Journal of Quality Measurement and Analysis
e-ISSN: 2600-8602
https://doi.org/10.17576/jqma.2201.2026.02

JQMA 22(1) 2026, 23-37
http://www.ukm.my/jqma

BINARY SEAGULLS OPTIMIZATION ALGORITHMS IN RANDOM FOREST CLASSIFIER FOR SPAM AND NON-SPAM EMAILS CLASSIFICATION

NATASHA CLARISSA MAHARANI, MOHAMAD MUSLIKH* & SYAIFUL ANAM

ABSTRACT

Spam detection is essential for maintaining secure and efficient email communication, as spam messages not only reduce productivity but also pose risks such as phishing, scams, and malware. Effective spam detection improves user experience by filtering out irrelevant messages, safeguarding against fraud, reducing server load, and lowering operational costs. This paper proposes the Binary Seagull Optimization Algorithm (BSOA), a swarm intelligence-based method inspired by seagull migration and hunting behaviors, as an optimizer for classifying spam and non-spam emails using the Spambase dataset from the UCI Machine Learning Repository. BSOA begins by initializing positions and evaluating the objective function, then simulates migration and attack behaviors to generate new candidate solutions, retaining the best ones across iterations. Experimental results show that BSOA improves test accuracy by 1.4% compared to standard Random Forest models, with corresponding gains in precision (1.4%), recall (2.3%), and F1-score (0.5%). Furthermore, BSOA demonstrates competitive performance against other binary metaheuristics, including the Binary Bat, Binary Dragonfly, Binary Gravitational Search, and Binary Whale Optimization Algorithms. Notably, it achieves the lowest computational times for multiple population sizes, such as 20 (630 s), 25 (760 s), and 40 (1297 s), highlighting its efficiency and robustness. These results suggest that BSOA is a promising tool for enhancing spam detection systems by balancing classification accuracy with computational efficiency.

Keywords: random forest classifier; swarm intelligent; binary seagulls optimization algorithm; feature selection

1. Introduction

Spam detection is crucial for several reasons (Hemalatha *et al.* 2022). Spam emails not only clutter users' inboxes with irrelevant or unwanted messages, causing annoyance and reducing productivity, but they also pose significant security risks (Hemalatha *et al.* 2022). Many spam emails are designed to deceive users into disclosing personal information, making financial transactions, or downloading malicious software (Aju & Joy 2022). Effective spam detection helps reduce unwanted content by filtering out irrelevant or unwanted emails, improving user productivity and reducing frustration (Das *et al.* 2021). It also protects against fraud by blocking phishing attempts, scams, and malware, safeguarding users from financial and security threats (Jena *et al.* 2023; Kaddoura *et al.* 2022; Sonare *et al.* 2023). Moreover, it enhances email system efficiency by saving storage space and reducing server load, leading to faster processing (Ghaleb *et al.* 2022; Kumar *et al.* 2023). For businesses, spam detection saves on operational costs and improves the overall user experience by ensuring that only legitimate emails are received (Ghaleb *et al.* 2022; Jain *et al.* 2023).

However, detecting spam is a challenging task due to several factors. The methods used by spammers to evade detection are constantly evolving. They often involve sophisticated techniques such as obfuscation, randomization, and the use of misleading content. Furthermore, the volume of spam emails is continually increasing, which can overwhelm traditional detection systems. Effective spam detection systems must adapt quickly to these changes and accurately

differentiate between legitimate and malicious content, making it a complex and ongoing challenge.

Previous studies have demonstrated that Random Forest outperforms other models in terms of generalization performance. It also offers faster learning speed and ease of implementation. Additionally, it does not require extensive parameter tuning during diagnostic tasks. Additionally, Random Forest is well-suited for handling high-dimensional data (Chaudhary *et al.* 2016), supports multi-class classification (Han *et al.* 2020), and is compatible with parallel processing (Ao *et al.* 2019). Its robustness allows it to learn effectively, even when faced with substantial data errors, while other algorithms tend to be significantly impacted by inaccuracies (Houssein *et al.* 2021). Random Forest has also been shown to achieve better performance and accuracy compared to models like Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Linear Discriminant Analysis (LDA) in various datasets (Chen *et al.* 2020). These characteristics make Random Forest an attractive option for spam detection, particularly in high-dimensional and noisy email datasets. However, when used alone, Random Forest can suffer from certain weaknesses. While it is generally robust to overfitting, it may struggle with very high-dimensional feature spaces and a large number of irrelevant features (Malekipirbazari *et al.* 2021). In such cases, its performance can degrade because it may focus on noise or irrelevant patterns, leading to suboptimal classification results. Additionally, Random Forest models can be computationally expensive, especially when dealing with large datasets (Manzali *et al.* 2024), and they may lack interpretability due to the complexity of decision trees and ensemble methods.

In spam detection, feature selection plays a pivotal role. This is because email data is typically high-dimensional, containing various textual features, metadata, and other attributes. Without proper feature selection, spam detection models may become overly complex. They can also suffer from overfitting or fail to identify the underlying patterns in the data. Feature selection can be framed as an optimization problem, where the goal is to select a subset of features that maximizes classification performance while minimizing redundancy and dimensionality (Houssein *et al.* 2023). Given the vast number of possible feature subsets, this problem becomes combinatorially complex and computationally expensive.

To address this challenge, metaheuristic algorithms have emerged as powerful tools for feature selection in machine learning. Their ability to efficiently explore large and complex search spaces makes them especially suitable for high-dimensional tasks such as spam detection. Metaheuristic-based feature selection has been successfully used in many areas, including image classification, bioinformatics, and text analysis. As machine learning tasks continue to grow in scale and complexity, these algorithms offer a powerful and flexible approach for selecting the most relevant features (Dökeroğlu *et al.* 2022; Agrawal *et al.* 2021).

A significant subset of these techniques falls under the category of swarm intelligence, which has gained considerable attention for its robustness in handling high-dimensional data—an area where conventional methods frequently falter (Nguyen *et al.* 2020; Rostami *et al.* 2021). Algorithms such as Particle Swarm Optimization (PSO), the Salp Swarm Algorithm (SSA) (Zivkovic *et al.* 2022), and the Grasshopper Optimization Algorithm (Wang *et al.* 2020) emulate the collective behavior of biological populations to explore the search space efficiently. These swarm-based methods are particularly valued for their global search capabilities, adaptability, and tendency to avoid premature convergence to local optima. As a result, they contribute not only to more accurate classification outcomes but also to reductions in computational complexity (Nguyen *et al.* 2020; Rostami *et al.* 2021). Comparative studies consistently demonstrate that swarm intelligence-based feature selection methods either outperform or remain competitive with other advanced metaheuristics and traditional algorithms across a wide range of benchmark datasets (Awadallah *et al.* 2022).

