Sustainable Machine Intelligence Journal

Journal Homepage: sciencesforce.com/smij

Sustain. Mach. Intell. J. Vol. 8 (2024) 56–98

Paper Type: Original Article

Mantis Search Algorithm Integrated with Opposition-Based Learning and Simulated Annealing for Feature Selection

Samia Mandour [1,*], Abduallah Gamal [1] and Ahmed Sleem [2]

[1] Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Zagazig, Sharqiyah, 44519, Egypt; Emails: samia.rmdan@fci.zu.edu.eg; abduallahgamal@zu.edu.eg.

[2] Department of Computer Science, Faculty of Computers and Informatics, Tanta University, Tanta, Egypt; ahmed.selim@ics.tanta.edu.eg.

Received: 20 Jan 2024 **Revised**: 15 May 2024 **Accepted**: 15 Jun 2024 **Published**: 18 Jun 2024

Abstract

Feature selection (FS) plays a vital role in minimizing the high-dimensional data as much as possible to aid in enhancing the classification accuracy and reducing computational costs. The purpose of the FS techniques is to extract the most effective subset features, which might enable the machine learning (ML) algorithms to better grasp the input data's patterns and improve their classification performance. Although several metaheuristic algorithms have been recently presented to solve this problem, they still suffer from several disadvantages, such as getting stuck in local optima, slow convergence speed, and a lack of population diversity, which prevent them from achieving the desired solutions in an acceptable time. Therefore, this study is presented to propose a new feature selection approach, namely OBMSASA, based on integrating the recently published mantis search algorithm with the opposition-based learning (OBL) method and simulated annealing (SA) to strengthen its exploration and exploitation operators. The OBL method aims to improve the exploration operator, making the algorithm able to avoid stagnation into local minima; meanwhile, the SA is used as a local search to further strengthen the exploration operator, thereby improving the convergence speed. The K-nearest neighbor algorithm is used to compute the accuracy of the selected feature. The proposed algorithm is assessed using 21 common datasets and compared to several rival optimizers in terms of several performance metrics, including convergence curve, average fitness, computational cost, length of selected features, and standard deviation, to observe its effectiveness and efficiency. The numerical findings demonstrated the proposed algorithm's superiority over its competitors. The source code is publicly accessible at https://drive.mathworks.com/OBMSASA.

Keywords: Data Mining; Feature Selection; Machine Learning; Mantis Search Algorithm; Simulated Annealing.

1 | Introduction

Data has recently grown in importance as a source for several disciplines, including data science and data mining. Handling enormous data dimensions is one of the difficult issues that data mining presents. The dimensionality of the data could make data mining difficult. Moreover, it requires large computation time and space expenditures. These enormous datasets are too big for conventional machine-learning techniques to handle. The dataset includes a set of instances or instances representing information regarding a certain case.

Corresponding Author: samia.rmdan@fci.zu.edu.eg

https://doi.org/10.61356/SMIJ.2024.8300

Every sample has a unique set of characteristics. The problem with the dataset is not just its enormous dimension sizes; it also includes attributes that are redundant or unimportant. Furthermore, the collected dataset can have a high level of noise, and the model might be complex. These issues raise computing costs and reduce the accuracy of machine learning (ML) techniques. Therefore, the feature selection (FS) as a preprocessing step is used to pick the best subset of the valuable characteristics to reduce the computational cost and improve the classification accuracy of the ML classifiers. The use of feature selection to lessen the effects of data dimensionality has proven to be quite effective. Locating the optimal selected feature (OSF) is a NP-hard optimization problem because it requires observing a huge number of combinations to reach the best one that could simultaneously optimize both the number of selected features and classification accuracy. Several feature selection techniques have been presented in the literature, which is divided into three categories: Filter, wrapper, and embedding techniques [1]. The filter method assesses the chosen subset of features based on the data's properties so, we can say that it always focuses on the broad features of the data. In contrast, the wrapper methods employ a ML classifier to observe the quality of the selected features. Those methods yield more accurate results than filters, but they are expensive in terms of computational cost. Embedded techniques are a combination of filters and wrapping methods. When using embedded approaches, feature selection happens concurrently with the classifier during the training phase [2]. Although wrapper procedures are slower, they yield better results than filter methods. Due to the effectiveness of wrapper-based FS techniques, they are extensively employed in the literature to optimize the FS problem for several fields. Among those techniques, the metaheuristic algorithms could achieve outstanding outcomes when applied to solve this problem due to their strong exploration and exploitation characteristics. The meta-heuristic techniques could achieve outstanding results for several optimization problems, including both continuous and combinatorial problems, in a reasonable amount of time. The majority of those techniques are based on two phases: exploration and exploitation. In the first phase, known as exploration, the search area is more extensively examined as the algorithm searches for the most promising places. The second step of the metaheuristic begins by scouring the most promising regions in greater detail to identify even better solutions.

According to some papers [3], Metaheuristic algorithms are categorized into five categories as follows: the first category is called evolutionary-based metaheuristic algorithms, these algorithms including optimization paradigms that are based on evolutionary mechanisms, such as biological genetics and natural selections, genetic algorithms (GA) [4], and differential evolution (DE) [5] are examples of evolutionary-based metaheuristic algorithms. Human-based metaheuristics are the second category, the foundation for introducing human-based metaheuristic methods is a mathematical simulation of a variety of human behaviors. Teaching-Learning-Based Optimization (TLBO) [6], Poor and Rich Optimization (PRO) [7], and Human Mental Search (HMS) [8] are examples of the Human-based metaheuristics. Swarm-based is the third category, they are designed to mimic the swarming habits of birds, mammals, and other natural organisms. some of the well-known algorithms that come to mind are Particle Swarm Optimization (PSO) [9], Ant Colony Optimization (ACO) [10], and Firefly Algorithm (FA) [11]. The mathematical representation of different physical laws and occurrences serves as the foundation for the building of physics-based metaheuristic algorithms, which is the fourth category of metaheuristics. Simulated Annealing (SA) [12] and the Gravitational Search Algorithm (GSA) [13] are two examples of widely recognized physics-based algorithms. Mathematics based is considered the final category of metaheuristic algorithms, it based on the mathematics mechanisms. Arithmetic optimization algorithm (AOA) [14], and sine cosine algorithm [15] are examples of the mathematics based metaheuristic.

Many metaheuristic algorithms are presented in this respect to address feature selection challenges. However, we argue that previous research has some shortcomings, including poor convergence, local optima trapping, and increased computation durations. The previous problems were the motivation to present our proposed model. The proposed opposition-based mantis search simulated annealing (OBMSASA) utilizes an improved version of the mantis search algorithm (MSA) based on an opposition-based learning method as the first phase, this mechanism improves the algorithm's ability to explore, making it able to provide better quality solutions. Secondly, the opposition-based mantis search algorithm (OBMSA) is hybridized with simulated annealing (SA). The convergence speed is increased by using SA as a local search to reinforce the exploratory operator even more. For assessing the performance of the proposed OBMSASA, twenty-one datasets were employed to track the effectiveness of the suggested technique. A comparative analysis was conducted using multiple recently published methods to address the feature selection issue, such as Discrete equilibrium optimizer combined with simulated annealing for feature selection (EOSA) [16], two-phase mutation gray wolf optimizer (TPGWO) [17], hybrid harris hawks optimization simulated annealing algorithm (HHOSA) [18], slime mould algorithm marine predators algorithm (SMAMPA) [19], sine cosine algorithm (SCA) [20], opposition based learning salp swarm algorithm (OBSA)[21], crossover cooperative whale optimization algorithm (CCWOA)[22], and the standard mantis search algorithm (MSA) [23].

The main contribution of this work is finding the best subset of features using a hybrid approach between an enhanced mantis search algorithm and simulated annealing addresses most of the constraints found in the previous studies. Since the SA can accept a subpar solution based on probability, it is hybridized with the Mantis Search Algorithm to escape from the local optima and improve population diversity as well. Moreover, using opposition-based learning in MSA can increase the diversity of the original population. Different and large-dimensional dataset sizes are utilized to assess the efficacy of the proposed method.

The remainder of the paper is arranged as follows: Section 2 discusses some recently published techniques for the FS; Section 3 briefly describes the K-nearest neighbor approach, the MSA algorithm, the opposition-based learning method, and simulated annealing (SA) as the main components of the proposed algorithm; Section 4 discusses the proposed algorithm; Section 5 provides numerical results and discussion; and the conclusions and recommendations for the future are presented in Section 6.

2 | Related Work

Large datasets present a significant challenge to machine learning techniques because of their high dimensionality, which might hinder data mining. Applications that use datasets with a lot of dimensions must therefore raise the classification parameters. Consequently, the classifier's performance considerably deteriorates. According to this principle, there is an urgent need to use methods for dimensionality reduction. Dimensional reduction is one well-liked method to get rid of noise and unnecessary features. It is a useful technique for increasing model generalization, reducing computational complexity, increasing precision, and reducing the amount of storage needed. One of the most popular techniques used in solving this issue is the feature selection process. A lot of well-known feature selection methods are used to solve the problem of high dimensionality; Metaheuristics is one of these methods that has become widespread recently. The different feature selection methods are indicated in Figure 1.

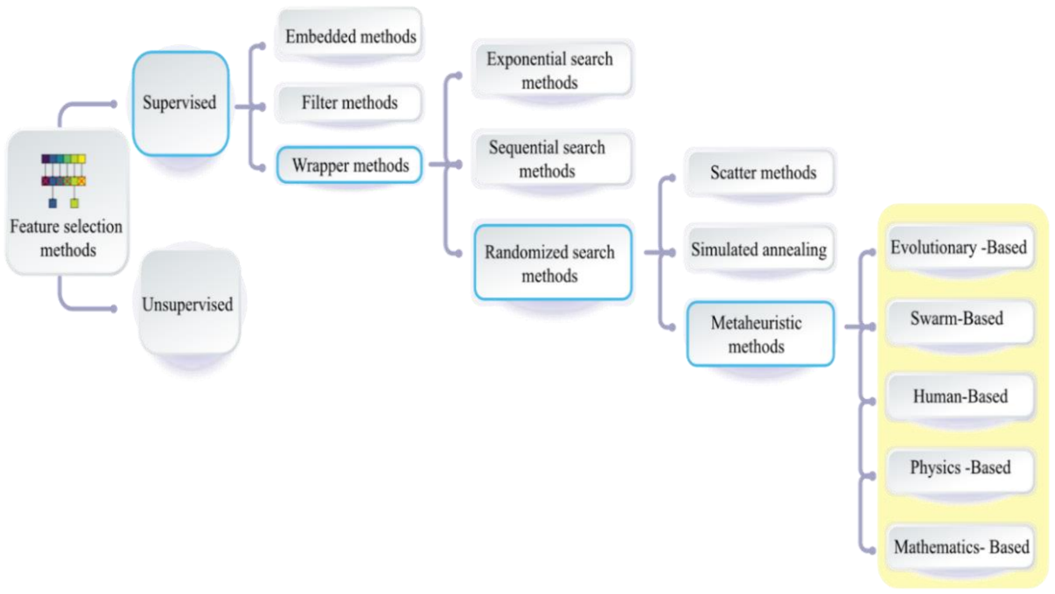


Figure 1. Feature selection methods techniques.

Feature selection methods aim to find the best subset of the features, keeping in mind the efficiency of classification as the priority. The problem of identifying the optimal subset of characteristics is classified as NP-Hard[24] The primary tasks carried out by metaheuristic algorithms can be illustrated in Figure 2.

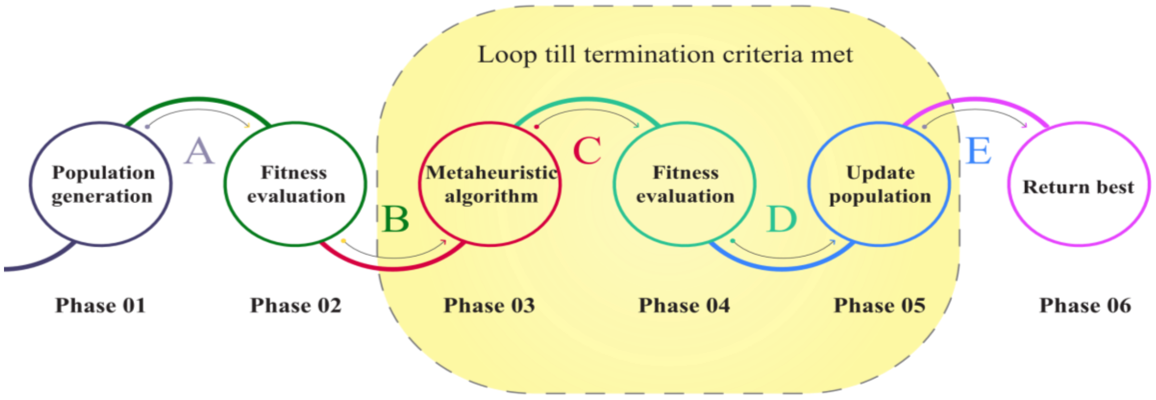


Figure 2. Feature selection cycle using metaheuristic algorithms.

To find the optimally selected feature, the scholars developed a number of metaheuristic approaches. Thus, we shall look into a number of those algorithms. Lately, to address the feature selection problem, a variation of the Vortex Search Algorithm (VSA) integrated with different chaotic maps has been investigated as a way to enhance the VSA operators and aid in controlling both exploitation and exploration [25]. This method's effectiveness was assessed using 24 UCI benchmark datasets. In [26], the authors suggested that MetaSCA stands for hybrid metaheuristic optimization. It is based on a golden sine strategy and a multilevel regulatory factor strategy for feature selection, as well as an enhanced sine cosine algorithm. Seven UCI datasets are employed in the evaluation process. The outcomes demonstrated that it attains superior performance in terms of accuracy and the ideal feature subset. This module's drawback was that it took a long time to extract the optimal feature subset from a large number of features; hence, it still requires a lot of work to enable a noticeable boost in the speed of the feature selection process. Another module was introduced by the authors of [27]. The goal of this study was to introduce a new feature selection (FS) technique by enhancing Gorilla Troops Optimizer (GTO) performance through the use of the GTO-BSA technique, a method for bird swarms (BSA). By using BSA, with a great ability to identify the viable regions that offer the optimum solution, the performance of GTO was improved. The testing results demonstrated that the suggested GTO-BSA method outperformed several existing metaheuristic algorithms in terms of results. One of the research's limitations is that it does not address all multi-objective challenges. Furthermore, in order to identify the best feature subsets from the NSL-KDD dataset, this research [28] suggests an innovative feature selection technique that makes use of a genetic algorithm (GA). Moreover, decision trees (DT) and logistic regression (LR) have been used in hybrid classification to improve accuracy (ACC) and detection rate (DR). This study optimized the chosen ideal features by applying and contrasting the performance of multiple meta-heuristic techniques. Despite providing good accuracy, the suggested task has certain drawbacks. This disadvantage is that, in addition to increasing complexity, the suggested method takes longer to converge, which could be expensive computationally. Another method in [29], in which the Sine-Cosine hybrid optimization algorithm is combined with a modified whale-optimization approach to handle feature selection and achieve high accuracy, is called SCMWOA. On 19 datasets, SCMWOA is evaluated. The results demonstrate how accurate the SCMWOA algorithm is.

Jun Li et al. [30] provide a better-hybridized salp swarm algorithm that is based on the first two stages of the TLSSA teaching-learning-based optimization technique. Although TLSSA produces higher results, it is limited to use on four feature selection datasets. Furthermore, a modified SSA method known as quantized SSA (QSSA) is recommended by [31] to increase performance. The quantization operator, a mathematical operator, is integrated into the fundamental SSA in the proposed method to choose the best features from benchmark datasets while maintaining accuracy. To lessen the dimensionality of agricultural disease detection, Sonal Jain et al. [32] presented a binary version of the memetic salp swarm optimization method (MSSOA), which finds the ideal number of characteristics for the best classification accuracy. The findings show that the suggested approach performs better than the other algorithms in terms of achieving accurate classification and minimizing the size of the feature.

Amel Ali Alhussan et al. [33] proposed an innovative feature selection technique that uses the KNN classifier and the binary version of the waterwheel plant's method of prey selection (bWWPA) as communication to find the optimal feature combination. Thirty datasets from the UCI machine learning repository were used in experiments to test the robustness and stability of the suggested bWWPA approach. A nonlinear binary grasshopper whale optimization algorithm (NL-BGWOA) is an amalgamated algorithm that is put forth by the authors of [34]. The suggested method maximizes the breadth of exploration in the target region by expressing a new position update strategy that combines the position variations of the whale and grasshopper populations. For assessment, ten different high-dimensional UCI datasets are used. NL-BGWOA produces good results; however, when it comes to datasets with fewer features, its fitness and accuracy of classification still need to be improved. Mustafa Serter Uzer et al. [35] introduced a new binary hybrid optimization-based wrapper feature selection technique called BWPLFS is put forth. It combines the Lévy Flight, Particle Swarm Optimization, and Whale Optimization Algorithms. To assess the suggested algorithm's performance, some common benchmark datasets are taken from the UCI repository.

Mahmoud Ragab [36] introduced a binary combination of currently available meta-heuristic methods, the particle swarm optimization (PSO) algorithm and the firefly algorithm (FA), that combines the best aspects from each method to offer an optimized and effective method of addressing the feature selection problem and is used to handle high-dimensionality datasets. Moving to [37], the authors address the drawbacks of the standard grasshopper optimization algorithm (GOA) by strengthening its global optimization capability and hindering it from getting into the local optimum trap by integrating elite opposition-based learning and Gaussian bare-bones into the GOA. The suggested module still has a lengthy computation time, even though it achieves good results in terms of accuracy and obtaining the best subset of features.

To tackle feature selection problems, a lot of algorithms are introduced in this regard. One of them, EOSA, in the article [16] suggests a binary adjusting of the recently suggested meta-heuristic, discrete equilibrium optimizer (EO), boosted with simulated annealing (SA). This procedure is employed as a local search process to improve the exploitation capability. The authors use the proposed EOSA method on eighteen popular UCI datasets and compare it with many other algorithms. EOSA exhibits strong performance on several high-dimensional datasets. To tackle feature selection for classification issues based on wrapper approaches, Abdel-Basset et al. suggested a new Grey Wolf Optimizer algorithm incorporated with a two-phase mutation called (TMGWO) [17]. Another study introduced a hybrid variant of the Harris Hawks Optimization algorithm (HHOSA) that uses wrapper approaches to solve the FS issue for the sake of classification [18]. HHOSA relies on bit-wise processing and Simulated Annealing. Recently, an improved version of the slime mold algorithm called SMAMPA has been introduced to address FS problems [19]. This version relies on the Marine Predators Algorithm (MPA) operators, which play the role of a local search technique, so it helps SMA increase the rate of convergence and prevents the attraction to local optima. A sine-cosine algorithm is introduced to make an appropriate tradeoff between selecting the optimal subset of features and maximizing the accuracy of classification [38]. Some authors introduced an enhanced version of the salp swarm algorithm to tackle feature selection problems and choose the best subset of features in wrapper mode. The original SSA algorithm was modified in two key ways to address its shortcomings and make it suitable for feature selection issues [21]. Opposition-Based Learning (OBL) is used during the SSA's startup phase to increase the diversity of populations in the area of search, which is the first improvement. The creation and application of a new local search algorithm with SSA to enhance its exploitation constitutes the second enhancement. Another study presented a novel strategy called Horizontal Crossover and Cooperative-hunting-based WOA (CCWOA) to solve these shortcomings [22]. The WOA framework is strengthened by this algorithm by adding a weight, horizontal crossover approach, and cooperative learning methods.

With remarkable success, metaheuristic approaches were developed to address a wide range of contemporary and emerging issues. As a result, numerous academics used metaheuristic algorithms to quickly address FS problems. Nonetheless, we contend that prior research suffers from several flaws, such as:

- Poor convergence and getting trapped in LO.
- The lengthening of computation times.
- Unfortunately, huge data dimensions may have an impact on the algorithm's performance.
- It takes time to prepare the algorithm's parameters before having to select the ideal configuration.

A hybrid approach between an improved mantis search algorithm and simulated annealing is looking for the optimal subset of features to address the majority of the constraints discovered in the earlier investigations. Since the SA can accept an inferior solution depending on probability, it is hybridized with the Mantis Search Algorithm to escape from the local optima and enhance population diversity as well. Furthermore, applying opposition-based learning to MSA can broaden the initial population's diversity. Various and large-dimensional dataset sizes are employed to examine the effectiveness of the suggested approach.

3 | Mantis Search Algorithm

Abdel-Basset introduced a nature-inspired algorithm that mimics the physical and behavioral methods used by mantises to protect their prey and evade predators [23]. There are about 2400 kinds of this kind of bug around the globe, grouped into 434 genera. The bug is distinguished by its long body, triangular face with two antennae and exophthalmic eyes that are compound, and elastic neck that allows certain species to rotate the head around 180 degrees. Ants, scorpions, and wasps are the usual food sources for mantises. Small mantises can be consumed by large mantises as well.

3.1 | MSA's Mathematical Model

The three primary MSA stages are shown mathematically in this section and are explained in brief in the following order: The initial positions of the mantises (population initialization) are the initial stage, which is in charge of randomly assigning the mantises within the optimization's search space. (ii) The second step is the phase of exploration, or "looking for prey," which imitates the actions taken by the mantises to locate their prey. (iii) The third step is the phase of exploitation, which imitates the mantises' attacking behavior. (iv) The fourth phase indicates sexual cannibalism. (v) The final stage describes the method of retrieving the solutions that are located outdoors in the search space. All these stages will be mathematically formulated in the following sections.

3.1.1 | Initialization

The suggested algorithm begins with an initial group of mantises, just like population-based approaches do. In a mantis optimization methodology, every mantis stands for a potential solution for an optimization issue. A two-dimensional matrix m of size $Num \times D$ can represent a population of Num mantises (solutions) in a D-dimensional search space. A vector of the following form can be used to define the position of the mantis i for function assessment t:

$$m_i^t = [m_{i,1}^t, m_{i,2}^t, \dots, m_{i,D}^t] \tag{1}$$

Where i indicates the current solution that belongs to the set $\{1,2,\dots\dots,Num\}$; t stands for the function evaluation that is now in effect; D denotes the problem's dimension; and the i^{th} mantis' position can be represented by m_i^t. The i^{th}mantis's m_i^t initial vector in the search space can be randomly generated using the equation below:

$$m_i^t = m_j^l + r \times (m_j^u - m_j^l) \tag{2}$$

where $m_j^u\, and\ m_i^l$ stand for the j-dimension's upper and lower bounds, correspondingly. A random number $\in [0,1]$. Every time the mantis moves into a new place, the potency of the solution is assessed based on the fitness function. The following is an update of the current position. The mantis shifts to the new place when the solution quality there is superior to the one it is now in. If not, the MOA approach keeps the mantis at its current location.

