computers

MDPI

Article

An Improved Dandelion Optimizer Algorithm for Spam Detection: Next-Generation Email Filtering System

Mohammad Tubishat [1], Feras Al-Obeidat [1], Ali Safaa Sadiq [2], Seyedali Mirjalili [3]

[1] College of Technological Innovation, Zayed University, Abu Dhabi, United Arab Emirates, mohammad.tubishat@zu.ac.ae, Feras.Al-Obeidat@zu.ac.ae
[2] Department of Computer Science, Nottingham Trent University, Clifton Lane, Nottingham NG11 8NS, United Kingdom, Ali.Sadiq@ntu.ac.uk
[3] Centre of Artificial Intelligence Research and Optimisation, Torrens University Australia, Brisbane, Australia, ali.mirjalili@gmail.com
* Correspondence: ali.sadiq@ntu.ac.uk; (A.S)

Abstract: Spam emails have become a pervasive issue in recent years, as internet users receive increasing amounts of unwanted or fake emails. To combat this issue, automatic spam detection methods have been proposed, which aim to classify emails into spam and non-spam categories. Machine learning techniques have been utilized for this task with considerable success. In this paper, we introduce a novel approach to spam email detection by presenting significant advancements to the Dandelion Optimizer (DO) algorithm. DO is a relatively new nature-inspired optimization algorithm inspired by the flight of dandelion seeds. While DO shows promise, it faces challenges, especially in high-dimensional problems such as feature selection for spam detection. Our primary contributions focus on enhancing the DO algorithm. Firstly, we introduce a new local search algorithm based on flipping (LSAF), designed to improve DO's ability to find the best solutions. Secondly, we propose a reduction equation that streamlines the population size during algorithm execution, reducing computational complexity. To showcase the effectiveness of our modified DO algorithm, which we refer to as Improved DO (IDO), we conduct a comprehensive evaluation using the Spam base dataset from the UCI repository. However, we emphasize that our primary objective is to advance the DO algorithm, with spam email detection serving as a case study application. Comparative analysis against several popular algorithms, including Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Generalized Normal Distribution Optimization (GNDO), Chimp Optimization Algorithm (ChOA), Grasshopper Optimization Algorithm (GOA), Ant Lion Optimizer (ALO), and Dragonfly Algorithm (DA), demonstrates the superior performance of our proposed IDO algorithm. It excels in accuracy, fitness, and the number of selected features, among other metrics. Our results clearly indicate that IDO overcomes the local optima problem commonly associated with the standard DO algorithm, owing to the incorporation of LSAF and the reduction equation methods. In summary, our paper underscores the significant advancement made in the form of the IDO algorithm, which represents a promising approach for solving high-dimensional optimization problems, with a keen focus on practical applications in real-world systems. While we employ spam email detection as a case study, our primary contribution lies in the improved DO algorithm, which is efficient, accurate, and outperforms several state-of-the-art algorithms in various metrics. This work opens avenues for enhancing optimization techniques and their applications in machine learning.

Keywords: Dandelion Optimizer (DO), Cybersecurity, Optimization, feature selection, Trusted Emails, Next-Generation Spam Email Detection

Citation: To be added by editorial staff during production.

Academic Editor: Firstname Lastname

Received: date
Revised: date
Accepted: date
Published: date

1. Introduction

With the increasing use of the internet and online social networks (OSNs) applications, communication and exchange of information among users have similarly increased. Along with this increased communication comes the problem of spam, which is a frequent issue that users of these applications frequently face. One of the most common forms of spam is unsolicited emails or spam emails, which fill up email inboxes and take time for users to check and delete [1], [2]. The problem of spam is not limited to just email but also affects other network applications [3]. For instance, users of social networking sites often receive unwanted messages or comments from fake accounts or spammers. Such messages can be annoying, and harmful, and can lead to privacy breaches, identity theft, and financial losses [4].

To address this problem, spam filtering software is developed and employed to detect and remove spam emails [5]. However, these filtering systems may not be accurate all the time and may mistakenly classify legitimate emails as spam [6]. This can lead to users missing out on important information or communication, such as important emails for job offers, contracts to sign, important appointments, etc. Additionally, spammers can use various tactics to bypass these filters and send malicious emails that are designed to deceive users into disclosing their personal information, passwords, or financial details. Therefore, a robust and accurate spam detection method is necessary to detect and prevent these threats [7].

In this research paper, therefore, we propose an automatic spam email detection method based on improved Discrete Optimization (IDO) algorithm. Discrete Optimization (DO) is a well-known optimization algorithm that has been widely used in various fields, including computer science, engineering, and operations research. Besides, DO has a promising protentional use in classification-related problems, such as spam email detection methods. However, DO has a problem of being stuck in local optima [8]–[10]. To overcome this problem, we propose several improvements to DO, including the development of a new local search algorithm that works based on the use of feature flipping, called the Local Search Algorithm with Flipping (LSAF). LSAF will update the best solution, ensuring that the algorithm is not stuck in a local optimum. Also, this paper introduces the use of a mathematical formula in maintaining the population size, which will result in a maintained complexity and sustain the algorithm's accuracy accordingly.

On the other hand, the proposed algorithm employs a wrapper-based feature selection method. This method is a supervised learning method that picks the most relevant features to be used for classification tasks. This method seeks to choose the features that are most dominant to the classification task while ignoring irrelevant or redundant features that may affect the classification accuracy. By introducing such a method, the computational time and complexity will be maintained at an acceptable level and will improve the accuracy of the spam detection system.

Over the last few years, spam detection has been one of the active research areas, and diverse systems have been proposed and developed to address such a high-dimensional problem. Such approaches can be generally classified into three groups: rule-based approaches, content-based approaches, and machine learning-based approaches. The rule-based approaches use a set of predefined rules to be used in the process of identifying spam emails. Though, these approaches have constrained accuracy, which might lead to not identifying new or foreign spam emails [11]. In contrast, content-based approaches use the content of the email as a way to flag spam emails. In other words, these approaches analyze the email's text, images, links, and other relevant features to verify whether it is a spam email or not. Yet, such approaches may be ineffective as opposed to advanced spamming methods that are mainly designed to bypass content-based filters [12]. While on the other side, machine learning-based approaches use statistical and machine learning methods to learn from a data sample and learn to classify emails as spam or legitimate [13].

Such approaches have demonstrated promising results and are broadly used in current spam detection systems [14].

Though, regardless of the developments in spam detection approaches, such a problem stays a challenging issue forth. The reason is that spammers are always evolving their strategies as a way to skirt the detection systems. This scenario makes it essential to keep developing new and adaptive spam detection systems. Hence, over recent years, one of the research fields that has gained consideration was the use of optimization algorithms in developing spam email detection [15]. Optimization algorithms, such as the DO algorithm, have proven their efficacy in resolving several optimization problems [16], [17]. For instance, the use of such algorithms for spam detection implies the selection of relevant features and the optimization of the classifier's parameters to achieve high accuracy. Beyond algorithmic advancements, our work is designed with practicality in mind, offering insights into the seamless integration of our method into operational email filtering systems.

In this paper, therefore, to solve such a crucial problem, we propose an improved DO algorithm to be used for automated spam email detection methods. Our proposed algorithm employs the LSAF technique and the mathematical question for population size reduction as a way to resolve the issue of local optima and cut down the algorithm's complexity. Moreover, in our proposed algorithm a wrapper-based feature selection technique is used to select the most relevant features for spam classification.

To summarize, the main contributions of this paper are:

- **Improved DO Algorithm:** Introduction of the Improved Dandelion Optimizer (IDO) algorithm, addressing local optima issues and enhancing optimization.
- **Local Search Enhancement:** Introduction of the Local Search Algorithm with Flipping (LSAF) to improve solution quality within the IDO algorithm.
- **Population Size Reduction:** Proposal of a mathematical formula for population size reduction, reducing computational complexity in spam detection.
- **Efficient Feature Selection:** Application of a wrapper-based feature selection method to efficiently select relevant features in spam detection, enhancing classification accuracy.
- **Effective Case Study Application:** Demonstrated the practical application of the IDO algorithm through a case study on spam email detection, showcasing its efficiency and accuracy in a real-world scenario.

The rest of the paper is structured as follows: section 2 presents and discusses some of the recent related works on spam email detection methods, and more specifically the use of optimization algorithms in this context. While section 3, we provide a detailed description of the the native DO algorithm along with some extra visulizations of the algorithm's nature behaviour. Also, in section 3, the algorithm's steps are explained, including the feature selection process, and the optimization of the classifier's parameters. In section 4, the implementation details and the experimental setup of the case study applicatio for Spam Email Detection are also discussed. The results and performance evaluation of the proposed IDO algorithm in contrast with the other benchmarked methods are presented and discussed in section 5. Finally, section 6 concludes the paper and suggests future work and directions for improvement.

2. Related Works

Over the last few years, optimization algorithms have been used widely for feature selection in spam email detection methods. Numerous search studies have proposed several optimization-based spam detection approaches, and their efficiency and powers have been widely analyzed. For example, in Sokhangoee and [18], a spam detection method based on association-rule mining and a genetic algorithm is proposed. The method

achieved high accuracy in detecting spam emails, though it was suffering from high computational complexity. In contrast, [19] proposed a spam detection method based on the combination of the Harris Hawks Optimizer (HHO) and the KNN classifier. The method has demonstrated promising results in terms of accuracy and processing time. Though, the HHO algorithm by its nature is heavily dependent on the random initialization of its parameters, which may impact its stability and reproducibility.