While swarm intelligence algorithms have demonstrated strong performance in feature selection tasks, they are not without limitations. One significant drawback is slow convergence, particularly as the dimensionality of the feature space increases. For instance, the SSA can require a substantial number of iterations to reach an optimal or near-optimal solution (Qaraad

et al. 2022). Another key challenge lies in maintaining a proper balance between exploration—searching new areas of the solution space—and exploitation—refining known good solutions. An improper balance may lead to premature convergence on suboptimal solutions or incur excessive computational costs, as observed in PSO (Li *et al.* 2021). Furthermore, some swarm-based methods struggle with scalability. For example, the Artificial Bee Colony (ABC) algorithm is often inefficient when applied to high-dimensional datasets due to its high computational complexity (Rostami *et al.* 2021).

Among these innovations, the Seagull Optimization Algorithm (SOA) offers a unique approach to enhancing spam detection. SOA is inspired by the migration and attacking behaviors of seagulls, which allows it to balance exploration and exploitation effectively within the search space (Rahman *et al.* 2022). By mimicking natural behaviors, SOA can efficiently navigate through complex and high-dimensional feature spaces, which is crucial for identifying subtle patterns in spam detection. SOA addresses computationally expensive problems by balancing exploration and exploitation in the search space, making it highly effective for complex optimization tasks (Dhiman & Kumar 2019). The SOA outperforms many metaheuristic algorithms, such as the Emperor Penguin Optimizer, Collective Neurodynamic Optimization, Artificial Chemical Reaction Optimization, and Galaxy-based Search Algorithm in engineering problems (Dhiman & Kumar 2019). Unfortunately, SOA is designed for continuous optimization problems and is not inherently suited for binary feature selection tasks.

Binary SOA(BSOA) extends this concept to binary optimization problems, where solutions are represented in binary form. This adaptation has proven particularly adept in data mining applications, showcasing robust capabilities in tasks such as feature selection (Kumar *et al.* 2021). Its ability to effectively navigate and optimize binary search spaces makes it a valuable tool for feature selection and other data-intensive tasks. These advancements highlight the continuous evolution and refinement of PSO and its variants, aimed at addressing specific challenges and enhancing performance across diverse application domains (Cao *et al.* 2023). This paper proposes using BSOA for classifying spam and non-spam emails, leveraging its ability to navigate complex feature spaces and improve email classification performance. The method will be rigorously evaluated against state-of-the-art algorithms to assess its competitiveness, strengths, and areas for refinement in email classification and beyond. The main contribution of this research is as follows.

(1) We implement a hybrid model by integrating the BSOA with Random Forest to improve performance in feature selection and classification tasks.
(2) We utilize the Spambase dataset, which includes labeled email samples, to analyze the effectiveness of the proposed method in detecting spam.
(3) We assess and compare the performance of the proposed hybrid model against other binary metaheuristic algorithms, including the BBA, BDA, BGSA, and BWOA.

The paper is structured as follows. Section 2 describes the methodology of the SOA and its competitors, including the BSOA and other relevant metaheuristic algorithms used in spam detection. Section 3 presents the results and discussion, comparing the performance of regular Random Forest with Random Forest combined with BSOA. This section also includes an analysis of the accuracy and computational time between the BSOA-based approach and the competitor algorithms, such as the BBA, BDA, BGSA, and BWOA, as well as a new subsection that provides an approximate time complexity analysis of the proposed method. Finally, a limitations section has been added to discuss the use of a single dataset and to outline potential directions for future work.

2. Methodology

The research process for the SOA involves several steps. Figure 1 illustrates the workflow of the proposed method. Solid straight lines represent the sequential flow of processes. The dashed

lines in the flowchart indicate the outputs of the data-splitting step, namely the training and test datasets, which are subsequently used in the following processes. Initially, the dataset is imported into the working environment, and pre-processing steps are carried out to prepare the data. This includes splitting the data into training and testing sets to facilitate model evaluation. Following this, parameter initialization for the seagull algorithm is performed, setting values for the number of seagulls in the population and the maximum number of iterations. Given that results can vary with each experiment, repeated experiments are essential for consistency. To evaluate consistency and variation, the program is executed multiple times, focusing on computational time during training and accuracy during evaluation. In each iteration, the seagull algorithm is employed for feature selection and to identify the optimal model using a random forest classifier. The model's performance is subsequently evaluated on the test data, recording metrics such as training and testing accuracy, precision, recall, f_1 score, and computation time. After completing all iterations, the average values and distribution of these metrics are calculated. This data is used to identify optimal parameters and models for the specific application, ensuring reliable and robust feature selection and classification outcomes. Additionally, the performance of the BSOA is compared with various established binary optimization algorithms.