3.1.2 | Exploration Phase (Searching for Prey)

Mantis shrimp are divided into two groups: smashers and spearers. The smashers hunt their prey far away in their natural habitat, on the ground and the leaves and branches of trees, while the spearers wait for their prey to approach their burrows and strike. This novel algorithm simulates this phase in two ways: first, it attempts to mimic the actions of the smashers, which search other areas for their victim, and second, it mimics the behaviors of the spearers, which wait for their prey in a concealed position before pounces.

3.1.2.1 | Smasher's Exploration Behavior

Those attackers use a variety of step sizes, including long, little, and surprise orientations, to hunt food outside of their burrows. This behavior covers both short and large step sizes by incorporating the Levy flight and normal distribution, while the surprise orientation is randomized and imitated.

The levy flight produces small step sizes that require huge function assessments to reach the desired solution, making it impractical when used alone. In contrast, the normal distribution produces huge numbers that push the solution into distant positions and subsequently discard an extensive number of solutions. Hence, to simulate the behaviors of the smashers while they hunt for their victims, the authors study recombination between these to produce distinct sequences of numbers in this research, supporting both little and comparatively high numbers. Thus, the steps produced by hybridization are in between those of a very large one and a very small one. At last, the following is the mathematical framework for this behavior:

$$\vec{m}_i^{t+1} = \begin{cases} \vec{m}_i^t + \vec{\tau_1}.(\vec{m}_i^t - \vec{m}_a^t) + |\tau_2|.\vec{U}.(\vec{m}_a^t - \vec{m}_b^t), & n_1 \leq n_2 \\ m_i^t * \vec{U} + \left(\vec{m}_a^t + n_3.(\vec{m}_b^t - \vec{m}_c^t)\right).\left(1 - \vec{U}\right), & Otherwise \end{cases} \tag{3}$$

Where $\vec{m}_i^t$ is the location of the i^{th} mantis at the function assessment t, $|\tau_2|$ is a random number derived from the normal distribution, and $n_1, n_2,$ and n_3 are three randomly produced values between 0 and 1. $\vec{\tau_1}$ is a vector of values constructed using the Levy-flight approach. $\vec{m}_a^t$, $\vec{m}_b^t$ and $\vec{m}_c^t$ are three mantis, which were selected in a random manner from the current population, such that $\vec{m}_a^t \neq \vec{m}_b^t \neq \vec{m}_c^t \neq \vec{m}_i^t$. a binary vector $\vec{\text{U}}$ is produced regarding to the next formula:

$$\vec{U} = \begin{cases} 0 & \vec{n_4} < \vec{n_5} \\ 1 & otherwise \end{cases} \tag{4}$$

Where $\vec{n_4}$ and $\vec{n_5}$ are two random vectors including values between 0 and 1. The first mathematical formula in Eq. (3) imitates the hybrid movements, whereas the second formula imitates the sudden orientation of the movements.

3.1.2.2 | Smasher's Exploration Behavior

These predators' means of establishing archives that include the locations of several burrows, where they wait for prey to approach and strike, are used to imitate their exploratory habits (Smasher behavior). Every mantis' local-best solutions are assigned to this archive, which is filled by randomly selecting one solution from inside it to replace the old one. With the help of their 180-degree rotating eyes located in their heads, these predators use them to survey their surroundings. The formula below is used to imitate this behavior:

$$\vec{m}_i^{t+1} = \vec{m}_i^t + \alpha.(\vec{m}_{ar}' - \vec{m}_a^t) \tag{5}$$

To allow the mantis to cross the ambush's distance, α is a parameter that controls its position. This parameter can be mathematically defined as described below:

$$\alpha = \cos(\pi n_5) . \left(1 - \frac{t}{MaxT}\right) \tag{6}$$

wherein n_5 is a number that is generated at random in the range of [0,1]. The function assessments' maximum number is denoted by $MaxT$. Mimicking the movements of prey as they seek their prey in their surroundings; nevertheless, this behavior can be determined to bring the prey inside ambush range employing the subsequent formula:

$$\vec{m}_i^{t+1} = \vec{m}'_{ar} + (n_2 * 2 - 1) * \mu * \left(m_j^l + n \times \left(m_j^u - m_j^l\right)\right) \tag{7}$$

where the upper and lower boundaries for the j-dimension are denoted, respectively, by m_j^u and m_j^l. A number chosen at random between in the range of [0,1] is called n. $\vec{m}'_{ar}$ represents a solution that was chosen at random from the repository to symbolize the i^{th} mantis' burrow. μ, a distance parameter that regulates the prey's position, is calculated using the subsequent formula:

$$\mu = \left(1 - \frac{t}{MaxT}\right) \tag{8}$$

Furthermore, the following mathematical formulation represents the actions of ambush behavior mantises and their prey:

$$\vec{m}_i^{t+1} = \begin{cases} \vec{m}_i^t + \alpha . (\vec{m}'_{ar} - \vec{m}_a^t) \, , & m_2 \le m_3 \\ \vec{m}'_{ar} + (n_2 * 2 - 1) * \mu * \left(m_j^l + n \times \left(m_j^u - m_j^l\right)\right), & Otherwise \end{cases} \tag{9}$$

In this case, the two numbers, m_2 and m_3, are chosen at random in the range of [0,1] for exchanging the behaviors of ambush hunters and prey. Lastly, there are the two exploratory behaviors of spearers (Eq. (9)) and ambushers (Eq. (3)). The recycling control factor (RCF) is a factor that can be used to balance various behaviors during the optimization process. The following is the mathematical formulation for this factor:

$$\text{RCF} = \left(1 - \frac{\left(t\%\left(\frac{\text{MaxT}}{\text{P}}\right)\right)}{\left(\frac{\text{MaxT}}{\text{P}}\right)}\right) \tag{10}$$

Where P represents an integer.

3.1.3 | Attack the Target: Stage of Exploitation

The two phases of a mantis's prey-catching action are their approach and the sweeping motion [39]. A mantis raises and spreads its arms during the first stage, known as the approach phase. The mantis gathers its prey at a fast speed and drags it in to consume it during the second phase, known as the sweeping phase. It's interesting to note that the mantis may gauge its distance from its target before choosing to sweep (strike) [40]. The mantis hovers at an appropriate angle before the strike and strikes the prey quickly. A mantis will frequently fix its error with a similar pause if it initially misjudges the velocity of its prey. As a result, two essential components of the hunting process's effectiveness are the assessment of the distance that exists between the predator and the target, or striking distance, and the velocity of the assault, or strike speed. To model this behavior analytically, the following three steps must be taken:

- Calculating the strike distance(d_{st}).
- Determining the striking speed (v_{st}).
- The mantis tries to strike again in this instance, taking into account the strike's failure.

i). Calculating the strike distance(d_{st}).

The following formula can be used to determine the mantis's strike distance at the function execution t.

$$tan\, \beta_{1_i}^t = d_{st_i}^t / B_i^t \; and \; \tan \alpha_{1_i}^t = d_{st_i}^t / \left(1 - B_i^t\right) \tag{11}$$

$$tan\,\beta_{2_i}^t = b_i^t/mi_{R_i}^t \; and \; \tan\alpha_{2_i}^t = b_i^t/mi_{L_i}^t \tag{12}$$

Because $L_{1_i}^t$ and $L_{2_i}^t$ are parallel, then $\beta_{1_i}^t = \beta_{2_i}^t$ and $\alpha_{1_i}^t = \alpha_{2_i}^t$. Therefore, $d_{st_i}^t/B_i^t = b_i^t/mi_{R_i}^t$ and $d_{st_i}^t/(1-B_i^t) = b_i^t/mi_{L_i}^t$. Let $b = 0.5 * B_i^t$, then, the above equations become $2mi_{R_i}^t * d_{st_i}^t = B^{2}{}_i^t$ and $2mi_{L_i}^t * d_{st_i}^t = B_i^t - B^{2}{}_i^t$, these relations can be indicated in (Figure 5) in [23].

We get $d_{st_i}^t$ after a certain amount of computations.

$$d_{st_i}^t = \left(m^* - \vec{m}_i^t\right) \tag{13}$$

Where $\vec{m}_i^t$ is the i^{th} mantis's present position, and m^* is the exact position of the prey or the solution that was found to be the best by far.

ii). Determining the striking speed (v_{st}).

The mantis attacks its prey using its frontal legs. By stabilizing its rear legs and expanding its forelegs out as far as it can in the direction of the prey, it updates its location. The concept of the sigmoid function can be used to quantitatively approximate the speed at which a mantis strikes its prey with its front legs. The velocity of striking is determined using the following equation:

$$v_{st} = \frac{1}{1+e^{r\rho}} \tag{14}$$

Where ϱ, a constant value, indicates the gravity acceleration ratio of the mantis's strike, and V is the mantis's strike velocity. The number r is produced to regulate the acceleration caused by gravity rate, and it ranges from -1 to 1. To capture the prey, every mantis gets updated using the formula below:

$$m_i^{t+1} = \left(m_i^t + m^*\right)/2.0 + v_{st}(m^* - m_i^t) \tag{15}$$

The mantis shifts its location between where it is now, m_i^t , and the target's position to minimize the space between them to expedite its assault process. m_i^{t+1} indicates the new location of mantis i during the function assessment t, and v_{st} sets the mantis's striking velocity. A mantis's strike may occasionally miss, in which case it must alter its path before trying again. As a result, the mantis changes its course in response to the directions of two randomly chosen mantises from the entire population.

$$m_i^{t+1} = m_i^t + n_1.(m_a^t - m_b^t) \tag{16}$$

where two mantises, X and Y, were chosen at random from the existing population. The mantis fell into a trap of the local optimality as a result of the mantis strike failing. To keep the algorithm from slipping into the local optimum trap, the subsequent mathematical formula is suggested:

$$m_i^{t+1} = m_i^t + e^{2r} * \cos(2r\pi) * \left|m_i^t - \vec{m}'_{ar}\right| + (n * 2 - 1) * \left(m_j^u - m_j^r\right) \tag{17}$$

This formula is applied in MSA with a probability of failure; for two reasons, the first one avoids getting into local minima, and the second accelerates the convergence speed to the best solution. This probability of failure can be expressed as follows:

$$f_b = a * (1 - \frac{t}{MaxT}) \tag{18}$$

Where A is a predetermined, fixed value that governs both exploration and exploitation operators and ranges from 0 to 1.

3.1.4 | Sexual Cannibalism

Sexual cannibalism is the term for the act of the female praying mantises eating the male during or after copulation. This behavior is formulated as follows:

$$m_i^{t+1} = m_i^t + r_1.\left(m_i^t - m_a^t\right) \tag{19}$$

In the case of praying mantises, m_i^t stands for the female, and m_a^t, a randomly chosen answer, symbolizes the male who is drawn to the female and mates with it before being devoured. An imprisoned female initiates the process of attracting a partner by acting on a possibility $Prob_t$ whose worth progressively decreases with repetition.

$$Prob_t = r_2 * \mu \tag{20}$$

The mating process and the creation of new progeny are expressed by the uniform crossover operator, which is derived from the operators of genetics and is given by the formula that follows:

$$m_i^{t+1} = m_i^t * \vec{U} + \left(\vec{m}_{11}^t + n_3.\left(-\vec{m}_{11}^t + \vec{m}_i^t\right)\right).\left(1 - \vec{U}\right), \tag{21}$$

where $\vec{m}_{11}^t$ symbolizes the male that mates with the female. Following or throughout the mating process, the female will use the following equation to consume the male:

$$m_i^{t+1} = m_a^t * \cos(2\pi r) * \mu \tag{22}$$

In this case, m_a^t stands for the male, µ for the male's consumed portion, and $\cos(2\pi r)$ gives the female the freedom to spin the male around throughout the eating process.

To express the prior behavior in all of its phases, utilize the MSA framework (Algorithm 1).

Algorithm 1 The steps of MSA

```
Input: MaxT, Num, A, $a$, $P_c$, $\rho$, $p$, and $Prob$
Output : $\vec{m}^*$
1.  The initialization process of mantises, $\vec{m}_i^t (i = 1,2, \dots\dots, Num)$, using Eq.(1)
2.  Assess every $\vec{m}_i$ to find the best one ($\vec{m^*}$).
3.  $t = 1$; //the function assessment.
4.  while ($t < MaxT$)
5.  n: a random number in a range [0,1].
6.  if n<p %%% Exploration phase
7.  $n_4$: a random number in a range [0,1].
8.  Updating the recycling factor, $RCF$, using Eq. (10)
9.  for $i=1:Num$
10. if $r_4$<RCF %% Smashers' behavior
11. Adapting $\vec{m}_i^{t+1}$ by Eq. (3)
12. Else  %% Spearers' behavior
13. Adapting $\vec{m}_i^{t+1}$ by Eq. (9)
14. End if
15. $t = t + 1$
16. Assess the mantis, $\vec{m}_i^{t+1}$, and exchange $\vec{m}_i^t$ with, $\vec{m}_i^{t+1}$ if it is superior.
17. End for
18. Else %%% Exploitation stage
19. for $i=1:Num$
20. for $j=1:D$
21. $n_2$: a random number in a range [0,1].
22. if $n_2 < n_4$
23. Updating $m_{ij}^{t+1}$ using Eq. (16)
24. Else
25. Updating $m_{ij}^{t+1}$ using Eq. (15)
26. if $r_4 < P_f$
27. Updating $m_{ij}^{t+1}$ using Eq. (17)
28. End
29. End if
30. End for
31. $t = t + 1$
32. Assess the mantis, $\vec{m}_i^{t+1}$, and exchange $\vec{m}_i^t$ by, $\vec{m}_i^{t+1}$ if it is superior.
33. A probability $f_b$ of failure is updated using Eq. (18)
34. End for
35. End If
36. if n< $P_c$ %%% %%% Sexual cannibalism
37. for $i=1:N$
38. $n_3$: a random number in a range [0,1].
```

```
39. n4: a random number in a range [0,1].
40. if n3< n4 %% Mating phase
41. Updating $m_i^{t+1}$ using Eq. (21)
42. Else
43. if n4< $P_t$ %% Attraction of partners
44. Updating $m_i^{t+1}$ using Eq. (19)
45. Else %% Cannibalism process
46. Updating $m_{ij}^{t+1}$ using Eq. (22)
47. End
48. End if
49. $t = t + 1$
50. Assess the mantis, $\vec{m}_i^{t+1}$, and exchange $\vec{m}_i^{t}$ with $\vec{m}_i^{t+1}$ if it is superior.
51. Adapting $P_t$ with Eq. (20)
52. End for
53. End if
54. End while
```

4 | Proposed Algorithm

In this section, the suggested algorithm OBMSASA is going to be thoroughly examined and clarified. The feature in FS is binary; if it is picked, it is set to one; if not, it is set to zero. The goal of the Harris Hawks optimization technique is to resolve continuous issues that defy the binary character of the FS issue. Our suggested method consists of two primary steps: firstly, the integration of the MSA algorithm in conjunction with the Opposition Based Learning technique (OBMSA) for FS. Secondly, the integration of the SA and OBMSA will be covered in the second step. MSA becomes trapped in local optima, just like a lot of other metaheuristics do. Consequently, the SA seeks to keep the MSA algorithm out of local optima. The framework of OBMSASA is displayed in Figure 3, and Algorithm 2 expresses the framework of OBMSASA.

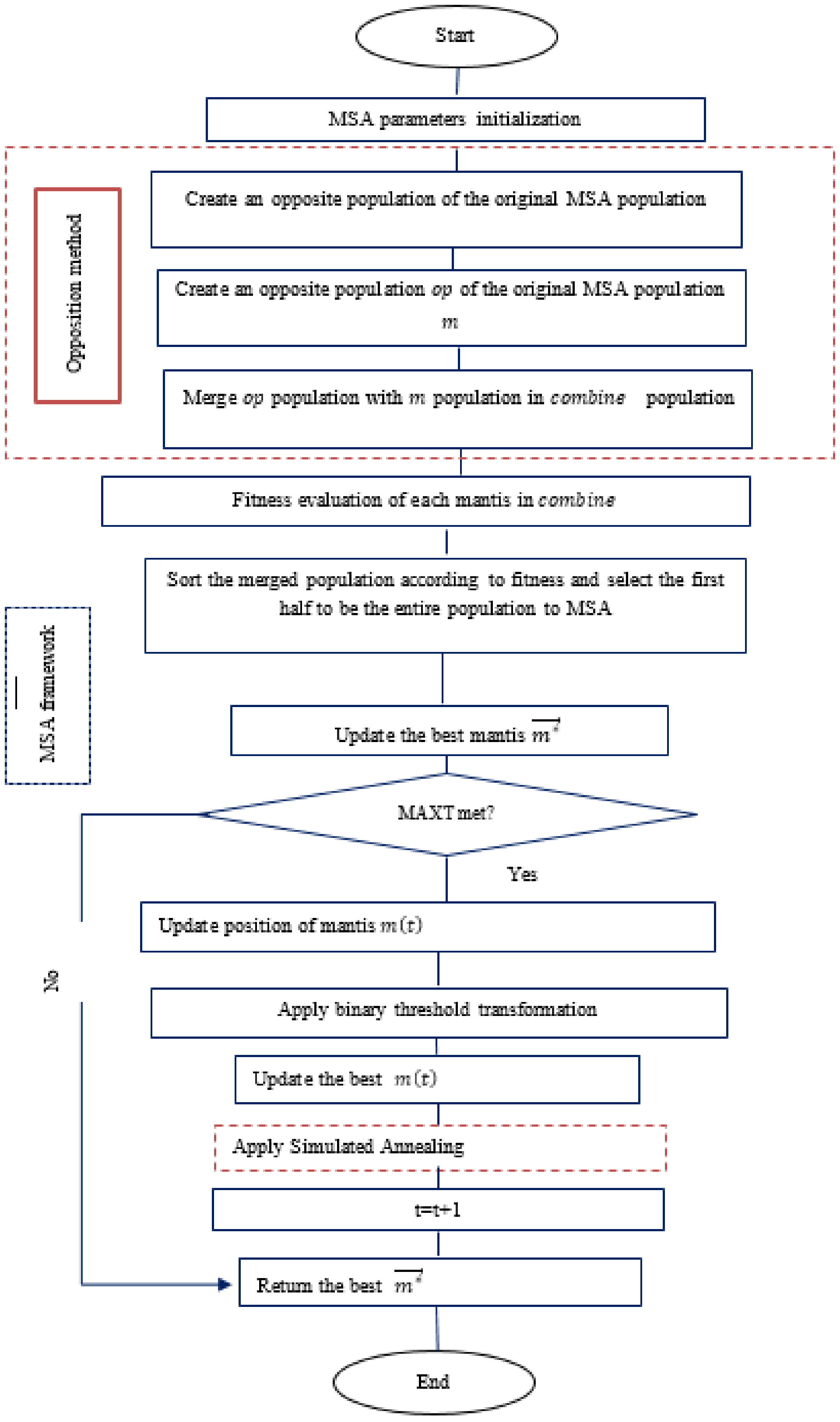

Start
MSA parameters initialization
Opposition method
Create an opposite population of the original MSA population
Create an opposite population op of the original MSA population m
Merge op population with m population in combine population
Fitness evaluation of each mantis in combine
Sort the merged population according to fitness and select the first half to be the entire population to MSA
MSA framework
Update the best mantis m
MAXT met?
Yes
No
Update position of mantis m(t)
Apply binary threshold transformation
Update the best m(t)
Apply Simulated Annealing
t=t+1
Return the best m
End

Figure 3. The OBMSASA framework.

4.1 | Mantis Search Algorithm for Solving FS Problem

There are two primary phases to the suggested OBMSASA for handling the FS problem. Initialization, transformation function, K- nearest neighbor (KNN) classifier, and assessment are the steps that make up the first stage. Additionally, the opposition-based learning method is used to raise the standard of the solution. On the other hand, the hybridization of the first stage with simulated annealing occurs in the second stage.

4.1.1 | Initialization

In this stage, a random population of agents for searching (m mantis) is formed. Every Mantis in the population stands for a potential solution. A potential solution is represented by a vector of dimensions D. The size of a dataset's features is represented by d. The vector's values can all be either 1 or 0, signifying whether or not the feature is chosen.

4.1.2 | KNN

A Metaheuristic algorithm is used to produce newly discovered samples for dimensionality reduction. This new sample is not labeled yet. The primary goal of classification is to assign a class to newly discovered samples that lack a label for a particular class. In this context, numerous classifiers have been employed. KNN classifier [41] is among the most popular. Because it is simple to use and only requires one parameter, K, to specify a number of neighbors, this can be shown in Figure 4.

