IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 31 January 2024, accepted 15 February 2024, date of publication 19 February 2024, date of current version 1 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3367440

RESEARCH ARTICLE

An Opposition-Based Great Wall Construction Metaheuristic Algorithm With Gaussian Mutation for Feature Selection

FAROUQ ZITOUNI[1], ABDULAZIZ S. ALMAZYAD[2], GUOJIANG XIONG[3], ALI WAGDY MOHAMED[4,5], AND SAAD HAROUS[6]

[1]Department of Computer Science and Information Technology, Kasdi Merbah University, Ouargla 30000, Algeria
[2]Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
[3]Guizhou Key Laboratory of Intelligent Technology in Power System, College of Electrical Engineering, Guizhou University, Guiyang 550025, China
[4]Operations Research Department, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza 12613, Egypt
[5]Applied Science Research Center, Applied Science Private University, Amman 11931, Jordan
[6]Department of Computer Science, College of Computing and Informatics, University of Sharjah, Sharjah, United Arab Emirates

Corresponding author: Farouq Zitouni (zitouni.farouq@univ-ouargla.dz)

This work was supported by King Saud University, Riyadh, Saudi Arabia, through the Researchers Supporting Program under Grant RSPD2023R809.

ABSTRACT The feature selection problem involves selecting a subset of relevant features to enhance the performance of machine learning models, crucial for achieving model accuracy. Its complexity arises from the vast search space, necessitating the application of metaheuristic methods to efficiently identify optimal feature subsets. In this work, we employed a recently proposed metaheuristic algorithm named the Great Wall Construction Algorithm to address this challenge – a powerful optimizer with promising results. To enhance the algorithm's performance in terms of exploration, exploitation, and avoidance of local optima, we integrated opposition-based learning and Gaussian mutation techniques. The proposed algorithm underwent a comprehensive comparative analysis against ten influential state-of-the-art methodologies, encompassing seven contemporary algorithms and three classical counterparts. The evaluation covered 22 datasets of varying sizes, ranging from 9 to 856 features, and included the utilization of six distinct evaluation metrics related to accuracy, classification error rate, number of selected features, and completion time to facilitate comprehensive comparisons. The obtained numerical results underwent rigorous scrutiny through several non-parametric statistical tests, including the Friedman test, the post hoc Dunn's test, and the Wilcoxon signed ranks test. The resulting mean ranks and p-values unequivocally demonstrate the superior efficacy of the proposed algorithm in addressing the feature selection problem. The Matlab source code for the proposed approach is available for access via the link "https://www.mathworks.com/matlabcentral/fileexchange/159728-an-opposition-based-gwca-for-the-fs-problem".

INDEX TERMS Feature selection problem, great wall construction metaheuristic algorithm, opposition-based learning, Gaussian mutation.

I. INTRODUCTION

In the era of big data and complex datasets, Machine Learning (ML) has emerged as a powerful tool for extracting valuable insights and making data-driven decisions [1], [2], [3], [4]. However, with the ever-increasing dimensionality of data, the curse of dimensionality has become a significant challenge in developing accurate and efficient predictive models [5]. This is where the crucial role of Feature Selection (FS) comes into play. FS, also known as attribute selection or variable selection, is the process of identifying and choosing the most relevant and informative subset of features from a vast pool of

The associate editor coordinating the review of this manuscript and approving it for publication was Diego Oliva.

input variables [6], [7]. The primary objective is to enhance the performance of ML algorithms by eliminating irrelevant, redundant, or noisy features that might negatively impact model accuracy, increase computational costs, and/or reduce interpretability.

The importance of FS lies in its ability to not only improve predictive model performance but also enhance the efficiency and generalization of ML algorithms [8], [9], [10]. By selecting a subset of the most discriminative features, FS not only reduces the risk of overfitting but also mitigates the computational burden associated with processing large volumes of data. Moreover, in many real-world applications, interpreting the model's decision-making process is crucial to gain trust and acceptance. By selecting a concise set of meaningful features, FS facilitates model interpretability, enabling domain experts and non-technical users to comprehend the factors influencing the model's predictions. This emphasizes the significance of FS in ML. Whether it is in the realm of predictive modelling, classification, regression, or any other ML task, FS serves as a critical preprocessing step to unlock the full potential of ML algorithms. Through careful selection of relevant features, data scientists can build more accurate, efficient, and interpretable models, paving the way for actionable insights and informed decision-making.

In this context, FS methods can be broadly categorized into filter, wrapper, and embedded techniques [6], [11], [12]. Each approach has its strengths and weaknesses, and the choice of method depends on the nature of the dataset and the specific ML algorithm being used. In other words, filter, wrapper, and embedded techniques are three broad categories of FS methods used to identify the most relevant and informative subset of features from a high-dimensional dataset. Each approach follows a distinct strategy to evaluate and select features, and the choice of method depends on the specific characteristics of the data and the ML algorithm being employed. Subsequently, we present a concise overview of the operational principles underlying each technique in the following points.

1) *Filter Techniques:* Filter techniques involve the independent evaluation of each feature based on some statistical or ranking criterion. These methods do not consider the ML algorithm used for the final model. Instead, they rank or score features individually and select the top-ranked ones. Filter techniques are computationally efficient and can be applied as a preprocessing step before running any specific ML algorithm.
2) *Wrapper Techniques:* Wrapper techniques assess the quality of feature subsets by using the ML algorithm's performance as a criterion. These methods create subsets of features, train a model on each subset, and evaluate its performance using a chosen evaluation metric. They are computationally more intensive compared to filter techniques since they involve training multiple models for different feature subsets.
3) *Embedded Techniques:* Embedded techniques incorporate FS into the model training process itself. These methods combine FS with the algorithm's learning process, exploiting the inherent capabilities of the learning algorithm to identify important features during training. As a result, FS is seamlessly integrated into the model building process, leading to more efficient and accurate models.