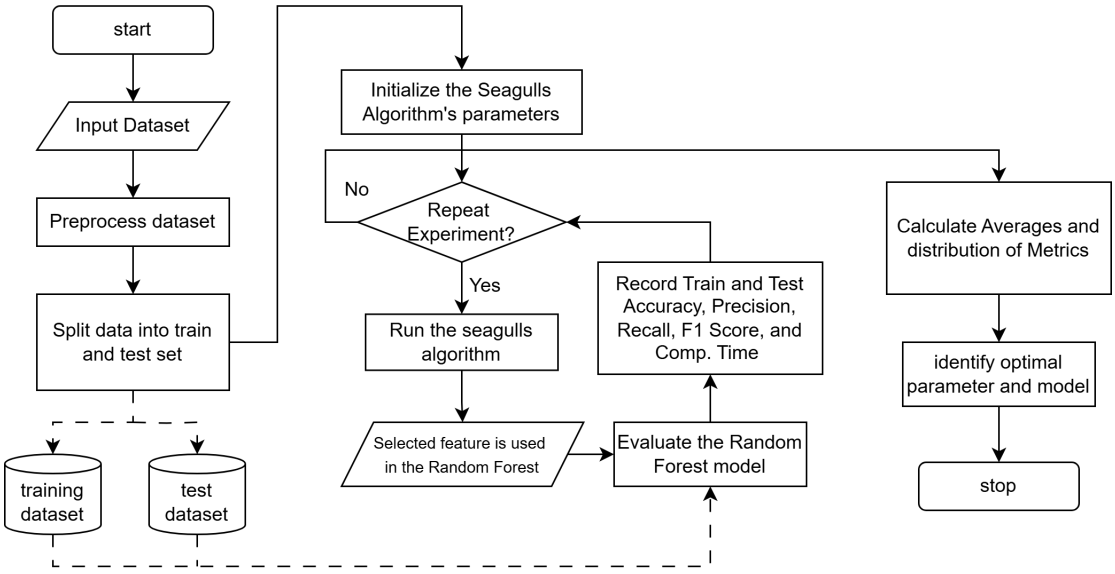

Figure 1: The research steps

2.1. *Input and preprocess data*

The dataset used for this research was sourced from the UCI Machine Learning Repository (Hopkins *et al.* 1999) and comprises 4,601 rows with 57 features, which can be seen in Table 1. The column numbers (0–56) indicate feature indices. The dataset consists of 57 predictive attributes and one target class. Attributes 1–48 correspond to the frequency of specific words that appear in the e-mail, expressed as the percentage of words in the message that match a given word. Attributes 49–54 represent the frequency of certain characters, measured as the percentage of characters in the e-mail that match the specified character. Attribute 55 measures the average length of uninterrupted sequences of capital letters, while attribute 56 captures the length of the longest uninterrupted sequence of capital letters. Attribute 57 represents the total number of capital letters in the e-mail. Finally, attribute 58 is the class attribute, which indicates whether the e-mail was classified as spam (1) or non-spam (0).

These features include various aspects of email content, such as word frequencies, character frequencies, and additional attributes. In the preprocessing phase, the dataset's quality is evaluated to ensure it is ready for analysis. An initial review revealed that the dataset is clean with no missing values, thus no imputation was required. However, a closer examination identified 391 duplicate entries, which were removed to preserve the integrity of the analysis and prevent potential bias. Since this study employs Random Forest as the base classifier, normalization is not applied, as Random Forest algorithms are inherently robust to differences in feature scaling. The subsequent step involves dividing the dataset into training and testing subsets, with 80% allocated for training and 20% for testing. This split is designed to provide a robust training set that captures diverse patterns while also allowing for a sufficient test set to objectively evaluate the model's performance.

Table 1: Sample of dataset

No	0	1	2	3	4	5	$\cdots$	54	55	56	57(Target)
0	0.00	0.64	0.64	0.0	0.32	0.00	$\cdots$	3.756	61	278	1
1	0.21	0.28	0.50	0.0	0.14	0.28	$\cdots$	5.114	101	1028	1
2	0.06	0.00	0.71	0.0	1.23	0.19	$\cdots$	9.821	485	2259	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
4600	0.00	0.00	0.65	0.0	0.00	0.00	$\cdots$	1.250	5	40	0

2.2. *Initializing seagulls optimization algorithm's parameters*

The next step involves initializing the parameters for the SOA. To ensure an unbiased evaluation of the optimal parameters, the program executes a series of experiments with varying parameter values. Key parameters such as the number of seagulls and the maximum number of iterations are systematically adjusted to assess their impact on the algorithm's performance. Meanwhile, the control parameter v is set to 1, as recommended in Kumar *et al.* (2021), as it acts as a velocity scaling factor that stabilizes the search dynamics; preliminary trials with other values showed no notable improvement. In this research, particular focus was placed on varying the number of seagulls, which directly influences the population size within the algorithm. The number of seagulls was tested at different levels: 10, 15, 20, 25, 30, 35, 40, 45, and 50. This iterative approach aims to identify the most effective parameter setting for achieving the best results in feature selection and optimization.

2.3. *Binary seagulls optimization algorithm (BSOA)*

The algorithm employed for feature selection in this research is the BSOA, as introduced by Kumar *et al.* (2021). It is a variant of the traditional SOA, adapted to handle binary optimization problems. The BSOA employs a spiral behavior to explore the solution space. This behavior is characterized by generating a control parameter θ, which is uniformly distributed between 0 and 2π. This parameter determines the direction in which the spiral evolves. Each feature is represented as a binary number in the binary space $\{b_1, b_2, \cdots ., b_m\}$. To align with the research objective of determining whether to select a particular feature, each entry in this space is represented by a value of $b_j = 0$ or $b_j = 1$, where $j \in 1, 2, 3, \ldots, m$ and m denotes the total number of features. Here, the $b_j = 0$ value denotes an unselected feature, while $b_j = 1$ signifies a selected feature. To create the spiral path, the algorithm generates a random radius r within the range $[0, 1]$ for each features. This radius controls the distance of the spiral from the origin. Using this radius and the control parameter θ, the algorithm calculates the spiral coordinates x and y as follows:

$$x = r \cdot \cos\theta \quad \text{and} \quad y = r \cdot \sin\theta \tag{1}$$

where B is a control parameter that scales the radius of the spiral. The cosine and sine functions determine the x and y coordinates of the spiral path, respectively. This approach ensures that the search process covers a wide range of possible solutions by moving along a spiral trajectory. In the BSOA, the position update of a seagull combines the current position with information from the best solution found so far and the spiral behavior. The updated position is computed using Eq. (2).

$$\boldsymbol{np_i} = \boldsymbol{p_i} + v \cdot (\boldsymbol{p_i} - \boldsymbol{bsol}) + (x + y) \cdot \mathbf{1} \tag{2}$$

with $\boldsymbol{p_i}$ representing the current position of the i-th seagulls, $\boldsymbol{bsol}$ as the best solution found, v as a coefficient that influences how much the current position should be adjusted based on the difference between the current position and the best solution, and x and y as the spiral coordinates computed in Eq. (1). The term $v \cdot (\boldsymbol{p_i} - \boldsymbol{bsol})$ drives the seagull towards the best-known solution, while $E \cdot (x + y)$ introduces exploratory behavior via the spiral path. This combination allows the algorithm to balance exploration and exploitation during the search process. The variable v is initialized and adjusted in the optimization process using Eq. (3).

$$v = v \left(1 - \frac{t}{max\_iter}\right) \tag{3}$$

The algorithm update the position for each seagulls by using Eq. (4),

$$\boldsymbol{T_r} = \frac{1}{1 + e^{-\boldsymbol{np_i}}} \tag{4}$$

to convert continuous values into probabilities. After computing the new position, the algorithm applies a binary threshold to ensure that the feature subset remains binary. The updated position is thresholded using Eq. (5).

$$p_i = \begin{cases} 1, & \text{if } T_r > \text{rand}(numfeatures) \\ 0. & \text{otherwise} \end{cases} \tag{5}$$

This process ensures that the seagull's position is adjusted in a binary manner, making it suitable for feature selection tasks in machine learning. The pseudocode for the BSOA is presented in Pseudocode 2.1.