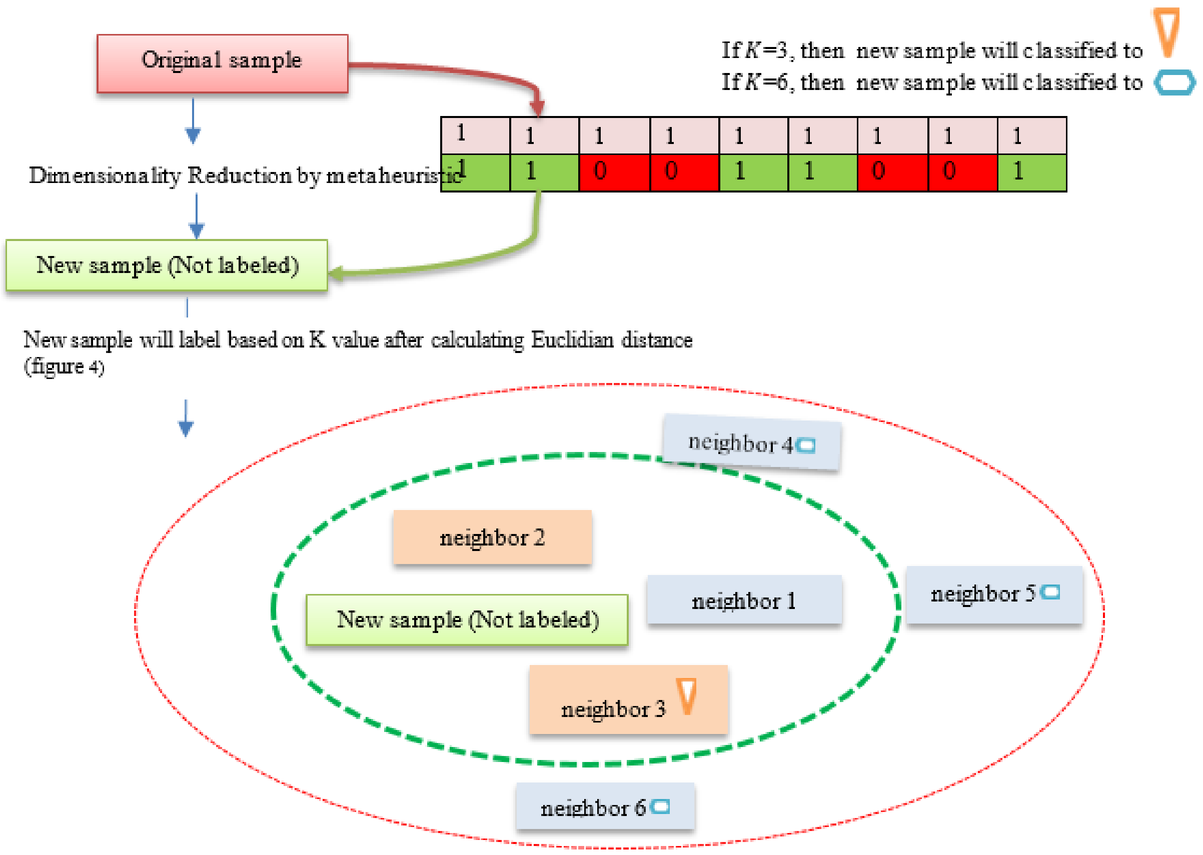

Original sample
If K=3, then new sample will classified to
If K=6, then new sample will classified to
1 1 1 1 1 1 1 1 1
1 1 0 0 1 1 0 0 1
Dimensionality Reduction by metaheuristic
New sample (Not labeled)
New sample will label based on K value after calculating Euclidian distance (figure 4)
neighbor 4
neighbor 2
neighbor 1
New sample (Not labeled)
neighbor 5
neighbor 3
neighbor 6

Figure 4. The representation of a new possible Mantis labeling process.

To allow the classifier to recognize the unique characteristics of the data, the relationship between the values of the attributes, and the label of the class, we must first train the classifier. We are unable to determine whether or not the classifier is successfully trained in real life. Thus, it is standard procedure to reserve some data that is labeled as a training dataset and some as a dataset for testing. The new database on which the classifier has not been trained still poses a dilemma. It is still necessary to ensure that the classifier performs well, and work has been done to use the training database to train the classifier to achieve better performance, the testing dataset is preserved at a distance. Each of the samples for the dataset being tested needs to use Euclidean distance to find its K nearest neighbors from the dataset used for training, as shown in Figure 5.

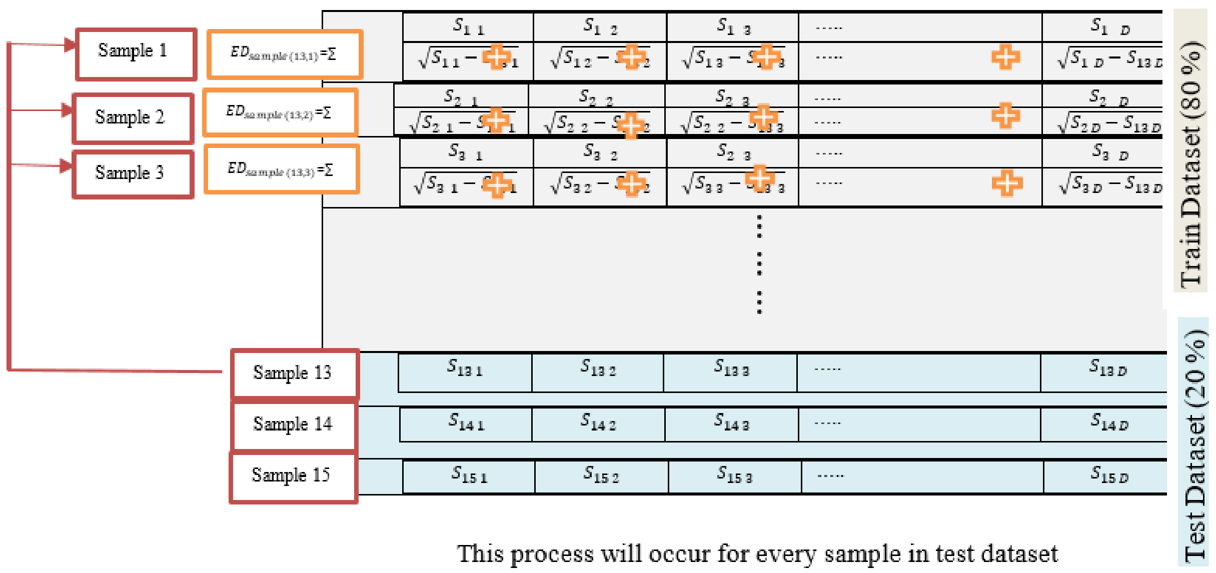

Sample 1
Sample 2
Sample 3
Sample 13
Sample 14
Sample 15
$ED_{sample(13,1)}=\Sigma$
$ED_{sample(13,2)}=\Sigma$
$ED_{sample(13,3)}=\Sigma$
Train Dataset (80 %)
Test Dataset (20 %)
This process will occur for every sample in test dataset

Figure 5. The representation Euclidian distance calculation.

$$Eclud\_Distance(ED) = \sqrt{\sum_{j=1}^{D}(Strain_j - Stest_j)} \tag{23}$$

here Euclidean distance is represented by $Eclud\_Distance$. The size of an attribute in a given dataset is denoted by D, and j =1,…. D. The $j^{th} the$ attribute in the sample from a training dataset is called $Strain_j$. The $j^{th} the$ attribute in the sample from the dataset being tested is called $Stest_j$,. One measure that shows how well the classifier predicts the class labels is the accuracy of classification. This can be calculated by dividing the proportion of accurate occurrences by the overall number of instances in the data set being tested. Conversely, the classification rate of error is calculated by dividing the overall number of cases detected in the testing dataset by the percentage of inaccurate instances.

4.1.3 | Assessment

The accuracy rate of classification derived from the KNN classifier is used to evaluate a solution's efficiency. A solution that optimizes the rate of classification accuracy is the optimal one. There will be two competing objectives in the fitness function used to evaluate the mantis population: maximizing one and minimizing the second. To reduce the two goals, the function of fitness will focus on decreasing the rate of error in classification instead of accuracy. The function of fitness is made with the following two goals in mind:

$$fitness = weight_1 \times Error\ rate + weight_2 \times \frac{|slected_f\#|}{|D|} \tag{24}$$

$$Error\ rate = (1 - accur) \tag{25}$$

$$weight_1 \in [0,1],\ weight_2 = 1 - weight_1 \tag{26}$$

here $(1 - accur)$represents the classification rate of error and $accur$ denotes the accuracy of the classification determined using KNN. The total number of features selected is shown by the variable $|\ slected_f\#|$. The size of a dataset's features is denoted by |D|. The two weight parameters for every aim are denoted by $weight_1$ and $weight_2$. Decreasing the error of classification (the maximization of classification accuracy) takes precedence over decreasing the quantity of the chosen attributes.

4.1.4 | Binary Representation (Transformation Functions)

The searching agents, or solutions, of metaheuristic approaches, are expressed by real values and are intended to solve continuous optimization issues. For the algorithm to adjust to the nature of FS, the search agent values need to be converted into binary values. For this, the transformation functions are accountable. The S and V shapes are two of the most significant and commonly used transformation functions [42].

Table 1. S-shape and V-shape transfer functions.

	TF Name	Mathematical Formula			TF Name	Mathematical Formula	
S-shape	S_1	$\frac{1}{1+e^{-2m}}$	(27)	V-shape	V_1	$\lvert \mathrm{erf}(\frac{\sqrt{\pi}}{2}m)\rvert = \left\lvert \frac{\sqrt{2}}{\pi}\int_0^{\frac{\sqrt{\pi}}{2}m} e^{-t^2}\,dt \right\rvert$	(31)
	S_2	$\frac{1}{1+e^{-m}}$	(28)		V_2	$\lvert \tanh m \rvert$	(32)
	S_3	$\frac{1}{1+e^{(-\frac{m}{2})}}$	(29)		V_3	$\lvert \frac{m}{\sqrt{1+\sqrt{m}^2}} \rvert$	(33)
	S_4	$\frac{1}{1+e^{(-\frac{m}{3})}}$	(30)		V_4	$\lvert \frac{2}{\pi}\mathrm{arc\ tan}\,(\frac{\sqrt{\pi}}{2}m)\rvert$	(34)

4.1.5 | Opposition Based Learning Method (OBL)

The initialization step for the majority of metaheuristic algorithms usually starts with a population that is produced randomly within the boundaries and has no prior knowledge of the search area. On the other hand, we can identify better solutions in the process of initialization potentially lessen the computing load, and improve global convergence if the starting population is produced by better methods. It is more beneficial to take into account both opposition and unpredictability, as opposed to pure randomness. An effective method for finding solutions in the opposite direction of the existing placements is opposition-based learning (OBL), which helps to improve an algorithm's search capabilities. The OBL strategy's primary concept is described as follows:

$$m'{}_i^{it} = (ub + lb) - m_i^{it} \tag{35}$$

Both ub and lb are expressing the lower and upper bounds, m_i^{it} is the original value. Actually, and $m'{}_i^{it}$ denotes the opposite value of m_i^{it} . The OBL technique can be applied at several updating stages to improve search ability as well as to improve the initialization population's quality.

4.1.6 | MSA and SA Hybridization

An algorithm called simulated annealing (SA) was introduced as a single solution to mimic the annealing process of metals. Metals are hardened by annealing, a physical process that involves heating the metal to an elevated temperature and allowing it to cool gradually. Initial temperature ($Temp_0$), final temperature ($Temp_{final}$), and cooling rate τ are the initial parameters of SA. The maximum temperature is called the beginning temperature, and it is progressively lowered to the end temperature by the rate of cooling. The algorithm starts with a randomly generated solution. It depends on the present solution being gradually improved. Iterations select a new adjacent solution to the present solution. If the new, nearby solution proves to be superior, the current one is changed accordingly. Additionally, if an adjacent solution proves to be superior, the best solution is upgraded. When the target temperature has been met, the algorithm terminates.

$$Temp = Temp * \tau \ , \tau \in [0,1] \tag{36}$$

To overcome the LO, SA is an algorithm based on probabilities that can accept the worst solution instead of the current surrounding solution. The likelihood that a worse alternative will be accepted depends on the degree to which it is worse and on how much the current temperature value is, which is as follows:

$$exp\left(\frac{-\Delta}{Temp}\right) \le rand \tag{37}$$

where Δ represents the fitness differences between the existing fitness and the new fitness that results from the surrounding solution. The temperature right now is T. The value of the exponent for raising e to is $\left(\frac{-\Delta}{Temp}\right)$, and exp is the function of the exponential form.

The SA is used to enhance the MSA algorithm's effectiveness and keep it from entering local optima to achieve even greater gains. Because the SA algorithm usually admits better solutions, it is capable of accepting worse ones depending on how likely they are to be worse and how high the temperature is at the moment. The SA algorithm can now start after the MSA iteration is complete. SA begins with a mantis location bunny created from the initial hybridization of OBL with MSA (OBMSA), as opposed to a solution generated at random.

The overall algorithm of the proposed can be expressed by the following Algorithm (2).

Algorithm 2 The steps of OBMSASA

```
Input: MaxT, Num, A, $a$, $P_c$, $\rho$, $p$, and $Prob$
Output : $\vec{m}^*$
1.  The initialization process of mantises, $\vec{m}_i^t (i = 1,2, \dots \dots, Num)$, using Eq.(1)
2.  Assess every $\vec{m}_i$ to find the best one ($\vec{m}^*$).
3.  $t = 1$; //the function assessment.
4.  while ($t < MaxT$)
5.  n: a random number in a range [0,1].
6.  if n<p %%% Exploration phase
7.  $n_4$: a random number in a range [0,1].
8.  Updating the recycling factor, RCF, using Eq. (10)
9.  for $i=1:Num$
10. if $r_4$<RCF %% Smashers' behavior
11. Adapting $\vec{m}_i^{t+1}$ by Eq. (3)
12. Else   %% Spearers' behavior
13. Adapting $\vec{m}_i^{t+1}$ by Eq. (9)
14. End if
15. $t = t + 1$
16. Assess the mantis, $\vec{m}_i^{t+1}$, and exchange $\vec{m}_i^t$ with, $\vec{m}_i^{t+1}$ if it is superior.
17. End for
18. Else %%% Exploitation stage
19. for $i=1:Num$
20. for $j=1:D$
21. $n_2$: a random number in a range [0,1].
22. if $n_2 < n_4$
23. Updating $m_{ij}^{t+1}$ using Eq. (16)
24. Else
25. Updating $m_{ij}^{t+1}$ using Eq. (15)
26. if $r_4 < P_f$
27. Updating $m_{ij}^{t+1}$ using Eq. (17)
28. End
29. End if
30. End for
31. $t = t + 1$
32. Assess the mantis, $\vec{m}_i^{t+1}$, and exchange $\vec{m}_i^t$ by, $\vec{m}_i^{t+1}$ if it is superior.
33. A probability $f_b$ of failure is updated using Eq. (18)
34. End for
35. End If
36. if n< $P_c$ %%%%%% Sexual cannibalism
37. for $i=1:N$
38. $n_3$: a random number in a range [0,1].
39. $n_4$: a random number in a range [0,1].
40. if $n_3 < n_4$  %% Mating phase
41. Updating $m_i^{t+1}$ using Eq. (21)
42. Else
43. if $n_4 < P_t$ %% Attraction of partners
44. Updating $m_i^{t+1}$ using Eq. (19)
45. Else %% Cannibalism process
46. Updating $m_{ij}^{t+1}$ using Eq. (22)
47. End
```

```
48. End if
49. $t = t + 1$
50. Initialize $Temp_0$, $Temp_{final}$, and $\tau$.
51. $solution_{best} = \vec{m}_i^{t+1}$
52. $best\_fit = fitness(solution_{best})$
53. Temp=$Temp_0$.
54. While (Temp<Temp $_{final}$ )
55. $num_{prob}$=random number between [1,3].
56. If $num_{prob} == 1$
57. If rand<rand
58. New solution=$solution_{best}$.
59. Else
60. New solution=$random\ solution\ from\ the\ population$ .
61.  $bits_0 = find(New\ solution == 0)$;
62. $bits_1 = find(New\ solution == 1)$;
63. $length_0 = length(\ bits_0)$.
64. $length_1 = length(\ bits_1)$.
65. If $length_0 \cong 0$ && $length_1 \cong 0$
66. $index_0 = random[1, length_0]$
67. $index_1 = random[1, length_1]$
68. %% swap between the two index
69. New solution($\mathrm{bits}_0(index_0) = 1$).
70. New solution($\mathrm{bits}_0(index_1) = 0$).
71. End
72. Else if  $num_{prob} == 2$
73. If rand < rand
74. New solution=$solution_{best}$.
75. Else
76. New solution=$random\ solution\ from\ the\ population$ .
77. $bits_0 = find(New\ solution == 0)$;
78. $length_0 = length(\ bits_0)$.
79. If $length_0 \sim = 0$
80. $index = random[1, length_0]$
81. New solution($\mathrm{bits}_0(index) = 1$).
82. End
83. Else if $num_{prob}$==3
84. If rand<5
85. New solution=$solution_{best}$.
86. Else
87. $New\ solution = random\ solution\ from\ the\ population$ .
88. end
89. $bits_1 = find(New\ solution == 1)$;
90. $length_1 = length(\ bits_1)$.
91.  $if\ length_1 \cong 0$
92. $index = random[1, length_1]$
93. New solution($\mathrm{bits}_1(index) = 0$).
94. end
95. if $num_{prob} == 4$
96. if rand <0
97. New solution=$solution_{best}$.
98. Else
99. $New\ solution = random\ solution\ from\ the\ population$ .
100. End
101. For d= 1 : Dim
102. $New\ solution(d) = 1 - New\ solution(d)$.
103. End
104. End
105. Selected indices=find (New solution $== 1$)
106. If length (selected indices==0)
107.  J=random (Dim)
108. $New\ solution(\ j) = 1$
109. $selected\ indices = find(New\ solution == 1)$;
110. end
111. New selected features =length (selected indices).
112. Evaluate the fitness  of the  $solution_{new\_selected\_features}$
113. If  $fitness(solution_{new\_selected\_features}) < best\_fit$
114. $best_{fit} = fitness(solution_{new\_selected\_features})$
```

```
115. $best\ solution = solution_{new\_selected\_features}$
116. $Else$
117. $\Delta = fitness(solution_{new\_selected\_features}) - best_{fit}$
118. if $(exp\left(\frac{-\Delta}{T}\right) \leq rand)$
119. $best\ solution = solution_{new\_selected\_features}$
120. $End$
121. $End$
122. $Update\ Temp_0 = colling\ rate * Temp_0$. $\%\%\%\ colling\ rate = 0.93$
123. End while SSA
124. End for %% for loop for all population
125. End while OBMSASA
```

5 | Results

This section examines the comparative outcomes of the proposed improved algorithm and most of the well-known meta-heuristic (MH) algorithms. Using MATLAB R2023 (b) on a personal computer running Windows 7/64-bit / Intel Core (TM) i7-3840QM, 2.80 GHz, and 16 GB RAM, all the algorithms used were coded and run in the same way. Each approach is implemented in the MATLAB 2023b environment.

5.1 | Dataset Description

The efficiency of the suggested OBMSASA algorithm was verified by experiments and assessments using twenty-one datasets as benchmarks. The UCI repository is the source of the datasets [43]. We concentrate on datasets that have a high dimension size (number of features), a high number of occurrences, or both. There are anywhere from 72 to 14980 instances. Additionally, there are 10 and 7129 features. Table 2 presents an explanation of the dataset.

Table 2. An explanation of the datasets.

Datasets				
ID	Name	Features #	Instances #	Classes #
1	Fri_c0_1000_10	10	1000	2
2	Page blocks	10	5473	2
3	Clean1	168	476	2
4	DNA	180	3186	3
5	Wisconsin	30	569	2
6	Segment	19	2310	7
7	Fri_c1_1000_10	10	1000	2
8	liver_numeric2	11	583	2
9	Ionosphere	35	351	2
10	SpectEW	44	267	2
11	WDBC	31	596	2
12	Glass	10	214	7
13	Australian	15	690	2
14	fri_c1_1000_25	26	1000	2
15	fri_c2_1000_25	26	1000	2
16	Spambase	57	4601	2
17	Eeg-eye-state	14	14980	2
18	Waveform	40	5000	3
19	Leukemia	7129	72	2
20	Pendigits	16	10992	2
21	Optdigits	64	5620	10

5.2 | Influence of V-shaped and S-shaped Transfer Functions on a Number of Recently Metaheuristic Algorithms

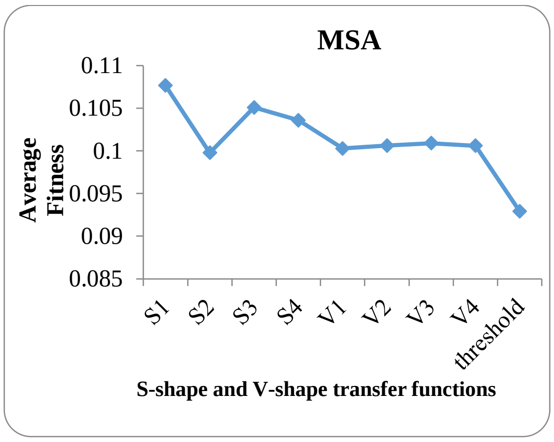
Recently, a lot of new metaheuristic algorithms have been introduced in the last few years, achieving great success in solving global optimization problems. Accordingly, many researchers have taken advantage of the power of these algorithms to solve FS problems, due to their power in obtaining many appropriate solutions to the problems in a reasonable time. Seven recently published algorithms that have not yet been used to solve feature selection problems were selected for navigation to select the fittest one to be the basic core of our model, such as (Mantis Search Algorithm (MSA) [23], Nutcracker Optimizer (NOA) [44], Young's Double Slit Experiment optimizer (YDSE) [45], Spider Wasp Optimizer (SWO) [46], Sinh-Cosh Optimizer (SCHO) [47], Exponential Distribution Optimization (EDO) algorithm [48], Zebra Optimization Algorithm (ZOA) [49]). These algorithms are converted to binary versions to meet the nature of the feature selection problem. The transformation from continuous to binary is done by using nine transformation methods: eight functions are introduced in Table 2, and the last method is a threshold method. After applying these transformation functions the results will be navigated to select the best transfer function, which achieves the best result. A sample of seven feature selection datasets is used to evaluate performance. Tables (3-9) mirror the performance of the aforementioned algorithm, in order. Then tracked the algorithms' performance on the FS problem in terms of fitness so that a winner could be chosen, to be the place of study.

Table 3 indicates the influence of different S-shape and V-shape functions, and also threshold method is used besides both S, and V shapes for transformation, so we operate the MSA algorithm on nine transformation methods. The first 4 columns represent the effect of four S-shape functions, while the following 4 columns represent the V-shape formulas, and the last column in the table represents the effect of the threshold method on MSA on seven selected datasets. The total average of fitness (Total AVG), and total standard deviation (Total STD) of each transformation method are calculated, by observing the result, we can say that the threshold method is the best transformation method for MSA.

Table 3. Influence of V-shaped and S-shaped transfer functions on MSA.