The FS problem, known to be $\mathcal{NP}$-hard, is increasingly tackled using Metaheuristic Algorithms (MAs) [7], [13], [14] instead of exact methods due to several compelling reasons. One key factor is the exponential increase in the number of possible feature subsets with the growing dimensionality of data. Exact methods typically suffer from combinatorial explosion, making them computationally infeasible for large-scale datasets. By contrast, MAs excel at efficiently exploring complex search spaces, providing near-optimal solutions within a reasonable time frame. Their ability to strike a balance between exploration and exploitation [15], [16], [17] allows them to effectively navigate through vast feature subsets and discover promising combinations that yield improved model performance. Moreover, MAs are inherently adaptive, making them suitable for a wide range of optimization problems, including FS, without relying on domain-specific knowledge. As a result, the use of MAs has become a preferred approach in addressing the FS problem, offering researchers a practical and scalable solution to enhance the accuracy, efficiency, and interpretability of ML models.

The pivotal role of FS in the ML process is evident in the seven-step framework, playing a crucial part in refining prediction accuracy. Thus, numerous scholars have dedicated extensive efforts to this phase, as evidenced by various research works. For instance, the study referenced in [18] addresses cancer classification, employing the kernel Shapley value rooted in cooperative game theory for feature extraction from high-dimensional gene expression data. Another notable work, referenced as [19], focuses on cancer prediction and combines spider monkey optimization with cuckoo search algorithm for hybridized feature selection. Additionally, [20] and [21] contribute valuable insights into FS across diverse ML classification tasks.

In the context of our research, we have harnessed the power of a cutting-edge metaheuristic algorithm, known as the Great Wall Construction Algorithm [22], to address the $\mathcal{NP}$-hard FS problem. This algorithm has garnered considerable attention for its exceptional performance across a wide spectrum of challenges, including both constrained and unconstrained benchmark problems. To further bolster its capabilities, we have taken the initiative to augment the fundamental version of this algorithm by introducing several key enhancements. These additions are strategically designed to amplify its prowess in exploring solution spaces, exploiting promising regions, and adeptly steering clear of local optimums, all of which are critical attributes for effective problem-solving. The enhanced algorithm underwent a comprehensive evaluation by being juxtaposed with ten influential metaheuristic algorithms commonly employed in solving feature selection problems. This comparative study encompassed key metrics such as classification accuracy and fitness value. The results unequivocally demonstrate the superior performance of the proposed enhanced algorithm, surpassing the effectiveness of the other metaheuristics across these evaluative criteria. The following three points summarize the main improvements added to the Great Wall Construction Algorithm:

- We employed an efficient opposition-based learning technique [23] to enrich our approach. This technique enhanced exploration and diversification through the generation of opposite or complementary solutions, facilitated escape from local optimums by offering alternative starting points or directions in the search space, and expedited convergence, enhancing the speed of our metaheuristic algorithm.
- We incorporated Gaussian mutation [24] into our approach to bolster local search capabilities and prevent entrapment in local optimums.
- We used the step function to discretize continuous values into a binary range, as it offers a straightforward and easily implementable method.

The paper is structured into six distinct sections, each contributing to a comprehensive understanding of our research. Section II provides an overview of the current state-of-the-art metaheuristic-based approaches designed to address the FS problem, shedding light on the latest advancements in this field. In Section III, we delve into the fundamental concepts and methodologies underpinning the development of our solution, establishing the theoretical groundwork for our approach. Section IV is dedicated to presenting our proposed solution in detail, elucidating the various steps involved and discussing their significance in tackling the FS problem. The experimental aspect is addressed in Section V, where we present the results of our empirical study and conduct a comparative investigation to evaluate the performance of our solution. Finally, in Section VI, we conclude by summarizing our primary contributions and offering insights into potential future directions for this research.

II. RELATED WORK

Several survey papers have been published to investigate and review studies addressing the FS problem [7], [13]. In this section, we present a comprehensive overview of metaheuristic-based FS methodologies that have been published recently. Our emphasis lies in elucidating the introduced algorithms, the transfer functions, the classifier and the metrics employed for evaluating their efficacy, and the diverse advantages and disadvantages of each approach. By illuminating these facets, we aim to provide a good understanding of the evolving landscape of FS techniques and their practical implementation across a spectrum of datasets. In the comprehensive landscape of FS algorithms, a multitude of innovative approaches have been explored to address the challenges posed by high-dimensional datasets. The algorithms will be categorized into two approaches for FS, specifically binary and hybrid metaheuristic methods.

The algorithm outlined in [25] employs the binary bat algorithm for FS problem resolution, incorporating S and V shape transfer functions. It utilizes the support vector machine classifier, yielding an accuracy of 98.25%. While excelling with large datasets, this algorithm experiences a slower convergence time. In [26], the binary grasshopper optimization algorithm is utilized to address the FS problem, integrating S and V shape transfer functions. It incorporates the k-nearest neighbours classifier, achieving an accuracy of 97.9%. This algorithm boasts a swift convergence time and effective FS, but its performance is constrained in high-dimensional datasets. The algorithm in [27] employs the binary grey wolf optimizer for FS problem-solving, utilizing S and V shape transfer functions with the k-nearest neighbours classifier, resulting in an accuracy of 84.20%. While demonstrating rapid convergence and effective FS, it may become entangled in local optimums. In [28], the binary firefly algorithm is applied for FS, incorporating an aggregation function and k-nearest neighbours, naive Bayes, and linear discriminant analysis classifiers, achieving an accuracy of 97.78%. This algorithm exhibits fast convergence and robust FS, but it may face challenges with local optimums. Furthermore, [29] utilizes binary particle swarm optimization for FS, incorporating the sigmoid transfer function and the decision tree classifier, achieving an accuracy of 98.17%. While excelling with small datasets, it encounters limitations with larger datasets and potential entrapment in local optimums. The algorithm in [30] employs S-shaped and V-shaped gaining–sharing knowledge-based algorithms for FS problem-solving, utilizing S and V shape transfer functions. It integrates the k-nearest neighbours classifier, resulting in an accuracy of 99.6%. This algorithm performs well with high-dimensional datasets and adaptive parameter tuning, although additional parameter tuning may be necessary. In [31], the binary sine-cosine algorithm is utilized for FS problem resolution, employing S and V shape transfer functions with the k-nearest neighbours classifier, achieving an accuracy of 98.23%. It proves efficient for high-dimensional problems but may require fine-tuning. The algorithm in [32] uses the binary Giza pyramids construction algorithm for FS, incorporating S and V shape transfer functions with the k-nearest neighbours classifier, achieving an accuracy of 98.75%. This algorithm demonstrates swift convergence and strong performance with large datasets, yet it may not be suitable for smaller datasets. For [33], the binary ant lion algorithm is employed for FS problem-solving, using S and V shape transfer functions with the k-nearest neighbours classifier, resulting in an accuracy of 96.37%. While effectively handling high dimensionality or non-linearity, it may suffer from slow convergence and potential entrapment in local optimums, requiring significant computational resources for optimal performance. The work described in [34] applies the binary salp swarm algorithm for FS, utilizing S and V shape transfer functions with the k-nearest neighbours classifier, achieving an accuracy of 95.26%. While adept at addressing problems with complex search spaces or multiple objectives, this algorithm can be sensitive to parameter choices and may demand substantial computational resources for optimal performance. The algorithm outlined in [35] utilizes the binary cuckoo search algorithm for FS problem resolution, employing the sigmoid transfer function. It incorporates the optimum-path forest classifier, achieving an accuracy of 97.33%. While effectively managing problems with multimodal search spaces or noisy objective functions, it may be sensitive to parameter choices and susceptible to local optimums. Finally, the binary equilibrium optimizer in [36] is employed for FS problem-solving, utilizing S and V shape transfer functions with the k-nearest neighbours classifier, achieving an accuracy of 97.01%. This algorithm has demonstrated effectiveness in finding global optimums, performing well with a relatively small population size. However, it may be sensitive to parameter choices and could require a larger population size for optimal performance.