Pseudocode 2.1.

Initialize $num\_seagulls$
Set $v = 1$, $t = 0$, $max\_iter = 50$
while $t < max\_iter$ **do**
  Calculate fitness value of each seagull
  for $i = 1$ to $num\_seagulls$ **do**
    $\theta = \text{rand}[0, 2\pi]$
    $r = \text{rand}[0, 1]$
    $x = r \cdot \cos(\theta)$
    $y = r \cdot \sin(\theta)$
    $\boldsymbol{np_i} = \boldsymbol{p_i} + v \cdot (\boldsymbol{p_i} - \boldsymbol{bsol}) + (x + y) \cdot \mathbf{1}$
    $\boldsymbol{T_r} = \frac{1}{1+e^{-\boldsymbol{np_i}}}$
    $v = v \cdot (1 - \frac{t}{max\_iter})$
    Update the position $\boldsymbol{p_i}$ using Eq. (5)
  end for
  Compute the fitness of every seagull
  Increment t by 1

Return the best solution
end while

2.4. *Comparison with competitor algorithm*

To evaluate the performance of the Binary Seagulls Optimization Algorithm (BSOA), we conducted a comparative analysis with several established optimization algorithms. This comparison aims to benchmark the effectiveness of BSOA in feature selection against other algorithms, providing insights into its relative strengths and weaknesses. The competitor algorithms selected for comparison are:

(1) Binary Bat Algorithm (BBA) (Ardiyansa *et al.* 2024): This algorithm is an adaptation of the Bat Algorithm that handles binary optimization problems by using bat-inspired behaviors and echolocation. The key equations used in this algorithm are Eqs. (6) and (7),

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} + (\boldsymbol{p_{best}} - \boldsymbol{x}_i^t) \cdot \beta \tag{6}$$

as the velocity update where $\boldsymbol{v}_i^t$ is the velocity of bat with and index i at iteration t, $\boldsymbol{p_{best}}$ as the global best position, $\boldsymbol{x}_i^t$ as the current position and β is a random number in $[0, 1]$.

$$x_{i,j}^{t+1} = sigmoid(v_{i,j}^t) > r, for j = 1, 2, ..., m \tag{7}$$

where $sigmoid(v) = \frac{1}{1+e^{-v}}$ and r is a random treshold. The algorithm utilizes three key parameters which are Loudness $A = 0.5$, pulse rate $r = 0.5$, and a fixed number of bats.

(2) Binary Dragonfly Algorithm (BDA) (Mafarja *et al.* 2017): It is a variant of the Dragonfly Algorithm tailored for binary optimization, leveraging the swarming behavior of dragonflies. Its key equation includes Eq. (8),

$$x_{i,j}^{t+1} = sigmoid(S_{i,j}^t) > r, \text{for } j = 1, 2, ..., m \tag{8}$$

where $\boldsymbol{S}_i^t = w \cdot \boldsymbol{S}_i^{t-1} + c_1 \cdot \boldsymbol{A} + c_2 \cdot \boldsymbol{F} + c_3 \cdot \boldsymbol{E} + c_4 \cdot \boldsymbol{G} + c_5 \cdot \boldsymbol{O}$, with w is the inertia weight, $\boldsymbol{A}$, $\boldsymbol{F}$, $\boldsymbol{E}$, $\boldsymbol{G}$, and $\boldsymbol{O}$ are attraction, food source, enemy, gravity, and obstacle factors, respectively. The algorithm uses parameters including cognitive and social coefficients $c_1 = c_2 = 2.0$, inertia weight $w = 0.9$.

(3) Binary Gravitational Search Algorithm (BGSA) (Somu *et al.* 2020): A binary version of the Gravitational Search Algorithm that models the gravitational interaction between agents for optimization. The fundamental equations are Eqs. (9) and (10),

$$F_{ij} = G \cdot \frac{M_i \cdot M_j}{R_{ij} + \epsilon} \tag{9}$$

as the gravitional force where G is the gravitational constant, M_i and M_j are masses of agents i and j, R_{ij} is the Euclidean distance, and ϵ is a small value to prevent division by zero.

$$\boldsymbol{v}_i^t = w \cdot \boldsymbol{v}_i^{t-1} + \sum_j F_{ij} \cdot (\boldsymbol{x_j} - \boldsymbol{x_i}) \tag{10}$$

as the velocity update. This algorithm also used position update with the same logic as in BBA. Parameters include gravitational constant $G = 100/(1+t)$, inertia weight $w = 0.8$.

(4) Binary Whale Optimization Algorithm (BWOA) (Hussien *et al.* 2020): An adaptation of the Whale Optimization Algorithm for binary problems, inspired by the hunting strategies

of whales. The key equations are Eqs. (11) and (12),

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{p_i} + A \cdot D \cdot \mathbf{1} \tag{11}$$

as the equation that represents how the whales encircling its prey where

$$D = |C \cdot \boldsymbol{p_{best}} - \boldsymbol{x_i}|,$$

A and C are coefficients calculated as $A = 2a \cdot r - a$ and $C = 2 \cdot r$.

$$\boldsymbol{x}_i^{t+1} = D \cdot e^{bl} \cdot \cos(2\pi l) \cdot \mathbf{1} + \boldsymbol{p_{best}} \tag{12}$$

as the spiral update where l is a random number in [–1, 1]. The parameters are a decreases from 2 to 0, r is random, and $b = 1$.

To ensure the reproducibility of the experimental results, several key factors were carefully controlled throughout the study. All algorithms, including BSOA and its competitors (BBA, BDA, BGSA, and BWOA), were implemented using consistent settings for population size, number of iterations, and stopping criteria. Each algorithm was executed 20 times, and the results reported represent the average across these runs to reduce the impact of randomness and provide a more reliable performance assessment. Additionally, all experiments were conducted on the same hardware environment to ensure that computational time comparisons were not influenced by external processing variations. The dataset used for training and testing was held constant across all experiments, with a fixed train-test split and random seed initialization to ensure consistency.