On the other hand [20] introduced a spam detection method based on the Horse Herd Optimization Algorithm (HOA) with a KNN classifier. Their method gained high accuracy in detecting spam emails, nonetheless, its performance could be heavily impacted by the sensitivity of the HOA algorithm, which comes from the nature of its parameter settings.

Another attempt [21] proposed the use of the Symbiotic Organisms Search (SOS) algorithm in the spam email detection mechanism. Their proposed approach has demonstrated high accuracy in detecting spam emails and performed well in contrast with other optimization-based approaches. Yet, the introduced computational cost could be relatively high, which bounds its feasibility with a large-scale spam detection problem.

On the other hand, the authors in [22] suggested the use of the sine–cosine algorithm (SCA) in detecting spam emails. The proposed approach has performed well in terms of accuracy and processing time. Though the performance could be limited by the nature of the SCA algorithm's sensitivity, due to the way its parameter settings. Hence, such a method will not be a reliable option, especially when it comes to the highly sensitive nature of the detection process of spam email filtering mechanisms.

The authors in [23] introduced the Water Cycle Optimization (WCO) algorithm in conjunction with Simulated Annealing (SA) to be used in detecting spam emails. Though their proposed method has demonstrated high accuracy in detecting spam emails, its computational complexity was relatively high.

From the presented methods and approaches, we can find out the potential use of optimization algorithms in spam email detection. Though, their performance varies depending on specific algorithmic features, parameter settings, and computational complexity. Additional research is therefore needed as a way to develop more efficient and effective optimization-based spam detection methods.

Table 1. Comparison of the Nature-Inspired Metaheuristic Algorithms.

Nature-Inspired Metaheuristics				
	Evolutionary Algorithms	**Swarm-Based Algorithms**	**Physical-Based Algorithms**	**Other Metaheuristics**
Population	Genetic Algorithms	Particle Swarm Algorithms	Simulated Annealing	Grey Wolf Optimization
Individual	Differential Evaluation Strategies	Firefly Algorithms	Harmony Search	Artificial Bee Colony Algorithm
Optimization Strategy	Evolutionary Programming	Ant Colony Optimization Algorithm	Memetic Algorithms	Imperialist Competitive Algorithm

In Table 1 a comparison of some of the common nature-inspired metaheuristic algorithms based on their population, individual, and optimization strategies are listed. The evolutionary types of algorithms, such as genetic algorithms and differential evaluation strategies, generally depend on the concept of natural selection to optimize solutions over a population of individuals. While, swarm-based algorithms, such as particle swarm

optimization and firefly algorithms, mimic the collective behaviour of social swarms to optimize the given solutions. Physical-based algorithms, such as simulated annealing and harmony search, are inspired by physical phenomena like thermal energy and musical harmony to optimize solutions. Other metaheuristic algorithms, such as grey wolf optimization [24], artificial bee colony algorithm, and imperialist competitive algorithm, draw inspiration from various sources to optimize solutions.

It's important to note that the choice of metaheuristic algorithm is heavily dependent on the specific optimization problem at hand. For instance, swarm-based algorithms are often used for optimization problems that require the exploration of a large search space, while physical-based algorithms are often used for optimization problems that require the optimization of a continuous function. In addition, hybrid metaheuristic algorithms that combine different techniques from different categories have been proposed to achieve better performance in optimization problems.

In order to demonstrate some of the key analysis aspects that could be used in comparing optimization techniques, below are some analysis points that could be used to highlight the competency of the related works:

Performance comparison: In addition to listing the strengths and weaknesses of each algorithm. This comparison can be done based on various metrics such as accuracy, precision, recall, F1 score, etc. The comparison can also be done on different datasets to evaluate the generalizability of the algorithms.

Impact of feature selection: Many of the algorithms mentioned in the related works section use feature selection techniques to improve the accuracy of spam detection. This analysis could demonstrate the impact of feature selection on the performance of the algorithms. This analysis could also include comparing the performance of algorithms with and without feature selection and also comparing different feature selection techniques.

- **Analysis of false positives and false negatives:** False positives and false negatives are common errors in spam detection. An analysis of the false positives and false negatives generated by each algorithm could be used on each of these algorithms to compare and contrast them. This analysis could help identify the specific types of emails that are misclassified by each algorithm and suggest improvements to reduce these errors.
- **Robustness analysis:** The robustness of the algorithms could be analyzed by testing their performance under different scenarios such as varying spam densities, different types of spam, and changes in the email dataset. This analysis could help evaluate the generalizability of the algorithms and identify scenarios where they may not perform well.
- **Comparison with traditional spam detection methods:** Such comparison could compare the performance of the optimization algorithms with traditional rule-based and content-based spam detection methods. This comparison could help evaluate the effectiveness of optimization algorithms in improving the accuracy of spam detection.
- **Analysis of computational efficiency:** Optimization algorithms can be computationally expensive, especially when dealing with large datasets. The computational efficiency of each algorithm could be analyzed and compared with their run times on different datasets. This analysis could help identify the most efficient algorithms and suggest improvements to reduce their computational cost.
- On the other hand, DO is a relatively new optimization algorithm that has been applied to various optimization problems, including feature selection and classification tasks, which has the potential to be used for spam detection. As with any other optimization algorithm, DO has some limitations, which are listed as follows:
- **Premature Convergence:** DO tends to converge prematurely to local optima, which can result in suboptimal solutions [8]. This is a common problem in many optimization algorithms and the DO algorithm is no exception.

- **Sensitivity to Initialization:** DO's performance can be sensitive to the initial population's quality and diversity [25]. Poor initialization can lead to premature convergence, while good initialization can improve the algorithm's performance.
- **Lack of Diversity:** DO does not have mechanisms to maintain population diversity, which can cause premature convergence and limit the algorithm's exploration capabilities [26].
- **Limited Search Space Exploration:** DO's search capabilities are limited, as it only explores a small portion of the search space at each iteration. This can result in suboptimal solutions and can make it difficult to find the global optimum [27].
- **Computational Complexity:** DO's computational complexity can be high, particularly for large-scale problems. The algorithm involves evaluating fitness functions, which can be computationally expensive, and the algorithm's complexity can increase with the problem's dimensionality [28].
- **Lack of Theoretical Analysis:** DO's theoretical analysis is still limited, and there are few theoretical guarantees of its convergence and performance under different conditions. This makes it difficult to understand the algorithm's behaviour and to design effective parameter settings [29].

In summarizing the performance evaluation of the DO algorithm, it has exhibited encouraging outcomes in certain applications; however, researchers need to acknowledge its limitations and drawbacks when considering its application to their specific optimization problems. To enhance the algorithm's effectiveness, researchers should investigate strategies to address and overcome these limitations.

While many optimization techniques have been utilized in literature for feature selection in spam email detection, the No Free Lunch Theorem (NFL) [30] suggests that no single solution can be applied to all problems and outperform all other algorithms. Hence, researchers continue to investigate the use of the most recent optimization algorithms for spam email detection, including DO.

However, as mentioned earlier, DO is susceptible to local optima, which limits its effectiveness. To address this, this paper proposes two main improvements to combine with the DO algorithm to enhance its performance and overcome its weaknesses.

To conclude this section, Table 2 provides a summary of several optimization algorithms, including Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Hill Climbing, Simulated Annealing, and Tabu Search. The strengths and weaknesses of each algorithm are listed, as well as their effectiveness in email spam detection. The table suggests that PSO, GA, ACO, and ABC have shown promising results in email spam detection, particularly for feature selection and email classification. However, each algorithm has its limitations and requires careful parameter tuning for optimal performance. Hill Climbing, Simulated Annealing, and Tabu Search have been used successfully for email classification but may not be as effective as other optimization algorithms for feature selection. Overall, the table provides a useful reference for researchers to choose an appropriate optimization algorithm for their email spam detection problem based on their specific requirements and constraints.

Table 2. Summary of Optimization Algorithms Application in Email Spam Detection.

Optimization Algorithm	Description	Strengths	Weaknesses	Effectiveness in Email Spam Detection
Particle Swarm Optimization (PSO)	A population-based optimization algorithm that involves particles moving around in the search space to find the best solution.	Good for feature selection, can handle high-dimensional data, easy to implement.	Can get stuck in local optima, sensitive to parameter settings.	Has shown promising results in email spam detection, particularly for feature selection and email classification.

