G}$ by applying the greedy criterion. Should the trial vector $x_{i,G+1}$ yield a cost function value lower than that of the target vector $x_{i,G}$, then the trial vector $x_{i,G+1}$ takes the place of the target vector. Otherwise, the value of the original target vector is maintained.

2) *SFO Algorithm*

Shadravan et al. [23] introduced an innovative algorithm known as the Sailfish Optimizer (SFO), in which the injured sardine that showcases the optimal fitness value is marked, pinpointing its location by P^i_{srdinj} during the ith iteration. With every iteration, the sardines' and sailfish's locations are adjusted. For each iteration, the sailfish's position is updated by utilizing information from the named elite fish $P_{Slfbesti}$ and the injured sardine's position according to a predetermined criterion.

In each iteration, the locations of sailfish and sardines are updated, and the new position is represented by $i + 1$. The elite and the injured are responsible for adjusting the sailfish's position to a new one, represented by P^{i+1}_{Slf}, as shown in Equation 4:

$$P^{i+1}_{Slf} = P^i_{Slfbest} - \sigma_i \left(\text{rnd} \cdot \left(\frac{P^i_{Slfbest} + P^i_{srdinj}}{2} - P^i_{Slf} \right) \right). \tag{4}$$

Where rnd $\in (0, 1)$ is determined randomly, and σ_i is a coefficient derived by Equation 5:

$$\sigma_i = 3 \cdot \text{rand} \cdot PrD - PrD \tag{5}$$

In every cycle, the prey density PrD, indicative of the available prey count, is calculated using Equation 6. As the prey count diminishes with group hunting activities, the value accordingly decreases:

$$PrD = 1 - \frac{N_{Slf}}{N_{Slf} - N_{srd}}. \tag{6}$$

The numbers for sailfish and sardines are denoted by N_{Slf} and N_{srd}, respectively, and can be determined using Equation 7:

$$N_{Slf} = N_{srd} \cdot Prcent. \tag{7}$$

Where $Prcent$ represents the population of sardines that comprises the original sailfish population, the initial number of sardines is assumed to be higher than the initial number of sailfish.

In each iteration, the locations of the sardines are adjusted following Equation 8:

$$P^{i+1}_{Srd} = \text{rand} \cdot (P^i_{Slfbest} - P^i_{Srd} + ATK). \tag{8}$$

The old and new locations of the sardine are denoted by P^i_{Srd} and P^{i+1}_{Srd}, respectively. Meanwhile, the representation of the strength of the sailfish's attack (ATK) at one-by-one iteration is determined by Equation 9:

$$ATK = A \cdot (1 - (2 \cdot itr \cdot k)). \tag{9}$$

ATK plays a pivotal role in dictating how many sardines adjust their positions and how far they move. Reducing ATK can enable the search agents' convergence. The determination of the sardines' number by adjusting their positions and the variables' number related to the sardines are calculated employing Equations 10 and 11:

$$\gamma = ATK \cdot N_{Srd} \tag{10}$$

$$\delta = ATK \cdot v \tag{11}$$

N_{Srd} denotes the number of sardines, and v refers to the variables. Whenever a sardine exceeds a sailfish fitness level, the sailfish will change its location to pursue the sardine. However, this results in the sardine being eliminated from its population.

B. *Transfer (Mapping) Functions (TFs)*

The DESFO algorithm generates a solution consisting of continuous values; it cannot be directly applied to an FS problem without modifications. Therefore, mapping or Transfer Functions (TFs) are required to convert continuous values to binary format, i.e., 0 or 1. TFs [21] dictate how decision variables transition between 0 and 1. When choosing a TF for converting continuous values to binary, several key factors need to be considered:

- Values derived from the TF should fall between 0 and 1, indicating the agent's likelihood of altering its present position.
- Should the alarm value fall below the safety threshold (ST), the likelihood of the TF changing its current position in the subsequent iteration should increase. This is under the assumption that an agent with an alarm value exceeding ST is likely veering too far from the best solution.
- In case the alarm value is low, the TF should offer a low likelihood of altering the current position.

- As the alarm value approaches ST, the TF-induced likelihood should increase, encouraging agents to correct their course and quickly return to their previous best position in future iterations.

consider the important capability of (TFs) to transform the current search process into a binary representation for each Y, as described in Equation 12:

$$y_{i,j}^{t+1,bin} = \begin{cases} -y_{i,j}^{t,bin} & \text{if rnd} < TF(y_{i,j}^{t+1}) \text{ and TF is V-shaped,} \\ y_{i,j}^{t,bin} & \text{if rnd} \geq TF(y_{i,j}^{t+1}) \text{ and TF is V-shaped,} \\ 0 & \text{if rnd} < TF(y_{i,j}^{t+1}) \text{ and TF is S-shaped,} \\ 1 & \text{if rnd} \geq TF(y_{i,j}^{t+1}) \text{ and TF is S-shaped.} \end{cases} \tag{12}$$

Where $y_{i,j}^{t+1,bin}$ represents the value in the j-th dimension for individuals in the current iteration $t + 1$, and rnd is a randomly chosen number from [0, 1]. $TF(y_{i,j}^{t+1})$ represents the probability value determined by employing a specific TF to each continuous value of the j-th component of agent i. According to Equation 12, two scenarios are encountered: (first) if the TF has an S-shape and rnd is less than the given probability value, the j-th dimension of the individual is updated to 0; otherwise, it is updated to 1. If the TF is V-shaped, if rnd is less than the provided probability value, the value of the j-th dimension is inverted; otherwise, it remains the same. Utilizing S-shaped and V-shaped TFs, as detailed in Table I

Figure 1 displays the variants of these Transfer Functions, categorizing them into two categories; S-shaped and V-shaped TFs.

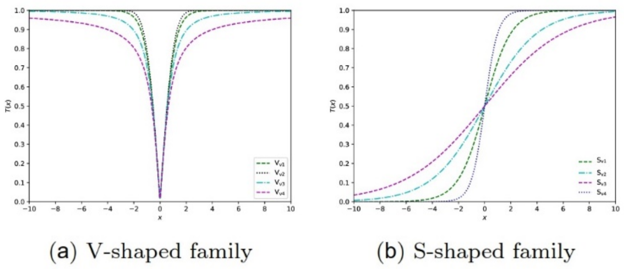

Figure 1. (V-shaped TF) and (S-shaped TF) families.

C. Machine Learning (ML) Classifiers

This study utilized three of the most effective ML classifiers for FS purposes: K-NN, SVM, and RF. Subsequent subsections detail each of these methods.

1) K-Nearest Neighbor Classifier (k-NN)

The K-Nearest Neighbor (k-NN) classifier [32] is widely recognized for its power in pattern recognition and ML fields. Its ease of implementation makes it a preferred choice over more complex supervised learning techniques [33]. K-NN is widely used in fields such as healthcare, forestry, image/video analysis, and finance for its ability to classify patterns. It creates classification rules from training data and assigns labels to unlabeled data in test sets based on the nearest training samples. Choosing the right "k" is crucial for accuracy and is typically found through trial and error. In the empirical studies conducted, the k-NN classifier, utilizing a Euclidean distance metric, was tested with a k value of 5 [34], [19], and the chosen feature subsets were evaluated for their effectiveness.

2) Random Forest Classifier (RF)

The Random Forest (RF) classifier [35] is an ML classifier widely used in various computing tasks such as label distribution learning, action recognition, detection, visual tracking, facial expression analysis, image classification, and time series forecasting. It consists of multiple decision trees and is known for its robustness against data labeling errors, ability to handle various classes, FS support, parallel computation, minimalism in tuning parameters, and efficient numerical and categorical data handling. RF remains famous for its simplicity, interpretability, computational efficiency, and its method of improving classification by breaking down data into smaller subsets for optimized learning [36], [37], [38].

The number of trees and their maximum depth affect RF algorithm performance, particularly for real-time applications. A hyper-heuristic approach can optimize these parameters, enhancing speed and accuracy. In tests, using ten estimators with a maximum depth of 5 showed significant improvements in classification accuracy. Despite some drawbacks, the RF model's capability to accurately model complex relationships demonstrates its value.

3) Support Vector Machine Classifier (SVM)

The Support Vector Machine (SVM) algorithm [39] stands as a prominent wrapper-based classifier utilized in data science, known for effectively segregating multiple classes. The technique relies on using hyper-planes to separate different groups. A significant benefit of SVM is its ability to provide reliable accuracy with low computational effort. It achieves this by applying a non-linear function, ϕ, to shift the original data into a higher dimensional space, where it seeks to linearly divide the data along a hyper-plane that maximizes separation margins between classes. However, challenges include choosing the proper base function and fine-tuning its parameters [40]. Finding the optimal decision plane essentially becomes an optimization problem, where a kernel function is tasked with determining the most suitable higher-dimensional space for achieving linear division of categories via a non-linear transformation.