BMSA										
Dataset id	Fitness	S1	S2	S3	S4	V1	V2	V3	V4	Threshold
1	AVG	0.155338	0.126973	0.122273	0.118163	0.12757	0.124448	0.12019	0.126625	0.12504
	STD	0.021303	0.011566	0.014084	0.01766	0.014225	0.017697	0.015745	0.011832	0.018711
2	AVG	0.040317	0.041731	0.039933	0.037902	0.040047	0.038147	0.039581	0.039193	0.04015
	STD	0.004404	0.003342	0.003489	0.002809	0.00273	0.003753	0.003483	0.004947	0.00342
3	AVG	0.054868	0.047699	0.065478	0.063313	0.044136	0.034936	0.047857	0.034853	0.026131
	STD	0.018134	0.0207	0.021594	0.017193	0.024105	0.024607	0.025229	0.020843	0.02058
4	AVG	0.157857	0.139931	0.153794	0.157402	0.149881	0.146718	0.150533	0.149466	0.11364
	STD	0.018433	0.009374	0.011694	0.010328	0.010309	0.011508	0.012915	0.006803	0.008282
5	AVG	0.245957	0.253814	0.267309	0.254486	0.248463	0.270467	0.258306	0.258368	0.254039
	STD	0.027087	0.044611	0.042388	0.038765	0.038889	0.032512	0.041031	0.052176	0.038746
6	AVG	0.001957	0.002271	0.002479	0.002741	0.00219	0.002032	0.001953	0.002056	0.001897
	STD	0.001127	0.001139	0.000844	0.000818	0.000983	0.000836	0.000814	0.000667	0.000555
7	AVG	0.097303	0.08602	0.08429	0.091068	0.089588	0.087505	0.08785	0.093548	0.08939
	STD	0.012817	0.015744	0.017446	0.012383	0.01502	0.00861	0.012039	0.019039	0.01816
Total AVG		0.10765671	0.099777	0.105079	0.103582	0.100268	0.100608	0.100896	0.100587	**0.092898**
Total STD		0.01475786	0.015211	0.015934	0.014279	0.01518	0.014218	0.015894	0.016615	0.015493

Bold Values indicate the best average fitness values.



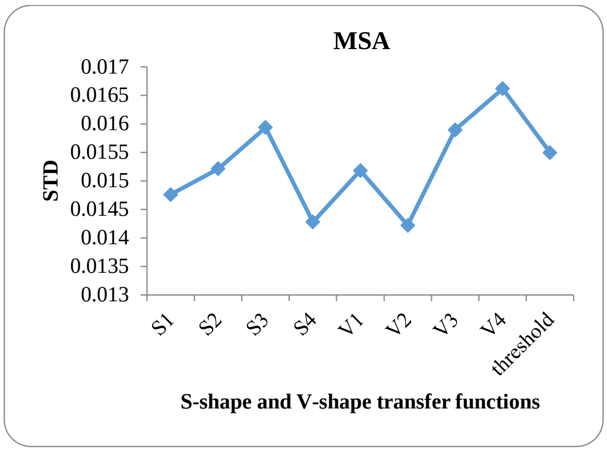


Figure 6. MSA total average for the average fitness values.

Figure 7. MSA total average for STD values.

Figure 6 indicates the influence of the nine transformation methods on the performance of MSA, in terms of total average. The threshold method outperforms all its peers. The total average STD is figured out in Figure 7.

Looking closely at the results obtained from Table 4, we can say that the threshold method is the best transformation method for NOA in terms of total average.

Table 4. Influence of V-shaped and S-shaped transfer functions on NOA.

BNOA										
Dataset id	Fitness	S1	S2	S3	S4	V1	V2	V3	V4	threshold
1	AVG	0.156665	0.119548	0.12514	0.12955	0.121035	0.124495	0.124943	0.12747	0.12049
	STD	0.01694	0.015619	0.018526	0.018344	0.02467	0.017824	0.016317	0.019179	0.015042
2	AVG	0.0409	0.040314	0.039635	0.040621	0.039419	0.040245	0.040652	0.04049	0.041441
	STD	0.002942	0.003698	0.00352	0.002976	0.003037	0.003035	0.002631	0.003217	0.003206
3	AVG	0.067001	0.065275	0.063745	0.067732	0.03064	0.032855	0.029279	0.038923	0.029877
	STD	0.018589	0.017236	0.022693	0.022668	0.027149	0.027946	0.027682	0.02917	0.013238
4	AVG	0.16978	0.161102	0.160039	0.160887	0.151118	0.151849	0.15096	0.149662	0.114245
	STD	0.016295	0.013836	0.011685	0.008763	0.008561	0.009996	0.009893	0.010963	0.010857
5	AVG	0.240684	0.292382	0.291835	0.283722	0.259937	0.254648	0.255513	0.256185	0.269701
	STD	0.028735	0.049	0.059397	0.041809	0.031369	0.036558	0.034524	0.031476	0.040554
6	AVG	0.002942	0.003722	0.003162	0.003145	0.002167	0.001842	0.002139	0.002056	0.002355
	STD	0.001327	0.001116	0.001062	0.000944	0.001124	0.00065	0.000908	0.000597	0.00121
7	AVG	0.100123	0.095775	0.09107	0.089833	0.090723	0.089433	0.08869	0.092505	0.0931
	STD	0.015067	0.018968	0.011271	0.015063	0.016878	0.01342	0.021593	0.014486	0.010018
Total AVG		0.11115643	0.11116	0.110661	0.110784	0.099291	0.099338	0.098882	0.101042	**0.095887**
Total STD		0.01427071	0.017068	0.018308	0.015795	0.016113	0.015633	0.016221	0.015584	0.013446

Bold Values indicate the best average fitness value.

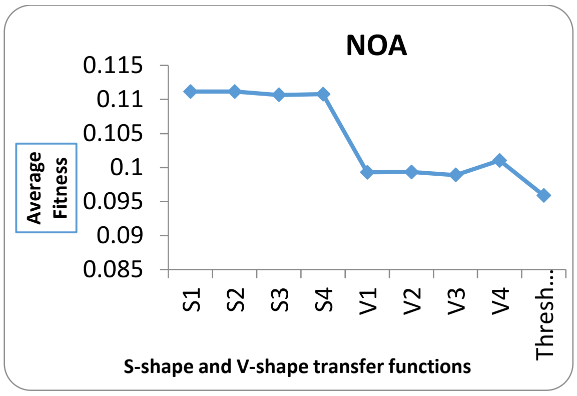


Figure 8. NOA total average for the average fitness values

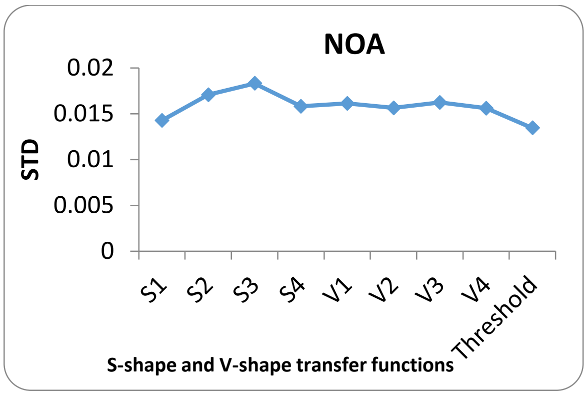


Figure 9. NOA total average for the standard deviation values

Figure 8 indicates the influence of the ninth transformation method on the performance of NOA, in terms of total average fitness. The threshold method outperforms all its peers. The total average STD is figured out in Figure 9.

Looking closely at the results obtained from Table 5, we can say that the threshold method is the best transformation method for YDSE in terms of total average.

Table 5. Influence of V-shaped and S-shaped transfer functions on YDSE.

BYDSE										
Dataset id	Fitness	S1	S2	S3	S4	V1	V2	V3	V4	Threshold
1	AVG	0.154038	0.12628	0.118755	0.131973	0.131925	0.12598	0.128455	0.12494	0.119005
	STD	0.025278	0.014553	0.01463	0.01672	0.016651	0.016258	0.014219	0.015649	0.017018
2	AVG	0.041648	0.039845	0.039838	0.039793	0.040748	0.040336	0.038676	0.038947	0.039924
	STD	0.004543	0.003912	0.003382	0.004052	0.003829	0.003315	0.002966	0.002688	0.003754
3	AVG	0.059703	0.067612	0.058103	0.066348	0.046128	0.052625	0.055903	0.05958	0.039896
	STD	0.021832	0.022637	0.016755	0.019983	0.019133	0.018174	0.017454	0.014874	0.018687
4	AVG	0.171248	0.1646	0.157991	0.156724	0.136577	0.135353	0.13482	0.134845	0.135627
	STD	0.01557	0.006154	0.007695	0.010698	0.017818	0.013868	0.014835	0.013018	0.00661
5	AVG	0.253632	0.309232	0.28554	0.265548	0.248844	0.260833	0.272166	0.260724	0.256482
	STD	0.031562	0.056712	0.04368	0.04788	0.044815	0.034317	0.019531	0.041698	0.046139
6	AVG	0.002141	0.004214	0.003752	0.003141	0.001951	0.002058	0.001953	0.00222	0.002402
	STD	0.001018	0.000844	0.001308	0.000908	0.000694	0.00088	0.000833	0.000771	0.000893
7	AVG	0.095173	0.091518	0.09003	0.093643	0.089585	0.0931	0.085378	0.094338	0.088695
	STD	0.015475	0.018122	0.014166	0.016266	0.014762	0.01115	0.014664	0.015296	0.018203
Total AVG		0.111083	0.114757	0.107716	0.108167	0.099394	0.101469	0.102479	0.102228	**0.097433**
Total STD		0.016468	0.017562	0.014517	0.016644	0.016815	0.013995	0.012072	0.014856	0.015901

Bold Values indicate the best average fitness values

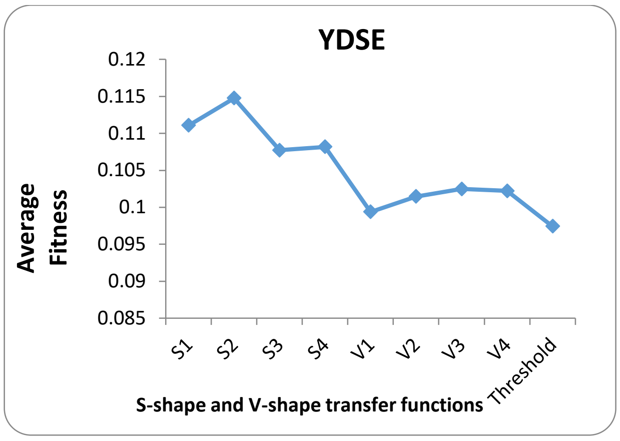


Figure 10. YDSE total average of fitness values.

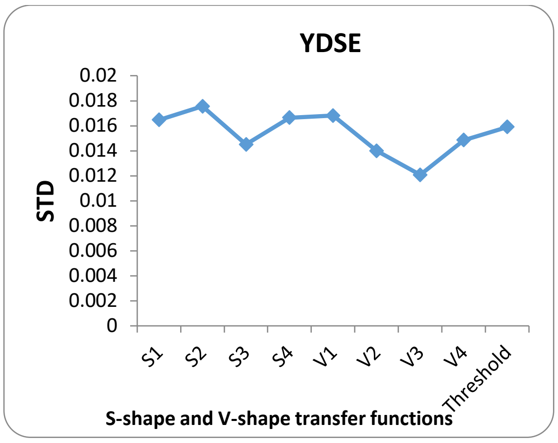


Figure 11. YDSE total average of STD values.

Figure 10 indicates the influence of the nine transformation methods on the performance of YDSE, in terms of total average fitness. The threshold method outperforms all its peers. The total average STD is figured out in Figure 11.

Results from Table 6 indicate that a threshold method performs better than other functions for algorithm SWO.

Table 6. Influence of V-shaped and S-shaped transfer functions on SWO.

BSWO										
Dataset id	Fitness	S1	S2	S3	S4	V1	V2	V3	V4	Threshold
1	AVG	0.170873	0.124153	0.128558	0.12534	0.125535	0.124003	0.12311	0.12717	0.12633
	STD	0.024308	0.018303	0.016008	0.016884	0.012473	0.020689	0.015941	0.016911	0.015379
2	AVG	0.040583	0.041367	0.039943	0.041131	0.040286	0.041074	0.040586	0.03935	0.040319
	STD	0.003022	0.003131	0.003195	0.003773	0.003448	0.003706	0.004582	0.003771	0.003659
3	AVG	0.066081	0.062694	0.062137	0.063727	0.065151	0.060175	0.049862	0.043406	0.04159
	STD	0.016402	0.02255	0.016795	0.020771	0.02374	0.02565	0.017374	0.018871	0.017595
4	AVG	0.180998	0.163187	0.164455	0.163405	0.155522	0.15325	0.152178	0.149823	0.132147
	STD	0.015495	0.009989	0.009366	0.008235	0.015046	0.008372	0.011878	0.013311	0.010715
5	AVG	0.253711	0.324885	0.288239	0.2682	0.274422	0.27276	0.272854	0.27698	0.270295
	STD	0.026821	0.035848	0.054628	0.040542	0.031813	0.035334	0.052076	0.045274	0.045571
6	AVG	0.002513	0.003528	0.003823	0.003487	0.002165	0.002511	0.002241	0.002432	0.001895
	STD	0.001007	0.00062	0.00083	0.001142	0.000694	0.001018	0.000807	0.001165	0.000465
7	AVG	0.109728	0.091868	0.092208	0.096025	0.09053	0.092505	0.09003	0.09162	0.09419
	STD	0.02024	0.019346	0.020814	0.016669	0.011683	0.01265	0.015048	0.01501	0.014222
Total AVG		0.117784	0.115954	0.111337	0.108759	0.107659	0.106611	0.104409	0.104397	**0.100967**
Total STD		0.015328	0.015684	0.017376	0.015431	0.014128	0.015346	0.016815	0.01633	0.015372

Bold Values indicate the best average fitness values.

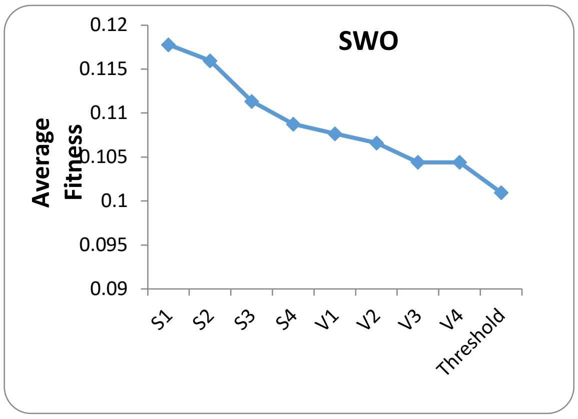


Figure 12. SWO total average of fitness values.

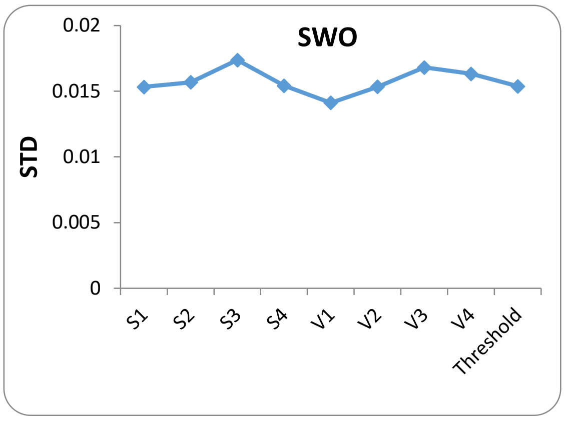


Figure 13. SWO total average of fitness values.

Figure 12 indicates the influence of the nine transformation methods on the performance of SWO, in terms of total average fitness. The threshold method outperforms all its peers. The total average STD is figured out in Figure 13.

Results from Table 7 indicate that the second V-Shape formula performs better than other functions for algorithm SCHO in terms of the average fitness.

Table 7. Influence of V-shaped and S-shaped transfer functions on SCHO.

BSCHO										
Dataset id	Fitness	S1	S2	S3	S4	V1	V2	V3	V4	Threshold
1	AVG	0.163258	0.132818	0.124148	0.125835	0.138003	0.133303	0.132263	0.142708	0.138403
	STD	0.030124	0.020254	0.015601	0.01661	0.025384	0.021024	0.014844	0.025394	0.0294
2	AVG	0.041738	0.039483	0.040333	0.039957	0.043191	0.042667	0.041784	0.042269	0.041467
	STD	0.003515	0.003014	0.002434	0.002116	0.004672	0.002594	0.00334	0.004194	0.002954
3	AVG	0.068543	0.067842	0.06705	0.074473	0.03664	0.024843	0.045953	0.028419	0.015333
	STD	0.013216	0.025273	0.015947	0.023741	0.038943	0.025796	0.035872	0.031402	0.023235
4	AVG	0.187118	0.164527	0.165861	0.164101	0.12299	0.123318	0.115444	0.120895	0.150341
	STD	0.014133	0.009224	0.010541	0.008009	0.020562	0.020253	0.017394	0.019962	0.015894
5	AVG	0.25878	0.296368	0.287124	0.290673	0.308218	0.292789	0.326595	0.299287	0.282384
	STD	0.03651	0.050003	0.047736	0.038721	0.053797	0.055777	0.045799	0.054946	0.056775
6	AVG	0.002752	0.003615	0.003803	0.003429	0.00294	0.002617	0.002618	0.002966	0.002889
	STD	0.001619	0.000834	0.001267	0.001002	0.001825	0.000974	0.001187	0.001281	0.001468
7	AVG	0.103043	0.090428	0.095428	0.088895	0.099035	0.098245	0.092303	0.091215	0.101708
	STD	0.016509	0.017835	0.01795	0.01403	0.018981	0.012992	0.014777	0.015589	0.018462
Total AVG		0.11789	0.113583	0.111964	0.11248	0.107288	**0.10254**	0.108137	0.103966	0.104646
Total STD		0.016518	0.018063	0.015925	0.01489	0.023452	0.019916	0.01903	0.021824	0.02117

Bold Values indicate the best average fitness values.

Figure 14 indicates the influence of the nine transformation methods on the performance of SCHO, in terms of total average fitness. The V-Shape second transformation method outperforms all its peers. The total average STD is figured out in Figure 15.

Results from Table 8 indicate that the first V-Shape formula performs better than other functions for algorithm EDO in terms of the average fitness.

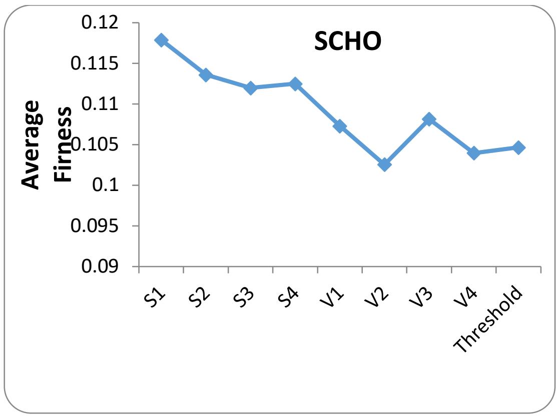


Figure 14. SCHO total average of fitness of values

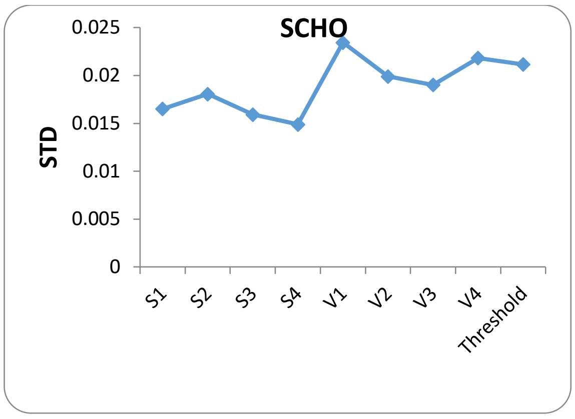


Figure 15. SCHO total average of fitness values.

Table 8. Influence of V-shaped and S-shaped transfer functions on EDO.

EDO										
Dataset id	Fitness	S1	S2	S3	S4	V1	V2	V3	V4	Threshold
1	AVG	0.178938	0.131033	0.128905	0.131038	0.123118	0.13356	0.133465	0.13213	0.127965
	STD	0.02664	0.018439	0.015774	0.015377	0.011161	0.018974	0.013548	0.016413	0.019211
2	AVG	0.042186	0.04176	0.042079	0.038916	0.039378	0.040736	0.040871	0.040843	0.0409
	STD	0.00336	0.004026	0.003737	0.00229	0.003586	0.003783	0.003874	0.003537	0.003563
3	AVG	0.051519	0.054754	0.064579	0.063668	0.057582	0.060482	0.07322	0.064177	0.069366
	STD	0.02566	0.024086	0.025056	0.022957	0.021617	0.025715	0.023862	0.02579	0.016385
4	AVG	0.180293	0.164649	0.165884	0.167557	0.172648	0.171365	0.172529	0.171317	0.154907
	STD	0.016389	0.007474	0.012893	0.008189	0.010434	0.014485	0.009554	0.007103	0.012797
5	AVG	0.256519	0.273182	0.273729	0.278861	0.260724	0.286474	0.272775	0.25656	0.295905
	STD	0.040662	0.050895	0.036708	0.049554	0.041728	0.029536	0.035641	0.040518	0.050873
6	AVG	0.003021	0.003626	0.00309	0.003464	0.002904	0.00293	0.002718	0.002746	0.002957
	STD	0.002065	0.00169	0.001029	0.00122	0.001406	0.001198	0.00138	0.001395	0.000783
7	AVG	0.109133	0.10117	0.08963	0.095028	0.096618	0.099638	0.093395	0.093993	0.10429
	STD	0.024661	0.01786	0.013998	0.018408	0.013007	0.017286	0.017457	0.014947	0.018823
Total AVG		0.117372	0.110025	0.109699	0.111219	**0.107567**	0.113598	0.11271	0.108824	0.113756
Total STD		0.019919	0.017782	0.015599	0.016856	0.014706	0.015854	0.015045	0.015672	0.017491

Bold Values indicate the best average fitness values.

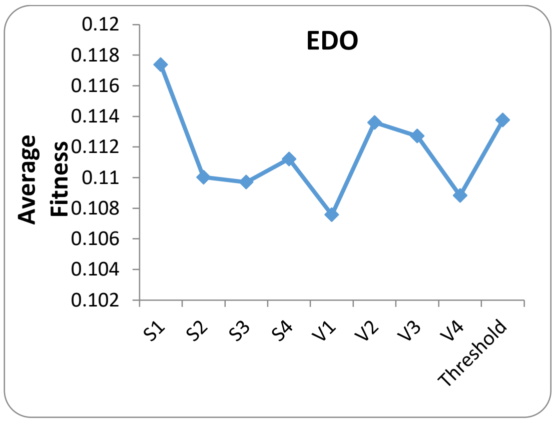


Figure 16. EDO total average of fitness values for 7 datasets.