Optimization Algorithm	Description	Strengths	Weaknesses	Effectiveness in Email Spam Detection
Genetic Algorithm (GA)	A population-based optimization algorithm that involves creating a population of potential solutions and then applying selection, crossover, and mutation operations to evolve the population over generations.	Can handle non-linear and non-convex problems and can find multiple optimal solutions.	Can be slow, requires careful parameter tuning, and may suffer from premature convergence.	Has been used successfully for email spam detection, particularly for email classification.
Ant Colony Optimization (ACO)	An optimization algorithm that uses pheromone trails to guide the search process.	Good for feature selection, can handle high-dimensional data, and can find global optima.	Can be slow, sensitive to parameter settings, and may suffer from premature convergence.	Has shown promising results in email spam detection, particularly for feature selection and email classification.
Artificial Bee Colony (ABC)	An optimization algorithm that involves employed bees, onlooker bees, and scout bees to explore the search space.	Good for finding global optima, easy to implement.	Can be slow, sensitive to parameter settings, and could suffer from premature convergence.	Has been used successfully for email spam detection, particularly for email classification.
Hill Climbing	A local search algorithm that iteratively improves the current solution by making small changes to it.	Simple and fast, can handle large datasets.	Can get stuck in local optima and could not find global optima.	Has been used successfully for email classification but may not be as effective as other optimization algorithms for feature selection.
Simulated Annealing	An optimization algorithm that starts with a high "temperature" and then gradually decreases it to find the best solution.	Able to find global optima, and manage noisy data.	Can be slow, and sensitive to parameter settings.	Has been used successfully for email classification but may not be as effective as other optimization algorithms for feature selection.
Tabu Search	A metaheuristic algorithm that is based on the concept of intensification and diversification.	Able to solve non-linear and non-convex problems, also finding global optima.	Can be slow and requires careful parameter tuning.	Has been used successfully for email classification but may not be as effective as other optimization algorithms for feature selection.

3. Dandelion Optimizer

The DO algorithm is inspired by the flight of dandelion seeds, as they grow and travel through the air [31]. This optimization algorithm utilizes mathematical models of the three stages of dandelion seed flight: rising, descending, and landing.

- **The rising phase:** During the rising phase, dandelion seeds are influenced by a pulling force in the weather that is both sunny and windy. A vortex forms above the seed, causing it to ascend into the air.
- **The descending phase:** Once the seed reaches a certain height, it enters the descending phase, where it falls steadily towards the ground.
- **The landing phase:** During the landing phase, dandelion seeds fall randomly due to the influence of wind and weather, ultimately landing in one location to sprout new dandelions.

By modelling these stages, the DO algorithm attempts to replicate the behaviour of dandelion seeds in order to optimize various functions. Nevertheless, it is noteworthy to highlight that the DO algorithm holds some limitations, such as its likelihood to be trapped in the local optima solutions. Therefore, it is important to explore some of the potential ways for boosting the algorithm's performance when applying such an algorithm in solving some of the complex optimization problems.

As a way to demonstrate the movement patterns of dandelion seeds, a simulated movement trajectory is presented in Figure 1. We have implemented a flight path simulation for a dandelion seed, considering the prevailing wind speed and direction. We started by identifying the seed's initial position, the wind speed and direction. Besides, in the developed simulation, we specified the number of iterations to 50 and the step size as 0.1, which can be adjusted according to the simulated scenario. The visualized seed's flight trajectory is generated with a little circle marker that indicates the starting position of the dandelion seed as the x and y-axis equal zero.

Subsequently, the simulation iterates over the specified number of 50 iterations. At each iteration, the new position of the seed is calculated based on factors such as wind speed, direction, and step size. A line is then plotted to depict the trajectory of the dandelion seed from its previous position to its current location. The simulation continues until the predetermined number of iterations is reached.

This example effectively demonstrates how mathematical models can be utilized to simulate the flight path of dandelion seeds under varying wind conditions. It provides a tangible illustration of the application of mathematical simulations in understanding and analyzing the behaviour of dandelion seeds in response to different wind parameters.

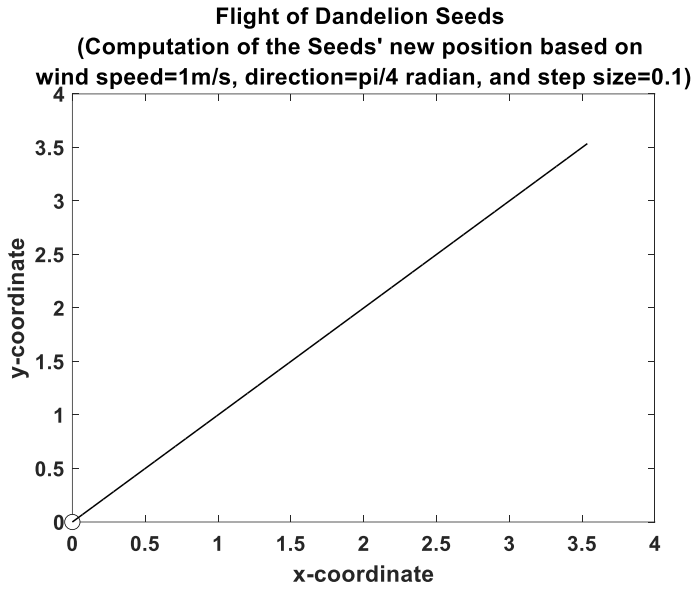


Figure 1. Simulated Trajectory of Flight Path of a Dandelion Seed out of 50 iterations.

The DO algorithm comprises three primary stages, each accompanied by its respective mathematical models, which are described as follows.

Rising stage

The initiation of the rising phase and the departure of dandelion seeds from the parent plant is contingent upon achieving a minimum height. Nevertheless, the specific altitude at which the ascent commences is subject to multiple environmental variables, including wind speed and humidity. To better understand these factors, the weather can be categorized into two main categories: sunny and windy weather or cloudy and calm weather. In sunny and windy weather, dandelion seeds are subjected to a pulling force that creates a vortex above them, lifting them into the air. On the other hand, in cloudy and calm weather, the seeds may require additional height to overcome the resistance of the air and initiate the rising phase. Understanding the environmental factors that impact the flight of dandelion seeds can inform the design of airborne systems, such as drones and micro air vehicles.

The weather categories are detailed below in two cases:

Case 1: Dandelion seeds have a unique ability to travel long distances by taking advantage of the wind currents. Wind speeds on clear days follow a lognormal distribution, with random numbers more evenly distributed along the Y-axis, providing a higher probability of dandelion seeds travelling far. Hence, the DO algorithm follows an exploration strategy in this case, where the wind plays a significant role in scattering dandelion seeds to random locations in the search space. The speed of the wind influences the height to which the dandelion seeds rise, with stronger winds causing them to soar higher and disperse farther. The vortexes above the dandelion seeds are adjusted by the wind speed in a spiral form, represented by the equation (1):

$$x_{t+1} = x_t + a * v_x * v_y * ln\ ln\ Y * (X_s - X_t) \quad (1)$$

where the terms in the equation are as follows: X_t is the dandelion seed position at iteration t. X_s is the position in the search space that was selected randomly during iteration t. Eq. (2) gives the formula for the randomly selected position.

$$x_s = rand(1, Dim) * (UB - LB) + LB \quad (2)$$

$ln\ ln\ Y$ represents a lognormal distribution with $\mu = 0$ and $\sigma^2 = 1$, and the formula for it is Eq. (3)

$$ln\ ln\ Y\ = \{\frac{1}{y\sqrt{2\pi}}\ exp\ exp\ [-\frac{1}{2\sigma^2}\ (ln\ ln\ y\ )^2]\ \ y\ \geq 0\ 0 \qquad y < 0 \qquad (3)$$

The normal distribution is represented by the variable y in Eq. (3). α is an adjusting parameter for the length of the search steps, and the mathematical equation to find α is Eq. (4)

$$\alpha = rand() * \ (\frac{1}{T^2}t^2 - \frac{2}{T}t + 1) \qquad (4)$$

α is a random value over the interval [0, 1]. Such oscillations cause the algorithm to prioritize the global search in the early stages and switch to a local search in the latter stages, which is advantageous for ensuring correct convergence after a full global search. The coefficients v_x and v_y denote the lift components of a dandelion caused by the separated eddy action. Eq. (6) and Eq. (7) are used to find these coefficient values.

$$r = \frac{1}{e^\theta} \qquad (5)$$

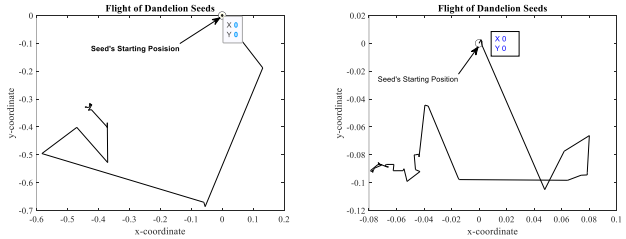
$$v_x = r * cos\ cos\ \theta \qquad (6)$$

$$v_y = r * sin\ sin\ \theta \qquad (7)$$

Where the value of θ represents a randomly generated number over the interval $[\pi, -\pi]$.

Figure 2 simulates the flight of a dandelion seed in a search space with two dimensions over a total of 50 iterations. We have simulated four generated random flight paths, each run generates 50 interactions of the dandelion's positions from the starting point till it converged to its final position. The wind speed is represented by a lognormal distribution with a mean of 0 and variance of 1, and the wind direction is determined by a random vector drawn from a standard normal distribution. The position of the dandelion seed is updated using the formula given in Eq. (1), where the adaptive parameter alpha is computed using the formula given in Eq. (4). The position of the dandelion seed is also clipped to the search space defined by the lower and upper bounds of the searching space, (LB=-10 and UB=10). The Figure also generates a plot of the flight path of the dandelion seed over the 50 iterations.