4. Methodology of the proposed DESFO

This study introduces a method that combines eight (TFs), including V-shaped and S-shaped families, with the

TABLE I. V-shaped and S-shaped Transfer Functions (TFs) [21]

V-shaped Family	S-shaped Family
V1: $TF(y) = \left\lvert \mathrm{erf}\left(\frac{\sqrt{\pi}}{2}y\right)\right\rvert = \left\lvert \frac{\sqrt{2}}{\pi}\int_0^{\frac{\sqrt{2\pi}}{y}} e^{-t^2}dt \right\rvert$,	S1: $TF(y) = \frac{1}{1+e^{-2y}}$,
V2: $TF(y) = \lvert\tanh(y)\rvert$,	S2: $TF(y) = \frac{1}{1+e^{-y}}$,
V3: $TF(y) = \left\lvert \frac{y}{\sqrt{1+y^2}} \right\rvert$,	S3: $TF(y) = \frac{1}{1+e^{-y/2}}$,
V4: $TF(y) = \left\lvert \frac{2}{\pi}\arctan\left(\frac{\pi}{2}y\right)\right\rvert$.	S4: $TF(y) = \frac{1}{1+e^{-y/3}}$.

DESFO, merging the DE and SFO algorithms. This FS approach uses K-NN, RF, and SVM in several steps: initialization, position updates, binary mapping or conversion, and fitness assessment. The DESFO algorithm runs for 100 iterations, split among SFO and DE at 50 iterations for both. Initially, DE takes the lead for the first 50 iterations, focusing on optimizing to find the best solution, which is then further refined by SFO to improve classification accuracy by selecting relevant features. Each phase is described in detail in subsequent sections.

A. *Initial Population Generation*

The initial stage of applying the DESFO algorithm involves creating a starting set of Y positions, symbolizing possible solutions within a space defined by D dimensions. The size of this population is established based on a particular formula as shown in Eq. 13:

$$Y = \mathrm{Round}(2 \times \sqrt{D} + 10). \quad (13)$$

Y represents the comprehensive count of positions, whereas D symbolizes dimensionality. The matrix describing positions is defined as follows:

$$K = \begin{bmatrix} k_{1,1} & k_{1,2} & \dots & k_{1,p} \\ k_{2,1} & k_{2,2} & \dots & k_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ k_{Y,1} & k_{Y,2} & \dots & k_{Y,p}. \end{bmatrix}$$

The initial population, symbolized by K, is created within specific boundaries as follows in Eq. 14:

$$K_i^u = u(0,1) \times (UB - LB) + LB. \quad (14)$$

B. *Position Update in DESFO Algorithm*

The position is updated according to the SFO and DE techniques, as detailed in sections 3.1.1 and 3.1.2. Following the update, the position is converted to binary values, as outlined in section 4.3.

C. *Converting Continuous Positional Values to Binary in DESFO for FS*

In the DESFO algorithm, converting continuous positional values to binary is crucial for effectively solving FS problems. FS typically requires binary decision-making: features are either selected, marked with a value of '1', or otherwise marked with a value of '0'. This binary structure is necessary for evaluating subsets of features in ML classification tasks.

1) *Binary Representation in FS*

In a typical FS problem, a binary vector represents the selection of features:

- 1 indicates that a particular feature has been selected.
- 0 indicates that a particular feature has been excluded.

The length of the binary vector equals the number of features in the dataset, meaning each element corresponds to a feature. This binary representation allows for the optimization of feature subset selection.

2) *Role of Transfer Functions (TFs)*

The DESFO algorithm, however, generates continuous values as part of its optimization process. Therefore, TFs convert these continuous positional values into binary values (0 or 1). TFs are crucial in determining how decision variables (features) transition from 0 to 1 based on probabilistic outcomes.

There are two main families of TFs utilized in DESFO:

1) **S-shaped Transfer Functions**: These TFs provide a smooth probabilistic transition, pushing continuous values closer to the boundary towards 0 or 1, favoring generalization. The output of an S-shaped TF is a probability between 0 and 1, which is compared to a random number. If the number is smaller than the TF output, the binary value is assigned a 0; if not, it receives a 1.
2) **V-shaped Transfer Functions**: These TFs focus on flipping binary values and are used for fine-tuned adjustments during FS. For V-shaped TFs, in case the rand number is smaller than the TF output, the

binary value is inverted (0 becomes one and vice versa); otherwise, the value remains unchanged.

3) Binary Conversion Process

In each iteration of DESFO, the continuous positions of features are updated based on the evolutionary strategies of DE and SFO. These continuous values are then converted into binary form for FS evaluation. The steps for binary conversion are as follows:

1) **Apply the Transfer Function (TF)**: For each continuous value in the solution vector, the appropriate TF (either S-shaped or V-shaped) is applied.
2) **Random Number Comparison**: A random number is produced within the range of [0, 1], which compared then to the TF output:
   - For **S-shaped TFs**: in case the rand number is smaller than the TF output, the binary value is set to 0; otherwise, it is set to 1.
   - For **V-shaped TFs**: in case the rand number is smaller than the TF output, the current binary value is inverted; otherwise, it remains unchanged.
3) **Binary Representation**: The continuous values are thus transformed into a binary vector, where each element represents the selection or exclusion of a corresponding feature.

4) Mathematical Representation of Binary Conversion

The binary conversion process for DESFO can be mathematically described as follows:

$$P^i_{\text{bin}} = \begin{cases} 1, & \text{if rnd} < T_F(P^i) \text{ for S-shaped TFs,} \\ 0, & \text{otherwise,} \\ -P^i_{\text{bin}}, & \text{if rnd} < T_F(P^i) \text{ for V-shaped TFs,} \\ P^i_{\text{bin}}, & \text{otherwise.} \end{cases}$$

Where:

- P^i_{bin} is the binary position for feature i.
- P^i is the continuous positional value for feature i.
- rnd is a randomly generated number in the range $[0, 1]$.
- $T_F(P^i)$ is the transfer function applied to the continuous value P^i.

5) Impact on Feature Selection

The TF choice directly influences the exploration-exploitation trade-off during FS. S-shaped TFs, due to their smooth transition, are suited for tasks requiring broader generalization and aggressive feature reduction. On the other hand, V-shaped TFs enable fine-tuned adjustments, making them ideal for refining smaller feature subsets.

By employing these TFs, the DESFO algorithm effectively navigates the binary search space, leading to improved classification accuracy and dimensionality reduction by selecting only the most relevant features.

D. Fitness Evaluation

Balancing feature set size and accuracy is crucial. A smaller feature set requires more precise classifiers like k-NN, RF, and SVM but risks reducing accuracy due to fewer features [40]. This implies a trade-off between the attribute length and the selection of optimal attributes, indicating a balance might be necessary between accuracy and feature set magnitude.

When evaluating an algorithm's performance, it's important to consider the equilibrium between feature-length and accuracy, as expressed in Eq. 15:

$$FIT = \alpha_1 \times (1 - \text{accuracy}) + \alpha_2 \times \frac{|D^*|}{|D|}. \qquad (15)$$

The two coefficients, α_1 and α_2 represent the weight coefficients, where α_1 varies in the range from 0 to 1. α_2 is obtained by subtracting α_1 from 1. The coefficients were established after comprehensive experimentation. $|D^*|$ represents the selected features, while $|D|$ signifies the overall features in the initial dataset.

E. Flowchart of DESFO

Figure 2 shows the steps and procedures of the DESFO algorithm.

F. Analysis of DESFO Computational Complexity for FS Across Different Classifiers

The DESFO algorithm exhibits computational complexity that scales with the dataset's number of features and ML classifier type used for FS. This subsection analyzes how the computational cost of DESFO scales when applied to different ML models (k-NN, RF, SVM), focusing on how features (D) number and the dataset size affect the overall complexity.

1) Overall Complexity of DESFO

The total computational complexity of the DESFO algorithm consists of several main components: initialization, DE steps, SFO updates, binary transformation, and fitness evaluation using ML models. The overall complexity for all generations (denoted as MaxGens) is given by:

$$\text{MaxGens} \times (D \times \text{NP} \times 3 + \text{NP} \times f(D)).$$

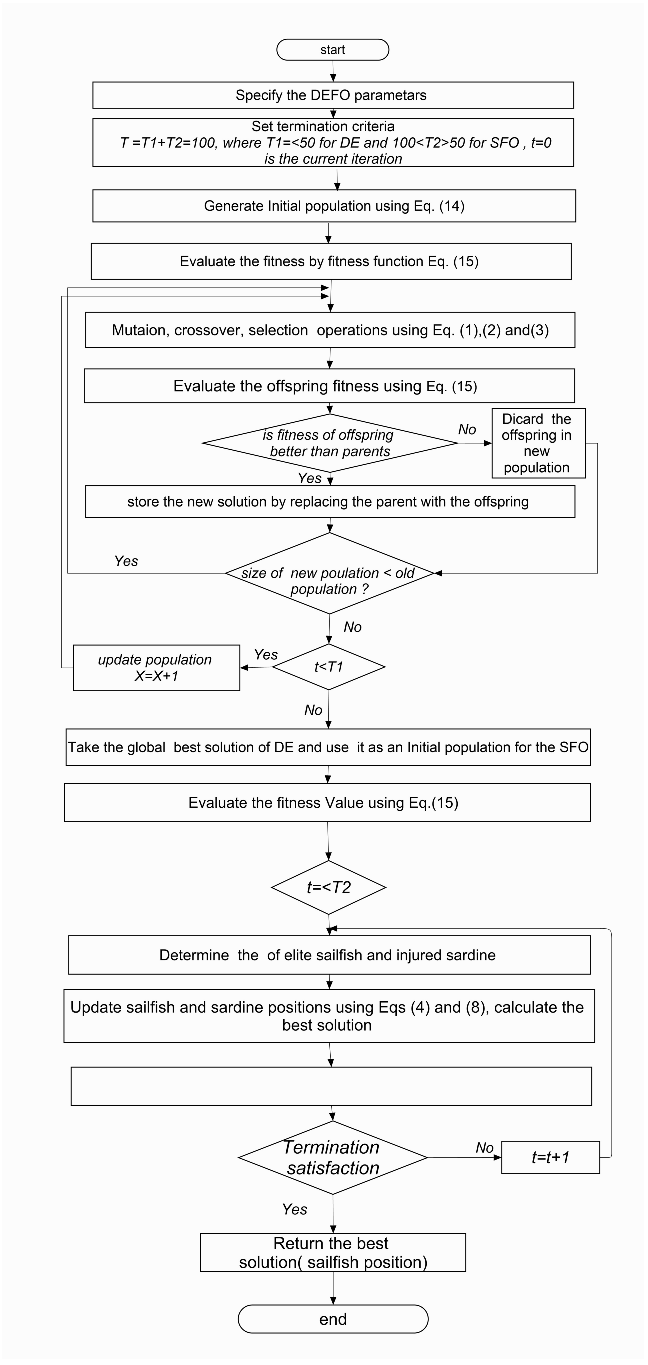
start
Specify the DEFO parametars
Set termination criteria
T =T1+T2=100, where T1=<50 for DE and 100<T2>50 for SFO , t=0 is the current iteration
Generate Initial population using Eq. (14)
Evaluate the fitness by fitness function Eq. (15)
Mutaion, crossover, selection operations using Eq. (1),(2) and(3)
Evaluate the offspring fitness using Eq. (15)
is fitness of offspring better than parents
No
Dicard the offspring in new population
Yes
store the new solution by replacing the parent with the offspring
Yes
size of new poulation < old population ?
No
update population X=X+1
Yes
t<T1
No
Take the global best solution of DE and use it as an Initial population for the SFO
Evaluate the fitness Value using Eq.(15)
t=<T2
Determine the of elite sailfish and injured sardine
Update sailfish and sardine positions using Eqs (4) and (8), calculate the best solution
Termination satisfaction
No
t=t+1
Yes
Return the best solution(sailfish position)
end

Figure 2. DESFO flowchart.

Where:

- D is the number of features in the dataset.
- NP is the population size.
- $f(D)$ is the classifier's computational complexity for fitness evaluation.
- MaxGens represents the iterations num of the algorithm.

2) *Impact of Number of Features (D)*

The computational complexity is highly sensitive to the dataset's number of features (D). As FS is central to the DESFO algorithm, the number of features directly influences the DE and SFO components and the binary transformation process. Specifically:

- **DE and SFO:** Both the DE algorithm's mutation and crossover operations and the SFO algorithm's position adjustment steps involve evaluating D dimensions for each individual in the population. Hence, the complexity of these steps scales linearly with D, resulting in a complexity of $O(\text{NP} \times D)$ for both algorithms.
- **Binary Conversion:** Each continuous value in the solution vector must be mapped to a binary value using TFS, further contributing to a complexity of $O(\text{NP} \times D)$.
- **Fitness Evaluation:** The computational complexity of evaluating the fitness of the binary vector depends on the chosen classifier and scales with D. Since FS involves subsets of features, larger values of D require more computational resources to assess the fitness of each solution. The complexity of this step is represented as $O(f(D)) \times \text{NP}$, where $f(D)$ is the complexity of the classifier.

Thus, the total complexity scales linearly with D in the DE and SFO phases, and the fitness evaluation complexity depends on the specific classifier.

3) *Impact of Dataset Size (Number of Instances)*

The size of the dataset, represented by the number of instances or samples, also affects the computational complexity, particularly during the fitness evaluation step. In FS, the fitness of each solution is evaluated based on the accuracy of the classifier on a subset of features. Larger datasets require more computational resources for training and testing classifiers.

- **k-NN Classifier:** The complexity of the k-NN classifier is proportional to the number of instances N in the dataset because it involves calculating the distance between the test sample and every training sample. The complexity of fitness evaluation with k-NN is $O(N \times D)$. Hence, the overall complexity for fitness evaluation in DESFO with k-NN scales as $O(N \times \text{NP} \times f(D))$.
- **Random Forest (RF) Classifier:** The complexity of Random Forest depends on trees numbers (T) and the maximum depth of each tree (L). Each tree

is built using a subset of features, and the fitness evaluation with RF scales with $O(T \times N \times \log(N))$. This means that, while the number of instances influences the computational complexity, Random Forest scales more efficiently than k-NN.

- **SVM Classifier:** The complexity of SVM is primarily influenced by the number of support vectors and the kernel used. For linear kernels, the complexity scales with $O(N \times D)$, while for non-linear kernels, the complexity can be higher due to the kernel computation. Therefore, the fitness evaluation in DESFO with SVM can range from $O(N \times D)$ for linear kernels to $O(N^2 \times D)$ for non-linear kernels, depending on the dataset and kernel choice.

4) *Scaling with Number of Features and Dataset Size*

The complexity of the DESFO algorithm increases with the number of feature (D) and instances in the dataset (N). As shown:

- For k-NN, the complexity scales linearly with both N and D, i.e., $O(N \times D \times \mathrm{NP})$.
- For Random Forest, the complexity depends on the number of trees and the size of the dataset, generally scaling as $O(T \times N \times \log(N))$.
- For SVM, the complexity ranges from $O(N \times D)$ for linear kernels to $O(N^2 \times D)$ for non-linear kernels, making it the most computationally expensive classifier when applied to large datasets with non-linear kernels.

5) *Summary of Complexity Analysis*

The overall computational complexity of the DESFO algorithm, when applied to FS, is influenced by:

- (D) feature number, with linear scaling in the DE and SFO components and the binary transformation phase.
- (N) record number primarily affects the fitness evaluation based on the classifier used.
- The classifier's inherent complexity, where k-NN, RF, and SVM exhibit different computational characteristics depending on the dataset size and number of features.

Thus, for larger feature sets and datasets, the choice of classifier significantly affects the overall computational cost of DESFO in FS tasks.

5. Results and Analysis

The experiments and results analysis using the DESFO algorithm are demonstrated in this section.

A. *Datasets Overview*

To thoroughly assess and confirm the effectiveness of the methods introduced in this paper, we utilized 14 diverse datasets from the UCI ML repository [41], spanning various domains (such as biology, politics, electromagnetic, gaming, physics, chemistry, and artificial intelligence) in all our experiments. These datasets play a crucial role in effectively substantiating the techniques proposed in this study, considering the varying numbers of instances and features they contain. The specifics of these datasets are presented in Table II.

B. *Experimental Setup*

This paper carefully selects the adopted model and MHA configurations to ensure robust performance evaluation. For the k-NN model, the Euclidean distance metric was employed with a neighborhood size of $k = five$. The SVM model used a polynomial kernel with a degree of $d = two$, while the RF model consisted of $n = ten$ estimators with a maximum tree depth of $d = five$. Given the stochastic nature of MHAs, each experiment was repeated thirty times, and the average performance metrics were recorded across all runs.

The proposed DESFO algorithm had the following settings: a population size of ten and a maximum of one hundred iterations. The feature count for each benchmark determined the problem's dimensionality. The continuous search space was defined within the $[-1, 1]$ domain, providing a sufficiently broad yet controlled exploration space.

The outcomes were validated using a cross-over strategy, where each benchmark was randomly split, with 20% allocated for testing and 80% for training. The documented outcomes reflect the average values of the fitness, accuracy, and the chosen attributes' number over the thirty runs. The remaining parameters for DESFO followed the standard settings from prior studies, as outlined in Table III. All experimentations were performed utilizing Python on a high-performance computing setup with a Dual Intel® Xeon® Gold 5115 2.4 GHz CPU and 128 GB RAM running on Microsoft Windows Server 2019.

C. *Evaluation Metrics*

The effectiveness of the DESFO algorithm is evaluated using eight TFs, and three highly regarded ML classifiers: K-NN, RF, and SVM. These evaluations were performed individually across 30 trials for each benchmark. Specific metrics are used to assess the FS approach.

1) *Mean Accuracy*

Conducting the procedure independently across 30 iterations can establish the precise rate of data categorization (μ_{Acc}).

TABLE II. Overview of the data collections utilized in our research.

Benchmark	**# Records**	**# Features**	**Domain**
BreastCancer	699	9	Biology
BreastEW	569	30	Biology
Exactly2	1000	13	Biology
IonosphereEW	351	34	Electromagnetic
KrVsKpEW	3196	36	Game
Lymphography	148	18	Biology
M-of-n	1000	13	Biology
PenglungEW	73	325	Biology
SonarEW	208	60	Biology
Tic-tac-toe	958	9	Games
Vote	300	16	Politics
WaveformEW	5000	40	Physics
WineEW	178	13	Chemistry
Zoo	101	16	Artificial Intelligence

TABLE III. Parameter setup for the proposed algorithm.