On the other hand, the algorithm introduced in [37] employs the binary chaotic bat algorithm to address the FS problem, utilizing V shape transfer functions. It incorporates random forest and k-nearest neighbours classifiers, achieving an accuracy of 96.97%. This algorithm adeptly manages challenges posed by intricate search spaces or multiple objectives. The integration of chaotic dynamics enhances its search capacity, preventing entrapment in local optimums. Nonetheless, sensitivity to the choice of chaotic function and a potential necessity for a sizable population size for optimal performance are noteworthy considerations. In [38], a similar approach is taken with the utilization of the binary chaotic dragonfly algorithm, incorporating chaotic transfer functions and the k-nearest neighbours classifier, resulting in an accuracy of 96.72%. While effectively handling complex search spaces or multiple objectives, this algorithm also displays sensitivity to the chosen chaotic function. The binary chaotic vortex algorithm, detailed in [39], leverages chaotic transfer functions and the k-nearest neighbours classifier, achieving an accuracy of 97.45%. Performance relies heavily on parameter settings, such as population size and maximum iteration number, necessitating careful tuning. However, computational expenses may arise, particularly for large datasets, due to multiple fitness function evaluations. In [25], the binary chaotic black hole algorithm integrates chaotic transfer functions and the k-nearest neighbours classifier, boasting an accuracy of 98.33%. Although promising for FS, its performance is contingent on parameter settings and specific applications. The binary chaotic moth–flame optimization algorithm, outlined in [40], applies chaotic transfer functions and the k-nearest neighbours classifier, yielding an accuracy of 96.62%. While effective in handling complex search spaces or multiple objectives, sensitivity to the chaotic function, potential slow convergence, and susceptibility to local optimums are potential drawbacks. The work in [41] introduces the fractional chaotic order marine predator algorithm, utilizing the k-nearest neighbours classifier with an accuracy of 97.13%. This promising FS method incorporates fractional calculus to enhance exploration and exploitation abilities, but computational expenses and potential reliance on a large population size are considerations. The island-based genetic algorithm in [42] employs support vector machine, k-nearest neighbours, decision tree, and multilayer perceptron classifiers, achieving an accuracy of 93.51%. Combining global and local search techniques enhances its search capability, but computational expenses are a concern. The optimizer described in [43] introduces the quantum whale optimization algorithm, utilizing the k-nearest neighbours, linear discriminant classifier, support vector machine, and decision tree classifiers, achieving an accuracy of 98.75%. Quantum-inspired operators enhance its search capability, but sensitivity to parameters and potential need for a large number of iterations are noted. Lastly, the approach in [44] combines the technique for order of preference by similarity to ideal solution with the binary JAYA algorithm, incorporating time-varying transfer functions. Using the Gaussian Naïve Bayes classifier, it attains an accuracy of 98.08%. While a hybrid algorithm effectively handling multiple objectives, it may require a substantial number of iterations and pose computational expenses for large-scale problems. This survey of diverse FS algorithms highlights their unique strengths and limitations, offering a rich spectrum of choices for researchers addressing the complexities of FS in various domains.

In the realm of FS, it is essential to recognize that achieving perfection remains elusive. Despite the proposal of numerous commendable solutions and their exceptional performance, the field continually calls for enhancements. This reality aligns with the principle articulated in the No-Free-Lunch theorem [45], emphasizing that no universally superior solution exists. Therefore, the door remains open for the exploration and development of new algorithms and solutions to address the ever-evolving challenges of the FS problem. In this vein, several promising avenues for further investigation emerge, including the exploration of algorithms such as the remora optimization algorithm [46] and the dynamic Harris Hawks optimization with a mutation mechanism [47]. These avenues promise to contribute valuable insights and advancements to the ongoing quest for optimizing FS methodologies.

III. BACKGROUND

In this section, we introduce the various concepts employed in the proposed methodology for addressing the FS problem. First, in Sections III-A, III-B, and III-C, we explain the working principles of the concepts related to MAs. Then, in Sections III-D and III-E, we describe the ML algorithm and metrics used to evaluate the performance. Finally, in Section III-F, we give the mathematical formulation of the FS optimization problem.

A. GREAT WALL CONSTRUCTION ALGORITHM

The Great Wall Construction Algorithm (GWCA) represents a novel metaheuristic optimizer introduced by Ziyu Guan and his colleagues [22]. Its draws inspiration from the historical competition and elimination mechanisms observed among workers during the construction of the ancient Great Wall. The GWCA optimizer incorporates these principles into its optimization strategy. Besides, the algorithm prioritizes performance-driven methodologies over metaphorical aspects, leveraging the competitive spirit of the workforce that contributed to the Great Wall's construction. With this unique approach, the GWCA algorithm aims to efficiently tackle complex optimization problems while emulating the effectiveness and resource management exhibited during the historical construction process. Table 1 summarizes the parameters utilized in the definition of the GWCA algorithm. In the following sections, we describe the phases of the GWCA optimizer.