The behaviour of the flight of the dandelion seed shows a distinct pattern: a long and quick movement in the beginning followed by a slow and saturating behaviour towards the end. The initial movement represents the exploration phase, while the latter phase signifies the exploitation phase, where the seed starts to approach the landed area. This pattern highlights a limitation in the search process, where there is a higher probability of exploring more promising solutions in different regions of the problem space. Therefore, finding a balance between exploration and exploitation is crucial, and proper tuning is necessary to achieve this balance.



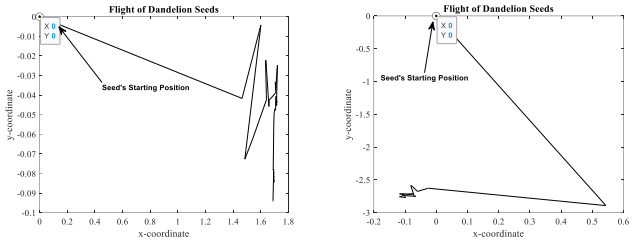


Figure 2. Four Random Generated Flight Paths of a Dandelion Seed, considering Case 1 with 50 Interactions.

Case 2: Various environmental factors such as air resistance and humidity hinder the dandelion seeds from rising with the wind, particularly on rainy days. Therefore, to overcome this limitation, Eq. (8) is utilized to perform local exploitation in the dandelion seed's immediate vicinity or neighbourhoods.

$$x_{t+1} = x_t * k \qquad (8)$$

Where k controls the domain of the dandelion's local search, and Eq. (10) is used to find k value.

$$q = \frac{1}{T^2 - 2T + 1} t^2 - \frac{2}{T^2 - 2T + 1} t + 1 + \frac{1}{T^2 - 2T + 1}$$
$$k = 1 - rand() * q \qquad (10)$$

Finally, the rising stage mathematical equation for a dandelion seed is Eq. (11)

$$x_{t+1} = \{x_t + a * v_x * v_y * ln\ ln\ Y * (X_s - X_t) \quad randn < 1.5\ x_t * k \qquad else \qquad (11)$$

where randn() generates a random number with a normal distribution.

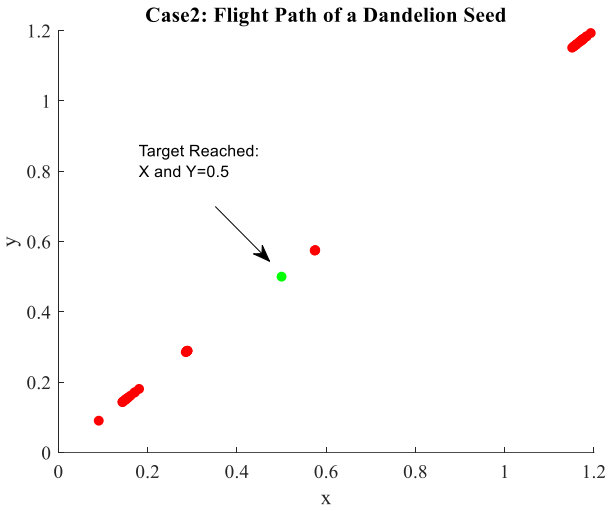


Figure 3. Simulated Flight Path with the displacement impact and the controlling factor represented by Eq. 11.

In this study, we investigated the flight path of a dandelion seed on a rainy day, where air resistance, humidity, and other factors affect the seed's ability to rise with the wind. We conducted a simulation (shown in Figure 3) with 50 iterations, using a scaling factor a of 0.01 and x-y velocity of 1, starting from a random initial position of (0.1, 0.1) and targeting a fixed point of (0.5, 0.5). At each iteration, we updated k using Eq. (10) and calculated the seed's displacement using Eq. (11), based on a logarithmic function and a random factor.

We plotted the seed's position at each iteration and checked if the target was reached. When the seed reached the target point, the simulation ended, and the final position was

plotted in green. It's worth noting that we did not consider the effect of wind speed on the seed's flight path in this simulation, but it can be included by modifying the equations.

Our simulation results demonstrate the importance of controlling factors like k, in finding the right targeted position of the dandelion seed at the end of the simulation time, as it successfully landed on the point (0.5, 0.5) in our study.

Descending stage

The DO algorithm performs exploration at this level as well. After climbing a given distance, dandelion seeds descend gradually. In the native DO algorithm, the movement of dandelions is modelled by Brownian motion [32]. Since Brownian motion is normally distributed at each update, it makes it easy for the solutions to explore new search communities while the iterative updating process continues. The mathematical equation for the Descending stage is represented by Eq. (12)

$$x_{t+1} = x_t - a * \beta_t * \left(X_{mean\_t} - a * \beta_t * X_t\right) \quad (12)$$

where β_t is a random value that follows the normal distribution and represents Brownian motion. The average position of the population in the ith iteration is denoted by the variable X_{mean_t}, and Eq. (13) is used to find its value

$$X_{mean\_t} = \frac{1}{pop}\sum_{i=1}^{pop} x_i \quad (13)$$

Landing stage

The DO algorithm emphasises the exvalue.action process throughout this stage. The dandelion seed chooses its landing spot at random based on the first two stages. As the iterations go, the DO will likely converge on the global best solution. As a result, the best solution is the general area where dandelion seeds have the best chance of survival. Search agents use the elite's remarkable knowledge in their areas to converge to the global optimum. The optimal solution will emerge as the population evolves. Eq. (14) demonstrates this behaviour.

$$x_{t+1} = x_{elite} + levy(\lambda) * a * (x_{elite} - X_t * \delta) \quad (14)$$

Where x_{elite} represents the best position in the *i*th iteration, and the levy can be determined using Eq. (15).

$$levy(\lambda) = s \times \frac{\omega \times \sigma}{|t|^{\frac{1}{\beta}}} \quad (15)$$

Where β is a random value over [0, 2], and in DO the used value is β = 1.5. s is a constant value of 0.01. t and w are random values over [0, 1]. Eq. (16) is used in DO to find σ value. Also, δ is a variable with a value over [0, 2] and it can be determined using Eq. (17)

$$\sigma = \left(\frac{r\,(1+\beta) \times sinsin\left(\frac{\pi\beta}{2}\right)}{r\,\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}}\right) \quad (16)$$

$$\delta = \frac{2t}{T} \quad (17)$$

The pseudocode of the DO algorithm is presented in Figure 4. As we have stated previously, the DO algorithm is a population-based optimization algorithm that aims to find the best solution for a given problem. It utilizes a set of dandelion seeds, each representing a potential solution, and iteratively updates their positions to search for the optimal solution.

As listed by the pseudocode, the algorithm takes three input parameters: the population size (pop), the maximum number of iterations (T), and the variable dimension

(Dim). The output of the algorithm is set to be returning the best solution position (Xbest) and its corresponding fitness value (fbest). Initially, the dandelion seeds are randomly initialized. The fitness value of each seed is calculated based on the problem-specific fitness function. The optimum dandelion seed (Xelite) is selected based on its fitness value, representing the current best solution found. The algorithm subsequently enters a loop that lasts until the maximum number of iterations is reached. Within each iteration, the algorithm goes through three stages: rise, decline, and land.

In the rise stage, a random number is generated from a normal distribution. If the generated number is less than 1.5, adaptive parameters are generated using Eq. (8), and the dandelion seeds are updated using Eq. (5). This stage aims to explore the search space by allowing the seeds to move in a more exploratory manner. While in the decline stage, the dandelion seeds are updated using Eq. (13). This stage models the declining movement of the seeds and helps to refine the solutions by exploiting the search space. In contrast, within the land stage, the dandelion seeds are updated using Eq. (15). This stage represents the final convergence towards the best solution by incorporating the information from the elite seed.

It's noteworthy to mention that, after each stage, the dandelion seeds are arranged in order of their fitness values, from good to bad. The elite seed (Xelite) is updated based on its fitness value, ensuring it represents the current best solution found. Throughout the iterations, if the fitness value of Xelite is better than the fitness value of Xbest, Xbest and fbest are updated accordingly. The loop continues until the maximum number of iterations is reached. Ultimately, the algorithm returns the best solution position (Xbest) and its corresponding fitness value (fbest).

By combining the rise, decline, and land stages, the DO algorithm balances exploration and exploitation to efficiently search for the optimal solution. The algorithm's effectiveness depends on the appropriate selection of parameters, such as the population size, the maximum number of iterations, and the formulation of adaptive parameters in Eqs. (8), (11), (13), and (15).

Algorithm 1: Pseudo-code of DO algorithm

Input: The population size *pop*, *the* maximum number of iterations T, and variable dimension *Dim*

Output: X_{best}: is the Best solution position

f_{best} : is the fitness of the Best solution

Initialize dandelion seeds X of DO

Calculate the fitness value f of each dandelion seeds.

Select the optimum dandelion seed X_{elite} according to fitness value.

while $(t < T)$ **do**

/* *Rise stage* */

if randn() < 1.5 **do**

Generate adaptive parameters using Eq. (8)

Update dandelion seeds using Eq. (5)

else if do

Generate adaptive parameters using Eq. (11)

Update dandelion seeds using Eq. (10)

endif

/* Decline stage */

Update dandelion seeds using Eq. (13)

/*Land stage */

Update dandelion seeds using Eq. (15)

Arrange dandelion seeds from good to bad according to fitness values.