Technique	Coefficients
SFO	Ratio between sailfish and sardines pp = 0.1
	A = 1
	ε = 0.0001
DE	Crossover Rate (Cr)= 0.9
	Scaling Factor (F)= 0.5

$$\mu_{Acc} = \frac{1}{30}\frac{1}{m}\sum_{k=1}^{30}\sum_{r=1}^{m}\text{match}(\text{PL}_r, \text{AL}_r). \tag{16}$$

Where μ_{Acc} symbolizes the mean accuracy, whereas m represents the overall number of samples within the testing dataset. The symbol PLr denotes the predicted class label for a given sample, while ALr represents the reference class label. A specific function, named match (PLr, ALr), is used to compare these two labels. If PLr matches ALr, the match (PLr, ALr) function returns a value of 1; if they do not match, the value is 0.

2) Mean Fitness

The μ_{Fit} is utilized to assess the average outcomes of fitness obtained by implementing the suggested method across 30 separate trials. The optimal outcome is represented by the smallest value, which is determined by evaluating the fitness as follows:

$$\mu_{Fit} = \frac{1}{30}\sum_{i=1}^{30} f_*^k. \tag{17}$$

The symbol μ_{Fit} denotes the average fitness value, whereas f_*^i signifies the optimal fitness obtained in each iteration in the 30 i-th iterations.

3) Mean Selected Number of Features

The metric symbolized by μ_{Feat} stands for the mean selected features, calculated independently The method was applied 30 times as follows :

$$\mu_{Feat} = \frac{1}{30}\sum_{i=1}^{30}\frac{|d_*^i|}{|D|}. \tag{18}$$

Where $|d_*^i|$ indicates the chosen attributes and the count of attributes in the optimum solution for each of the i-th iterations, with $|D|$ representing the total features' number utilized from the datasets.

D. DESFO Behavior Assessment Utilizing Eight TFs

DESFO was initially developed for continuous optimization and required adaptation for discrete space searches. To address the FS problem, its effectiveness was evaluated using eight different transformation functions on 14 benchmark datasets.

The various TFs underwent evaluation using DESFO alongside K-NN, RF, and SVM classifiers, focusing on the mean accuracy (μ_{Acc}), mean fitness value (μ_{Fit}), and average number of selected features (μ_{Feat}). The outcomes of these evaluations are detailed in Tables IV through XII.

The methods were developed using eight transfer functions (TFs), which are categorized into two groups: four V-shaped TFs labeled as Vv1, Vv2, Vv3, and Vv4, and four S-shaped TFs named Sv1, Sv2, Sv3, and Sv4. Thus, in the subsequent discussions, the proposed techniques are referred to as ”DESFO-TF,” with TF representing any of the eight TFs mentioned. The abbreviations W, T, and L found at the bottom of the tables denote the number of times each method wins, ties, or loses compared to the other competitors. Examining and comparing the results mentioned, the most suitable DESFO variant for each classifier will be determined based on the TF that shows the best performance for that particular classifier.

1) Evaluation of DESFO-TFs using the k-NN classifier

Table IV presents the μ_{Acc} of DESFO across eight TFs, utilizing the k-NN classifier. DESFO-Vv4 significantly outperformed in 8 of the 14 datasets, and DESFO-Vv3, DESFO-Sv1, DESFO-Sv2, and DESFO-Vv1 followed closely behind, excelling in 7, 7, 7, and 6 of 14 datasets, respectively. As a result, when considering the mean accuracy, DESFO-Vv4 is ranked highest among the evaluated methods.

Table V presents the mean Fitness Value μ_{Fit} for DESFO across eight TFs, utilizing the k-NN classifier. It was noted that DESFO-Sv4 demonstrated notable performance in 7 out of the 14 datasets, and DESFO-Vv3, DESFO-Sv3, and DESFO-Vv1 followed closely, showing significant results in 6, 6, and 5 out of 14 datasets, respectively. Hence, when considering the mean fitness values, DESFO-Sv4 is the top-performing method among all the TFs methods.

Table VI presents the mean features selected μ_{Feat} for DESFO across eight TFs, using the k-NN classifier for analysis. DESFO-Sv1 and DESFO-Sv4 demonstrated remarkable performance in 7 of the 14 datasets reviewed. Following them closely were DESFO-Sv3 and DESFO-Vv4, each showing significant outcomes in 6 and 5 out of the 14 datasets, respectively. Therefore, based on the average features selected, DESFO-Sv1 and DESFO-Sv4 emerge as the leading methods among all the TFs considered.

2) Evaluation of DESFO-TFs using the RF Classifier

Table VII presents the mean accuracy (μ_{Acc}) of DESFO across eight TFs, utilizing the RF classifier. It was found that DESFO-Vv4 significantly outperformed in 10 of the 14 datasets, with DESFO-Vv3 and DESFO-Sv4 closely behind, excelling in 9 and 8 of the 14 datasets, respectively. As a result, when considering the mean accuracy, DESFO-Vv4 is ranked highest among the evaluated methods.

Table VIII presents the mean Fitness Value (μ_{Fit}) for DESFO across eight TFs, utilizing the RF classifier. It was noted that DESFO-Sv1 demonstrated notable performance in 8 out of the 14 datasets, and DESFO-Vv3, DESFO-Sv3, and DESFO-Sv4 followed closely, showing significant results in 6, 6, and 5 out of 14 datasets, respectively. Hence, when considering the mean fitness values, DESFO-Sv1 is the top-performing method among all the TF methods.

Table IX presents the mean features selected (μ_{Feat}) for DESFO across eight TFs, using the RF classifier for analysis. DESFO-Sv1 demonstrated remarkable performance in 8 out of the 14 datasets reviewed. Following closely were DESFO-Sv3 and DESFO-Sv4, each showing significant outcomes in 7 of the 14 datasets. Therefore, based on the average features selected, DESFO-Sv1 emerges as the leading method among all the TFs considered.

3) Evaluation of DESFO-TFs using the SVM Classifier

Table X presents the mean accuracy (μ_{Acc}) of DESFO across eight TFs utilizing the SVM classifier. It was found that DESFO-Sv4 significantly outperformed in 10 out of the 14 datasets, with DESFO-Sv1 and DESFO-Sv3 closely behind, excelling in 9 out of the 14 datasets for both of them. As a result, when considering the mean accuracy, DESFO-Sv4 is ranked highest among the evaluated methods.

Table XI presents the mean Fitness Value (μ_{Fit}) for DESFO across eight TFs, utilizing the SVM classifier. It was noted that DESFO-Sv3 demonstrated notable performance in 8 out of the 14 datasets, and DESFO-Sv4 and DESFO-Sv2 followed closely, showing significant results in 7 and 6 out of 14 datasets, respectively. Hence, when considering the mean fitness values, DESFO-Sv3 is the top-performing method among all the TF methods.

Table XII presents the mean features selected (μ_{Feat}) for DESFO across eight TFs, using the SVM classifier for analysis. Both DESFO-Sv2 and DESFO-Sv3 demonstrated remarkable performance in 7 out of the 14 datasets for both of them. Following closely were DESFO-Sv1 and DESFO-Sv4, each showing significant outcomes in 5 and 6 of the 14 datasets. Therefore, based on the average features selected, DESFO-Sv2 and DESFO-Sv3 emerge as the leading methods among all the TFs considered.

E. The Overall Evaluation and Discussion

Table XIII shows the overall evaluation of the DESFO behavior with the eight TFs and ML classifiers regarding the best accuracy, fitness value, and selected feature results.

According to TableXIII, the optimal combinations of the DESFO variant and the three classifiers alongside the eight TFs have been identified, setting the stage for subsequent experiments in this section. The models that emerged as the most effective in terms of μ_{Acc} are DESFO-Vv4–RF and DESFO-S-ve4–SVM. When considering μ_{Fit}, the most effective models identified are DESFO-S-ve1 and DESFO-S-ve3. Furthermore, in terms of μ_{Feat}, DESFO-S-ve1 stands out as the most effective model.

F. Impact of Feature Reduction on Performance and Efficiency

FS often requires a trade-off between reducing the number of selected features and maintaining classification accuracy. This subsection examines scenarios where reduced feature sets lead to performance gains and highlights the roles of V-shaped and S-shaped TFs in achieving these outcomes. By balancing exploration and exploitation, these transfer functions facilitate feature reduction strategies that align with dataset characteristics and classifier requirements.

1) *V-shaped Transfer Functions for Precision-Driven Feature Selection*

V-shaped transfer functions excel in datasets where selective feature retention is critical for classification accuracy. By supporting incremental adjustments, these functions allow for fine-grained exploration in FS, leading to a compact feature set without the unnecessary inclusion of redundant features. This is particularly beneficial in high-dimensional datasets where only a few features are highly informative.

For instance, in datasets like *IonosphereEW* and *WaveformEW* in Table XIV), DESFO with V-shaped transfer functions achieved high accuracy while significantly reducing feature counts. The precision-oriented nature of V-shaped functions helps classifiers like SVM and k-NN achieve optimal performance with a smaller, targeted subset of features, avoiding overfitting and reducing computational cost. This is especially advantageous when interpretability and efficient model training are priorities.