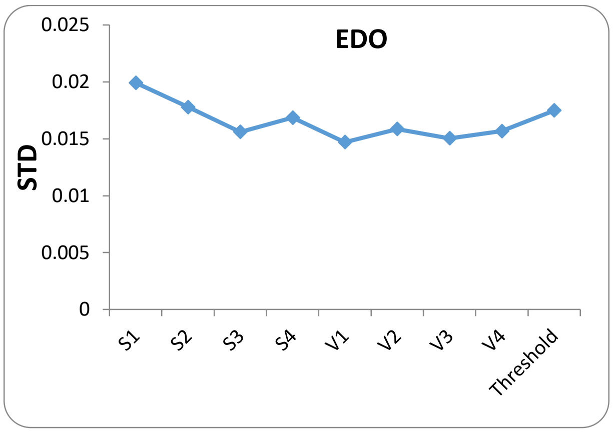


Figure 17. EDO total average of fitness values for 7 datasets.

Figure 16 indicates the influence of the nine transformation methods on the performance of EDO, in terms of total average fitness. The V-Shape first transformation method outperforms all its peers. The total average STD is figured out in Figure 17.

Results from Table 9 indicate that the threshold formula performs better than other functions for algorithm ZOA in terms of average fitness.

Table 9. Influence of V-shaped and S-shaped transfer functions on ZOA.

BZOA										
Dataset id	Fitness	S1	S2	S3	S4	V1	V2	V3	V4	Threshold
1	AVG	0.151968	0.132913	0.124445	0.130543	0.12806	0.124595	0.129545	0.131028	0.134293
	STD	0.0185	0.015798	0.018307	0.013607	0.013723	0.022187	0.015369	0.017029	0.024505
2	AVG	0.042015	0.040907	0.040305	0.040155	0.040259	0.0401	0.041264	0.04034	0.04206
	STD	0.003277	0.004699	0.002932	0.004241	0.002888	0.003845	0.003392	0.003665	0.004287
3	AVG	0.059161	0.05824	0.072377	0.074277	0.048436	0.053039	0.045411	0.05654	0.029601
	STD	0.018875	0.022413	0.02362	0.019558	0.020794	0.019786	0.021654	0.015327	0.029665
4	AVG	0.163392	0.16081	0.159092	0.157906	0.143909	0.137721	0.13916	0.145386	0.119009
	STD	0.017092	0.011558	0.011506	0.009539	0.015292	0.010427	0.017089	0.01372	0.01878
5	AVG	0.247322	0.282535	0.27972	0.287604	0.265465	0.269436	0.272182	0.264741	0.286183
	STD	0.027258	0.047421	0.041589	0.029877	0.036032	0.0396	0.03804	0.050452	0.039943
6	AVG	0.002406	0.003814	0.00291	0.003543	0.002782	0.002113	0.002273	0.002271	0.00262
	STD	0.001118	0.001183	0.001324	0.001644	0.001095	0.000797	0.001179	0.001202	0.001427
7	AVG	0.109478	0.10033	0.098545	0.093695	0.091515	0.096365	0.08904	0.09137	0.103985
	STD	0.01939	0.014643	0.013562	0.012361	0.015497	0.01622	0.014374	0.015646	0.02015
Total AVG		0.11082	0.111364	0.111056	0.112532	0.102918	0.103338	0.102696	0.104525	**0.102536**
Total STD		0.015073	0.016816	0.01612	0.012975	0.015046	0.016123	0.015871	0.01672	0.019822

Bold Values indicate the best average fitness values.

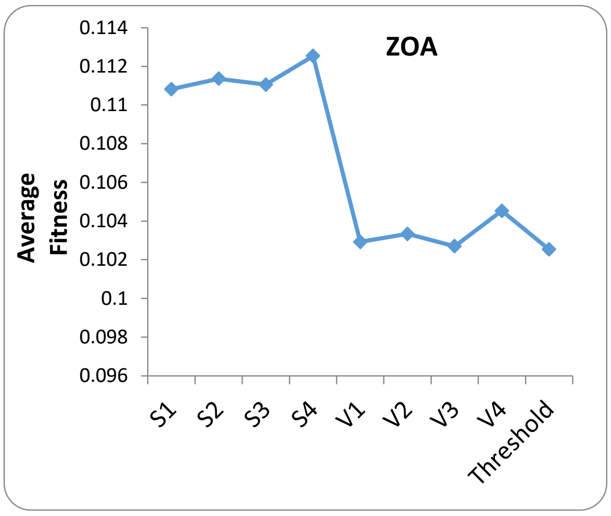


Figure 18. ZOA total average of fitness values for 7 datasets.

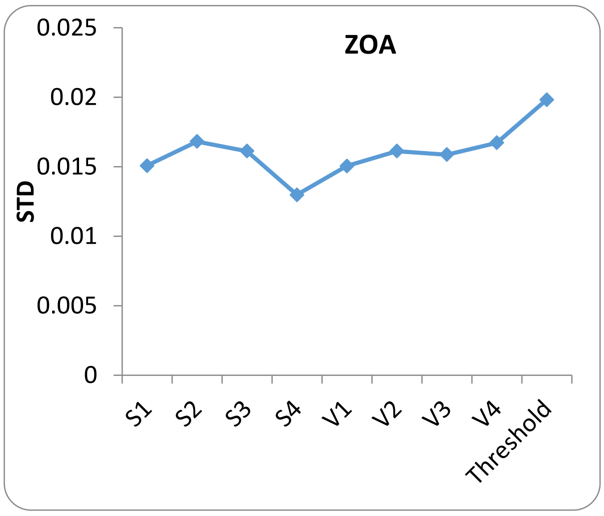


Figure 19. ZOA total average of fitness values for 7 datasets.

Figure 18 indicates the influence of the nine transformation methods on the performance of ZOA, in terms of total average fitness. It is clear that the S-shape fourth transformation method outperforms all its peers. The total average STD is figured out in Figure 19.

By tracking past performance, the Algorithm mantis search was chosen to be the subject of the study.

5.3 | Tuning of Parameters

Any algorithm's performance can be impacted by how its parameter values are configured. In practice, a lot of experiments are needed to investigate the impact of parameter adjustment on the suggested algorithm. As a result, the parameter values are determined by trial and error or by following the advice of earlier research. The suggested algorithm's efficacy is contrasted with that of other algorithms already in use. We evaluate each method over twenty separate runs. Furthermore, for every experiment, a maximum number of iterations is fixed at thirty. We observe that adding more search agents doesn't substantially change the findings, so we limit the number of mantis or search agents to 5. In this case, 80% of every dataset is used for training while the remaining 20% is used for testing, as indicated by [50-53]. To guarantee the same ranking of the instances number across all algorithms, the dataset's instances were randomly seeded before splitting. Compared to other classifiers, the KNN classifier using the Euclidean distance measurement has just one parameter (k) that needs to be tuned, making it a popular wrapping approach. The optimal outcomes are attained when k = 5, and other earlier research also supports this number [54-56]. In [57], $Weight_1$ and $Weight_2$have values that are 0.01 and 0.99, correspondingly.

A comparison is made between the suggested algorithm OBMSASA and some well-known methods, such as hybrid Harris Hawks with Simulated Annealing(HHSA)[18], hybrid Slime Mould algorithm with Marine Predators algorithm(SMAMPA)[19], Two-phase Mutation Gray wolf algorithm(TMGWO)[17]. hybrid Opposition Based with Salp Swarm algorithm (OBSSA)[21], hybrid Crossover and Cooperative with Whale Optimization algorithm (CCWOA)[22], Hybrid Equilibrium Optimizer with Simulated Annealing (EOSA)[16], Sine Cosine Algorithm(SCA), and the standard Mantis Search Algorithm(MSA)[23]. All previous algorithms used in the comparison are in the form of binary versions. The parameter configuration is introduced in Table 10.

Table 10. The parameter configuration.

Parameter	Value
Number of mantis $\boldsymbol{N}$	5
Number of function evaluations	30
Dimension number $\boldsymbol{D}$	Number of features
$\boldsymbol{weight_1}$	0.99
$\boldsymbol{weight_2}$	0.01
K	5
$\boldsymbol{\lambda}$	1.5
Starting temperature $\boldsymbol{Temp_0}$	30
End temperature $\boldsymbol{Temp_{final}}$	0.01
Cooling rate $\boldsymbol{\tau}$	0.5

5.4 | Metrics of Performance

5.4.1 | Accuracy

It is a measure of efficiency that assesses how well the classifier chooses the best subset of attributes after executing the algorithm N rounds. One can compute the optimal categorization accuracy as follows:

$$BestAccura = Maximum\ Accura_i^* \tag{38}$$

where $Accura_i^*$ represents the optimal rate of classification at run i after the algorithm has been run M rounds. i represents the method's i^{th}run, and i =1,...,N. One can calculate the average accuracy of classification as follows:

$$AvgAccura = \frac{1}{N}\sum_{i=1}^{N} Accura_i \tag{39}$$

The final accuracy can be expressed by $Accura$.

5.4.2 | The Selected Features Number

It is about how big the features that are chosen for a solution are. Here we consider two distinct measurements. The first metric, called Selected Features (SF), measures how big the chosen features are in a solution that has the highest fitness value. The Average Selected Features ($Avg\ slected_f$), the second metric, can be computed in the manner described below.

$$Avg\ slected_f = \frac{1}{N}\sum_{i=1}^{N}\frac{slected_{f_i}}{D}$$

The best subset of features obtained by the algorithm is expressed by $ected_f$.

5.4.3 | Fitness Function

There are three fitness metrics used: best value, average value, and worst value. The lowest fitness value reached after executing the algorithm N times is represented by the best value or best fitness.

$$BestFitness = Minimum\ Fitness_i^* \tag{40}$$

where $Minimum\ Fitness_i^*$ is the lowest fitness value reached when running the algorithm N times at run i. The total of all fitness values obtained by executing the algorithm N times is represented by the average fitness ($AvgFitness$), which is then divided by the total number of runs (N). The calculation for it is as follows:

$$AvgFitness = \frac{1}{N}\sum_{i=1}^{N} Fitness_i \tag{41}$$

The greatest fitness value attained after executing the algorithm M times is known as ($worstFitness$), and it can be calculated as follows:

$$WorstFitness = Maximum\ Fitness_i^* \tag{42}$$

where $Fitness_i$ represents the final value of fitness obtained.

5.5 | Evaluation of the Suggested Algorithm OBMSASA and other Algorithms

5.5.1 | Fitness Function

The current subsection examines how well the HHOBSA algorithm performs in comparison to a number of other metaheuristic algorithms, including CSA, CCWOA, OBSSA, TPGWO, EOSA, SAMPA, HHOSA, and MSA. The optimal, average, and worst fitness values as well as the standard deviation attained by each method are listed in Table 12. The average value of fitness (AVG) is what's gained when all of the dataset's features are chosen. It aids in quantifying the progress achieved by every method in the table. Based on the table's results, it is evident that in the majority of the datasets, OBMSASA outperforms all other methods. When compared to other algorithms, it can achieve more promising solutions. In 16 of the 21 datasets, we can see that OBMSASA can outperform its counterparts. Figures 20-40 express the convergence curves for all algorithms on all 21 datasets.

By observing Figures (20-40), we notice that the proposed OBMSASA outperforms its peers in terms of the convergence curve to the average fitness in Figs. (20-22,24-34,36,39), so, the proposed algorithm came in first place, outperforming 16 datasets out of 21 datasets, achieving the minimum value of the total average fitness, which is 0.092476. Furthermore, EOSA comes in second place with a value of 0.097641, concerning the total average fitness, but it achieves superiority on only one dataset (Spambase). Exploring the numerical results in Table 11, we can see that, although TPGWO reaches the best total average fitness on three datasets (DNA, Waveform, Optdigits), it comes in the third rank. CCOWA comes in the last rank with a value of 0.124347. Fig.24 shows that TPGWO Algorithm came in first place, followed by Algorithm OBSSA, followed by Algorithm HHOSA in third place, then Algorithm EOSA in fourth place, and the proposed algorithm OBMSASA came in fifth place. By analyzing Fig. 36 the superiority went to EOSA, followed by TPGWO, then HHOSA in third place, MSA in fourth place, followed by SMAMPA, and the proposed OBMSASA came in sixth place. Fig.38 shows that TPGWO converges to the best solution, followed by EOSA, then SMAMPA, and MSA came in fourth place, while OBMSASA came in fifth place. Moving to Fig. 39, HHOSA is the best one and converges to the best solution, Algorithm OBMSASA is exhausted in reaching the best solution, occupying second place in achieving it, while CCWOA is followed to arrive at the best solution. Lastly, Fig.41 shows that TPGWO outperforms all algorithms, then EOSA follows it, followed by OBMSASA in the third place. The total average of best values of all algorithms for all datasets, which is shown in Fig. 42(a, b), goes to the proposed OBMSASA with a value of 0.063793, while EOSA came in second place, and CCWOA comes in the last rank with a value of 0.081452. Moving to fig.43(a, b), we can say that the proposed OBMSASA outperforms all other algorithms, achieves the total average fitness on all datasets with a value of 0.092476, while EOSA came in second place, and CCWOA comes in last rank with a value of 0.124347. The superior results obtained by OBMSASA result from the advantages of the integration of the SA, which boosts the convergence of OBMSASA, and the opposition-based learning method helps the algorithm to explore the search space more effectively, reaching to best solution areas.

Table 11. The results of best fitness, average fitness, worst fitness, and STD for all algorithms.

Dataset ID	Criteria (Fitness)	CSA	CCWOA	OBSSA	TPGWO	EOSA	SMAMPA	HHOSA	MSA	OBMSASA
1	Best	0.09905	0.0941	0.0941	0.0842	0.10795	0.09905	0.0842	0.10995	0.09015
	AVG	0.131062	0.158397	0.135885	0.127137	0.131822	0.135358	0.12324	0.132817	**0.121478**
	Worst	0.16735	0.20105	0.1654	0.1634	0.16045	0.1743	0.15845	0.1624	0.14755
	STD	0.02126	0.023281	0.01792	0.01683	0.014226	0.019439	0.016281	0.014726	0.017411
2	Best	0.033863	0.034768	0.034673	0.033863	0.032053	0.033863	0.034768	0.034768	0.033673
	AVG	0.043623	0.040426	0.041007	0.041379	0.039961	0.040534	0.040205	0.041301	**0.039335**
	Worst	0.050437	0.046342	0.046532	0.048342	0.046342	0.047532	0.045627	0.047437	0.048247
	STD	0.003693	0.003011	0.002849	0.003485	0.003567	0.003072	0.003325	0.003983	0.003044
3	Best	5.95E-05	5.95E-05	0.002738	0.003274	0.012028	0.016254	5.95E-05	5.95E-05	5.95E-05
	AVG	0.081747	0.061805	0.031933	0.03077	0.058143	0.058711	0.031405	0.055968	**0.01599**
	Worst	0.150299	0.119274	0.065919	0.055855	0.116893	0.11058	0.084083	0.098432	0.095932
	STD	0.025946	0.028202	0.017752	0.014551	0.023617	0.025711	0.026943	0.023823	0.026915
4	Best	0.147816	0.112847	0.086591	0.082872	0.091308	0.120784	0.085923	0.124501	0.098083
	AVG	0.179087	0.163374	0.111434	**0.10822**	0.120149	0.152335	0.118027	0.158444	0.130176
	Worst	0.21276	0.203599	0.142655	0.130997	0.155141	0.175735	0.163251	0.180845	0.162192
	STD	0.013642	0.020229	0.014693	0.010975	0.012493	0.014596	0.019919	0.011058	0.01588
5	Best	0.235411	0.209984	0.184243	0.183618	0.183618	0.212484	0.183306	0.185181	0.132451
	AVG	0.34875	0.277213	0.288947	0.285276	0.277241	0.294217	0.291963	0.276073	**0.227845**
	Worst	0.443832	0.316069	0.417467	0.417467	0.341184	0.367862	0.365362	0.342747	0.286891
	STD	0.04971	0.029584	0.048249	0.055684	0.043021	0.040971	0.044875	0.03779	0.036836
6	Best	0.001053	0.002105	0.001053	0.001053	0.001053	0.001053	0.001053	0.001053	0.001053
	AVG	0.002505	0.004929	0.002939	0.00245	0.002343	0.002927	0.002648	0.002358	**0.002289**
	Worst	0.005338	0.009586	0.005301	0.004248	0.005338	0.006917	0.004774	0.004774	0.004774
	STD	0.001053	0.002148	0.000945	0.000948	0.000993	0.001468	0.001149	0.000977	0.001132
7	Best	0.05745	0.06835	0.06835	0.0733	0.0723	0.06835	0.0624	0.0535	0.0515
	AVG	0.095175	0.121352	0.096962	0.094352	0.093497	0.096368	0.093432	0.094585	**0.085798**
	Worst	0.1505	0.18915	0.1228	0.1218	0.12675	0.12875	0.13665	0.1228	0.11685
	STD	0.01819	0.027336	0.01406	0.013794	0.013364	0.016013	0.016678	0.016281	0.014357
8	Best	0.199293	0.224897	0.200293	0.182224	0.182224	0.216362	0.183224	0.208828	0.191759
	AVG	0.237029	0.261961	0.235438	0.238985	0.238936	0.242114	0.236443	0.234149	**0.232982**
	Worst	0.282638	0.283638	0.275103	0.276103	0.267569	0.293172	0.275103	0.265569	0.266569
	STD	0.021524	0.016581	0.018748	0.017326	0.021303	0.015464	0.022609	0.015917	0.021485
9	Best	0.015025	0.043605	0.015613	0.015319	0.015025	0.030345	0.015025	0.015319	0.000588
	AVG	0.051177	0.088772	0.07236	0.044557	0.040069	0.066065	0.055676	0.0576	**0.036917**
	Worst	0.100765	0.142605	0.116672	0.07395	0.101353	0.128168	0.099588	0.100765	0.071008
	STD	0.022721	0.022283	0.024862	0.016027	0.020796	0.020957	0.02181	0.019489	0.015391
10	Best	0.095669	0.137573	0.118894	0.113894	0.05922	0.11753	0.061038	0.097487	0.113439
	AVG	0.194304	0.233315	0.197095	0.173572	0.174439	0.179873	0.170807	0.187676	**0.168912**
	Worst	0.264237	0.338499	0.246921	0.264691	0.263782	0.230515	0.246921	0.266964	0.226878

	STD	0.038108	0.042874	0.028468	0.045927	0.042837	0.034644	0.043041	0.033947	0.030452
11	Best	0.009428	0.030283	0.010428	0.009428	0.009761	0.018855	0.018189	0.011428	0.001
	AVG	0.038166	0.049364	0.038047	0.043737	0.040177	0.042036	0.036931	0.038468	**0.031827**
	Worst	0.062661	0.072422	0.072755	0.097372	0.070755	0.071755	0.0539	0.064661	0.063327
	STD	0.014189	0.011078	0.016716	0.019342	0.017499	0.015381	0.010748	0.013698	0.013903

Bold Values indicate the best average fitness values.

Dataset ID	Criteria (Fitness)	CSA	CCWOA	OBSSA	TPGWO	EOSA	SMAMPA	HHOSA	MSA	OMSASA
12	Best	0.144762	0.193016	0.191905	0.143651	0.148095	0.148095	0.170556	0.148095	0.167222
	AVG	0.24918	0.269489	0.245881	0.255688	0.240907	0.247008	0.258008	0.240981	**0.235156**
	Worst	0.335556	0.360238	0.358016	0.380476	0.355794	0.332222	0.336667	0.309762	0.309762
	STD	0.048929	0.040646	0.036368	0.052231	0.047189	0.041639	0.039354	0.039124	0.041887
13	Best	0.079627	0.101149	0.081056	0.093975	0.073882	0.081056	0.081056	0.060963	0.072453
	AVG	0.126778	0.18345	0.127803	0.130747	0.121681	0.131702	0.125629	0.119482	**0.111953**
	Worst	0.181491	0.263261	0.159969	0.24677	0.218075	0.175031	0.175031	0.159969	0.152081
	STD	0.025535	0.051497	0.019265	0.03127	0.024178	0.022594	0.025931	0.022084	0.017834
14	Best	0.0511	0.0812	0.0911	0.0709	0.08535	0.08575	0.061	0.06635	0.0705
	AVG	0.110763	0.177725	0.159778	0.103775	0.100885	0.145058	0.110835	0.124523	**0.099355**
	Worst	0.1798	0.28105	0.22555	0.1691	0.1212	0.2202	0.16455	0.19625	0.13485
	STD	0.025861	0.045708	0.028285	0.018945	0.011834	0.030254	0.026336	0.02812	0.015517
15	Best	0.061	0.08615	0.06555	0.0511	0.04575	0.0808	0.07585	0.05605	0.05525
	AVG	0.093728	0.160252	0.135332	0.085848	0.085255	0.11556	0.107013	0.104662	**0.08208**
	Worst	0.1394	0.2428	0.18635	0.11585	0.1303	0.1818	0.15465	0.1604	0.1204
	STD	0.017174	0.042558	0.028256	0.018801	0.019146	0.022985	0.018132	0.02459	0.01645
16	Best	0.070051	0.079291	0.076261	0.053184	0.059418	0.065278	0.059921	0.06708	0.076507
	AVG	0.09025	0.099136	0.092898	0.072219	**0.069532**	0.082033	0.07579	0.076617	0.087375
	Worst	0.113597	0.12058	0.117726	0.093175	0.083443	0.095233	0.103514	0.089502	0.10121
	STD	0.009915	0.010745	0.010215	0.008655	0.006142	0.008176	0.007777	0.006114	0.006877
17	Best	0.041615	0.037427	0.036051	0.036382	0.03506	0.035444	0.034399	0.035391	0.029122
	AVG	0.052195	0.043626	0.04621	0.042049	0.040261	0.042379	0.041623	0.041442	**0.03743**
	Worst	0.080778	0.049269	0.056379	0.051252	0.044643	0.049322	0.047286	0.047009	0.042393
	STD	0.008827	0.002732	0.004973	0.002974	0.002893	0.003159	0.003059	0.003054	0.003353
18	Best	0.15472	0.1555	0.15078	0.13048	0.13221	0.15278	0.14141	0.14363	0.14137
	AVG	0.17516	0.179651	0.178562	**0.150158**	0.151876	0.16502	0.168567	0.16526	0.174106
	Worst	0.19656	0.19904	0.20078	0.1676	0.16539	0.1847	0.18172	0.18051	0.19284
	STD	0.010732	0.011087	0.010735	0.008164	0.008556	0.008483	0.008956	0.008258	0.011001
19	Best	0.000203	1.40E-06	4.21E-06	0.001659	0.000227	3.51E-05	1.40E-06	3.09E-05	1.40E-06
	AVG	0.000492	0.007825	0.003614	0.004152	0.002811	0.000661	**4.16E-06**	0.003245	0.000195
	Worst	0.001124	0.071124	0.074318	0.072478	0.070971	0.002459	9.82E-06	0.071181	0.000783
	STD	0.000192	0.021418	0.013499	0.012905	0.012876	0.000578	2.07E-06	0.012863	0.000206
20	Best	0.004825	0.00545	0.005901	0.005276	0.005101	0.005901	0.005276	0.005276	0.002151
	AVG	0.006562	0.008182	0.006834	0.006225	0.0061	0.007015	0.006555	0.006428	**0.004539**
	Worst	0.008879	0.011306	0.008429	0.007427	0.007427	0.008226	0.009228	0.007252	0.006728
	STD	0.000916	0.001222	0.000657	0.000554	0.000642	0.000654	0.000892	0.000561	0.001225
21	Best	0.012245	0.012728	0.012984	0.009816	0.009716	0.012103	0.014021	0.012216	0.011322
	AVG	0.023453	0.021048	0.022194	**0.014046**	0.014377	0.017883	0.018028	0.016962	0.01626
	Worst	0.032873	0.029576	0.029719	0.018368	0.020598	0.021848	0.02476	0.022686	0.02469
	STD	0.005117	0.003789	0.003565	0.002226	0.002882	0.00257	0.0026	0.002391	0.003007
Rank		8	9	7	3	2	6	4	5	1
Average Fitness		0.111009	0.124347	0.10815	0.097873	0.097641	0.10785	0.100611	0.103764	**0.092476**
Average best Fitness		0.072108	0.081452	0.072789	0.065689	0.064826	0.076294	0.065366	0.068436	**0.063793**

Bold Values indicate the best average fitness values

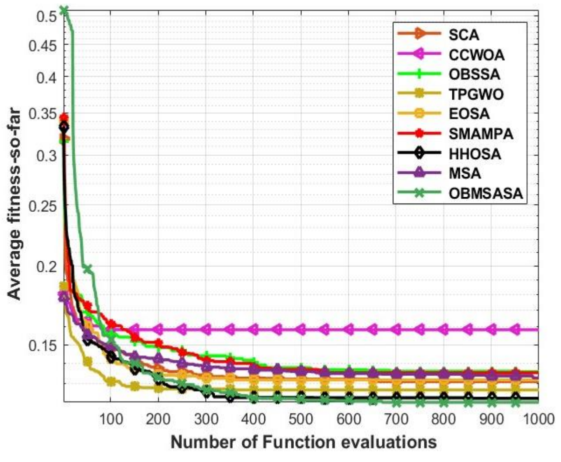


Figure 20. The average of fitness values for Fri_c0_1000_10 dataset.