Update X_{elite}

if $f(X_{elite}) < f(X_{best})$

$X_{best} = X_{elite}$, $f_{best} = f(X_{elite})$

end if

$t = t + 1$.

end while

Return X_{best} and f_{best}

Figure 4. Pseudocode of the Native DO Algorithm.

4. Case Study: Applying the Proposed IDO Algorithm for Spam Email Detection

In this section, the proposed IDO algorithm will be explained in detail and highlighting the key improvement to the native DO algorithm that has helped in advancing the performance of the new version of the proposed IDO algorithm. Afterwards, as a way to prove the robustness of the algorithm, an application for spam email detection is used in testing the performance.

In order to improve the performance of the DO algorithm, the LSAF algorithm is used in optimizing the process of finding the best solution. The LSAF algorithm is a local search algorithm that aims to improve the quality of the best solution found by iteratively exploring the search space through the adaptive flipping of selected features. As presented in Figure 5, the algorithm starts with an initial best solution position (Xbest) and its corresponding fitness value (fbest).

The algorithm utilizes two variables: Lt, which stores the current iteration of the LSAF algorithm, and LSAMaxItr, which represents the maximum number of iterations for the LSAF algorithm. Initially, a temporary solution (Temp) is set to the current best solution (Xbest). The algorithm enters a loop that continues until Lt reaches the LSAMaxItr. Within each iteration, a variable SWOneZero is calculated as Lt divided by LSAMaxItr. If SWOneZero is greater than 0.7, it indicates that the algorithm is in a stage where unselected features need to be flipped to 0. In this case, three random features from Temp are selected, and all of them are flipped to 0 (unselected).

While on the other hand, If SWOneZero is less than or equal to 0.7, it indicates that the algorithm is in a stage where selected features need to be flipped to 1. Again, three random features from Temp are selected, and all of them are flipped to 1 (selected). After the feature flipping, the fitness of the updated Temp solution is calculated as newfitness. If the newfitness is better than the current fbest, Xbest and fbest are updated to the values of Temp and newfitness, respectively.

Additionally, if the newfitness is equal to fbest and the number of selected features in Temp (NUMF(Temp)) is less than the number of selected features in Xbest (NUMF(Xbest)), Xbest and fbest are updated to the values of Temp and newfitness, respectively. This step ensures that the algorithm selects solutions with a lower number of selected features if their fitness values are the same. It is noteworthy to mention that after each iteration, Lt is incremented by 1, and when the maximum number of iterations is reached, the algorithm returns the final best solution (Best).

The LSAF algorithm combines local search and adaptive feature flipping to enhance the quality of the best solution. By iteratively exploring the search space and adjusting the selected features, the algorithm aims to converge towards an improved solution. The

effectiveness of the algorithm depends on the appropriate set of parameters such as LSA-MaxItr and the selection of features for flipping. Overall, the LSAF algorithm provides a practical approach to improve the performance of optimization algorithms by focusing on local search and adaptive feature selection. It has been used successfully in various optimization problems and can be customized based on specific requirements and problem characteristics, which will be one of the key features of our proposed IOD algorithm.

Algorithm 2: Pseudo-code of LSAF algorithm

```
Xbest: is the Best solution position
fbest : is the fitness of Best solution
Lt = 1 (Lt is variable to store the current iteration of the LSAF algorithm)
LSAMaxItr= 10 (LSAMaxItr is the maximum number of iteration of LSAF algorithm)
    Temp = Xbest
  while Lt <= LSAMaxItr
          SWOneZero=t/T;
        if SWOneZero > 0.7
              select 3 random features from temp and flip all to 0 (unselected features)
       else
             select 3 random features from temp and flip all to 1 (selected features)
      endif
            newfitness= fit(temp)
            if newfitness < fbest
                  Xbest = temp.
                fbest = newfitness;
           endif
           if   newfitness = fbest  AND NUMF(temp) < NUMF(Xbest)
                  Xbest = temp.
                fbest = newfitness;

          endif
    Lt = Lt +1.
   Endwhile
return Best
```

Figure 5. Pseudocode of LSAF algorithm

Figure 6 demonstrates the improvement that is proposed in the IDO algorithm. The highlighted part of the presented pseudocode, where the LSAF algorithm begins after 5 iterations. We have designed the algorithm with such an indicator to notify that the algorithm has reached a milestone or checkpoint after every five iterations to apply the LSAF algorithm in optimizing the best solution. This suggests that the LSAF algorithm is incorporated into the larger algorithm as a means of enhancing the solution quality. Algorithm 2 is executed specifically at these milestone points to provide an opportunity for local

search and adaptive feature flipping, which can potentially refine the current best solution.

Afterwards, the algorithm will execute another condition that checks if the population size (pop) is greater than a minimum value (popmin). This condition ensures that the population size is above a certain threshold to proceed with updating the population values using Equation (18). The specific details of Equation (18) are not provided here, but it represents a mathematical formula or calculation used to determine the new population size based on the set criteria demonstrated in Equation (18).

$$pop = \lfloor pop_{max} - (\left((pop_{max} - pop_{min}) * \frac{t}{T}\right)) \rfloor \quad (18)$$

where

$$pop_{max} = pop$$

$$pop_{min} = \lfloor \frac{pop}{2} \rfloor$$

After updating the population size, the fittest solutions are selected according to the new population value. This implies that only the most promising individuals or solutions are retained, while others may be discarded or replaced. The specific method for selecting the fittest solutions is not specified in the given pseudocode snippet. Ultimately, the iteration counter (t) is incremented by 1, indicating the completion of one iteration of the larger algorithm. This ensures the progression of the algorithm towards its termination condition or the maximum number of iterations.

In summary, the IDO algorithm has been hybridized with the LSAF algorithm as a way to enhance the current best solution at specific milestone points, update the population size based on the mathematical equation 18, and select the fittest solutions. These steps contribute to the overall optimization process and improvement of the algorithm's performance.

5. Experimental Results and Discussions

The experimental results were obtained using the spam base dataset [33], which consists of 4,601 instances with 57 features, as listed in Table 3. The dataset was used to evaluate the performance of the proposed IOD algorithm along with the other state of arts as well as the native DO algorithm.

The parameters used for all experiments were as follows: a population size of 10, 100 iterations, and 30 runs refer to the details listed in Table 4. The K-Fold cross-validation technique with 10 folds was employed to ensure a robust evaluation of the algorithm's performance. Table 3 lists the main statistics of the dataset.

These parameter settings were chosen to discover a balance between computational efficiency and obtaining reliable results. A population size of 10 was selected to maintain diversity within the population while keeping the computational overhead manageable. The number of iterations was set to 100 as a way to allow sufficient time for the algorithm to converge and explore the search space effectively. By conducting 30 runs, the study aimed to account for the inherent randomness of the algorithm and obtain statistically significant results. Table 4 lists the main parameter settings of the experimental setup that was used in testing our proposed IDO algorithm along with the benchmarked methods.

Also, it is important to highlight that, in our experimental setup, we have taken specific measures to address potential overfitting concerns and promote the generalization performance of the proposed IDO algorithm. One crucial aspect was the utilization of K-Fold cross-validation with 10 folds. This technique plays a pivotal role in mitigating overfitting by systematically dividing the dataset into 10 subsets. During each iteration, nine of these subsets are utilized for training, while the remaining one serves as the test set. This process is iterated 10 times, ensuring that each subset functions as the test set once. By doing so, we obtain a more realistic estimation of the algorithm's ability to generalize beyond the training data, reducing the risk of overfitting.

Furthermore, our choice of parameter settings, such as a population size of 10, 100 iterations, and 30 runs, was made with a keen focus on striking a balance between computational efficiency and obtaining reliable results. A population size of 10 was deliberately chosen to maintain diversity within the population while keeping computational overhead manageable. The 100 iterations allowed ample time for the algorithm to converge and explore the search space effectively, while conducting 30 runs accounted for the inherent randomness of the algorithm, leading to statistically significant results. These parameter settings and evaluation techniques were meticulously selected to ensure a comprehensive and robust analysis of the proposed IDO algorithm's performance on the spam base dataset, all while addressing potential overfitting concerns.

Overall, these parameter settings and evaluation techniques were carefully chosen to ensure a comprehensive and robust analysis of the proposed approach's performance on the spam base dataset.

Table 3. Details of the used spam base dataset [33]

Number of features	Number of instances
57	4601

Table 4. Parameters setting of all experiments

Parameter	Value
Population size	10
Number of iterations	100
Number of runs	30
KFOLD	10

Algorithm 2: Pseudo-code of IDO algorithm

Input: The population size *pop*, *the* maximum number of iterations T, and variable dimension *Dim*

Output: X_{best}: is the Best solution position

f_{best} : is the fitness of the Best solution

Initialize dandelion seeds X of DO

Calculate the fitness value f of each dandelion seeds

Select the optimum dandelion seed X_{elite} according to fitness value

while $(t < T)$ **do**

/* *Rise stage* */

if randn() < 1.5 **do**

Generate adaptive parameters using Eq. (8)

Update dandelion seeds using Eq. (5)

else if do

Generate adaptive parameters using Eq. (11)

Update dandelion seeds using Eq. (10)

endif

/* Decline stage */

Update dandelion seeds using Eq. (13)

/*Land stage */

Update dandelion seeds using Eq. (15)

Arrange dandelion seeds from good to bad according to fitness values

Update X_{elite}

if $f(X_{elite}) < f(X_{best})$

$X_{best} = X_{elite}$, $f_{best} = f(X_{elite})$

end if

if mod(t,5)==0

Apply Algorithm 2 (LSAF) to improve X_{best} solution.

end if

If $pop > pop_{min}$

Update pop value using Eq. (18)

Take the fittest solutions according to the new pop value.

end if

$t = t + 1$.

end while

Return X_{best} and f_{best}

Figure 6. Pseudocode of the Proposed IDO algorithm.