3. Result and Discussion

3.1. *Comparative analysis of Random Forest with BSOA and regular Random Forest*

The evaluation of precision, recall, and f_1-score revealed that the Random Forest model optimized with BSOA performed exceptionally well, achieving precision and recall values ranging from 0.95 to 0.97 for both classes. Additionally, it recorded high f_1-scores of 0.95 for class 1 and 0.96 for class 0, as presented in Table 2. These results indicate that the Random Forest model with BSOA excels in accurately classifying instances in both categories. Furthermore, the Random Forest model with BSOA outperformed the standard Random Forest model in these evaluations. Table 2 highlights improvements in macro averages: a 0.0144 increase in precision, a 0.0235 increase in recall, and a 0.0058 increase in f_1-score. Both micro and macro averages provides a more comprehensive assessment of performance, where micro averages emphasize overall accuracy across all instances, while macro averages ensure that each class is treated equally, preventing performance bias towards the majority class. These findings suggest that the proposed model offers superior performance compared to the regular Random Forest.

To confirm the robustness of these improvements, a paired t-test was conducted on the accuracy values of both models. The results ($t = 8.54, p < 0.001$) demonstrate that the performance gains achieved by the Random Forest model with BSOA are statistically significant. These findings strongly suggest that the proposed model offers superior performance compared to the regular Random Forest.

3.2. *Influence of parameter to computational time of Random Forest with BSOA*

The Binary Seagulls Algorithm was executed with variations in the number of seagulls parameter. The result suggest that the distribution of computational time is affected by this parameter. Through experimentation, it was observed that the computation time showed variations, as illustrated in Figure 2 and detailed in Table 3. BSOA shows an increase in computation time as

Table 2: Performance of the proposed algorithm and normal random forest on precision, recall, and f_1-score

Label	Meaning	Random Forest with BSOA			Reguler Random Forest		
		Presicion	Recall	f_1-score	Presicion	Recall	f_1-score
0	The given mail is not spam	0.9536	0.9793	0.9663	0.9397	0.9698	0.9541
1	The given mail is spam	0.9711	0.9539	0.9532	0.9563	0.9164	0.9359
	Macro average	0.9624	0.9666	0.9598	0.9480	0.9431	0.9540
	Weighted average	0.9610	0.9684	0.9607	0.9468	0.9470	0.9463

the number of seagulls increases, a trend observed in other algorithms as well. Compared to other algorithms, BSOA has small computational time when the number of particles is between 20 and 40. BBA,BGSA, WOA has smaller computational time when the number of particle are between 10 and 15 and also between 45 and 50. BDA has the biggest computational time compared to the other algorithms.

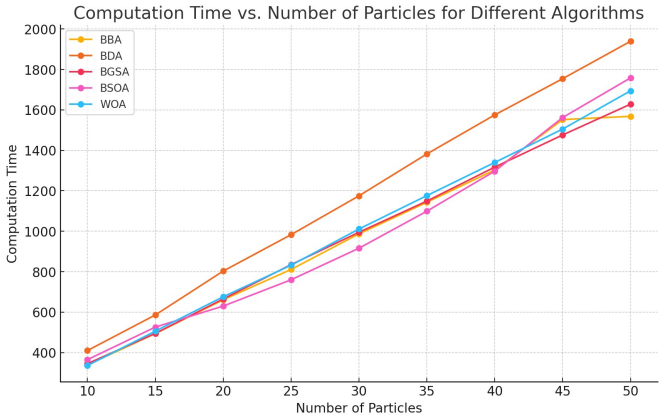

Figure 2: Distribution of computational time with different number of particles

To further evaluate the observed performance differences, paired t-tests were conducted across seven runs for each particle setting (10–40). The comparison between BSOA and BDA yielded a highly significant difference ($t = -5.22$, $p \approx 0.002 < 0.01$), indicating that BDA significantly outperformed BSOA in terms of computational cost. In contrast, the comparisons of BSOA with BBA ($t = -1.47$, $p \approx 0.19$) and BGSA ($t = -1.81$, $p = 0.12$) did not reveal statistically significant differences. For BWOA, the test produced a borderline significant result ($t = -2.18$, $p \approx 0.05$), suggesting that BWOA may slightly outperform BSOA, although the evidence is weaker. Overall, these findings confirm that while BSOA demonstrates competitive performance, BDA provides a statistically superior outcome under the tested conditions.

Overall the BSOA has smaller computational time compared to its competitor. These insights are critical for understanding the performance and efficiency of these algorithms under varying conditions. They aid in selecting the appropriate algorithm based on the specific requirements and constraints of the problem at hand. By optimizing computational time, BSOA demonstrates its advantage in scenarios where quick processing is essential. This enhances its appeal for practical applications in machine learning and optimization tasks.

3.3. *Influence of parameter to accuracy of Random Forest with BSOA*

The testing accuracy can be seen in Figure 3. For testing accuracy, BSOA performs well, with accuracy rising from 93.82% to 96.08% as the number of particles increases, peaking at 50 particles. BDA achieves the highest testing accuracy overall, with a peak of 96.20% at various

Table 3: Computational time of different number of particles in different algorithms

Number of particles	BBA	BDA	BGSA	BSOA	BWOA
10	339s	410s	345s	365s	**336s**
15	**494s**	586s	495s	526s	506s
20	661s	803s	665s	**630s**	676s
25	811s	983s	835s	**760s**	832s
30	987s	1175s	995s	**916s**	1011s
35	1143s	1382s	1149s	**1099s**	1176s
40	1303s	1575s	1317s	**1297s**	1340s

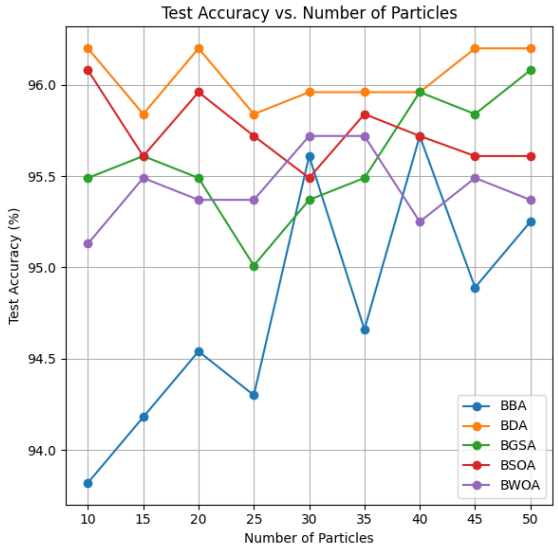

Figure 3: Test accuracy of different algorithms

particle sizes, showcasing its robustness. BGSA and BWOA also deliver strong results but fall short of BDA's peak performance.