1) INITIALIZATION

Equation 1 is employed to initialize the individuals in the first population, where the parameter λ governs the growth rate of the logistic map (set to 4), and the parameter α is a uniformly distributed random number within the range [0, 1] (excluding the values 0.25, 0.5, 0.75, and 1).

$$
\begin{aligned}
X_{i,j}^{(0)} &= \varphi_{i,j} \times \left(\mathrm{UB}_j - \mathrm{LB}_j\right) + \mathrm{LB}_j \\
\varphi_{i,j} &= \begin{cases} \alpha, & i = 1 \\ \lambda \varphi_{i-1,j}\left(1 - \varphi_{i-1,j}\right), & 1 < i \leq N \end{cases} \\
& i \in \{1, \ldots, N\} \text{ and } j \in \{1, \ldots, D\}
\end{aligned} \tag{1}
$$

2) EXPLOITATION

Equation 2 is employed to exploit the search space during the swarming process, where the parameter k is a uniformly distributed random number sampled from a uniform distribution over the set {0, 1}, and the parameter ϵ is an infinitely small number set to 2.22E-16.

$$
\begin{aligned}
X_{i,j}^{(t+1)} &= \alpha_1 \times \upsilon \times X_{i,j}^{(t)} + R_{i,j}^{(t)} + X_{\mathrm{b},j}^{(t)} \\
\upsilon &= \left(\frac{T \times \mathrm{TL}}{m} - g \times \frac{H(t)}{\sin(\theta)}\right) \times C(t) \times \mathbb{G}(t, P, Q) \\
H(t) &= 1 - \frac{t}{T_{\max}} \\
C(t) &= \log\left((C_{\max} - C_{\min}) \times \frac{T_{\max} - t}{T_{\max}} + C_{\min}\right) \\
R_{i,j}^{(t)} &= (-1)^k \times \alpha_2 \times \left(X_{\mathrm{b},j}^{(t)} - X_{i,j}^{(t)}\right) \\
\mathrm{TL} &= 1 - \frac{t}{T_{\max}} + \epsilon
\end{aligned} \tag{2}
$$

3) EXPLORATION

Equation 3 is employed to explore the search space during the swarming process, where the parameter ϵ is an infinitely small number set to 2.22E-16.

$$
\begin{aligned}
X_{i,j}^{(t+1)} &= X_{i,j}^{(t)} + \alpha_3 \times T_1 + \alpha_4 \times \upsilon \times \mathrm{sign}(T_2) \times T_3 \\
T_1 &= X_{\mathrm{b},j}^{(t)} - X_{i,j}^{(t)} \\
T_2 &= f\left(X_{\mathrm{n}\langle i\rangle}^{(t)}\right) - f\left(X_i^{(t)}\right) \\
T_3 &= X_{\mathrm{n}\langle i\rangle,j}^{(t)} - X_{i,j}^{(t)} \\
\upsilon &= m \times g \times \frac{H(t)}{\sin(\theta)} \times C(t) \times \mathbb{G}(t, P, Q) \\
H(t) &= 1 - \frac{t}{T_{\max}} + \epsilon \\
C(t) &= \log\left((C_{\max} - C_{\min}) \times \frac{T_{\max} - t}{T_{\max}} + C_{\min}\right) \\
\mathrm{sign}(x) &= \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}
\end{aligned} \tag{3}
$$

4) BALANCE BETWEEN EXPLOITATION AND EXPLORATION

Equation 4 is employed to bias the search towards better solutions, promoting convergence towards the optimal or near-optimal solutions in the search space, and overcome the issue of getting trapped in local optimums during the optimization process.

$$
\begin{aligned}
X_{i,j}^{(t+1)} &= X_{i,j}^{(t)} + 2 \times \alpha_5 \times T_1 + T_2 \times \mathbb{G}(t, P, Q) \\
T_1 &= X_{\mathrm{b},j}^{(t)} - X_{i,j}^{(t)} \\
T_2 &= M_{i,j} - X_{i,j}^{(t)}
\end{aligned} \tag{4}
$$

5) SELECTION

Algorithm 1 is used to determine which individuals from the current population are more likely to be chosen to appear in the next generation (i.e., eliminate the worst solutions). The worst solution are replaced with new ones generated using Equation 5. It is worth mentioning that the coefficients $r_1, \ldots, r_D$ are uniformly distributed random numbers within the range [0, 1].

$$
\begin{aligned}
X &= [r_1 \times T_1, \ldots, r_D \times T_D] \\
&\begin{cases} T_1 = (\mathrm{UB}_1 - \mathrm{LB}_1) + \mathrm{LB}_1 \\ \vdots \\ T_D = (\mathrm{UB}_D - \mathrm{LB}_D) + \mathrm{LB}_D \end{cases}
\end{aligned} \tag{5}
$$

6) GWCA'S PSEUDOCODE AND TIME COMPLEXITY

This section provides a comprehensive overview of the GWCA, focusing on both its pseudocode representation and its time complexity. The time complexity of a function evaluation is $O(D)$, and the time complexity of the swarming behaviour is $O(T_{\max} \times N \times D)$. Since the function evaluation step is included into the swarming loops, it means that the

TABLE 1. The parameters used in the GWCA.