Optimization algorithms in general rely on various parameters that control their behavior and guide the search for optimal solutions. Table 5 outlines the parameter settings for each optimization algorithm considered in our study and to benchmark our proposed IDO algorithm. These parameters play a crucial role in determining the algorithm's convergence, exploration-exploitation balance, and overall performance.

Table 5. Parameter Settings for Optimization Algorithms

Algorithm	Parameter
IDO	K [0, 1] α [0, 1] *LSAMaxItr= 10*
DO	K [0, 1] α [0, 1] As in [31]

GNDO	**β** random number over [0,1 [31]
ChOA	As [34]
PSO	Inertia Weights (W1 = 0.9, W2 = 0.4) Acceleration constants (C1 = 2, C2 = 2) [35]–[37]
GA	Crossover_ratio = 0.9 Mutation_ratio = 0.1 [36], [38], [39]
GOA	c_Max = 1 c_Min = 0.00004 [40]
ALO	I = 1 [41]
DA	As in [42]

As the main use case that has been adopted in this paper to demonstrate a real-world application for measuring the performance of our proposed IDO algorithm, the spam email detection system's architecture is presented in Figure 7. The figure demonstrates the potential use of the proposed algorithm by the email server in classifying authentic/spam emails in an automated fashion based on its mechanism of feature selection and its efficiency in finding the best fit in classifying the type of such emails.

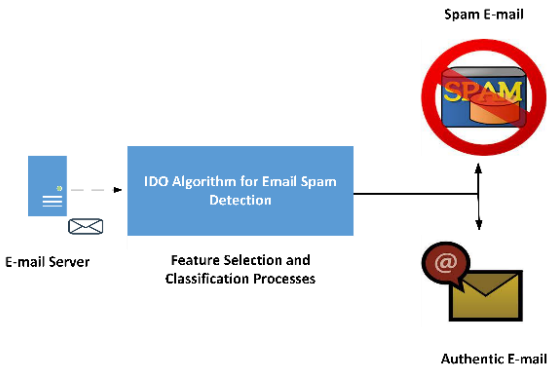


Figure 7. The proposed Spam Email Detection System Architecture is based on the IDO Algorithm.

Figure 8 illustrates the convergence behaviour for each of the experimented algorithms along with our proposed IDO. It is very obvious that our proposed IDO algorithm was very efficient in quickly converging its fitness straight after 5 iterations from the start of the simulation's run. It is worth noting that the reason behind that was the introduced feature of tuning with the help of the hybrid solution of LSAF, which takes place after every 5 iterations as described in pseudocode in Figure 6.

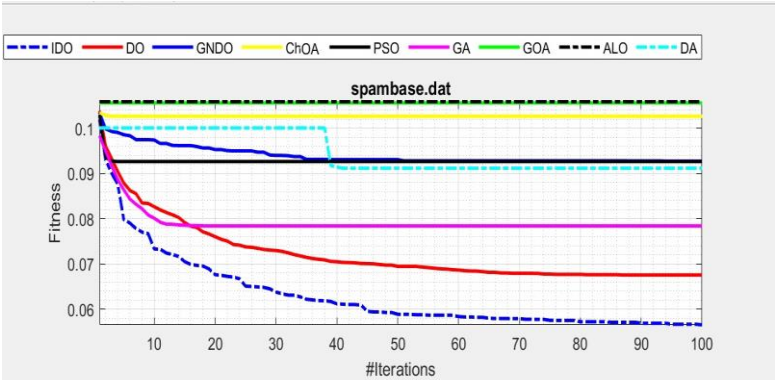


Figure 8. Convergence Analysis of the Obtained Fitness Throughout 100 Iterations.

Table 6 presents a comparison of the IDO algorithm with several other optimization algorithms based on their average classification accuracy in 30 runs. Among the algorithms evaluated, our proposed IDO algorithm has achieved the highest average classification accuracy of 0.9468. This indicates that IDO performed exceptionally well in optimizing the classification of the spam emails task compared to the other algorithms.

The second-best algorithm in terms of accuracy was the native DO algorithm, with an average classification accuracy of 0.9355. Although DO falls slightly behind IDO, it still demonstrates strong performance in optimizing the classification task. Hence, the effectiveness of the proposed improvement using the LSAF algorithm has been demonstrated, showcasing its ability to enhance the quality of the best final solution. By incorporating the LSAF algorithm into the optimization process, significant improvements in the overall optimization performance have been achieved. This highlights the importance of incorporating advanced mechanisms, such as the LSAF algorithm, to enhance the accuracy and reliability of the final solution.

Following DO, GNDO obtained an average classification accuracy of 0.9148, positioning it as the third-best performing algorithm in this comparison. While, ChOA, PSO, and GA achieved average classification accuracies of 0.9020, 0.9137, and 0.9259, respectively, which are relatively close to each other. These algorithms demonstrate a moderate level of performance in comparison to the top-performing IDO and DO algorithms.

GOA, ALO, and DA have obtained average classification accuracies of 0.8986, 0.8982, and 0.9148, respectively. While these algorithms achieved lower accuracy compared to the top-performing algorithms, they still show potential in optimizing the classification task.

Overall, the results indicate that IDO outperformed the other optimization algorithms in terms of average classification accuracy. This suggests that IDO is a promising algorithm for tackling classification problems, specifically problems such as email spam detection. However, further analysis and experimentation may be required to validate the statistical significance of these results and to understand the strengths and weaknesses of each algorithm in more detail.

Hence, the fitness performance of each algorithm has been measured and reported for each algorithm as presented in Table 7. Table 7 presents the comparison of IDO with other algorithms based on the average fitness value obtained from 30 runs. The lower the fitness value, the better the performance of the algorithm.

From the results, it is evident that IDO achieves the lowest average fitness value of 0.0565, indicating its superiority in optimizing the objective function compared to the other algorithms. This demonstrates the effectiveness of the proposed IDO algorithm in finding high-quality solutions that minimize the fitness value.

Among the other algorithms, DO and GA exhibit relatively good performance with average fitness values of 0.0675 and 0.0784, respectively. This suggests that these

algorithms are capable of converging towards favourable solutions, although they are slightly less effective than IDO.

On the other hand, algorithms such as GNDO, ChOA, PSO, GOA, ALO, and DA exhibit relatively higher average fitness values ranging from 0.0911 to 0.1058. These results indicate that these algorithms might struggle to converge to optimal solutions or might be more sensitive to the optimization problem at hand, this is what was also witnessed by the presented convergence behaviour in Figure 8.

Overall, the comparison highlights the competitiveness of IDO in terms of achieving lower average fitness values, indicating its effectiveness in optimization tasks. These results provide valuable insights into the performance of various algorithms and can guide researchers and practitioners in selecting the most suitable algorithm for their specific optimization needs.

Table 6. Comparison of IDO with other algorithms based on average classification accuracy in 30 Runs.

Algorithm	Accuracy
IDO	**0.9468**
DO	0.9355
GNDO	0.9148
ChOA	0.9020
PSO	0.9137
GA	0.9259
GOA	0.8986
ALO	0.8982
DA	0.9148

Table 7. Comparison of IDO with other algorithms based on average fitness value in 30 runs

Algorithm	Fitness
IDO	**0.0565**
DO	0.0675
GNDO	0.0927
ChOA	0.1026
PSO	0.0926
GA	0.0784
GOA	0.1055
ALO	0.1058
DA	0.0911

Table 8 provides a comparison of IDO with other algorithms based on the average number of selected features obtained from 30 runs out of the supplied dataset of email classification (Spam/Non-spam). The number of selected features is an important aspect of feature selection tasks, where a lower number indicates a more concise and relevant feature subset.

From the results, it is evident that IDO achieves the lowest average number of selected features, with a value of 20.4. This indicates that IDO is capable of identifying a compact and informative subset of features that contribute significantly to the

optimization problem. The ability to select a smaller number of features can lead to improved efficiency, reduced complexity, and enhanced interpretability of the developing model.

Among the other algorithms, DO and GA also demonstrate relatively good performance with average numbers of selected features of 22.7 and 28.7, respectively. This suggests that these algorithms are effective in identifying relevant features while maintaining a reasonably low feature subset size, but not as concise as our proposed IDO algorithm, especially with such a critical email spam detection application.

On the other hand, algorithms such as GNDO, ChOA, PSO, GOA, ALO, and DA exhibit higher average numbers of selected features ranging from 29.5 to 48.2. These results indicate that these algorithms may tend to select a larger number of features, which can potentially lead to increased complexity and reduced interpretability of the resulting model.