The testing accuracy as seen in Table 4 varies slightly, with values ranging from 93.82% to 96.20%. The Random Forest models with different optimization algorithms generally maintain high accuracy on the test data as well. BDA has the highest testing accuracy, reaching 96.20% at several particle settings. This suggests that the BDA might be slightly better at generalizing to unseen data compared to the others in this experiment. BSOA and BGSA also show strong performance, with testing accuracy reaching 96,08%, BSOA's performance is close to BDA, but BDA has a slight edge in test accuracy.

Overall, the performance of all algorithms remains relatively stable across different particle sizes, with only minor variations in test accuracy. This suggests that the choice of particle size has a manageable effect on model performance. While BSOA performs excellently, it is slightly outperformed by BDA in terms of peak accuracy. Statistical analysis using paired t-tests across the nine runs confirmed that the difference between BSOA and BDA is significant ($t = -4.68$, $p = 0.0016$), indicating that BDA achieves consistently higher performance. On the other hand, BSOA significantly outperforms BBA ($p = 0.0046$) and BWOA ($p = 0.0167$), while the difference between BSOA and BGSA is not statistically significant ($p = 0.181$). BGSA and BWOA also show commendable results, demonstrating that various optimization approaches can be effective, with BDA leading slightly in this analysis.

Table 4: Test accuracy of different algorithms

Number of particles	BBA	BDA	BGSA	BSOA	BWOA
10	93.82%	96.20%	95.49%	96.08%	95.13%
15	94.18%	95.84%	95.61%	95.61%	95.49%
20	94.54%	96.20%	95.49%	95.96%	95.37%
25	94.30%	95.84%	95.01%	95.72%	95.37%
30	95.61%	95.96%	95.37%	95.49%	95.72%
35	94.66%	95.96%	95.49%	95.84%	95.72%
40	95.72%	95,96%	95.96%	95.72%	95.25%
45	94.89%	96.20%	95.84%	95.61%	95.49%
50	95.25%	96.20%	96.08%	95.61%	95.37%
mean	94.77%	96.04%	95.60%	95.73%	95.46%
Std.	0.5785	0.2140	0.2561	0.1776	0.1712

3.4. *Comparison between Random Forest with BSOA and competitor algorithms*

BSOA demonstrates a strong balance between accuracy and computational efficiency, offering competitive performance compared to other recent metaheuristic algorithms. This trade-off is essential in practical settings, especially in machine learning tasks like spam detection, where rapid decision-making is often required (Nssibi *et al.* 2023). The experimental results reveal that BSOA consistently achieves high classification accuracy while maintaining lower computational costs, particularly for a moderate number of particles.

Table 5: Training accuracy comparison of optimization algorithms

Number of particles	BBA	BDA	BGSA	BSOA	BWOA
10	99.76%	99.85%	99.82%	99.91%	99.76%
15	99.64%	99.88%	99.79%	99.85%	99.73%
20	99.79%	99.85%	99.67%	99.85%	99.85%
25	99.88%	99.85%	99.84%	99.85%	99.85%
30	99.82%	99.94%	99.82%	99.79%	99.85%
35	99.72%	99.94%	99.88%	99.85%	99.82%
40	99.91%	99.94%	99.79%	99.85%	99.91%
45	99.76%	99.85%	99.73%	99.73%	99.91%
50	99.85%	99.85%	99.88%	99.70%	99.82%
Mean	99.79%	99.88%	99.83%	99.83%	99.82%
Std.	0.0838	0.0427	0.0541	0.0779	0.0602

In contrast, the BDA, although achieving the highest test accuracy across most configurations, exhibits significantly higher computational time. This is consistent with the findings of Warnakulasooriya and Segev (2025), who observed that certain swarm-based algorithms may suffer from increased complexity as problem dimensionality grows. BDA incorporates complex movement and memory mechanisms that, while improving exploitation, also contribute to longer runtimes (Emambocus *et al.* 2021).

BBA proves to be the fastest algorithm in this comparison; however, its lower accuracy suggests it may not explore the solution space adequately, a limitation observed in simplified swarm-based models (Rostami *et al.* 2021). BGSA and BWOA show moderate computational time and accuracy, aligning with prior evaluations that note their stable but sometimes suboptimal search behavior in high-dimensional tasks (Qaraad *et al.* 2022).

The competitive performance of BSOA can be attributed to its unique migration and attack-inspired search mechanism, which allows it to adaptively navigate between global and local search zones. This dual behavior supports its ability to avoid premature convergence, a com-

mon challenge in binary feature selection problems (Dhiman & Kumar 2019). Furthermore, its scalability and parameter flexibility make it suitable for real-world problems where feature spaces are large and diverse. Notably, BSOA also demonstrates strong consistency, as reflected by its relatively low standard deviation (Table 3) in test accuracy, indicating stable performance across different particle counts.

Although not previously discussed, Table 5 shows that the training accuracy for all algorithms remained consistently high, hovering around 99%. However, the differences emerge during the testing phase, where accuracies range between 95% and 96%. This noticeable gap suggests a degree of overfitting, where the models may have learned patterns that are too tailored to the training data and struggle to generalize to new, unseen samples. To address this, future enhancements could explore strategies such as regularization or more robust cross-validation methods to reduce overfitting and improve generalization performance. Ultimately, the results suggest that while BDA is slightly more accurate, BSOA provides a much more efficient alternative, making it preferable in scenarios that require a balance between accuracy and execution time. These findings support the growing interest in biologically inspired binary optimization techniques that offer practical efficiency for high-dimensional classification tasks (Li *et al.* 2021). For example, in gene expression data classification—where thousands of genes are measured for a limited number of samples—the BSOA can be employed to select the most relevant subset of genes, reducing dimensionality while maintaining or improving classification accuracy. This highlights BSOA's potential not only in spam detection but also in critical domains like bioinformatics, where robust and efficient feature selection is essential.

3.5. *Computational complexity analysis*

The computational complexity of the proposed BSOA+RF method can be estimated based on the number of operations performed during its main phases. The Binary Seagull Optimization Algorithm (BSOA) operates on a population of N agents for I iterations. During the initialization phase, each of the N agents is randomly generated, which incurs a cost of $O(N)$. In each subsequent iteration, every agent undergoes three main operations: two fitness evaluations and one movement (position update). This results in approximately $3 \times I \times N$ basic operations throughout the search process, giving a total BSOA-side cost of about $N(1 + 3I)$.