Parameter	Signification
N	The population size
D	The dimensionality of the search space
LB	A D-dimensional vector representing the lower boundaries of the search space
UB	A D-dimensional vector representing the upper boundaries of the search space
$T_{\max}$	The maximum number of iterations
t	The current iteration
$X_{i,j}^{(t)}$	The component j of the individual i at iteration t
M_i	The memory of an agent i used to save its best location found so far
$X_{\mathrm{b},j}^{(t)}$	The component j of the best solution at iteration t
$X_{\mathrm{n}\langle i\rangle,j}^{(t)}$	The jth component of the nearest individual to agent i at iteration t
$f(.)$	The objective function to be minimized
$\alpha_1, \ldots, \alpha_5$	Uniformly distributed random numbers within the range $[0, 1]$
T	The force produced by the tool (thrust)
m	The weight of the rock
g	The gravitational acceleration
θ	The angle between the worker's position on the slope and the horizon (random within the range $[0, 80]$)
$\mathbb{G}(t, P, Q)$	The gamma distribution's probability density function – P controls the shape of the distribution ($P > 0$), and Q scales the distribution ($Q > 0$)
$C_{\min}$ and $C_{\max}$	Two constant numbers controlling the step size

Algorithm 1 The Selection Mechanism

Input: ρ: The percentage of individuals to be eliminated.
Input: $P = \{X_1, \ldots, X_N\}$: The population of individuals.
Output: $P = \{X_1, \ldots, X_N\}$: The updated population of individuals.

$k \leftarrow 1$;
while $k \leq \lceil \rho \times N \rceil$ **do**
    $P \leftarrow P - \left\{ \underset{i \in \{1,\ldots,|P|\}}{\operatorname{argmax}} \{f(X_i)\} \right\}$;
    $k \leftarrow k + 1$;
end
while $|P| < N$ **do**
    Generate a candidate solution X using Equation 5;
    $P \leftarrow P \cup \{X\}$;
end

Algorithm 2 Pseudocode of the GWCA

Input: Initialize the parameters of the GWCA.
Input: $P = \{X_1, \ldots, X_N\}$: The population of individuals.
Input: $M = \{M_1, \ldots, M_N\}$: The memory of individuals.
Output: X^*: The best solution.

for $i \leftarrow 1$ **to** N **do**
    for $j \leftarrow 1$ **to** D **do**
        $X_{i,j}^{(0)}$ is initialized using Equation 1;
        $M_{i,j} \leftarrow X_{i,j}$;
    end
end
for $t \leftarrow 1$ **to** $T_{\max}$ **do**
    for $i \leftarrow 1$ **to** N **do**
        Generate a random integer number $I \in \{1, 2, 3\}$;
        if $I = 1$ **then**
            for $j \leftarrow 1$ **to** D **do**
                $X_{i,j}^{(t)}$ is updated using Equation 2;
            end
        else if $I = 2$ **then**
            for $j \leftarrow 1$ **to** D **do**
                $X_{i,j}^{(t)}$ is updated using Equation 3;
            end
        else
            for $j \leftarrow 1$ **to** D **do**
                $X_{i,j}^{(t)}$ is updated using Equation 4;
            end
        for $j \leftarrow 1$ **to** D **do**
            $X_{i,j}^{(t)}$ is updated using Algorithm 3;
        end
        M_i is updated using Algorithm 4;
    end
    Replace undesired individuals using Algorithm 1;
end
$X^* \leftarrow \underset{i \in \{1,\ldots,N\}}{\operatorname{argmin}} \left\{ f\left(X_i^{(t)}\right) \right\}$;

time complexity of the GWCA is $O(n^4)$. The pseudocode depicted in Algorithm 2 describes the different steps of the GWCA.

B. OPPOSITION-BASED LEARNING

Opposition-Based Learning (OBL) [48] stands as an emerging notion within the field of MAs, drawing inspiration from the contrasting dynamics observed among different entities. The inception of the opposition concept in 2005 marked a significant milestone, garnering substantial attention from researchers over the subsequent decade. Diverse algorithms in the field of soft computing, including optimization techniques, reinforcement learning, artificial neural networks, and fuzzy systems, have embraced the principles of OBL to enhance and elevate their operational efficiency. At the core of OBL lies the foundational idea of concurrently examining the current solution and its contrasting counterpart

Algorithm 3 The Boundary Checker

Input: $X_{i,j}^{(t)}$: The solution to be checked.
Input: LB: The vector of lower boundaries.
Input: UB: The vector of upper boundaries.
Output: $X_{i,j}^{(t)}$: The checked solution.

if $X_{i,j}^{(t)} < LB_j$ **then**
  $X_{i,j}^{(t)} \leftarrow \mathrm{LB}_j$;
end
if $X_{i,j}^{(t)} > UB_j$ **then**
  $X_{i,j}^{(t)} \leftarrow \mathrm{UB}_j$
end

Algorithm 4 The Memory Updating Process

Input: $X_i^{(t)}$: The current solution.
Input: M_i: The memory to be updated.
Output: M_i: The updated memory.

if $\left(f\left(X_i^{(t)}\right) < f(M_i)\right)$ **then**
  for $j \leftarrow 1$ **to** D **do**
    $M_{i,j} \leftarrow X_{i,j}^{(t)}$;
  end
end

to achieve efficient problem-solving [49]. In simpler terms, when an optimization algorithm aims to discover the best possible outcome for an objective function, the incorporation of both a candidate solution and its opposite can be proven advantageous, thereby augmenting the algorithm's overall effectiveness.

Starting from January 2005, over 400 academic works have been disseminated pertaining to the concept of OBL [48]. These research contributions have found their home within various platforms including conferences, journals, and books, all situated within the domains of soft computing. Within this compilation, approximately 60% manifest as journal papers, 38% materialize as conference papers, while the remaining 2% comprise books or theses.

Definition 1: Let $X = (x_1, \ldots, x_D)$ be a candidate solution in the search space, where $x_j \in (\mathrm{LB}_j, \mathrm{UB}_j)$ and $j \in \{1, \ldots, D\}$. The opposite candidate solution of X is denoted by $\breve{X}$ and is computed by Equation 6 [49].

$$\breve{X} = \mathrm{LB} + \mathrm{UB} - X \tag{6}$$

Since the introduction of the initial OBL concept, a series of works have emerged. In this context, we delve into a straightforward yet highly efficient OBL approach, as detailed in the publication [23]. This technique serves as a cornerstone within our proposed algorithm, specifically designed to address the FS problem. In the following section, we describe the working principle of the OBL technique described in [23]. This approach employs a pair of algorithms, namely Algorithms 5 and 6, to calculate contrasting solutions. The goal is to minimize the waste of function evaluations. The choice between these algorithms depends on the diversity of the current population. When the diversity, computed using Equation 7, surpasses a predefined threshold, Algorithm 5 is executed. Conversely, if it falls below the threshold, Algorithm 6 is employed. On the one hand, Algorithm 5 has demonstrated its ability to accelerate the convergence speed of MAs by fully leveraging opposing information. On the other hand, Algorithm 6 has been shown to enhance the diversity of MAs by partially incorporating opposing information. Table 2 summarizes the parameters utilized in the definition of Algorithms 5 and 6, and Equations 7 to 15.