2) *S-shaped Transfer Functions for Aggressive Reduction in Redundant Features*

S-shaped transfer functions are designed for more aggressive feature reduction, which is advantageous in datasets with a high degree of feature redundancy. By pushing values toward binary extremes (0 or 1), S-shaped functions favor exploitation, leading to rapid convergence on a smaller feature subset. This characteristic is particularly beneficial for RF classifiers, which are resilient to feature count variations but benefit from fewer features for faster training and model simplicity.

In datasets like *PenglungEW* and *KrVsKpEW*, DESFO with S-shaped transfer functions demonstrated a high degree of feature elimination while maintaining competitive accuracy. This approach is useful in high-dimensional datasets, where reducing features minimizes computational load without compromising classifier performance, making it ideal for applications requiring fast processing or real-time analysis.

3) *Balancing Exploration and Exploitation for Optimal Feature Subsets*

The choice of V-shaped or S-shaped TFs highlights different strengths in FS, underscoring the need to align function selection with dataset and classifier characteristics. V-shaped functions offer controlled exploration for precise selection in complex feature spaces, while S-shaped functions enable decisive feature elimination for efficient model performance. By modulating the exploration-exploitation balance, DESFO adapts to varying data complexities and optimizes FS across multiple classifiers.

Table XIV illustrates the performance impact of V-shaped and S-shaped TFs on selected datasets, showcasing their roles in balancing accuracy and feature reduction.

As shown in Table XIV, DESFO with V-shaped transfer functions achieved high accuracy with a minimal feature set in *IonosphereEW* and *WaveformEW*, supporting precision in datasets with high feature variability. Conversely, DESFO with S-shaped functions enabled substantial feature reduction in *PenglungEW* and *KrVsKpEW*, underscoring the advantage of the aggressive reduction in datasets with significant redundancy.

6. Discussion

This study underscores the effectiveness of the DESFO algorithm as an advanced FS method when paired with various TFs and ML classifiers. Through extensive testing on benchmark datasets, the results indicate that DESFO optimizes both classification accuracy and computational efficiency, with performance highly influenced by the specific TF and classifier combinations used.

A. *Influence of Transfer Functions and Classifiers on Performance*

The results show that DESFO's performance in FS and classification accuracy varies significantly depending on the type of TF and classifier. V-shaped TFs, particularly Vv4, demonstrated strong performance with the k-NN and RF classifiers, often yielding the highest accuracy on high-dimensional datasets. This suggests that the V-shaped functions support DESFO in making fine-grained adjustments to feature subsets, enhancing precision in tasks that demand accurate, selective feature inclusion.

Conversely, S-shaped TFs, notably Sv4 and Sv3, showed superior outcomes when used with the SVM classifier. The smooth, probabilistic nature of S-shaped TFs, combined with SVM's kernel-based capabilities, resulted in both high accuracy and effective feature reduction. These TFs facilitated gradual changes in FS, allowing DESFO to explore broadly while converging efficiently. This finding highlights the adaptability of S-shaped TFs, making them particularly well-suited for tasks requiring broad generalization, such as complex, multidimensional data classification.

B. *Dimensionality Reduction and Computational Efficiency*

DESFO proved effective at dimensionality reduction across diverse datasets. Notably, Sv1 and Sv4 achieved the highest feature reduction in most benchmarks, especially with the RF classifier. This capacity to reduce the number of selected features while maintaining high accuracy is critical in enhancing computational efficiency and model interpretability, both of which are essential for high-dimensional applications.

The computational complexity analysis further confirms DESFO's suitability for handling complex datasets.

TABLE IV. The mean accuracy μ_{Acc} results using the eight TFs and K-NN with DESFO

Benchmarks	**Eval-Metric**	**V-ve1**	**V-ve2**	**V-ve3**	**V-ve4**	**S-ve1**	**S-ve2**	**S-ve3**	**S-ve4**
BreastCancer	μ_{Acc}	**0.9857**	**0.9857**	**0.9857**	**0.9857**	**0.9857**	**0.9857**	**0.9857**	**0.9857**
BreastEW	μ_{Acc}	0.9649	0.9649	0.9649	0.9649	0.9649	0.9649	**0.9658**	0.9649
Exactly2	μ_{Acc}	**0.7970**	0.7935	0.7880	0.7935	0.7880	0.7945	0.7870	0.7865
IonosphereEW	μ_{Acc}	0.9310	0.9310	0.9324	0.9324	**0.9563**	0.9493	0.9493	0.9535
KrVsKpEW	μ_{Acc}	0.9803	0.9788	0.9803	**0.9822**	0.9789	0.9819	0.9803	0.9814
Lymphography	μ_{Acc}	0.8333	0.8367	**0.8433**	0.8400	0.8333	0.8367	0.8400	0.8333
M-of-n	μ_{Acc}	0.9995	**1.0000**	**1.0000**	**1.0000**	0.9995	**1.0000**	0.9995	**1.0000**
PenglungEW	μ_{Acc}	0.6533	0.6533	0.6600	0.6533	**0.7333**	0.7067	0.6867	0.7200
SonarEW	μ_{Acc}	0.9786	0.9929	0.9857	0.9857	0.9857	0.9857	**0.9952**	0.9857
Tic-tac-toe	μ_{Acc}	**0.8542**	**0.8542**	**0.8542**	**0.8542**	**0.8542**	**0.8542**	**0.8542**	**0.8542**
Vote	μ_{Acc}	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
WaveformEW	μ_{Acc}	0.8445	0.8448	0.8454	**0.8480**	0.8436	0.8449	0.8453	0.8443
WineEW	μ_{Acc}	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Zoo	μ_{Acc}	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Score	(W/T/L)	1/5/8	0/6/8	1/6/7	1/7/6	2/5/7	0/6/8	2/5/7	0/6/8

TABLE V. The mean fitness μ_{Fit} results using the eight TFs and K-NN with DESFO

Benchmarks	**Eval-Metric**	**V-ve1**	**V-ve2**	**V-ve3**	**V-ve4**	**S-ve1**	**S-ve2**	**S-ve3**	**S-ve4**
BreastCancer	μ_{Fit}	**0.0201**	**0.0201**	**0.0201**	**0.0201**	**0.0201**	0.0204	**0.0201**	**0.0201**
BreastEW	μ_{Fit}	0.0366	0.0369	0.0366	0.0368	**0.0355**	0.0359	0.0356	0.0356
Exactly2	μ_{Fit}	**0.2062**	0.2097	0.2154	0.2093	0.2150	0.2084	0.2155	0.2161
IonosphereEW	μ_{Fit}	0.0710	0.0714	0.0698	0.0698	**0.0443**	0.0516	0.0519	0.0472
KrVsKpEW	μ_{Fit}	0.0254	0.0270	0.0256	**0.0232**	0.0267	0.0239	0.0256	0.0250
Lymphography	μ_{Fit}	0.1691	0.1659	**0.1593**	0.1631	0.1694	0.1662	0.1625	0.1691
M-of-n	μ_{Fit}	0.0053	**0.0048**	0.0049	**0.0047**	0.0054	0.0050	0.0055	0.0050
PenglungEW	μ_{Fit}	0.3468	0.3466	0.3400	0.3465	**0.2646**	0.2920	0.3127	0.2777
SonarEW	μ_{Fit}	0.0248	**0.0109**	0.0177	0.0181	0.0174	0.0173	**0.0082**	0.0172
Tic-tac-toe	μ_{Fit}	**0.1544**	**0.1544**	**0.1544**	**0.1544**	**0.1544**	**0.1544**	**0.1544**	**0.1544**
Vote	μ_{Fit}	0.0022	0.0023	0.0023	0.0026	**0.0019**	**0.0019**	**0.0019**	**0.0019**
WaveformEW	μ_{Fit}	0.1596	0.1592	**0.1591**	0.1563	0.1607	0.1594	0.1590	0.1600
WineEW	μ_{Fit}	0.0032	0.0032	**0.0031**	**0.0031**	**0.0031**	**0.0031**	**0.0031**	**0.0031**
Zoo	μ_{Fit}	0.0033	0.0033	0.0034	0.0033	**0.0032**	0.0033	**0.0032**	**0.0032**
Score	W/T/L	1/2/11	0/2/12	1/3/10	3/3/8	3/4/7	0/3/11	1/5/8	0/5/9

The dual-stage optimization strategy—where DE handles initial exploration and SFO fine-tunes through exploitation—enables DESFO to achieve a balance between comprehensive search and rapid convergence. This balance mitigates the risk of premature convergence while allowing a thorough exploration of feature subsets, which is particularly beneficial for high-dimensional FS tasks. DESFO's effectiveness in handling this balance is also evident in its ability to outperform other FS methods on a wide range of datasets.

C. Practical Implications and Applicability

The outcomes of this study have substantial implications for applications requiring robust feature selection. In fields like bioinformatics, where high-dimensional gene expression data are prevalent, DESFO's ability to identify key features without compromising accuracy is particularly valuable. Additionally, the successful pairing of S-shaped TFs with SVM for tasks requiring generalization suggests DESFO's potential in fields like text classification and medical diagnostics, where data complexity often requires broader generalization capabilities.