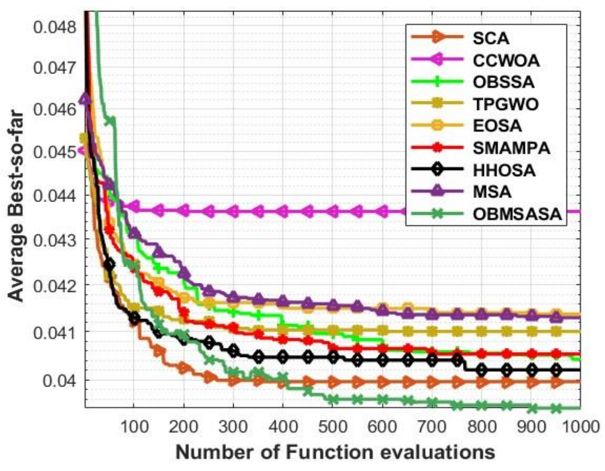


Figure 21. The average of fitness values for Page blocks dataset.

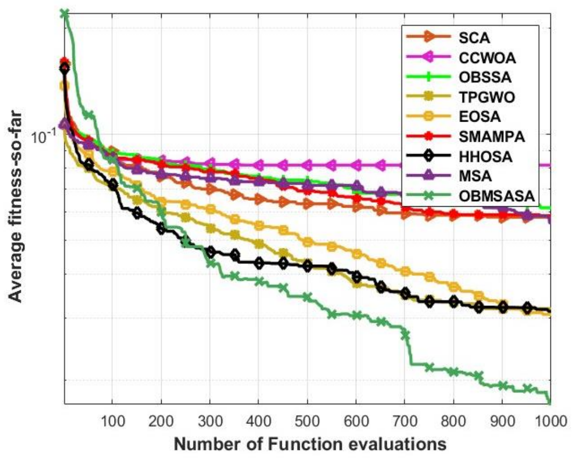


Figure 22. The average of fitness values for Clean1 dataset.

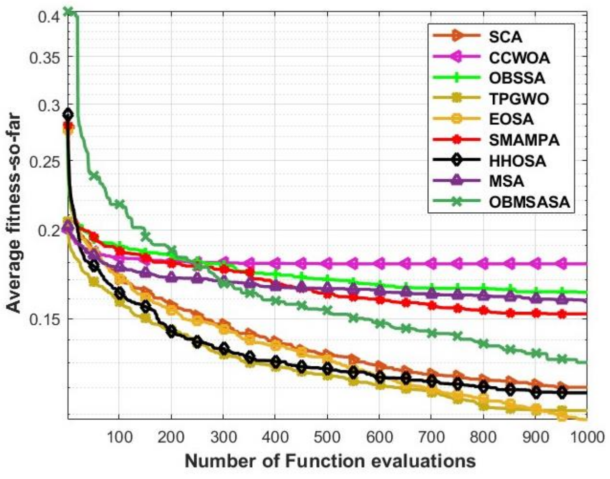


Figure 23. The average of fitness values for DNA dataset.

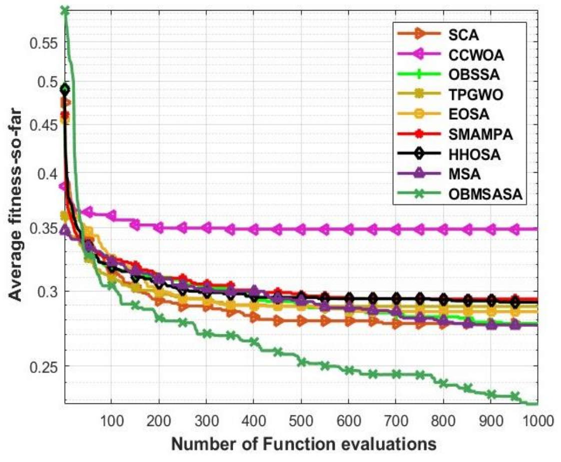


Figure 24. The average of fitness values for Wisconsin dataset.

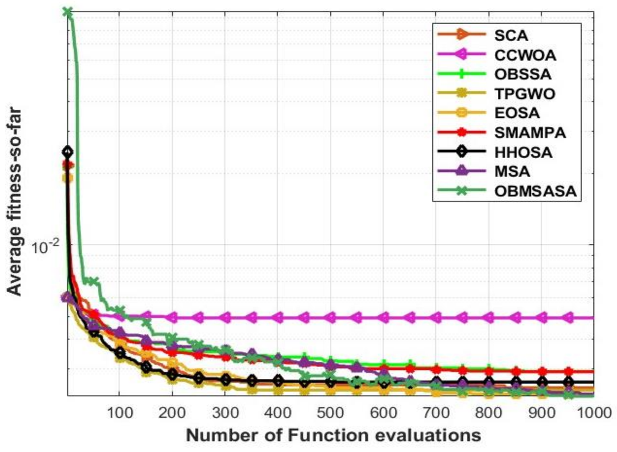


Figure 25. The average of fitness values for Segment dataset.

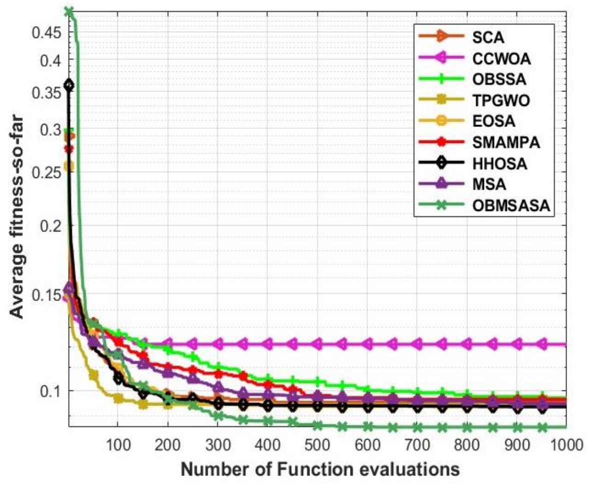


Figure 26. The average of fitness values for Fri_c1_1000_10 dataset

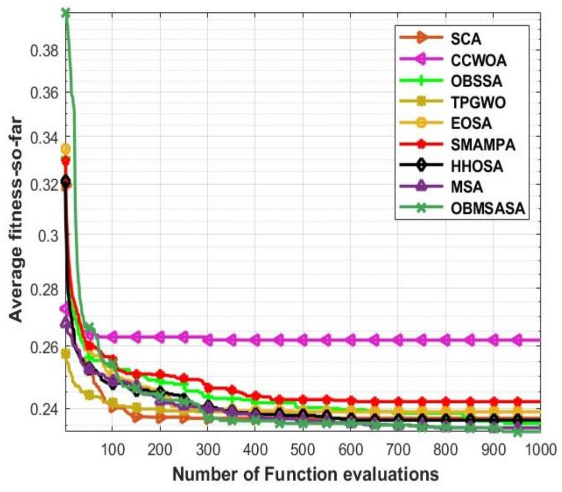


Figure 27. The average of fitness values for liver_numeric2 dataset.

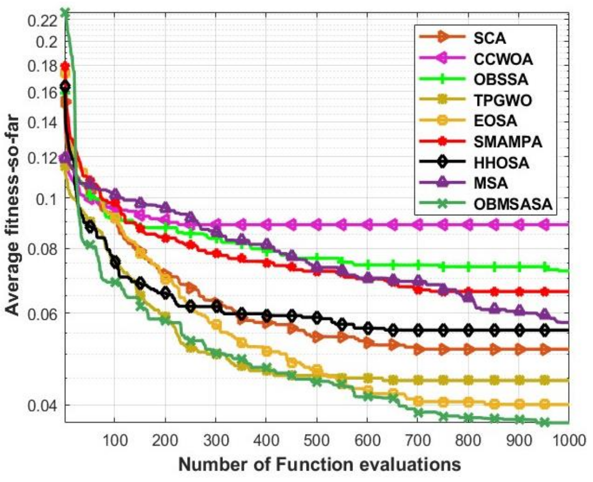


Figure 28. The average of fitness values for Ionosphere dataset.

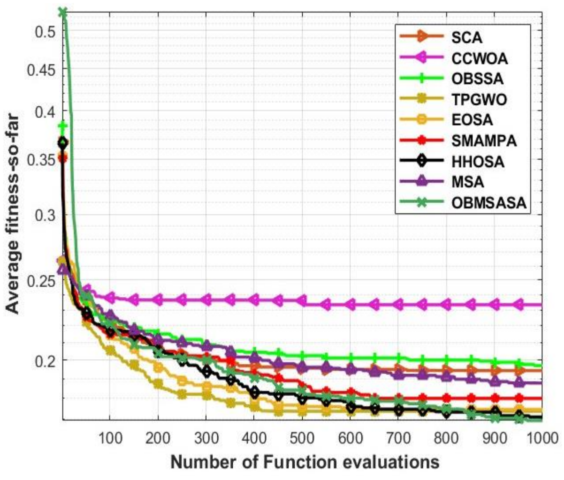


Figure.29. The average of fitness values for SpectEW dataset.

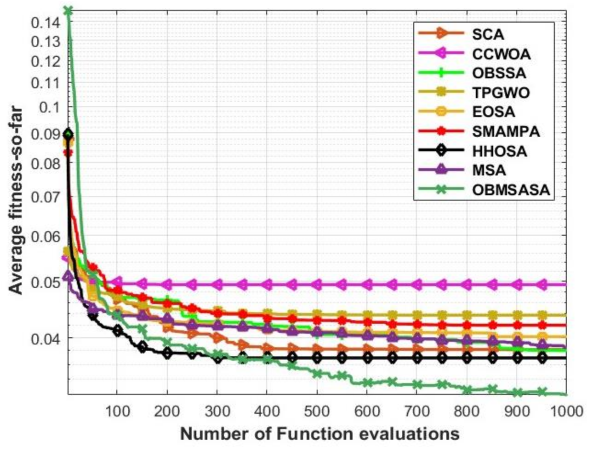


Figure 30. The average of fitness values for the WDBC dataset.

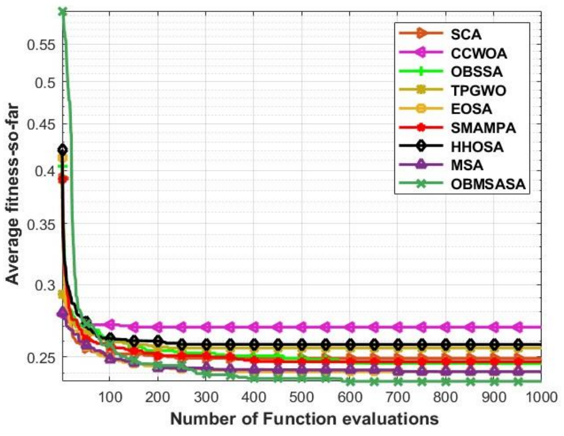


Figure 31. The average of fitness values for the Glass dataset.

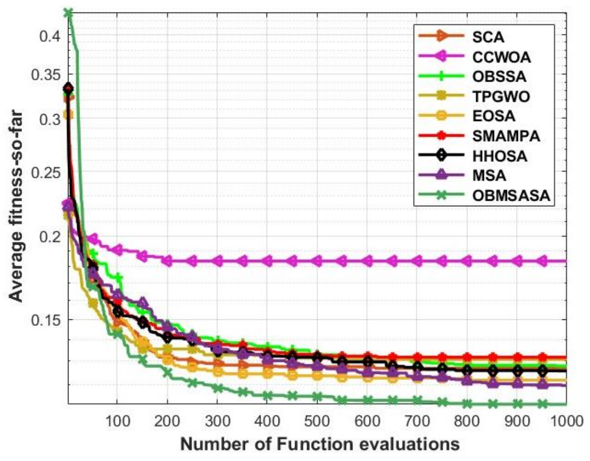


Figure 32. The average of fitness values for the Australian dataset.

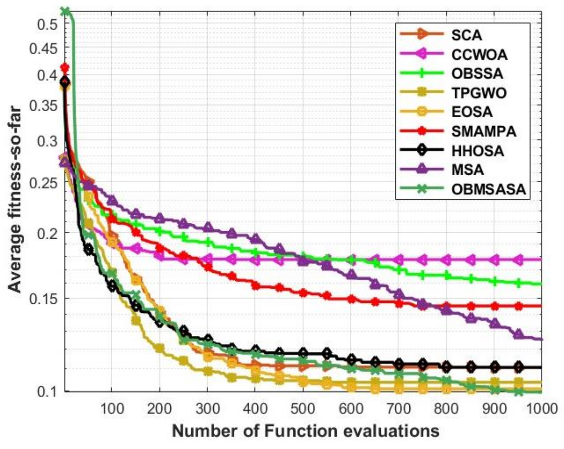


Figure 33. The average of fitness values for the Fri_c1_1000_25 dataset.

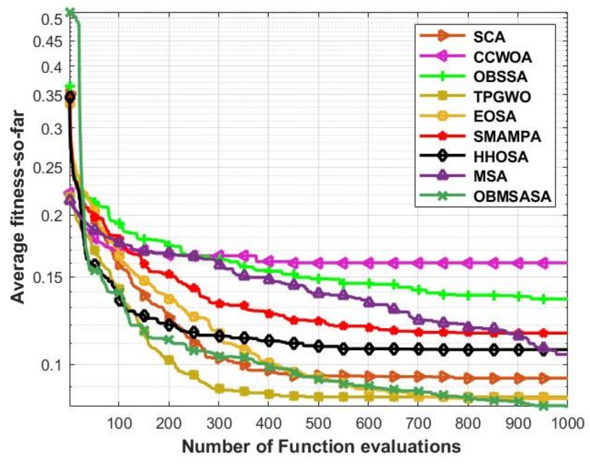


Figure 34. The average of fitness values for the Fri_c2_1000_25 dataset.

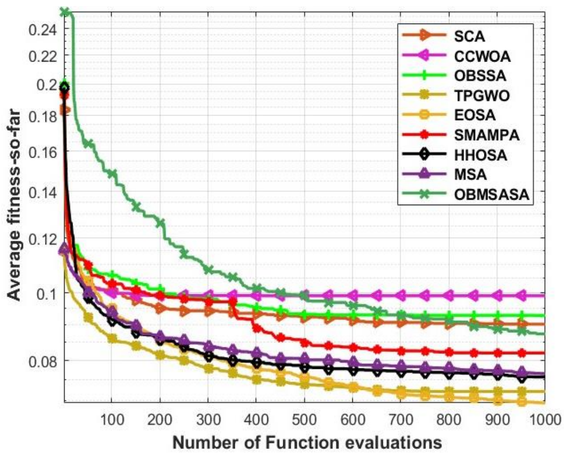


Figure 35. The average of fitness values for the Spambase dataset.

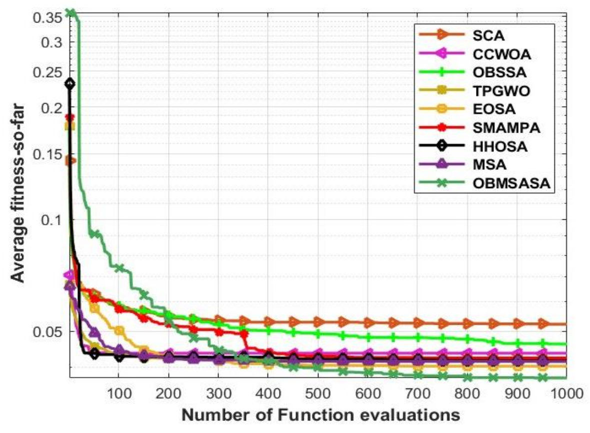


Figure 36. The average of fitness values for Eeg-eye –state dataset

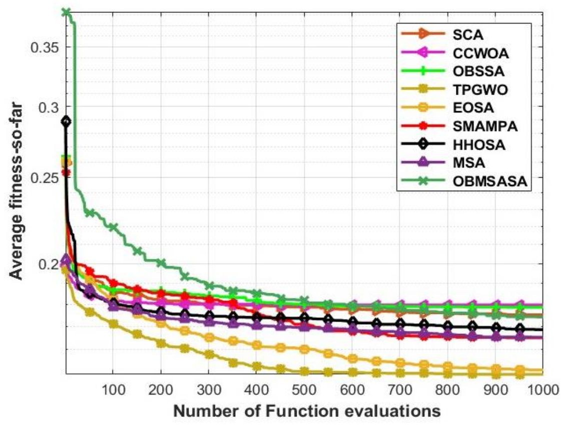


Figure 37. The average of fitness values for the Waveform dataset.

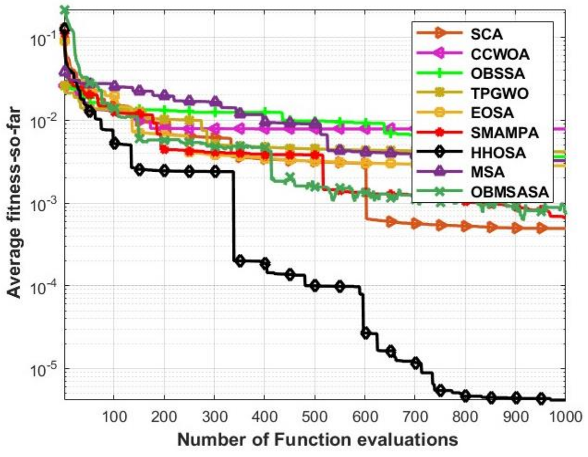


Figure 38. The average of fitness values for the Leukemia dataset.

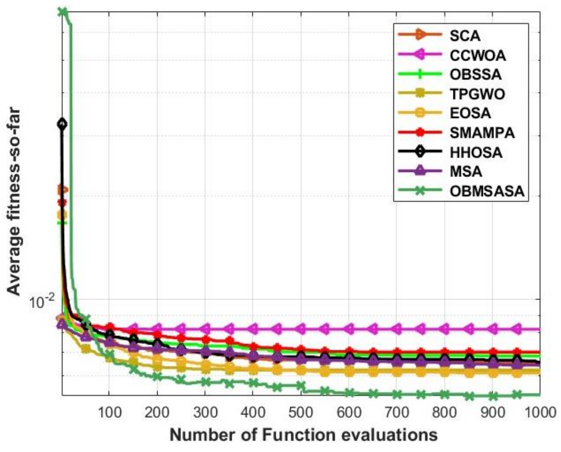


Figure 39. The average of fitness values for the Pendigits dataset

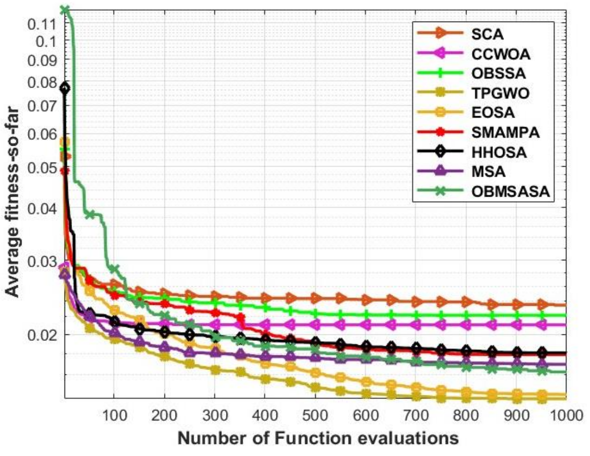


Figure 40. The average of fitness values for Optdigits dataset.