$$\mathrm{normDiv} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D}\sqrt{\frac{1}{D}\left(\frac{X_{i,j}-\bar{X}_j}{\mathrm{UB}_j-\mathrm{LB}_j}\right)^2} \tag{7}$$

$$\bar{X} = \frac{1}{N}(X_1 + \ldots + X_N)$$
$$\breve{X}_{i,j} = \mathbb{B}(\alpha,\beta) \times \left(\mathrm{UB}_j - \mathrm{LB}_j\right) + \mathrm{LB}_j$$
$$i \in \{1, \ldots, N\} \text{ and } j \in \{1, \ldots, D\}$$

$$\mathbb{B}(\alpha,\beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\,dt \tag{8}$$

$$\alpha = \begin{cases} \mathrm{spread} \times \mathrm{peak}, & \mathrm{mode} < 0.5 \\ \mathrm{spread}, & \text{otherwise} \end{cases} \tag{9}$$

$$\beta = \begin{cases} \mathrm{spread}, & \mathrm{mode} < 0.5 \\ \mathrm{spread} \times \mathrm{peak}, & \text{otherwise} \end{cases} \tag{10}$$

$$\mathrm{spread} = \left(\frac{1}{\sqrt{\mathrm{normDiv}}}\right)^{1+\mathbb{N}(0,0.5)} \tag{11}$$

$$\mathrm{peak} = \begin{cases} \dfrac{(\mathrm{spread}-2)\times\mathrm{mode}+1}{\mathrm{spread}\times(1-\mathrm{mode})}, & \mathrm{mode} < 0.5 \\ \dfrac{2-\mathrm{spread}}{\mathrm{spread}} + \dfrac{\mathrm{spread}-1}{\mathrm{spread}\times\mathrm{mode}}, & \text{otherwise} \end{cases} \tag{12}$$

$$\mathrm{mode} = \frac{\mathrm{UB}_j - X_{i,j}}{\mathrm{UB}_j - \mathrm{LB}_j} \tag{13}$$

$$\mathrm{spread} = 0.1 \times \sqrt{\mathrm{normDiv}} + 0.9 \tag{14}$$

$$\mathrm{mode} = \frac{X_{i,j} - \mathrm{LB}_j}{\mathrm{UB}_j - \mathrm{LB}_j} \tag{15}$$

C. GAUSSIAN MUTATION

The Gaussian Mutation (GM) [24] operator introduces random perturbations to the current solution by sampling from a Gaussian distribution. The GM is commonly utilized to make slight adjustments to the values of the solution variables. Algorithm 8 depicts the pseudocode of the GM operator.

Two parameters govern the extent of mutation: the mutation rate (γ) and the mutation strength (δ). The former determines the probability of mutation for each solution

TABLE 2. The parameters used in Algorithms 5 and 6.

Parameter	Signification
N	The population size
D	The dimensionality of the search space
LB	A D-dimensional vector representing the lower boundaries of the search space
UB	A D-dimensional vector representing the upper boundaries of the search space
$X_{i,j}$	The component j of the candidate solution i
$\check{X}_{i,j}$	The component j of the opposite candidate solution i
$\mathbb{B}(\alpha, \beta)$	The beta function – the values of α and β determine the shape of the beta function's graph ($\alpha > 0, \beta > 0$)
$\mathbb{N}(0, 0.5)$	The Gaussian distribution of mean 0 and standard deviation 0.5

variable (i.e., increasing the mutation rate raises the chances of mutation taking place); while the latter determines the magnitude of perturbations applied to the solution variables (i.e., higher mutation strength results in more significant variations across the search space). The normal distribution is referred to as $\mathbb{N}(\mu, \sigma)$, where μ and σ are its mean and its standard deviation, respectively.

D. K-NEAREST NEIGHBORS

The K-Nearest Neighbors (KNN) [50] is a simple yet effective ML algorithm used for classification and regression tasks. The working principle of KNN revolves around the idea of proximity-based prediction. Given a new data point, the algorithm identifies its k closest neighbours within the training dataset based on a chosen distance metric, often Euclidean distance. For classification, the majority class among these k neighbours is assigned to the new data point. In regression, the algorithm calculates the average or weighted average of the target values from the k neighbours to predict a continuous value. The key assumption is that similar data points are likely to have similar outcomes. The value of k, the number of neighbours, is a crucial parameter that influences the algorithm's performance and generalization. Smaller k values result in more flexible, potentially noisy predictions, while larger k values lead to smoother but potentially oversimplified predictions. KNN is easy to understand and implement, making it a valuable tool for various tasks, but its efficiency can decrease with larger datasets due to the need to calculate distances for each query point.

E. EVALUATION METRICS

In classification tasks in ML, various evaluation metrics are used to assess the performance of model's predictions [51]. These metrics provide insights into how well the model is classifying different classes and help quantify its strengths and weaknesses. These metrics provide a comprehensive view of a classifier's performance from different angles. The choice of metric depends on the specific characteristics of the problem, the class distribution, and the goals of the application. It is often recommended to consider multiple metrics to get a well-rounded assessment of a model's performance. Some common evaluation metrics for classification tasks are given in the following sections.

1) CONFUSION MATRIX

A confusion matrix provides a detailed breakdown of True Positives (TP) – the model identifies a positive case correctly, True Negatives (TN) – the model correctly identifies a negative case, False Positives (FP) – the model predicts a positive outcome when it should have predicted a negative outcome, and False Negatives (FN) – the model fails to predict a positive outcome when it should have, which are essential for calculating the subsequent metrics.

2) ACCURACY

Accuracy is the ratio of correctly predicted instances to the total number of instances in the dataset. While easy to understand, accuracy might not be suitable for imbalanced datasets where one class dominates the others. Its mathematical expression is given by Equation 16.