The versatility in DESFO's performance across different TF and classifier combinations suggests that practitioners can tailor the algorithm to specific data characteristics and performance objectives. For instance, precision-focused applications may benefit from V-shaped TFs with RF classifiers, while tasks needing generalization might lean towards S-shaped TFs with SVM. This flexibility allows for customization, optimizing DESFO's effectiveness across various ML domains.

D. Limitations and Future Research Directions

Despite the promising results, this study identifies some limitations. The computational complexity associated with

TABLE VI. The mean features' number μ_{Feat} results using the eight TFs and K-NN with DESFO

Benchmarks	**Eval-Metric**	**V-ve1**	**V-ve2**	**V-ve3**	**V-ve4**	**S-ve1**	**S-ve2**	**S-ve3**	**S-ve4**
BreastCancer	μ_{Feat}	**6.000**	**6.000**	**6.000**	**6.000**	**6.000**	6.300	**6.000**	**6.000**
BreastEW	μ_{Feat}	5.700	6.500	**5.500**	6.300	**2.400**	3.600	5.100	2.500
Exactly2	μ_{Feat}	6.800	6.900	7.200	**6.300**	6.600	6.500	**6.000**	6.100
IonosphereEW	μ_{Feat}	9.000	10.40	9.700	9.700	**3.500**	4.600	5.900	4.000
KrVsKpEW	μ_{Feat}	21.40	21.50	22.00	**20.10**	**20.80**	21.40	21.90	23.80
Lymphography	μ_{Feat}	7.300	7.500	7.600	8.500	**7.900**	8.100	7.400	**7.300**
M-of-n	μ_{Feat}	6.200	6.300	6.400	**6.100**	6.400	6.500	6.500	6.500
PenglungEW	μ_{Feat}	116.2	110.0	112.0	108.2	**20.00**	52.30	80.60	**17.10**
SonarEW	μ_{Feat}	21.80	23.10	21.10	23.60	**19.40**	**18.70**	20.90	**18.50**
Tic-tac-toe	μ_{Feat}	**9.00**	**9.00**	**9.00**	**9.00**	**9.00**	**9.00**	**9.00**	**9.00**
Vote	μ_{Feat}	3.500	3.700	3.600	4.100	**3.000**	**3.100**	**3.000**	**3.100**
WaveformEW	μ_{Feat}	22.80	22.10	24.00	23.20	23.40	23.50	**23.20**	23.30
WineEW	μ_{Feat}	4.100	4.100	**4.000**	**4.000**	**4.000**	**4.000**	**4.000**	**4.000**
Zoo	μ_{Feat}	5.200	5.300	5.400	5.200	**5.100**	**5.200**	**5.100**	**5.100**
Score	W/T/L	0/3/11	1/2/11	0/3/11	2/3/9	2/5/7	0/2/12	1/5/8	2/5/7

TABLE VII. The mean Accuracy μ_{Acc} results using the eight TFs and RF with DESFO

Benchmarks	**Eval-Metric**	**V-ve1**	**V-ve2**	**V-ve3**	**V-ve4**	**S-ve1**	**S-ve2**	**S-ve3**	**S-ve4**
BreastCancer	μ_{Acc}	**0.9857**	**0.9857**	**0.9857**	**0.9857**	**0.9857**	**0.9857**	**0.9857**	**0.9857**
BreastEW	μ_{Acc}	**0.9974**	0.9956	0.9930	0.9947	0.9939	0.9939	0.9956	0.9930
Exactly2	μ_{Acc}	**0.7650**	**0.7650**	**0.7650**	**0.7650**	**0.7650**	0.7645	**0.7650**	**0.7650**
IonosphereEW	μ_{Acc}	0.9704	**0.9732**	0.9704	**0.9732**	0.9718	0.9704	0.9704	0.9676
KrVsKpEW	μ_{Acc}	0.9486	0.9469	**0.9494**	0.9487	0.9467	0.9483	0.9470	0.9484
Lymphography	μ_{Acc}	0.8900	0.8833	**0.8933**	**0.8933**	0.8733	0.8833	0.8900	0.8767
M-of-n	μ_{Acc}	0.9935	0.9935	**0.9950**	**0.9950**	0.9825	0.9925	0.9870	0.9840
PenglungEW	μ_{Acc}	0.7533	0.7600	0.7467	0.7667	**0.7667**	0.7533	0.7600	**0.7933**
SonarEW	μ_{Acc}	0.9167	0.9238	0.9190	0.9286	0.9310	0.9286	0.9238	**0.9333**
Tic-tac-toe	μ_{Acc}	**0.8698**	**0.8698**	**0.8698**	**0.8698**	**0.8698**	**0.8698**	**0.8698**	**0.8698**
Vote	μ_{Acc}	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
WaveformEW	μ_{Acc}	0.8174	0.8176	0.8181	**0.8197**	0.8163	0.8196	0.8193	0.8193
WineEW	μ_{Acc}	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Zoo	μ_{Acc}	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
score	W/T/L	0/7/7	0/7/7	1/8/5	0/10/4	0/6/8	0/5/9	0/6/8	0/8/6

TABLE VIII. The mean fitness μ_{Fit} results using the eight TFs and RF with DESFO

Benchmarks	**Eval-Metric**	**V-ve1**	**V-ve2**	**V-ve3**	**V-ve4**	**S-ve1**	**S-ve2**	**S-ve3**	**S-ve4**
BreastCancer	μ_{Fit}	**0.0201**	**0.0201**	**0.0201**	**0.0201**	**0.0201**	0.0204	**0.0201**	**0.0201**
BreastEW	μ_{Fit}	0.0366	0.0369	0.0366	0.0368	**0.0355**	0.0359	0.0356	0.0356
Exactly2	μ_{Fit}	**0.2062**	0.2097	0.2154	0.2093	0.2150	0.2084	0.2155	0.2161
IonosphereEW	μ_{Fit}	0.0710	0.0714	**0.0698**	**0.0698**	**0.0443**	0.0516	0.0519	0.0472
KrVsKpEW	μ_{Fit}	0.0254	0.0270	0.0256	**0.0232**	0.0267	**0.0239**	0.0256	0.0250
Lymphography	μ_{Fit}	0.1691	0.1659	**0.1593**	0.1631	0.1694	0.1662	0.1625	0.1691
M-of-n	μ_{Fit}	0.0053	**0.0048**	0.0049	**0.0047**	0.0054	0.0050	0.0055	0.0050
PenglungEW	μ_{Fit}	0.3468	0.3466	0.3400	0.3465	**0.2646**	0.2920	0.3127	0.2777
SonarEW	μ_{Fit}	0.0248	**0.0109**	0.0177	0.0181	0.0174	0.0173	**0.0082**	0.0172
Tic-tac-toe	μ_{Fit}	**0.1544**	**0.1544**	**0.1544**	**0.1544**	**0.1544**	**0.1544**	**0.1544**	**0.1544**
Vote	μ_{Fit}	0.0022	0.0023	0.0023	0.0026	**0.0019**	**0.0019**	**0.0019**	**0.0019**
WaveformEW	μ_{Fit}	0.1596	0.1592	0.1591	**0.1563**	0.1607	0.1594	**0.1590**	0.1600
WineEW	μ_{Fit}	0.0032	0.0032	**0.0031**	**0.0031**	**0.0031**	**0.0031**	**0.0031**	**0.0031**
Zoo	μ_{Fit}	0.0033	0.0033	0.0034	0.0033	**0.0032**	0.0033	**0.0032**	**0.0032**
Score	(W/T/L)	1/2/11	0/2/12	1/3/10	3/3/8	3/5/6	0/3/11	1/5/8	0/5/9

TABLE IX. The mean features' number μ_{Feat} results using the eight TFs and RF with DESFO

Benchmarks	**Eval-Metric**	**V-ve1**	**V-ve2**	**V-ve3**	**V-ve4**	**S-ve1**	**S-ve2**	**S-ve3**	**S-ve4**
BreastCancer	μ_{Feat}	**5.000**	**5.000**	**5.000**	5.100	**5.000**	**5.000**	**5.000**	5.200
BreastEW	μ_{Feat}	12.70	12.80	12.000	12.20	**10.00**	**9.400**	11.10	**9.000**
Exactly2	μ_{Feat}	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
IonosphereEW	μ_{Feat}	15.10	16.00	14.500	14.60	10.50	**9.000**	14.30	**8.900**
KrVsKpEW	μ_{Feat}	15.50	17.90	17.700	17.10	15.90	**14.50**	15.80	15.20
Lymphography	μ_{Feat}	9.000	**8.500**	9.300	9.100	**7.300**	8.600	8.700	7.700
M-of-n	μ_{Feat}	6.400	6.400	6.500	**6.300**	7.000	**6.300**	6.500	6.700
PenglungEW	μ_{Feat}	156.2	150.2	142.20	155.6	**43.60**	81.80	105.5	**39.50**
SonarEW	μ_{Feat}	30.20	29.20	29.300	27.60	24.80	**17.40**	22.90	23.70
Tic-tac-toe	μ_{Feat}	**7.000**	**7.000**	**7.000**	**7.000**	**7.000**	**7.000**	**7.000**	**7.000**
Vote	μ_{Feat}	3.200	3.200	3.500	3.200	**2.000**	2.100	2.200	**2.000**
WaveformEW	μ_{Feat}	18.70	18.80	18.500	20.90	**16.90**	17.90	19.30	19.80
WineEW	μ_{Feat}	**3.000**	**3.000**	**3.000**	**3.000**	**3.000**	**3.000**	**3.000**	**3.000**
Zoo	μ_{Feat}	4.500	4.700	4.600	4.500	**4.000**	4.200	4.100	4.100
Score	(W/T/L)	0/4/10	0/4/10	0/4/10	0/4/10	1/7/6	1/6/7	0/4/10	3/4/7