In this wrapper-based setting, each fitness evaluation involves training and validating a Random Forest (RF) classifier on a subset of features. The training complexity of RF is approximately $O(T \times n \log n)$, where T is the number of trees and n is the number of samples. Since each agent is evaluated twice per iteration, the total evaluation cost becomes $2I(N \cdot T \cdot n \log n)$. In addition, there is an IN cost associated with the movement updates of the agents, which is relatively lightweight compared to the evaluation step but included here for completeness.

Combining these components, the total computational cost of the BSOA+RF pipeline can be expressed as:

$$N \; + \; 2I(N \cdot T \cdot n \log n) \; + \; IN.$$

By dropping constant multipliers and lower-order terms, the overall asymptotic complexity of the proposed approach simplifies to:

$$O(I\, N\, T\, n \log n).$$

This analysis is provided to make the underlying computational cost explicit and to complement the empirical runtime measurements reported in the results section.

3.6. *Limitations*

This study was conducted using a single spam email dataset that already provides structured feature representations compatible with the proposed BSOA-based feature selection pipeline. While additional datasets could provide broader validation, most other available spam datasets consist of raw text and would require extensive NLP preprocessing and substantial modifications to the current experimental design. Incorporating such datasets would shift the scope of this study from feature selection to end-to-end text processing, which was not the primary objective. Future work could explore extending the proposed approach to raw text datasets by integrating NLP-based feature extraction techniques.

4. Conclusion

In this research, BSOA is applied to develop a model for the Spambase dataset. The study aims to identify the most effective optimization strategy for spam detection by comparing BSOA with other advanced algorithms such as BBA, BDA, BGSA, and BWOA. The evaluation criteria include accuracy, precision, recall, and F1-score, with an emphasis on both training and testing phases. In the test phase, BSOA achieved a mean accuracy of 95.73% with a standard deviation of 0.1776, indicating both strong performance and low variance. In terms of computational time, BSOA showed efficient runtime across different particle sizes—for instance, 630 s (20 particles), 760 s (25 particles), and 1297 s (40 particles)—consistently outperforming BDA and BGSA. These results demonstrate that BSOA is a strong contender in spam detection, striking a good balance between high accuracy and manageable computational cost. Compared to BDA (96.04%, std 0.2140), BGSA (95.60%, 0.2561), BWOA (95.46%, 0.1712), and BBA (94.77%, 0.5785), BSOA offers a competitive and robust solution.

The number of particles used in the optimization process significantly affects both classification accuracy and computational time. Generally, increasing the number of particles results in improved accuracy, particularly during the testing phase, as observed across all algorithms evaluated. However, this performance gain comes with a corresponding increase in computational cost. BSOA stands out by maintaining relatively high accuracy while keeping computation time at a manageable level. In contrast, BDA—despite achieving the highest test accuracy—incurs the longest runtime, highlighting the inherent trade-off between accuracy and efficiency in optimization-based feature selection.

This research underscores the importance of choosing an appropriate optimization algorithm for spam detection, as well as the careful tuning of key parameters—such as the number of particles—to achieve optimal results. Thoughtful algorithm selection and configuration can significantly enhance the effectiveness of spam detection systems, leading to improved user productivity, stronger protection against security threats, and greater overall system efficiency. The competitive performance of BSOA can be attributed to its biologically inspired migration and attack mechanisms, which enable it to strike a balance between global exploration and local exploitation. This allows the algorithm to avoid premature convergence and efficiently navigate complex, high-dimensional search spaces. Given these strengths, the approach may also hold promise for other challenging classification problems, such as gene expression data analysis in the field of bioinformatics.

Future research could refine BSOA further by introducing a dynamic radius adjustment mechanism based on population diversity, iteration progress, or convergence behavior. Such enhancements could improve the balance between exploration and exploitation. Moreover, testing BSOA on more diverse and high-dimensional datasets would help validate its generalizability and scalability. Finally, the observed performance gap between training and testing phases suggests some degree of overfitting. Addressing this through regularization, robust cross-validation, or ensemble methods may enhance generalization and reliability in real-world applications.

Acknowledgements

This paper is funded by Professor Grant number 02172.9/UN10.F0901/B/KS/2024 of Brawijaya University, Malang, East Java, Indonesia.

References

Agrawal P., Abutarboush H.F., Ganesh T. & Mohamed A.W. 2021. Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). *IEEE Access* **9**: 26766–26791.

Aju O.G. & Joy A.A. 2022. An email spam filtering model using ensemble of machine learning techniques. *International Journal of Computer Applications Technology and Research* **11**(3): 66–71.

Ao Y., Li H., Zhu L., Ali S. & Yang Z. 2019. The linear random forest algorithm and its advantages in machine learning assisted logging regression modeling. *Journal of Petroleum Science and Engineering* **174**: 776–789.

Ardiyansa S.A., Maharani N.C., Anam S. & Julianto E. 2024. Optimizing heart attack diagnosis using random forest with bat algorithm and greedy crossover technique. *BAREKENG: Journal of Mathematics and Its Applications* **18**(2): 1053–1066.

Awadallah M.A., Al-Betar M.A., Braik M.S., Hammouri A.I., Doush I.A. & Zitar R.A. 2022. An enhanced binary Rat Swarm Optimizer based on local-best concepts of PSO and collaborative crossover operators for feature selection. *Computers in Biology and Medicine* **147**: 105675.

Cao L., Wang Z., Wang Z., Wang X. & Yue Y. 2023. An energy-saving and efficient deployment strategy for heterogeneous wireless sensor networks based on improved seagull optimization algorithm. *Biomimetics* **8**(2): 231.

Chaudhary A., Kolhe S. & Kamal R. 2016. An improved random forest classifier for multi-class classification. *Information Processing in Agriculture* **3**(4): 215–222.

Chen R.C., Dewi C., Huang S.W. & Caraka R.E. 2020. Selecting critical features for data classification based on machine learning methods. *Journal of Big Data* **7**: 52.

Das L., Ahuja L. & Pandey A. 2021. Existing spam filtering methods considering different technique: A review. *Proceedings of the 2021 International Conference on Technological Advancements and Innovations*, pp. 515–520.