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{16}$$

3) PRECISION

Precision measures the ratio of correctly predicted positive observations to the total predicted positives. It focuses on the correctness of positive predictions and helps in scenarios where false positives are costly. Its mathematical representation is defined by Equation 17.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{17}$$

4) RECALL

Recall calculates the ratio of correctly predicted positive observations to the actual positives. It is useful when the emphasis is on minimizing false negatives. Equation 18 provides its mathematical formulation.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{18}$$

5) F1-SCORE

The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall, which can be valuable when you need to consider both false positives and false negatives. Its mathematical formula is expressed in Equation 19.

$$\text{F1-score} = \frac{2 \times (\text{precision} \times \text{recall})}{\text{precision} + \text{recall}} \tag{19}$$

Algorithm 5 The First OBL Technique

Input: $P = \{X_1, \ldots, X_N\}$: The population of individuals.
Input: LB: Lower boundaries of the search space.
Input: UB: Upper boundaries of the search space.
Input: N: The population size.
Input: D: The dimensionality of the search space.
Input: $f(.)$: The objective function to be minimized.
Output: $P = \{X_1, \ldots, X_N\}$: The population of individuals.

Define a zero matrix $A = (a_{ij})_{1 \le i \le N, 1 \le j \le D}$;
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** D **do**
    $a_{ij} \leftarrow X_{i,j}$;
  end
end
Compute the covariance matrix C of A;
Compute the matrix V whose columns are the eigenvectors of C;
$U \leftarrow \emptyset$;
Compute normDiv using Equation 7;
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** D **do**
    if $rand(0, 1) \le 0.5$ **then**
      Compute mode using Equation 13;
      Compute spread using Equation 11;
    end
    else
      Compute mode using Equation 15;
      Compute spread using Equation 14;
    end
    Compute peak using Equation 12;
    Compute α using Equation 9;
    Compute β using Equation 10;
    Compute $\check{X}_{i,j}$ using Equation 8;
  end
  $X_i' \leftarrow \left(V^T \times X_i^T\right)^T$;
  $\check{X}_i' \leftarrow \left(V^T \times \check{X}_i^T\right)^T$;
  Compute U_1 using Algorithm 7 (X_i', $\check{X}_i'$, $C_r = 0.1$);
  Compute U_2 using Algorithm 7 (X_i', $\check{X}_i'$, $C_r = 0.9$);
  $\left(V \times U_1^T\right)^T$ is updated using Algorithm 3;
  $\left(V \times U_2^T\right)^T$ is updated using Algorithm 3;
  $U \leftarrow U \cup \left\{\left(V \times U_1^T\right)^T, \left(V \times U_2^T\right)^T\right\}$;
end
$U \leftarrow U \cup P$;
$P \leftarrow \emptyset$;
while $|P| < N$ **do**
  $B \leftarrow \left\{\underset{i \in \{1,\ldots,|U|\}}{\text{argmin}} \{f(U_i)\}\right\}$;
  $U \leftarrow U - \{B\}$;
  $P \leftarrow P \cup \{B\}$;
end

Algorithm 6 The Second OBL Technique.

Input: $P = \{X_1, \ldots, X_N\}$: The population of individuals.
Input: LB: Lower boundaries of the search space.
Input: UB: Upper boundaries of the search space.
Input: N: The population size.
Input: D: The dimensionality of the search space.
Input: $f(.)$: The objective function to be minimized.
Output: $P = \{X_1, \ldots, X_N\}$: The population of individuals.

Define a zero matrix $A = (a_{ij})_{1 \le i \le N, 1 \le j \le D}$;
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** D **do**
    $a_{ij} \leftarrow X_{i,j}$;
  end
end
Compute the covariance matrix C of A;
Compute the matrix V whose columns are the eigenvectors of C;
Compute the median m of the series: $f(X_1), \ldots, f(X_N)$;
$U \leftarrow \emptyset$;
Compute normDiv using Equation 7;
for $i \leftarrow$ **to** N **do**
  if $f(X_i) \ge m$ **then**
    for $j \leftarrow 1$ **to** D **do**
      if $rand(0, 1) \le 0.5$ **then**
        Compute mode using Equation 13;
        Compute spread using Equation 11;
      end
      else
        Compute mode using Equation 15;
        Compute spread using Equation 14;
      end
      Compute peak using Equation 12;
      Compute α using Equation 9;
      Compute β using Equation 10;
      Compute $\check{X}_{i,j}$ using Equation 8;
    end
    $X_i' \leftarrow \left(V^T \times X_i^T\right)^T$;
    $\check{X}_i' \leftarrow \left(V^T \times \check{X}_i^T\right)^T$;
    Compute U_1 using Algorithm 7 (X_i', $\check{X}_i'$, $C_r = 0.1$);
    Compute U_2 using Algorithm 7 (X_i', $\check{X}_i'$, $C_r = 0.9$);
    $U_1 \leftarrow \left(V \times U_1^T\right)^T$;
    $U_2 \leftarrow \left(V \times U_2^T\right)^T$;
    U_1 is updated using Algorithm 3;
    U_2 is updated using Algorithm 3;
    $U \leftarrow U \cup \left\{\text{argmin}\{f(U_1), f(U_2), f(X_i)\}\right\}$;
  end
  else
    $U \leftarrow U \cup \{X_i\}$;
  end
end
$P \leftarrow U$;

6) CLASSIFICATION ERROR RATE

The classification error rate in ML is a fundamental performance metric that quantifies the proportion of incorrectly classified instances in a dataset, comparing the number of misclassified data points to the total number of instances. It serves as a straightforward indicator of a classification model's accuracy, with lower error rates indicating better performance and higher rates reflecting lower accuracy. However, the classification error rate has limitations, such as not distinguishing between different

Algorithm 7 The Multiple Exponential Recombination Algorithm

Input: X_1: The first parent solution.
Input: X_2: The second parent solution.
Input: D: The dimensionality of the search space.
Input: C_r: Mutation probability.
Input: T: Length of exchanged segments ($T = 2$).
Output: X_3: The offspring solution.