TABLE X. The mean Accuracy μ_{Acc} results using the eight TFs and SVM with DESFO

Benchmarks	**Eval-Metric**	**V-ve1**	**V-ve2**	**V-ve3**	**V-ve4**	**S-ve1**	**S-ve2**	**S-ve3**	**S-ve4**
BreastCancer	μ_{Acc}	**0.9786**	**0.9786**	**0.9786**	**0.9786**	**0.9786**	**0.9786**	**0.9786**	**0.9786**
BreastEW	μ_{Acc}	0.9474	0.9474	0.9474	0.9474	**0.9544**	0.9500	0.9491	0.9526
Exactly2	μ_{Acc}	**0.7500**	**0.7500**	**0.7500**	**0.7500**	**0.7500**	**0.7500**	**0.7500**	**0.7500**
IonosphereEW	μ_{Acc}	0.9662	0.9648	**0.9676**	0.9662	0.9648	0.9648	**0.9676**	0.9620
KrVsKpEW	μ_{Acc}	0.9820	0.9833	0.9842	0.9831	0.9847	0.9833	0.9808	**0.9850**
Lymphography	μ_{Acc}	0.8533	0.8467	0.8533	0.8533	0.8467	0.8500	**0.8633**	0.8500
M-of-n	μ_{Acc}	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
PenglungEW	μ_{Acc}	0.8200	0.8267	0.8400	0.8400	**0.9267**	0.9000	0.8733	0.9200
SonarEW	μ_{Acc}	0.9476	0.9500	0.9452	0.9429	0.9429	0.9429	0.9429	**0.9548**
Tic-tac-toe	μ_{Acc}	**0.9062**	**0.9062**	**0.9062**	**0.9062**	**0.9062**	**0.9062**	**0.9062**	**0.9062**
Vote	μ_{Acc}	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
WaveformEW	μ_{Acc}	0.8767	0.8773	0.8778	0.8768	0.8765	0.8761	0.8777	**0.8785**
WineEW	μ_{Acc}	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Zoo	μ_{Acc}	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Score	(W/T/L)	0/7/7	0/7/7	0/8/6	0/7/7	2/7/5	0/7/7	1/8/5	3/7/4

TABLE XI. The mean Fitness value μ_{Fit} results using the eight TFs and SVM with DESFO

Benchmarks	**Eval-Metric**	**V-ve1**	**V-ve2**	**V-ve3**	**V-ve4**	**S-ve1**	**S-ve2**	**S-ve3**	**S-ve4**
BreastCancer	μ_{Fit}	**0.0262**	**0.0262**	**0.0262**	**0.0262**	**0.0262**	**0.0262**	**0.0262**	**0.0262**
BreastEW	μ_{Fit}	0.0548	0.0551	0.0550	0.0548	**0.0459**	0.0504	0.0518	0.0476
Exactly2	μ_{Fit}	**0.2483**	**0.2483**	**0.2483**	**0.2483**	**0.2483**	**0.2483**	**0.2483**	**0.2483**
IonosphereEW	μ_{Fit}	0.0370	0.0384	**0.0361**	0.0371	0.0380	0.0378	**0.0357**	0.0409
KrVsKpEW	μ_{Fit}	0.0241	0.0227	0.0220	0.0230	**0.0218**	0.0229	0.0246	**0.0211**
Lymphography	μ_{Fit}	0.1495	0.1557	0.1495	**0.1493**	0.1562	0.1525	**0.1397**	0.1530
M-of-n	μ_{Fit}	0.0051	**0.0048**	0.0050	0.0049	0.0051	0.0049	0.0050	0.0052
PenglungEW	μ_{Fit}	0.1812	0.1746	0.1616	0.1616	**0.0734**	0.1004	0.1275	**0.0801**
SonarEW	μ_{Fit}	0.0558	0.0536	0.0581	0.0605	0.0602	0.0598	0.0599	**0.0487**
Tic-tac-toe	μ_{Fit}	**0.1017**	**0.1017**	**0.1017**	**0.1017**	0.1018	**0.1017**	**0.1017**	**0.1017**
Vote	μ_{Fit}	0.0022	0.0026	**0.0020**	0.0021	**0.0019**	**0.0019**	**0.0019**	**0.0019**
WaveformEW	μ_{Fit}	0.1280	0.1271	**0.1269**	0.1281	0.1284	0.1286	0.1273	0.1272
WineEW	μ_{Fit}	0.0018	0.0018	**0.0015**	0.0018	**0.0015**	**0.0015**	**0.0015**	**0.0015**
Zoo	μ_{Fit}	0.0033	0.0033	0.0034	**0.0032**	0.0033	**0.0032**	**0.0031**	**0.0032**
Score	(W/T/L)	0/3/11	1/3/10	1/4/9	0/3/11	2/4/8	0/5/9	3/5/6	2/5/7

TABLE XII. The mean selected Features μ_{Feat} results using the eight TFs and SVM with DESFO

Benchmarks	Eval-Metric	V-ve1	V-ve2	V-ve3	V-ve4	S-ve1	S-ve2	S-ve3	S-ve4
BreastCancer	μ_{Feat}	**5.000**	8.000	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**	**5.000**
BreastEW	μ_{Feat}	8.200	3.500	8.700	8.000	2.300	2.700	4.200	**2.200**
Exactly2	μ_{Feat}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
IonosphereEW	μ_{Feat}	12.10	11.90	13.80	12.40	10.700	**10.00**	12.40	10.90
KrVsKpEW	μ_{Feat}	22.70	22.00	22.90	22.60	24.000	22.70	**20.10**	22.40
Lymphography	μ_{Feat}	7.800	**7.100**	7.800	7.400	8.000	7.200	7.900	8.100
M-of-n	μ_{Feat}	6.600	**6.200**	6.500	6.400	6.600	6.400	6.500	6.700
PenglungEW	μ_{Feat}	97.00	98.70	103.2	102.4	**26.400**	44.600	68.20	30.30
SonarEW	μ_{Feat}	23.90	24.70	23.30	23.50	21.700	**19.50**	19.90	23.20
Tic-tac-toe	μ_{Feat}	**8.000**	**8.000**	**8.000**	**8.000**	8.100	**8.000**	**8.000**	**8.000**
Vote	μ_{Feat}	3.500	4.200	3.200	3.300	**3.000**	**3.000**	**3.000**	**3.000**
WaveformEW	μ_{Feat}	23.70	22.50	23.60	24.70	24.500	23.90	24.70	**22.50**
WineEW	μ_{Feat}	2.400	2.300	**2.000**	2.300	**2.000**	**2.000**	**2.000**	**2.000**
Zoo	μ_{Feat}	5.200	5.300	5.500	5.100	5.200	5.100	**5.000**	5.100
Score	(W/T/L)	0/3/11	3/2/9	0/4/10	0/3/11	1/4/9	2/5/7	2/5/7	1/5/8

TABLE XIII. The three primary classifiers and their respective top-performing binary versions in the suggested DESFO algorithm

Classifier	Metric	Best-performing TF	# of Superiority Benchmarks
K-NN	μ_{Acc}	DESFO-V-ve4	8
	μ_{Fit}	DESFO-S-ve1	7
	μ_{Feat}	DESFO-S-ve1 and DESFO-S-ve4	7
RF	μ_{Acc}	DESFO-V-ve4	10
	μ_{Fit}	DESFO-S-ve1	8
	μ_{Feat}	DESFO-S-ve1	8
SVM	μ_{Acc}	DESFO-S-ve4	10
	μ_{Fit}	DESFO-S-ve3	8
	μ_{Feat}	DESFO-S-ve2 and S-ve3	7

TABLE XIV. Accuracy and Feature Reduction Achieved by V-shaped and S-shaped Transfer Functions Across Selected Datasets

Dataset	Classifier	TF	ACC (%)	Selected	Total
IonosphereEW	SVM	Vv4	97.2	9	34
WaveformEW	k-NN	Vv3	85.4	12	40
PenglungEW	RF	Sv1	76.7	30	325
KrVsKpEW	RF	Sv3	94.7	15	36

larger datasets, combined with the need for careful TF and classifier tuning, presents challenges. Future research could explore adaptive techniques that dynamically select the optimal TF-classifier combination, reducing the need for manual tuning and enhancing algorithm efficiency. Additionally, investigating DESFO's performance with other metaheuristic algorithms and classifiers could yield insights into it.