Dhiman G. & Kumar V. 2019. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. *Knowledge-Based Systems* **165**: 169–196.

Dökeroğlu T., Deniz A. & Kiziloz H.E. 2022. A comprehensive survey on recent metaheuristics for feature selection. *Neurocomputing* **494**: 269–296.

Emambocus B.A.S., Jasser M.B., Mustapha A. & Amphawan A. 2021. Dragonfly algorithm and its hybrids: A survey on performance, objectives and applications. *Sensors* **21**(22): 7542.

Ghaleb S.A.A., Mohamad M., Ghanem W.A.H.M., Nasser A.B., Ghetas M., Abdullahi A.M., Saleh S.A.M., Arshad H., Omolara A.E. & Abiodun O.I. 2022. Feature selection by multiobjective optimization: Application to spam detection system by neural networks and grasshopper optimization algorithm. *IEEE Access* **10**: 98475–98489.

Han S., Kim H. & Lee Y.S. 2020. Double random forest. *Machine Learning* **109**(8): 1569–1586.

Hemalatha M., Katta S., Santosh R.S. & Priyanka P. 2022. E-mail spam detection. *International Journal of Computer Science and Mobile Computing* **11**(1): 36–44.

Hopkins M., Reeber E., Forman G. & Suermondt J. 1999. Spambase. https://archive.ics.uci.edu/dataset/94/spambase (15 January 2024).

Houssein E.H., Gad A.G., Hussain K. & Suganthan P.N. 2021. Major advances in particle swarm optimization: Theory, analysis, and application. *Swarm and Evolutionary Computation* **63**: 100868.

Houssein E.H., Oliva D., Çelik E., Emam M.M. & Ghoniem R.M. 2023. Boosted sooty tern optimization algorithm for global optimization and feature selection. *Expert Systems with Applications* **213**: 119015.

Hussien A.G., Oliva D., Houssein E.H., Juan A.A. & Yu X. 2020. Binary whale optimization algorithm for dimensionality reduction. *Mathematics* **8**(10): 1821.

Jain A., Goel H., Joshi K., Gupta V.K., Shukla A. & Jain P. 2023. G-mail spam detection using natural language processing. *Proceedings of the 2023 4th IEEE Global Conference for Advancement in Technology*, pp. 1–6.

Jena D., Kumari A., Tejaswini K., Ankita A. & Kumar B. 2023. Malicious spam detection to avoid vicious attack. *Proceedings of the 2023 14th International Conference on Computing Communication and Networking Technologies*, pp. 1–7.

Kaddoura S., Chandrasekaran G., Popescu D.E. & Duraisamy J.H. 2022. A systematic literature review on spam content detection and classification. *PeerJ Computer Science* **8**: e830.

Kumar A., Kumar S., Kumar K. & Naib B.B. 2023. E-mail fraud detection. *International Journal of Emerging Science and Engineering* **11**(9): 1–7.

Kumar V., Kumar D., Kaur M., Singh D., Idris S.A. & Alshazly H. 2021. A novel binary seagull optimizer and its application to feature selection problem. *IEEE Access* **9**: 103481–103496.

Li D., Guo W., Lerch A., Li Y., Wang L. & Wu Q. 2021. An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization. *Swarm and Evolutionary Computation* **60**: 100789.

Mafarja M.M., Eleyan D., Jaber I., Hammouri A. & Mirjalili S. 2017. Binary dragonfly algorithm for feature selection. *Proceedings of the 2017 International Conference on New Trends in Computing Sciences*, pp. 12–17.

Malekipirbazari M., Aksakalli V., Shafqat W. & Eberhard A. 2021. Performance comparison of feature selection and extraction methods with random instance selection. *Expert System with Applications* **179**: 115072.

Manzali Y., Akhiat Y., Barry K.A., Akachar E. & El Far M. 2024. Prediction of student performance using random forest combined with Naïve Bayes. *The Computer Journal* **67**(8): 2677–2689.

Nguyen B.H., Xue B. & Zhang M. 2020. A survey on swarm intelligence approaches to feature selection in data mining. *Swarm and Evolutionary Computation* **54**: 100663.

Nssibi M., Manita G. & Korbaa O. 2023. Advances in nature-inspired metaheuristic optimization for feature selection problem: A comprehensive survey. *Computer Science Review* **49**: 100559.

Qaraad M., Amjad S., Hussein N.K. & Elhosseini M.A. 2022. An innovative quadratic interpolation salp swarm-based local escape operator for large-scale global optimization problems and feature selection. *Neural Computing and Applications* **34**(20): 17663–17721.

Rahman L.I.S., Pratiwi A.B. & Suprajitno H. 2022. Penerapan seagulls optimization algorithm untuk menyelesaikan open vehicle routing problem. *Contemporary Mathematics and Applications* **4**(1): 42–50.

Rostami M., Berahmand K., Nasiri E. & Forouzandeh S. 2021. Review of swarm intelligence-based feature selection methods. *Engineering Applications of Artificial Intelligence* **100**: 104210.

Somu N., Gauthama Raman M.R., Kaveri A., Akshay Rahul K., Krithivasan K. & Shankar Sriram V.S. 2020. IBGSS: An Improved Binary Gravitational Search Algorithm based search strategy for QoS and ranking prediction in cloud environments. *Applied Soft Computing* **88**: 105945.

Sonare B., Dharmale G.J., Renapure A., Khandelwal H. & Narharshettiwar S. 2023. E-mail spam detection using machine learning. *Proceedings of the 2023 4th International Conference for Emerging Technology*, pp. 1–5.

Wang D., Chen H., Li T., Wan J. & Huang Y. 2020. A novel quantum grasshopper optimization algorithm for feature selection. *International Journal of Approximate Reasoning* **127**: 33–53.

Warnakulasooriya K. & Segev A. 2025. Comparative analysis of accuracy and computational complexity across 21 swarm intelligence algorithms. *Evolutionary Intelligence* **18**: 18.

Zivkovic M., Stoean C., Chhabra A., Budimirovic N., Petrovic A. & Baanin N. 2022. Novel improved salp swarm algorithm: An application for feature selection. *Sensors* **22**(5): 1711.

Mathematics Department
Faculty of Science
Brawijaya University
East Java, INDONESIA
E-mail: natashacm@student.ub.ac.id, mslk@ub.ac.id, syaiful@ub.ac.id*

Received: 31 July 2024
Accepted: 20 September 2025

*Corresponding author