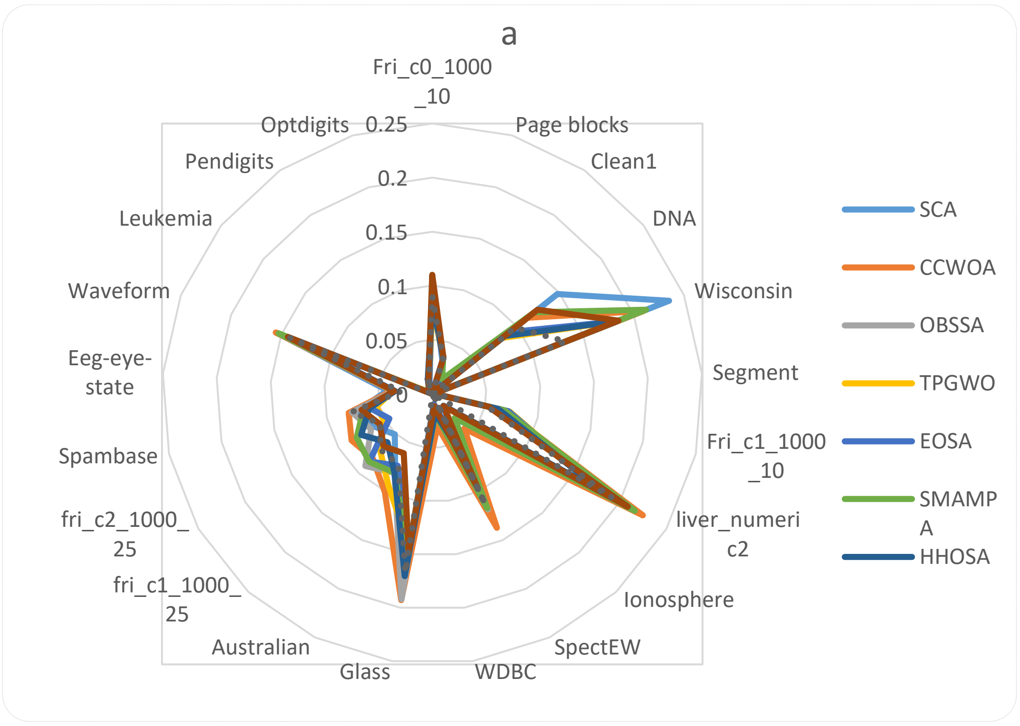


Figure 41 (a). The total of best values for all datasets.

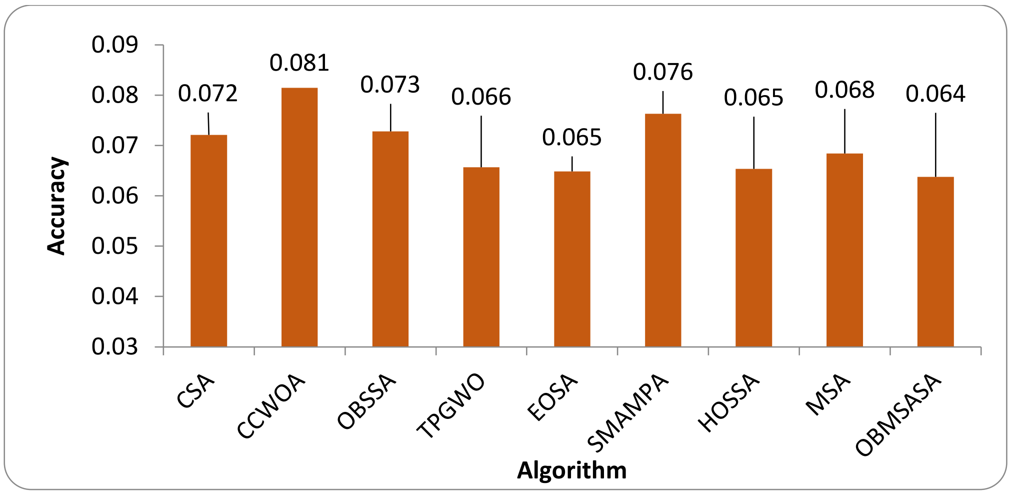


Figure 41 (b). The total of best values for all datasets.

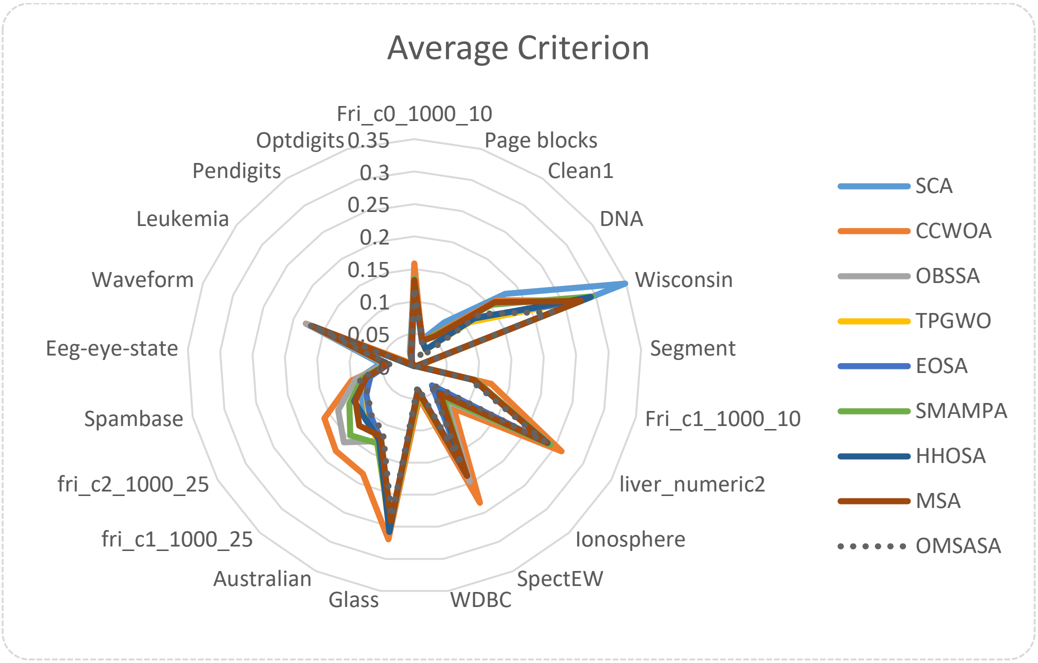


Figure 42 (a). The total average of fitness values for all datasets.

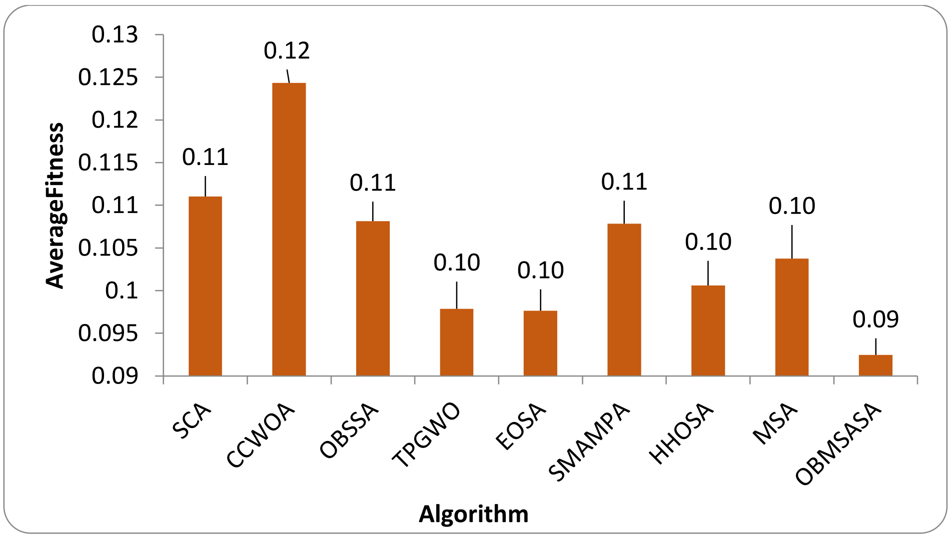


Figure 42 (b). The total average of fitness values for all datasets.

5.5.2 | Accuracy

within this subsection, we discuss the comparison between the proposed method and all other algorithms in terms of classification accuracy. According to Table 12, four criteria are used for assessment best classification accuracy, average classification accuracy, worst classification accuracy, and STD. The numerical results in the following table indicate that OBMSASA is the best model for achieving the best average classification rate. Figure 43 displays that the proposed OBMSASA achieves the total best classification accuracy rate with a value of 0.879874, followed by HHOSA, and EOSA comes in last place with a rate of 0.557036. Figure 44 shows that sixteen out of twenty-one datasets, OBMSASA is recording the maximum total average classification accuracy with a value of 0.909478667, followed by TPGWO, and then EOSA comes in the last rank with a value of 0.655656238.

Table 12. The results of classification accuracy criteria for all algorithms.

Dataset ID	Criteria (Accuracy)	CSA	CCWOA	OBSSA	TPGWO	EOSA	SMAMPA	HHOSA	MSA	OMSASA
1	Worst	0.835	0.805	0.785	0.84	0.6	0.83	0.845	0.84	0.855
	AVG	0.872833	0.846333	0.862833	0.877	0.677833	0.869167	0.880667	0.871667	**0.881167**
	Best	0.905	0.91	0.91	0.92	0.81	0.905	0.92	0.895	0.915
	STD	0.021603	0.023191	0.026771	0.017201	0.068086	0.019302	0.016543	0.014933	0.017503
2	Worst	0.95521	0.954296	0.946069	0.957038	0.914077	0.957038	0.957038	0.95521	0.954296
	AVG	0.963406	0.960481	0.961335	0.962249	0.937386	0.962828	0.963193	0.962188	**0.963985**
	Best	0.971664	0.969835	0.968921	0.968007	0.961609	0.969835	0.968921	0.968921	0.969835
	STD	0.00402	0.00363	0.004961	0.003085	0.014173	0.003215	0.003568	0.003991	0.00319
3	Worst	0.884211	0.852632	0.863158	0.936842	0.842105	0.894737	0.915789	0.905263	0.905263
	AVG	0.943158	0.922105	0.919649	0.970526	0.967368	0.945263	0.970526	0.948421	**0.985965**
	Best	0.989474	1	0.957895	1	1	0.989474	1	1	1
	STD	0.023554	0.025725	0.02237	0.017787	0.05627	0.025843	0.026283	0.023021	0.026563
4	Worst	0.846154	0.792779	0.799058	0.861852	0.298273	0.827316	0.844584	0.825746	0.841444
	AVG	0.88001	0.825118	0.832444	0.89079	0.489273	0.851701	**0.882051**	0.84584	0.873208
	Best	0.908948	0.855573	0.861852	0.915228	0.819466	0.882261	0.913658	0.875981	0.905808
	STD	0.012248	0.013502	0.014494	0.0141	0.177762	0.014398	0.018509	0.010169	0.016172
5	Worst	0.657895	0.552632	0.447368	0.578947	0.447368	0.631579	0.631579	0.657895	0.710526
	AVG	0.721053	0.65	0.648246	0.709649	0.532456	0.705263	0.70614	0.723684	**0.771053**
	Best	0.815789	0.763158	0.736842	0.815789	0.657895	0.789474	0.815789	0.815789	0.868421
	STD	0.043489	0.050855	0.071648	0.049103	0.056009	0.041695	0.045361	0.038941	0.037312
6	Worst	0.995671	0.993506	0.989177	0.997835	0.80303	0.995671	0.997835	0.997835	0.997835
	AVG	0.999206	0.998052	0.999062	0.999351	0.882973	0.999134	0.999062	0.999467	**0.999495**
	Best	1	1	1	1	0.995671	1	1	1	1
	STD	0.001204	0.002368	0.002178	0.001009	0.076396	0.001345	0.001091	0.000881	0.000931
7	Worst	0.85	0.815	0.845	0.88	0.69	0.875	0.865	0.88	0.885
	AVG	0.9075	0.881833	0.894167	0.9085	0.741833	0.906833	0.909833	0.9085	**0.9165**
	Best	0.945	0.935	0.935	0.93	0.795	0.935	0.94	0.95	0.95
	STD	0.018465	0.027433	0.022325	0.014212	0.026925	0.015892	0.016891	0.01672	0.014273
8	Worst	0.715517	0.715517	0.637931	0.724138	0.62069	0.706897	0.724138	0.732759	0.732759
	AVG	0.763506	0.738793	0.739655	0.762069	0.69454	0.758908	0.764368	0.766954	0.767241
	Best	0.801724	0.775862	0.801724	0.818966	0.767241	0.784483	0.818966	0.793103	**0.810345**
	STD	0.022145	0.017115	0.043394	0.017624	0.037929	0.015764	0.023124	0.016401	0.022297
9	Worst	0.9	0.857143	0.785714	0.928571	0.357143	0.871429	0.9	0.9	0.928571
	AVG	0.949524	0.912381	0.858095	0.956667	0.680952	0.935238	0.944762	0.943333	**0.96381**
	Best	0.985714	0.957143	0.928571	0.985714	0.857143	0.971429	0.985714	0.985714	1
	STD	0.023045	0.022738	0.037316	0.016129	0.18654	0.021464	0.02211	0.019668	0.015798
10	Worst	0.735849	0.584906	0.566038	0.735849	0.490566	0.773585	0.754717	0.735849	0.773585
	AVG	0.806289	0.75283	0.757233	0.828302	0.598113	0.822642	0.831447	0.815723	**0.832075**
	Best	0.90566	0.867925	0.886792	0.886792	0.754717	0.886792	0.943396	0.90566	0.886792

	STD	0.03836	0.066358	0.069499	0.046837	0.052939	0.035246	0.0443	0.034572	0.030685
11	Worst	0.938053	0.929204	0.920354	0.902655	0.80531	0.929204	0.946903	0.938053	0.938053
	AVG	0.962537	0.953392	0.954867	0.956932	0.89115	0.960177	0.963717	0.963422	**0.969322**
	Best	0.99115	0.973451	0.99115	0.99115	0.955752	0.982301	0.982301	0.99115	1
	STD	0.014266	0.011129	0.017752	0.019683	0.043176	0.015719	0.010738	0.013892	0.013892

Dataset ID	Accuracy	CSA	CCWOA	OBSSA	TPGWO	EOSA	SMAMPA	HHOSA	MSA	OBMSASA
12	Worst	0.666667	0.642857	0.571429	0.619048	0.380952	0.666667	0.666667	0.690476	0.690476
	AVG	0.752381	0.73254	0.750794	0.746032	0.546032	0.754762	0.743651	0.761111	**0.765873**
	Best	0.857143	0.809524	0.809524	0.857143	0.738095	0.857143	0.833333	0.857143	0.833333
	STD	0.049865	0.04085	0.044973	0.052438	0.104053	0.042477	0.039882	0.039784	0.042446
13	Worst	0.818841	0.73913	0.710145	0.753623	0.427536	0.826087	0.826087	0.84058	0.847826
	AVG	0.873913	0.81715	0.841304	0.870048	0.533333	0.869565	0.875362	0.881884	**0.888647**
	Best	0.92029	0.898551	0.913043	0.905797	0.731884	0.92029	0.92029	0.942029	0.927536
	STD	0.025645	0.05066	0.050217	0.031496	0.089937	0.022756	0.026189	0.022609	0.017799
14	Worst	0.82	0.725	0.61	0.83	0.59	0.78	0.835	0.805	0.865
	AVG	0.889667	0.8225	0.75	0.896833	0.718167	0.855833	0.8895	0.876333	**0.9015**
	Best	0.95	0.92	0.86	0.93	0.795	0.915	0.94	0.935	0.93
	STD	0.02616	0.045557	0.058339	0.019275	0.059051	0.03026	0.026664	0.028037	0.015928
15	Worst	0.86	0.76	0.71	0.885	0.685	0.82	0.845	0.84	0.88
	AVG	0.906833	0.8405	0.795	0.914833	0.7535	0.885333	0.893333	0.896167	**0.918667**
	Best	0.94	0.915	0.89	0.95	0.825	0.92	0.925	0.945	0.945
	STD	0.017394	0.042271	0.050034	0.018914	0.037832	0.022967	0.018399	0.024659	0.016606
16	Worst	0.890217	0.883696	0.88587	0.909783	0.38913	0.909783	0.902174	0.917391	0.903261
	AVG	0.912536	0.906703	0.911486	0.931304	0.464167	0.923949	**0.930616**	0.929964	0.916775
	Best	0.932609	0.927174	0.929348	0.95	0.816304	0.942391	0.946739	0.940217	0.927174
	STD	0.009928	0.010922	0.010463	0.008823	0.152159	0.008383	0.008074	0.006265	0.006753
17	Worst	0.9249	0.959613	0.950267	0.95761	0.525367	0.96028	0.961615	0.962617	0.960948
	AVG	0.955118	0.965866	0.962072	0.966834	0.582221	0.966644	0.967312	0.967279	**0.967**
	Best	0.966622	0.972296	0.972964	0.97263	0.967957	0.974299	0.974633	0.973632	0.975634
	STD	0.00951	0.002797	0.005748	0.003107	0.130408	0.003222	0.003222	0.003112	0.003313
18	Worst	0.806	0.804	0.803	0.835	0.3	0.819	0.822	0.826	0.809
	AVG	0.826867	0.825967	0.8241	0.853267	0.3571	0.840333	0.837567	**0.841**	0.8289
	Best	0.847	0.85	0.853	0.873	0.823	0.853	0.866	0.863	0.862
	STD	0.010881	0.011174	0.011081	0.008267	0.125152	0.009121	0.009507	0.008404	0.011177
19	Worst	**1**	0.928571	0.857143	0.928571	0.5	**1**	**1**	0.928571	**1**
	AVG	1	0.992857	0.985714	0.997619	0.690476	1	1	0.997619	1
	Best	1	1	1	1	1	1	1	1	1
	STD	0	0.021795	0.039347	0.013041	0.143267	0	0	0.013041	0
20	Worst	0.99545	0.993631	0.995905	0.997725	0.876251	0.99727	0.99636	0.99727	0.99818
	AVG	0.998044	0.997543	0.998226	0.998468	0.902002	0.998574	0.998514	0.998726	**0.999287**
	Best	1	1	0.999545	0.999545	0.999545	1	0.999545	1	1
	STD	0.001076	0.001509	0.000913	0.000514	0.038903	0.000651	0.000877	0.00067	0.000556
21	Worst	0.97153	0.978648	0.975979	0.986655	0.097865	0.983986	0.983986	0.985765	0.980427
	AVG	0.980961	0.986862	0.983155	0.99051	0.127906	0.989176	0.989739	**0.990985**	0.988582
	Best	0.991103	0.993772	0.992883	0.994662	0.987544	0.993772	0.993772	0.995552	0.993772
	STD	0.00496	0.003823	0.003642	0.002158	0.16236	0.002675	0.002671	0.002567	0.003002
AVG best accuracy		0.861335	0.82332	0.794081	0.860239	0.557036	0.860803	0.86843	0.865585	**0.879874**
AVG accuracy		0.89834	0.8728479	0.8680684	0.904180143	0.655656238	0.895301095	0.901969524	0.899541286	0.909478667

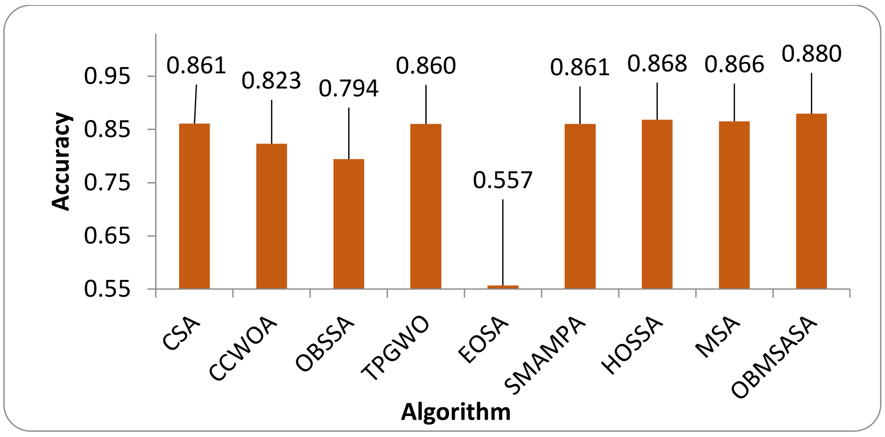


Figure 43. The total average of best accuracy for all datasets.

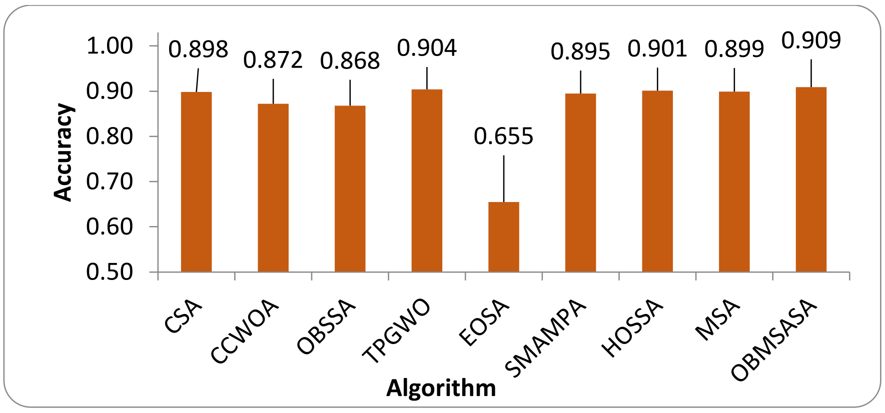


Figure 44. The total average of accuracy values for all datasets.

5.5.3 | Selected Features

The minimization of features is a crucial goal that is pursued to uphold the optimization of classification precision. It has been observed previously that OBMSASA surpasses alternative algorithms based on the appropriateness of values related to fitness and accuracy in classification. At this juncture, it is imperative to evaluate the capacity of OBMSASA to reduce features when juxtaposed with other metaheuristics. Table 13 presents the quantitative outcomes of the specified criteria regarding features. Performance monitoring incorporates four distinct criteria: the optimal subset of selected features, the average subset of selected features (AVG), the suboptimal subset of selected features, and the standard deviation (STD). Based on the numerical findings, it can be inferred that HHOSA demonstrates the most favorable average subset of selected features, registering a value of 12.280952. Following HHOSA, CSA performs well, while the proposed OBMSASA ranks third with a value of 25.728573. The overall average of selected features across all datasets is illustrated in Figure 45. Consequently, OBMSASA exhibits promising outcomes with the average selected features.

Table 13. The results of best SF, average SF, worst SF, and STD for all algorithms.