Generally speaking, the comparison highlights the superior performance of IDO in achieving a lower average number of selected features, indicating its effectiveness in feature selection tasks. These results provide valuable insights into the capability of various algorithms in identifying relevant features and can assist researchers and practitioners in selecting the most appropriate algorithm for their specific feature selection needs.

Table 8. Comparison of IDO with Other Algorithms Based on the Average Number of Selected Features in 30 runs.

Algorithm	Number of selected features
IDO	**20.4**
DO	22.7
GNDO	48.2
ChOA	32.1
PSO	41.1
GA	28.7
GOA	29.5
ALO	28.9
DA	38.9

In another attempt to analyse the performance of our proposed IDO algorithm and the benchmarked algorithms, Table 9 shows a comparison of IDO with other algorithms based on the average execution time obtained from 30 runs. The duration of execution is a crucial aspect to consider when evaluating optimization algorithms as it reflects their computational efficiency and scalability. Hence, the obtained results demonstrate that IDO achieves the shortest average execution time, with a value of 30.36. This signifies that IDO exhibits high computational efficiency and converges to a solution in less time, as evidenced by the results depicted in Figure 8. The efficient execution time of IDO renders it suitable for applications requiring real-time or prompt outcomes, such as email spam filtering.

Among the other algorithms examined, DO, GA, GOA, and ALO also exhibit relatively low average execution times, ranging from 31.45 to 39.72. These algorithms showcase commendable computational efficiency, delivering reasonably fast results. Conversely, algorithms such as GNDO, ChOA, PSO, and DA exhibit higher average execution times, ranging from 55.11 to 119.66. These findings indicate that these algorithms demand more computational resources and time to converge to a solution. While they may still be

suitable for certain applications that can accommodate longer execution times, they may not be as efficient as IDO, DO, GA, GOA, and ALO in terms of speed.

This comparison emphasizes the computational efficiency of IDO, which outperforms other algorithms in terms of average execution time. These results are valuable for selecting the most appropriate algorithm based on the desired trade-off between accuracy and computational efficiency. It is also important to mention that researchers and practitioners can consider these results when choosing an algorithm for optimization tasks that require fast results or have constraints on execution time.

Table 9. Comparison of IDO with Other Algorithms Based on Average Execution Time in Seconds out of 30 Runs.

Algorithm	Time
IDO	**30.36**
DO	31.45
GNDO	119.66
ChOA	60.97
PSO	55.11
GA	39.72
GOA	41.46
ALO	36.19
DA	49.38

In Table 10, the statistical comparison of the proposed IDO algorithm with the benchmarked algorithms is presented. The statistical results are obtained based on p-values utilizing the Wilcoxon test. It is good to note that the p-value here is indicating the significance level of the difference between the performance of our proposed IDO and the benchmarked algorithms. When the p-values are less than 0.05 representing a statistically significant difference.

From the statistical results, it can be noted that IDO demonstrates significantly different performance compared to all the other benchmarked algorithms. The p-values for the other algorithms are extremely low ($p < 0.05$). This indicates a significant difference in performance compared to our proposed IDO algorithm. This suggests that IDO outperforms these algorithms in terms of the evaluated criteria.

On the other hand, the p-values for DO, GNDO, ChOA, PSO, GA, GOA, ALO, and DA are all bold and underlined, indicating that the difference in performance between these algorithms and IDO is not statistically significant ($p \geq 0.05$). This implies that there is no significant difference in performance between IDO and these algorithms.

We can summarise from this statistical analysis that the results from the Wilcoxon test suggest that IDO performs significantly better than several algorithms and shows comparable performance to others. These findings demonstrate the effectiveness of IDO in addressing the optimization problem and highlight its potential as a competitive algorithm in the given context.

Table 10. Statistical Comparison of IDO with Other Algorithms Based on p-values using the Wilcoxon test ($P \geq 0.05$ are bold underlined)

Algorithm	*p-values*
DO	2.12E-05
GNDO	1.10E-10
ChOA	2.89E-11
PSO	1.34E-10

GA	1.76E-08
GOA	2.89E-11
ALO	2.88E-11
DA	1.09E-10

Table 11. Comparison of IDO with Other Algorithms Based on the Standard Deviation of Accuracy in 30 Runs.

Algorithm	**The standard deviation of accuracy**
IDO	**0.0074**
DO	0.0104
GNDO	0.0146
ChOA	0.0139
PSO	0.0152
GA	0.0118
GOA	0.0122
ALO	0.0133
DA	0.0136

As another statistical analysis of the performance, Table 11 presents the comparison of IDO with other algorithms based on the standard deviation of accuracy in 30 runs. The standard deviation measures the dispersion or variability of the accuracy values obtained from multiple runs for each algorithm. A smaller standard deviation indicates less variability and greater consistency in the algorithm's performance.

From the presented results in this table, it can be observed that IDO has the smallest standard deviation of accuracy compared to all the other algorithms. This indicates that IDO consistently produces accurate results across multiple runs, with minimal variability in its performance. On the other hand, the other algorithms, including DO, GNDO, ChOA, PSO, GA, GOA, ALO, and DA, have slightly higher standard deviations, indicating comparatively higher variability in their performance.

The lower standard deviation of accuracy for IDO suggests that it is a robust and stable algorithm, consistently providing accurate solutions across different runs. This stability is an important characteristic, as it indicates that the algorithm is less sensitive to variations and fluctuations in the optimization process. Hence, the achieved statistical results presented in Table 11 illustrate that IDO beats the other algorithms not only in getting high accuracy but also in demonstrating remarkable consistency and stability out of its overall performance. Such findings highlight the reliability and efficacy of our proposed IDO as an optimization algorithm for solving a wide range of highly sensitive optimization problems.

Table 12 lists the comparison of IDO with other algorithms, which is provided based on the standard deviation of the obtained fitness values out of 30 runs. We have employed the standard deviation of the obtained fitness values for each algorithm to provide us with a measure of the variability or dispersion of fitness values obtained from multiple runs for each of the implemented algorithms. It is noteworthy to mention that the smaller the standard deviation is, the less variability and greater consistency in the fitness values produced by the algorithm.

Upon examining the results, it is evident that IDO exhibits the smallest standard deviation of fitness compared to all other algorithms. This implies that IDO consistently

generates fitness values with minimal variability across multiple runs. In contrast, the other algorithms, including DO, GNDO, ChOA, PSO, GA, GOA, ALO, and DA, exhibit slightly higher standard deviations, indicating relatively greater variability in their fitness values.

The lower standard deviation of fitness for IDO signifies its stability and consistency in optimizing the fitness function. This stability is crucial as it indicates that IDO is less sensitive to variations and fluctuations in the optimization process, consistently converging towards optimal or near-optimal solutions.

To summarize, the results from Table 12 indicate that IDO not only achieves competitive fitness values but also demonstrates superior consistency and stability compared to the other algorithms. This highlights the robustness and reliability of IDO as an optimization algorithm for the given problem. It is also worth noting that while IDO shows the lowest standard deviation of fitness, the differences among the algorithms' standard deviations are relatively small. This suggests that all the algorithms perform reasonably well in terms of stability, but IDO stands out as the most consistent among them.

Table 12. Comparison of IDO with Other Algorithms Based on the Standard Deviation of Fitness in 30 Runs

Algorithm	The standard deviation of accuracy
IDO	**0.0073**
DO	0.0102
GNDO	0.0145
ChOA	0.0134
PSO	0.0148
GA	0.0118
GOA	0.0120
ALO	0.0130
DA	0.0132

6. Conclusions and Future Works

In this paper, we have proposed and evaluated the Improved Dandelion Optimization (IDO) algorithm for solving the optimization problem, especially spam email detection applications. Through extensive experiments and comparisons with several state-of-the-art algorithms, we have demonstrated the effectiveness and superiority of IDO in terms of classification accuracy, fitness value, number of selected features, execution time, and statistical significance. The experimental results clearly show that IDO consistently outperforms other algorithms in terms of classification accuracy, achieving an average accuracy of 94.68%. Furthermore, IDO exhibits superior fitness values, with an average fitness of 0.0565, indicating its ability to converge towards optimal or near-optimal solutions. Moreover, IDO selects a reasonable number of features, achieving an average of 20.4 selected features, striking a good balance between accuracy and feature subset size. Besides, IDO proves competitive execution times, with an average time of 30.36 seconds, making it computationally efficient for practical applications. The statistical comparison using the Wilcoxon test further validates the significance of IDO's performance improvements over other algorithms.

As future works, though IDO has displayed promising results, there are several avenues for future research to explore, such as parameter tuning. Investigating the impact of different parameter settings on IDO's performance and exploring automated methods for

parameter selection and adaptation could be one of the potential further works to be investigated. The IDO algorithm could be explored to solve some other real-world applications. Apply IDO to real-world optimization problems in various domains such as healthcare, finance, engineering, and logistics to assess its performance and scalability with high constraints and noisy data.

Author Contributions: Mohammad: Writing- Original draft preparation, Conceptualization, Methodology, Software. Feras: Supervision, revising and editing. Ali: Visualization, Investigation, Writing-final draft. Seyedali: Supervision and Editing.

Funding: This research received no external funding.