7. Conclusions and Future Work

In the study, the behavior of the DESFO algorithm is introduced, merging the DE and SFO algorithms with eight (TFs) divided into two categories: V-shaped and S-shaped. This merger was aimed at improving (FS) strategies. Classifiers such as K-NN, RF, and SVM were employed to assess the efficacy of the selected feature groups and determine classification accuracy. This study tested benchmarks featuring multi-scale features and multi-records to prove its success. Results from the three classifiers were juxtaposed against the performances of the eight V-shaped and S-shaped (TFs). Evaluation metrics included average fitness, accuracy, and the number of attributes chosen. The results revealed that considering mean accuracy, the DESFO algorithm, when paired with the RF classifier and V-shaped V4 (TFs), as well as with the SVM classifier and S-shaped V4 (TFs), performed better than all other configurations on 10 out of 14 benchmarks analyzed. For mean fitness functions, the DESFO technique performed best with the RF classifier and V-shaped V1 (TFs) and with the SVM classifier and S-shaped V3, leading in 8 of the 14 benchmarks. Additionally, regarding the average of chosen attributes, the DESFO algorithm, combined with the RF classifier and equipped with the S-shaped V1, was the standout performer in 8 out of the 14 benchmarks. Further exploration into the effectiveness of DESFO for (FS), utilizing various ML algorithms like Naive Bayes, Logistic Regression, Decision Trees (DT), and more, is warranted. Due to its established prowess in selecting features, the DESFO holds significant potential across multiple domains, including Engineering, software cost estimation, networking security, and healthcare.

Conflict of Interest

The authors assure that there are no conflicts of interest or personal relationships that might appear to affect the research presented in this study.

References

[1] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[2] B. Remeseiro and V. Bolon-Canedo, "A review of feature selection methods in medical applications," *Computers in Biology and Medicine*, vol. 112, p. 103375, 2019.

[3] D. T. Mosa, S. E. Sorour, A. A. Abohany, and F. A. Maghraby, "Ccfd: Efficient credit card fraud detection using meta-heuristic techniques and machine learning algorithms," *Mathematics*, vol. 12, no. 14, p. 2250, 2024.

[4] M. G. Gafar, A. A. Abohany, A. E. Elkhouli, and A. A. A. El-Mageed, "Optimization of gene selection for cancer classification in high-dimensional data using an improved african vultures algorithm," *Algorithms*, vol. 17, no. 8, p. 342, 2024.

[5] R. M. Hussien, A. A. Abohany, A. A. Abd El-Mageed, and K. M. Hosny, "Improved binary meerkat optimization algorithm for efficient feature selection of supervised learning classification," *Knowledge-Based Systems*, vol. 292, p. 111616, 2024.

[6] V. F. Rodriguez-Galiano, J. A. Luque-Espinar, M. Chica-Olmo, and M. P. Mendes, "Feature selection approaches for predictive modelling of groundwater nitrate pollution: An evaluation of filters, embedded and wrapper methods," *Science of the Total Environment*, vol. 624, pp. 661–672, 2018.

[7] A. A. A. El-Mageed, A. E. Elkhouli, A. A. Abohany, and M. Gafar, "Gene selection via improved nuclear reaction optimization algorithm for cancer classification in high-dimensional data," *Journal of Big Data*, vol. 11, no. 1, p. 46, 2024.

[8] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[9] A. A. Abd El-Mageed, A. G. Gad, K. M. Sallam, K. Munasinghe, and A. A. Abohany, "Improved binary adaptive wind driven optimization algorithm-based dimensionality reduction for supervised classification," *Computers & Industrial Engineering*, vol. 167, p. 107904, 2022.

[10] S. E. Abdallah, W. M. Elmessery, W. Z. Hassan, N. S. Al-Sattary, A. A. Abohany, and A. Elashry, "Appropriate and optimal classifier for beef quality discrimination by a low-cost optical apparatus," 2023.

[11] S. E. Sorour, L. Hassan, A. A. Abohany, and R. M. Hussien, "An improved binary crayfish optimization algorithm for handling feature selection task in supervised classification," *Mathematics*, vol. 12, no. 15, p. 2364, 2024.

[12] M. Abedi, G. H. Norouzi, and A. Bahroudi, "Support vector machine for multi-classification of mineral prospectivity areas," *Computers & Geosciences*, vol. 46, pp. 272–283, 2012.

[13] A. A. Zhigljavsky, *Theory of Global Random Search (Mathematics and its Applications)*. Springer, 1991.

[14] H. E. Abdelkader, A. G. Gad, A. A. Abohany, and S. E. Sorour, "An efficient data mining technique for assessing satisfaction level with online learning for higher education students during the covid-19," *IEEE Access*, vol. 10, pp. 6286–6303, 2022.

[15] A. H. Salem, S. M. Azzam, O. Emam, and A. A. Abohany, "Advancing cybersecurity: a comprehensive review of ai-driven detection techniques," *Journal of Big Data*, vol. 11, no. 1, p. 105, 2024.

[16] K. M. Sallam, A. A. Abohany, and R. M. Rizk-Allah, "An enhanced multi-operator differential evolution algorithm for tackling knapsack optimization problem," *Neural Computing and Applications*, vol. 35, no. 18, pp. 13 359–13 386, 2023.

[17] E. El-shafeiy, K. M. Sallam, R. K. Chakrabortty, and A. A. Abohany, “A clustering based swarm intelligence optimization technique for the internet of medical things,” *Expert Systems with Applications*, vol. 173, p. 114648, 2021.

[18] A. A. Abd El-Mageed, A. A. Abohany, and A. Elashry, “Effective feature selection strategy for supervised classification based on an improved binary aquila optimization algorithm,” *Computers and Industrial Engineering*, vol. 181, p. 109300, 2023.

[19] M. Mafarja *et al.*, “Binary dragonfly optimization for feature selection using time-varying transfer functions,” *Knowledge-Based Systems*, vol. 161, pp. 185–204, 2018.

[20] J. C. Bansal and K. Deep, “A modified binary particle swarm optimization for knapsack problems,” *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11 042–11 061, 2012.

[21] S. Mirjalili and A. Lewis, “S-shaped versus v-shaped transfer functions for binary particle swarm optimization,” *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, 2013.

[22] R. Storn and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[23] S. Shadravan, H. R. Naji, and V. K. Bardsiri, “The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems,” *Engineering Applications of Artificial Intelligence*, vol. 80, pp. 20–34, 2019.

[24] A. G. Gad, K. M. Sallam, R. K. Chakrabortty, M. J. Ryan, and A. A. Abohany, *An improved binary sparrow search algorithm for feature selection in data classification*. Springer London, 2022, vol. 34.

[25] I. Klyueva, L. Demidova, and A. Pylkin, “Hybrid approach to improving the results of the svm classification using the random forest algorithm,” in *Procedia Computer Science*, vol. 150, 2019, pp. 455–461.

[26] Y. Chen *et al.*, “A hybrid binary dragonfly algorithm with an adaptive directed differential operator for feature selection,” *Remote Sensing*, vol. 15, no. 16, p. 3980, 2023.

[27] H. Chantar *et al.*, “Feature selection using binary grey wolf optimizer with elite-based crossover for arabic text classification,” *Neural Computing and Applications*, vol. 32, pp. 12 201–12 220, 2020.

[28] A. Fatahi, M. H. Nadimi-Shahraki, and H. Zamani, “An improved binary quantum-based avian navigation optimizer algorithm to select effective feature subset from medical data: A covid-19 case study,” *Journal of Bionic Engineering*, vol. 21, no. 1, pp. 426–446, 2024.

[29] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[30] S. M. Azzam, O. E. Emam, and A. S. Abolaban, “An improved differential evolution with sailfish optimizer (desfo) for handling feature selection problem,” *Scientific Reports*, vol. 14, no. 1, p. 13517, 2024.

[31] A. A. Samir, A. R. Rashwan, K. M. Sallam, R. K. Chakrabortty, M. J. Ryan, and A. A. Abohany, “Evolutionary algorithm-based convolutional neural network for predicting heart diseases,” *Computers & Industrial Engineering*, vol. 161, p. 107651, 2021.

[32] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[33] Y. Wu, K. Ianakiev, and V. Govindaraju, “Improved k-nearest neighbor classification,” *Pattern Recognition*, vol. 35, no. 10, pp. 2311–2318, 2002.

[34] M. Mafarja and S. Mirjalili, “Whale optimization approaches for wrapper feature selection,” *Applied Soft Computing*, vol. 62, pp. 441–453, 2018.

[35] M. J. Zaki and W. Meira, *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

[36] R. Katuwal, P. N. Suganthan, and L. Zhang, “An ensemble of decision trees with random vector functional link networks for multi-class classification,” *Applied Soft Computing*, vol. 70, pp. 1146–1153, 2018.

[37] H. Cao, S. Bernard, R. Sabourin, and L. Heutte, “Random forest dissimilarity based multi-view learning for radiomics application,” *Pattern Recognition*, vol. 88, pp. 185–197, 2019.

[38] A. Criminisi, J. Shotton, and E. Konukoglu, “Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning,” *Foundations and Trends in Computer Graphics and Vision*, vol. 7, no. 2–3, pp. 81–227, 2012.

[39] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[40] A. Tharwat, A. E. Hassanien, and B. E. Elnaghi, “A ba-based algorithm for parameter optimization of support vector machine,” *Pattern Recognition Letters*, vol. 93, pp. 13–22, 2017.

[41] A. Frank, “Uci machine learning repository,” 2010. [Online]. Available: http://archive.ics.uci.edu/ml