Dataset ID	(Selected Features)	CSA	CCWOA	OBSSA	TPGWO	EOSA	SMAMPA	HHOSA	MSA	OMSASA
1	Best	4	4	4	4	1	4	4	4	4
	AVG	5.166667	6.266667	5.833333	5.366667	6.5	5.833333	5.1	5.766667	5.666667
	Worst	7	9	9	7	20	9	7	8	8
	STD	0.874281	1.229896	1.234094	0.808717	4.804811	1.234094	0.884736	0.897634	0.884087
2	Best	2	2	2	2	1	2	2	3	3
	AVG	3.733333	4.5	4.266667	3.633333	7.1	3.733333	3.766667	3.866667	4.666667
	Worst	5	8	7	5	12	6	5	6	7
	STD	0.73968	1.479748	1.201532	0.718395	3.555957	1.048261	0.678911	0.730297	1.124441

3	Best	12	1	71	34	1	42	1	1	1
	AVG	31.4	77.8	84.4	46.26667	44.13333	75.96667	37.4	82.4	10.86667
	Worst	65	138	98	61	168	107	132	145	78
	STD	12.50545	23.67073	6.79046	6.741193	69.86822	16.70739	40.28827	33.71135	17.75646
4	Best	11	58	80	40	1	42	4	31	6
	AVG	24.46667	107.1667	90.16667	59.7	48.86667	99.33333	22.63333	104.8667	21.33333
	Worst	51	166	101	118	180	148	169	161	98
	STD	10.37814	27.47674	5.977448	18.70488	70.51375	21.39341	43.03205	28.26809	24.18083
5	Best	1	1	9	2	1	3	1	1	2
	AVG	3.466667	7.2	14.76667	4.8	12.43333	7.766667	3.333333	8.066667	5.233333
	Worst	8	23	19	17	32	13	7	21	18
	STD	1.357821	6.104831	2.661129	2.721561	12.0907	3.103761	1.561019	5.112077	3.370085
6	Best	2	2	4	2	1	2	2	2	2
	AVG	3.266667	5.7	7.433333	3.433333	10.96667	3.933333	3.266667	3.666667	4.633333
	Worst	5	9	11	5	19	7	5	6	14
	STD	0.907187	1.878187	2.144493	0.678911	7.014681	1.311312	0.827682	1.124441	2.579539
7	Best	2	3	3	3	1	3	3	3	3
	AVG	3.6	4.366667	4.466667	3.766667	6.466667	4.133333	4.166667	4	3.733333
	Worst	5	7	7	5	15	6	6	6	5
	STD	0.674665	1.217214	1.074255	0.678911	3.980412	0.730297	0.985527	0.982607	0.639684
8	Best	1	1	2	2	1	1	1	1	1
	AVG	2.9	3.366667	3.933333	3.433333	6.733333	3.433333	3.166667	3.433333	3.266667
	Worst	5	7	7	6	14	6	6	7	6
	STD	0.844863	1.828573	1.387961	0.971431	4.076284	1.104328	1.261727	1.524135	1.172481
9	Best	2	1	11	3	1	2	1	2	2
	AVG	4.1	6.9	17.16667	5.633333	13.13333	6.633333	3.366667	5.1	4.366667
	Worst	8	15	22	11	34	16	6	11	9
	STD	1.493665	4.130208	2.93708	2.008316	12.5251	3.189242	1.217214	2.090207	1.473521
10	Best	1	1	6	4	1	1	2	6	3
	AVG	5.566667	9.933333	11.46667	7.9	11.5	9.433333	8.666667	11.53333	7.3
	Worst	10	22	16	13	25	14	19	18	16
	STD	2.192201	6.119124	2.596195	2.426151	9.198013	3.16972	4.936377	2.763473	3.249934
11	Best	1	2	6	2	1	4	2	2	2
	AVG	3.233333	9.666667	12.73333	3.3	14.13333	7.833333	3.033333	6.766667	6.033333
	Worst	6	23	20	6	30	14	6	14	15
	STD	1.222866	4.655981	3.609693	1.316998	12.35602	2.889736	0.964305	2.528231	3.242852

Dataset ID	(Selected Features)	CSA	CCWOA	OBSSA	TPGWO	EOSA	SMAMPA	HHOSA	MSA	OMSASA
12	Best	2	1	2	2	1	2	2	2	2
	AVG	3.633333	4.23333	4.3	3.83333	7.233333	3.8	3.8	4.033333	4.5
	Worst	6	8	7	7	20	6	6	6	7
	STD	1.098065	1.612095	1.118805	1.17688	4.775536	1.242911	1.063501	0.964305	1.358244
13	Best	1	1	1	1	1	1	1	1	1
	AVG	2.733333	3.4	4.966667	2.933333	7.833333	3.6	3.133333	3.566667	3.6
	Worst	4	11	9	5	24	6	5	6	6
	STD	1.172481	2.672981	2.042367	1.112107	6.454527	1.248447	0.973204	1.304722	1.328728
14	Best	2	2	5	2	1	3	3	3	3
	AVG	3.833333	5	10.7	4.1	9.866667	5.833333	3.6	5.233333	4.2
	Worst	5	22	16	5	25	11	5	9	5
	STD	0.592093	3.895178	2.793465	0.758856	9.863668	1.801978	0.563242	1.50134	0.550861
	Best	2	1	4	2	1	2	3	2	3

15	AVG	3.733333	5.866667	10.1	3.833333	12.06667	5.1	3.533333	4.666667	3.966667
	Worst	7	19	15	5	25	10	5	9	5
	STD	1.048261	4.083046	2.795933	0.833908	11.0514	1.78789	0.62881	1.647011	0.556053
16	Best	15	26	24	16	1	19	26	34	18
	AVG	20.86667	38.6	30.03333	24	23.96667	38.43333	40.46667	41.5	30.3
	Worst	33	51	36	35	81	51	55	50	45
	STD	3.857222	6.71899	2.785224	4.250761	25.05646	6.521494	8.997062	4.431471	7.278452
17	Best	9	13	10	11	1	12	11	11	12
	AVG	10.86667	13.76667	12.33333	12.9	17.56667	13.1	12.96667	12.66667	13.4
	Worst	13	14	14	14	59	14	14	14	14
	STD	1.136642	0.430183	1.212957	0.711967	15.52458	0.711967	0.614948	0.660895	0.563242
18	Best	10	20	17	14	1	15	15	26	8
	AVG	15.03333	29.43333	23.06667	19.56667	15.36667	27.8	31.03333	31.4	19.13333
	Worst	21	40	33	26	40	35	38	38	33
	STD	2.953471	6.179406	3.512866	2.800041	12.37206	4.342254	6.088334	3.450137	7.532611
19	Best	145	1	3396	1183	1	25	1	22	1
	AVG	350.6667	537.5	3480.767	1279.333	1693.467	471.5333	2.966667	633.1333	325.8667
	Worst	801	2789	3556	1548	7129	1753	7	2322	1657
	STD	136.8708	706.1939	38.98778	79.45475	3051.48	412.0221	1.473521	642.9884	462.2105
20	Best	6	6	6	6	1	7	6	7	8
	AVG	7.4	9.2	8.7	7.533333	12.3	8.966667	8.133333	8.266667	11.46667
	Worst	9	13	11	9	29	11	10	10	15
	STD	0.813676	1.627352	1.118805	0.681445	9.180752	1.033352	1.074255	0.73968	1.814374
21	Best	21	33	30	22	1	35	32	33	36
	AVG	29.46667	51.46667	35.53333	29.76667	16	45.86667	50.36667	51.43333	46.76667
	Worst	39	61	43	37	64	56	62	59	59
	STD	4.651535	7.977267	3.401149	3.720246	18.80389	5.001609	7.608903	4.973609	4.789956
Average SF		25.6730179	44.825398	184.625412	73.096809	95.125413	40.5746014	12.280952	49.3031745	25.728573
Rank		2	5	9	8	7	4	1	6	3

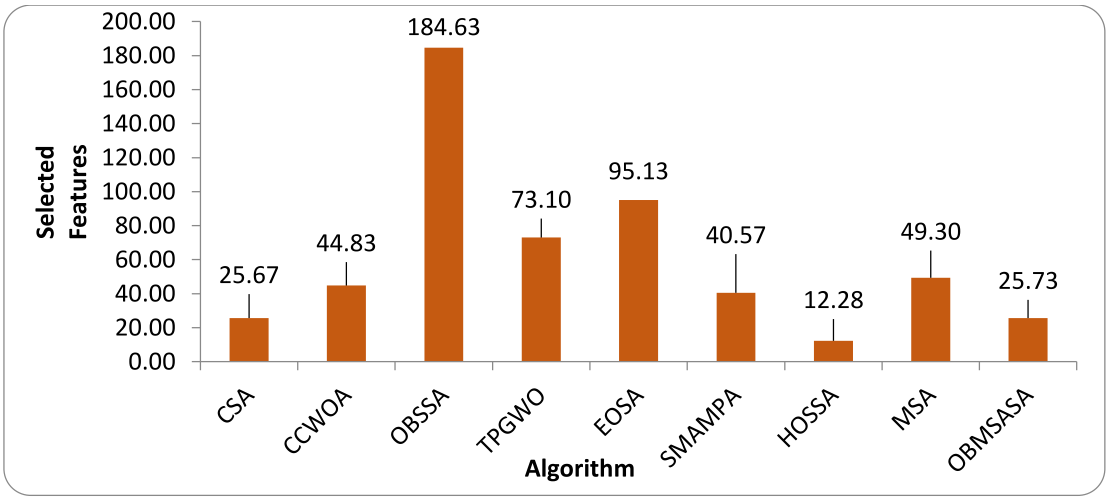


Figure 45. The total average of selected features for all datasets.

5.5.4 |Time Execution for OBMSASA

Table 14 indicates how much time each algorithm takes to execute all the datasets, from the numerical analysis, TPGWO comes in first place, followed by HHOSA, and OBMSASA comes in sixth place.

Table 14. The time execution for all algorithms.

Algorithm	CSA	CCWOA	OBSSA	TPGWO	EOSA	SMAMPA	HHOSA	MSA	OBMSASA
Time	1.981428	2.107838	2.023919	1.841559	1.943561	1.985466	1.966042	2.055254	2.008919

6 |Conclusions and Future Work

In the present investigation, a hybrid methodology integrating the Mantis Search Algorithm with the Opposition-based learning technique and SA algorithm OBMSASA is utilized to explore the optimal subset of features through a wrapper method. The algorithm put forward incorporates the use of KNN due to its widespread application, simplicity in implementation, and the presence of a solitary parameter for adjustment. The OBMSASA technique is utilized for 21 standardized datasets, with the potential for their dimensions to extend into the thousands. Within the context of the Feature Selection (FS) issue, the decision must be made whether to include a particular feature, resulting in a binary problem. Consequently, an adaptation function is integrated into the original MSA algorithm. The examination of the impact of V-shaped functions, S-shaped functions, and the threshold method on the proposed algorithm is initiated. Firstly, the selection of the Mantis Search algorithm was based on its demonstrated efficacy compared to numerous recently developed algorithms that have not been previously applied to address the research problem at hand, thus justifying its inclusion as the focal point of study. the threshold method demonstrates superior performance and rapid convergence towards the optimal solution throughout iterations in comparison to the S-shaped and V-shaped approaches. Secondly, to mitigate the risk of encountering local optima, the incorporation of the Opposition-based learning (OBL) technique within the framework of MSA is implemented in the initialization phase. OBL is used to better improve the spread of sample solutions in the research area. Thirdly, incorporating algorithm SA into algorithm MSA enhances the MSA algorithm's capacity to achieve optimal solutions as the SA algorithm is the local search component within the framework. The analysis of OBMSASA's performance is thoroughly scrutinized with seven highly esteemed metaheuristics published in this publication. The findings demonstrated the excellence of the suggested algorithm and its capacity to effectively address the issue under its remarkable skill in navigating the trade-off between exploration and exploitation, evading local optima, and enhancing population diversity. This superiority results from observing how well the algorithm performs with a number of parameters, including fitness, classification accuracy, and the chosen features. There are four numerical results for each criterion (best, worst, average, and STD). Future research should evaluate the suggested algorithm's performance using a variety of classifiers, such as support vector machines, and other classifiers. Another noteworthy development is the application of the FS classification to financial data, and the Internet of Things. One of the main drawbacks of the proposed algorithm is the computational time. To leverage computational resources and minimize processing time, we want to create a parallel version of OBMSASA, which will improve the algorithm's performance when handling large data dimension sizes.

Acknowledgments

The authors are grateful to the editorial and reviewers, as well as the correspondent author, who offered assistance in the form of advice, assessment, and checking during the study period.

Author Contributions

All authors contributed equally to this work.

Funding

This research was conducted without external funding support.

Data Availability

Not applicable.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Ghosh, A., A. Datta, and S. Ghosh, Self-adaptive differential evolution for feature selection in hyperspectral image data. Applied Soft Computing, 2013. 13(4): p. 1969-1977.

[2] Tang, J., S. Alelyani, and H. Liu, Feature selection for classification: A review. Data classification: Algorithms and applications, 2014: p. 37.

[3] Abdel-Basset, M., et al., Exponential distribution optimizer (EDO): A novel math-inspired algorithm for global optimization and engineering problems. Artificial Intelligence Review, 2023. 56(9): p. 9329-9400.

[4] Mirjalili, S. and S. Mirjalili, Genetic algorithm. Evolutionary algorithms and neural networks: Theory and applications, 2019: p. 43-55.

[5] Das, S. and P.N. Suganthan, Differential evolution: A survey of the state-of-the-art. IEEE transactions on evolutionary computation, 2010. 15(1): p. 4-31.

[6] Rao, R.V., V.J. Savsani, and D.P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-aided design, 2011. 43(3): p. 303-315.

[7] Moosavi, S.H.S. and V.K. Bardsiri, Poor and rich optimization algorithm: A new human-based and multi populations algorithm. Engineering applications of artificial intelligence, 2019. 86: p. 165-181.

[8] Mousavirad, S.J. and H. Ebrahimpour-Komleh, Human mental search: a new population-based metaheuristic optimization algorithm. Applied Intelligence, 2017. 47: p. 850-887.

[9] Wang, D., D. Tan, and L. Liu, Particle swarm optimization algorithm: an overview. Soft computing, 2018. 22: p. 387-408.

[10] Dorigo, M., M. Birattari, and T. Stutzle, Ant colony optimization. IEEE computational intelligence magazine, 2006. 1(4): p. 28-39.

[11] Yang, X.-S. and X. He, Firefly algorithm: recent advances and applications. International journal of swarm intelligence, 2013. 1(1): p. 36-50.

[12] Delahaye, D., S. Chaimatanan, and M. Mongeau, Simulated annealing: From basics to applications. Handbook of metaheuristics, 2019: p. 1-35.

[13] Rashedi, E., E. Rashedi, and H. Nezamabadi-Pour, A comprehensive survey on gravitational search algorithm. Swarm and evolutionary computation, 2018. 41: p. 141-158.

[14] Abualigah, L., et al., The arithmetic optimization algorithm. Computer methods in applied mechanics and engineering, 2021. 376: p. 113609.

[15] Gabis, A.B., et al., A comprehensive survey of sine cosine algorithm: variants and applications. Artificial Intelligence Review, 2021. 54(7): p. 5469-5540.

[16] Guha, R., et al., Discrete equilibrium optimizer combined with simulated annealing for feature selection. Journal of Computational Science, 2023. 67: p. 101942.

[17] Abdel-Basset, M., et al., A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection. Expert Systems with Applications, 2020. 139: p. 112824.

[18] Abdel-Basset, M., W. Ding, and D. El-Shahat, A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection. Artificial Intelligence Review, 2021. 54(1): p. 593-637.

[19] Ewees, A.A., et al., Enhanced feature selection technique using slime mould algorithm: A case study on chemical data. Neural Computing and Applications, 2023. 35(4): p. 3307-3324.

[20] Taghian, S. and M.H. Nadimi-Shahraki, Binary sine cosine algorithms for feature selection from medical data. arXiv preprint arXiv:1911.07805, 2019.

[21] Tubishat, M., et al., Improved Salp Swarm Algorithm based on opposition based learning and novel local search algorithm for feature selection. Expert Systems with Applications, 2020. 145: p. 113122.

[22] Chakraborty, S., et al., Horizontal crossover and co-operative hunting-based Whale Optimization Algorithm for feature selection. Knowledge-Based Systems, 2023. 282: p. 111108.

[23] Abdel-Basset, M., et al., Mantis Search Algorithm: A novel bio-inspired algorithm for global optimization and engineering design problems. Computer Methods in Applied Mechanics and Engineering, 2023. 415: p. 116200.

[24] Kohavi, R. and G.H. John, Wrappers for feature subset selection. Artificial intelligence, 1997. 97(1-2): p. 273-324.

[25] Gharehchopogh, F.S., I. Maleki, and Z.A. Dizaji, Chaotic vortex search algorithm: metaheuristic algorithm for feature selection. Evolutionary Intelligence, 2022. 15(3): p. 1777-1808.

[26] Sun, L., et al., A hybrid feature selection framework using improved sine cosine algorithm with metaheuristic techniques. Energies, 2022. 15(10): p. 3485.

[27] Kareem, S.S., et al., An effective feature selection model using hybrid metaheuristic algorithms for iot intrusion detection. Sensors, 2022. 22(4): p. 1396.

[28] Kunhare, N., R. Tiwari, and J. Dhar, Intrusion detection system using hybrid classifiers with meta-heuristic algorithms for the optimization and feature selection by genetic algorithm. Computers and Electrical Engineering, 2022. 103: p. 108383.

[29] El-Kenawy, E.-S.M., et al., Novel meta-heuristic algorithm for feature selection, unconstrained functions and engineering problems. IEEE Access, 2022. 10: p. 40536-40555.

[30] Li, J., et al., Teaching–learning guided salp swarm algorithm for global optimization tasks and feature selection. Soft Computing, 2023. 27(23): p. 17887-17908.

[31] Mahapatra, A.K., N. Panda, and B.K. Pattanayak, Quantized Salp Swarm Algorithm (QSSA) for optimal feature selection. International journal of information technology, 2023. 15(2): p. 725-734.

[32] Jain, S. and R. Dharavath, Memetic salp swarm optimization algorithm based feature selection approach for crop disease detection system. Journal of Ambient Intelligence and Humanized Computing, 2023. 14(3): p. 1817-1835.

[33] Alhussan, A.A., et al., A Binary Waterwheel Plant Optimization Algorithm for Feature Selection. IEEE Access, 2023.

[34] Fang, L. and X. Liang, A Novel Method Based on Nonlinear Binary Grasshopper Whale Optimization Algorithm for Feature Selection. Journal of Bionic Engineering, 2023. 20(1): p. 237-252.

[35] Uzer, M.S. and O. Inan, A novel feature selection using binary hybrid improved whale optimization algorithm. The Journal of Supercomputing, 2023: p. 1-26.

[36] Ragab, M., Hybrid firefly particle swarm optimisation algorithm for feature selection problems. Expert Systems, 2023.

[37] Xu, Z., et al., Enhanced Gaussian bare-bones grasshopper optimization: mitigating the performance concerns for feature selection. Expert Systems with Applications, 2023. 212: p. 118642.

[38] Hafez, A.I., et al. Sine cosine optimization algorithm for feature selection. in 2016 international symposium on innovations in intelligent systems and applications (INISTA). 2016. IEEE.

[39] Rivera, J. and Y. Callohuari, A new species of praying mantis from Peru reveals impaling as a novel hunting strategy in Mantodea (Thespidae: Thespini). Neotropical Entomology, 2020. 49(2): p. 234-249.

[40] Iwasaki, T., Predatory behavior of the praying mantis, Tenodera aridifolia I. Effect of prey size on prey recognition. Journal of Ethology, 1990. 8(2): p. 75-79.

[41] Altman, N.S., An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 1992. 46(3): p. 175-185.

[42] Mirjalili, S. and A. Lewis, S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm and Evolutionary Computation, 2013. 9: p. 1-14.

[43] Lichman, M., UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2013). 2017.

[44] Abdel-Basset, M., et al., Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems. Knowledge-Based Systems, 2023. 262: p. 110248.

[45] Abdel-Basset, M., et al., Young's double-slit experiment optimizer: A novel metaheuristic optimization algorithm for global and constraint optimization problems. Computer Methods in Applied Mechanics and Engineering, 2023. 403: p. 115652.

[46] Abdel-Basset, M., et al., Spider wasp optimizer: A novel meta-heuristic optimization algorithm. Artificial Intelligence Review, 2023: p. 1-64.

[47] Bai, J., et al., A sinh cosh optimizer. Knowledge-Based Systems, 2023. 282: p. 111081.

[48] Abdel-Basset, M., et al., Exponential distribution optimizer (EDO): a novel math-inspired algorithm for global optimization and engineering problems. Artificial Intelligence Review, 2023: p. 1-72.

[49] Trojovská, E., M. Dehghani, and P. Trojovský, Zebra optimization algorithm: A new bio-inspired optimization algorithm for solving optimization algorithm. IEEE Access, 2022. 10: p. 49445-49473.

[50] Mafarja, M., et al., Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. Knowledge-Based Systems, 2018. 145: p. 25-45.

[51] Rajamohana, S. and K. Umamaheswari, Hybrid approach of improved binary particle swarm optimization and shuffled frog leaping for feature selection. Computers & Electrical Engineering, 2018. 67: p. 497-508.

[52] Too, J., A.R. Abdullah, and N. Mohd Saad. A new co-evolution binary particle swarm optimization with multiple inertia weight strategy for feature selection. in Informatics. 2019. MDPI.

[53] Mafarja, M.M. and S. Mirjalili, Hybrid binary ant lion optimizer with rough set and approximate entropy reducts for feature selection. Soft Computing, 2019. 23(15): p. 6249-6265.

[54] Arora, S. and P. Anand, Binary butterfly optimization approaches for feature selection. Expert Systems with Applications, 2019. 116: p. 147-160.

[55] Emary, E., H.M. Zawbaa, and A.E. Hassanien, Binary ant lion approaches for feature selection. Neurocomputing, 2016. 213: p. 54-65.

[56] Agrawal, R., B. Kaur, and S. Sharma, Quantum based whale optimization algorithm for wrapper feature selection. Applied Soft Computing, 2020. 89: p. 106092.

[57] Faris, H., et al., An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. Knowledge-Based Systems, 2018. 154: p. 43-67.