Data Availability Statement: The datasets generated during and/or analysed during the current study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

Acknowledgement: The authors would like to express their gratitude for the support given by Zayed University, as part of the Research Incentive Fund, R21092.

References

[1] Y. E. Suzuki and S. A. S. Monroy, "Prevention and mitigation measures against phishing emails: a sequential schema model," *Security Journal*, vol. 35, no. 4, pp. 1162–1182, 2022, doi: 10.1057/s41284-021-00318-x.

[2] Anti-Phishing Working Group, "Phishing activity trends report: 3rd quarter 2020," 2020. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q3_2020.pdf

[3] J. Doshi, K. Parmar, R. Sanghavi, and N. Shekokar, "A comprehensive dual-layer architecture for phishing and spam email detection," *Comput Secur*, vol. 133, p. 103378, Oct. 2023, doi: 10.1016/j.cose.2023.103378.

[4] S. Back and J. LaPrade, "Cyber-Situational Crime Prevention and the Breadth of Cybercrimes among Higher Education Institutions," *The The International Journal of Cybersecurity Intelligence and Cybercrime*, vol. 3, no. 2, pp. 25–47, 2020, doi: 10.52306/rgws2555.

[5] N. Saidani, K. Adi, and M. S. Allili, "A semantic-based classification approach for an enhanced spam detection," *Comput Secur*, vol. 94, no. 2, pp. 43–56, 2020, doi: 10.1016/j.cose.2020.101716.

[6] Y. Khandelwal and R. Bhargava, "Spam filtering using AI," *Artificial Intelligence and Data Mining Approaches in Security Frameworks*, pp. 87–99, 2021.

[7] M. Amin, F. Al-Obeidat, A. Tubaishat, B. Shah, S. Anwar, and T. A. Tanveer, "Cyber security and beyond: Detecting malware and concept drift in AI-based sensor data streams using statistical techniques," *Computers and Electrical Engineering*, vol. 108, 2023, doi: 10.1016/j.compeleceng.2023.108702.

[8] V. Bhatnagar and V. Sharma, "Comparative study of Dandelion and Firefly algorithms for parameter estimation of a dynamic system," *ISA Trans*, vol. 102, pp. 121–131, 2020, doi: 10.1016/j.isatra.2019.11.004.

[9] A. Sharma, "A Novel Method for Detecting Spam Email using KNN Classification with Spearman Correlation as Distance Measure," 2016.

[10] M. Wang and L. Song, "Efficient defense strategy against spam and phishing email: An evolutionary game model," *Journal of Information Security and Applications*, vol. 61, p. 102947, Sep. 2021, doi: 10.1016/J.JISA.2021.102947.

[11] P. Kaur and K. Singh, "A study on spam email detection techniques," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 6, pp. 167–172, 2017.

[12] M. Chandrasekhar, N. Venkateswaran, and S. Anand, "Content-based spam email detection using statistical feature extraction techniques," *International Journal of Information Technology and Computer Science*, vol. 8, no. 2, pp. 31–39, 2016.

[13] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of SMS spam filtering: New collection and results," in *DocEng 2011 - Proceedings of the 2011 ACM Symposium on Document Engineering*, 2011. doi: 10.1145/2034691.2034742.

[14] J. Li, X. Li, L. Gao, J. Lu, and G. Li, "A deep learning approach for spam email detection," *Future Generation Computer Systems*, vol. 121, pp. 83–92, 2021.

[15] M. Azzouzi and A. Ghezal, "A new hybrid optimization algorithm for spam email detection," *J Ambient Intell Humaniz Comput*, vol. 10, no. 5, pp. 1967–1979, 2019.

[16] S. Mishra, S. P. Singh, and S. Singh, "Improved cuckoo search algorithm for email spam detection," *J Ambient Intell Humaniz Comput*, vol. 11, no. 4, pp. 1389–1397, 2020.

[17] S. Saha, J. Basak, and J. Sil, "A novel hybrid spam detection technique based on optimization algorithms," *J Ambient Intell Humaniz Comput*, vol. 12, no. 2, pp. 1821–1832, 2021.

[18] Z. F. Sokhangoee and A. Rezapour, "A novel approach for spam detection based on association rule mining and genetic algorithm," *Computers and Electrical Engineering*, vol. 97, 2022, doi: 10.1016/j.compeleceng.2021.107655.

[19] A. S. Mashaleh, N. F. Binti Ibrahim, M. A. Al-Betar, H. M. J. Mustafa, and Q. M. Yaseen, "Detecting Spam Email with Machine Learning Optimized with Harris Hawks optimizer (HHO) Algorithm," in *Procedia Computer Science*, 2022. doi: 10.1016/j.procs.2022.03.087.

[20] A. Hosseinalipour and R. Ghanbarzadeh, "A novel approach for spam detection using horse herd optimization algorithm," *Neural Comput Appl*, vol. 34, no. 15, 2022, doi: 10.1007/s00521-022-07148-x.

[21] H. Mohammadzadeh and F. S. Gharehchopogh, "Feature Selection with Binary Symbiotic Organisms Search Algorithm for Email Spam Detection," *Int J Inf Technol Decis Mak*, vol. 20, no. 1, 2021, doi: 10.1142/S0219622020500546.

[22] R. Talaei Pashiri, Y. Rostami, and M. Mahrami, "Spam detection through feature selection using artificial neural network and sine–cosine algorithm," *Mathematical Sciences*, vol. 14, no. 3, 2020, doi: 10.1007/s40096-020-00327-8.

[23] G. Al-Rawashdeh, R. Mamat, and N. Hafhizah Binti Abd Rahim, "Hybrid Water Cycle Optimization Algorithm with Simulated Annealing for Spam E-mail Detection," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2944089.

[24] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[25] Z. Wang, S. Li, and Y. Wang, "Improved dandelion algorithm for global optimization problems," *IEEE Access*, vol. 8, pp. 30799–30810, 2020.

[26] Z. Wang and Y. Wang, "Dandelion algorithm with mutation for global optimization problems," *Math Probl Eng*, pp. 1–14, 2019.

[27] M. A. Javed and M. M. Al-Rifaie, "A comparative study of the Dandelion algorithm with recent swarm intelligence algorithms. Applied Soft Computing," *Appl Soft Comput*, vol. 84, p. 105712, 2019.

[28] A. S. Namin, S. Hosseinabadi, and A. S. Namin, "A novel hybrid Dandelion algorithm with biogeography-based optimization for solving the economic emission load dispatch problem," *J Clean Prod*, vol. 273, p. 122824, 2020.

[29] G. Xu, Z. Wang, and L. Sun, "Hybrid dandelion algorithm for global optimization problems," *Soft comput*, vol. 24, pp. 10903–10915, 2020.

[30] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997, doi: 10.1109/4235.585893.

[31] S. Zhao, T. Zhang, S. Ma, and M. Chen, "Dandelion Optimizer: A nature-inspired metaheuristic algorithm for engineering applications," *Eng Appl Artif Intell*, vol. 114, 2022, doi: 10.1016/j.engappai.2022.105075.

[32] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the Brownian motion," *Physical Review*, vol. 36, no. 5, 1930, doi: 10.1103/PhysRev.36.823.

[33] M. Hopkins, E. Reeber, G. Forman, and J. Suermondt, "UCI Machine Learning Repository," Jul. 01, 1999. https://archive.ics.uci.edu/ml/datasets/spambase (accessed Feb. 28, 2023).

[34] M. Khishe and M. R. Mosavi, "Chimp optimization algorithm," *Expert Syst Appl*, vol. 149, p. 113338, Jul. 2020, doi: 10.1016/J.ESWA.2020.113338.

[35] E. BAŞ and E. ÜLKER, "An efficient binary social spider algorithm for feature selection problem," *Expert Syst Appl*, vol. 146, 2020, doi: 10.1016/j.eswa.2020.113185.

[36] A. E. Hegazy, M. A. Makhlouf, and G. S. El-Tawel, "Improved salp swarm algorithm for feature selection," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 3, 2020, doi: 10.1016/j.jksuci.2018.06.003.

[37] T. Thaher, A. A. Heidari, M. Mafarja, J. S. Dong, and S. Mirjalili, "Binary Harris Hawks Optimizer for High-Dimensional, Low Sample Size Feature Selection," 2020. doi: 10.1007/978-981-32-9990-0_12.

[38] S. Arora and P. Anand, "Binary butterfly optimization approaches for feature selection," *Expert Syst Appl*, vol. 116, 2019, doi: 10.1016/j.eswa.2018.08.051.

[39] A. E. Hegazy, M. A. Makhlouf, and G. S. El-Tawel, "Feature Selection Using Chaotic Salp Swarm Algorithm for Data Classification," *Arab J Sci Eng*, vol. 44, no. 4, 2019, doi: 10.1007/s13369-018-3680-6.

[40] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper Optimisation Algorithm: Theory and application," *Advances in Engineering Software*, vol. 105, pp. 30–47, 2017, doi: 10.1016/j.advengsoft.2017.01.004.

[41] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015, doi: 10.1016/j.advengsoft.2015.01.010.

[42] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput Appl*, vol. 27, no. 4, pp. 1053–1073, 2016, doi: 10.1007/s00521-015-1920-1.