$E_m \leftarrow T \times C_r$;
$E_s \leftarrow T \times (1 - C_r)$;
Generate a random integer number $n \in \{1, \ldots, D\}$;
$k \leftarrow 1$;
flag $\leftarrow 1$;
while $k \leq D$ **do**
  if *flag* $= 1$ **then**
    while $k \leq D$ ***and*** $rand(0, 1) \leq \frac{E_m}{E_m+1}$ **do**
      $j \leftarrow 0$;
      if $n \leq D$ **then**
        $j \leftarrow n$;
      end
      else
        $j \leftarrow n - D$;
      end
      $X_{3,j} \leftarrow X_{2,j}$;
      $k \leftarrow k + 1$;
      $n \leftarrow n + 1$;
    end
    flag $\leftarrow 0$;
  end
  else
    while $k \leq D$ ***and*** $rand(0, 1) \leq \frac{E_s}{E_s+1}$ **do**
      $j \leftarrow 0$;
      if $n \leq D$ **then**
        $j \leftarrow n$;
      end
      else
        $j \leftarrow n - D$;
      end
      $X_{3,j} \leftarrow X_{1,j}$;
      $k \leftarrow k + 1$;
      $n \leftarrow n + 1$;
    end
    flag $\leftarrow 1$;
  end
end

types of errors (e.g., false positives and false negatives) and not accounting for class imbalances. As a result, it is often used in combination with other metrics to provide a more comprehensive assessment of a model's classification capabilities. Equation 20 gives its mathematical expression.

$$\text{CER} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{20}$$

Algorithm 8 The Gaussian Mutation Process

Input: X_i: The solution to be mutated.
Input: γ: The mutation rate.
Input: δ: The mutation strength.
Input: μ: The Gaussian distribution's mean.
Input: σ: The Gaussian distribution's standard deviation.
Output: X_i: The mutated solution.

for $j \leftarrow 1$ **to** D **do**
  if $(rand(0, 1) < \gamma)$ **then**
    $X_{i,j} \leftarrow X_{i,j} + \mathbb{N}(\mu, \sigma) \times \delta$;
    $X_{i,j}$ is updated using Algorithm 3;
  end
end

F. MATHEMATICAL FORMULATION OF FS PROBLEMS

The FS problem is about selecting a subset of features from a larger set while aiming to achieve a certain optimization goal, such as improving model performance or reducing complexity. The mathematical formulation can vary based on the specific objective and constraints of the problem. In the following, we give a mathematical formulation for the FS problem.

In the FS problem, we assume a dataset with N instances and D features: $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where $\mathbf{x}_i$, is a D-dimensional feature vector, and a response variable y. The goal is to select a subset of features from the original D features that maximizes or minimizes a certain objective function. The objective function can be defined based on various criteria, such as model performance (e.g., accuracy, F1-score), model complexity (e.g., number of selected features), or other domain-specific considerations. The general FS problem, used in this work, is mathematically formulated using Equations 21, 22 and 23.

Minimize:

$$\alpha \times \text{CER} + (1 - \alpha) \times \frac{|R|}{|D|} \tag{21}$$

where CER represents the classification error rate computed using Equation 20, α is a random number sampled from the uniform distribution, $|R|$ denotes the number of selected features, and $|D|$ refers to the total number of features.

Subject to:

$$\sum_{j=1}^{D} x_{i,j} \leq K \ , \ i \in \{1, \ldots, N\} \tag{22}$$

where K is the maximum number of selected features (if there are constraints on limiting the number of selected features).

With:

$$x_{i,j} \in \{0, 1\} \ , \ i \in \{1, \ldots, N\} \ , \ j \in \{1, \ldots, D\} \tag{23}$$

where $x_{i,j}$ is a binary decision variable that represents whether feature j is selected for instance i. If $x_{i,j} = 1$, the feature is selected; if $x_{i,j} = 0$, the feature is not selected.

IV. PROPOSED ALGORITHM

In this section, we present and elucidate the algorithm that embodies our proposed methodology for addressing the FS problem. Algorithm 9 provides a comprehensive overview of the distinct steps involved in its formulation. This algorithm serves as a vital roadmap for understanding the intricacies of our method and its practical implementation. Through the following discussion, we aim to provide a clear and detailed account of our approach, allowing for a deeper insight into the methodology's inner workings.

In our algorithm, we have employed the transfer function defined by Equation 24 to facilitate the mapping of candidate solutions from a continuous space to a binary space. This transfer function is called the step transfer function, and it plays a pivotal role in transforming the real-valued outputs into binary decisions, allowing us to effectively navigate the discrete nature of the FS problem and make meaningful decisions based on the continuous input data.

$$Y_{i,j}^{(t)} = \begin{cases} 0, & \text{if } X_{i,j}^{(t)} \leq 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{24}$$

The various stages of the proposed algorithm can be elucidated as follows:

- Initially, the algorithm commences by initializing and inputting the values of controlling parameters, which are detailed in Table 4.
- Lines 1 to 6 of the algorithm involve the initialization of the initial population of candidate solutions through the utilization of chaotic maps. This approach aims to promote diversity within the population, facilitating exploration across a broad spectrum of values and potentially covering diverse regions within the solution space.
- Between lines 7 and 37, the algorithm carries out the swarming process in an interactive manner. The cessation of this process can be determined by various stopping criteria, such as a predefined maximum number of generations, a set maximum for function evaluations, or a threshold for objective function values, among other possibilities.
- Within the algorithmic framework, specifically in lines 8 to 11, the execution of either Algorithm 5 or 6 is determined based on the population diversity's value. Algorithm 5 is designed to expedite the convergence speed of the proposed algorithm by fully exploiting opposing information, while Algorithm 6 aims to amplify the diversity of the algorithm by selectively incorporating opposing information. Subsequently, in the span of lines 12 to 34, the swarming behavior of the GWCA is considered, orchestrating movement within the search space. It is worth pointing out that the transition between the preceding phases is conducted randomly, contributing an element of stochasticity to the algorithmic process.
- In line 35, Gaussian mutation is executed to enhance the diversity of solutions and avoid getting trapped in local optimums.
- It is worth highlighting that lines 10, 30, 33, and 36 are used to save the best solution encountered by the various agents during the swarming process. This process plays a crucial role in guiding the algorithm toward better solutions over successive iterations.
- Finally, at line 38, the best solution found so far is returned, representing the set of selected features.

We scrutinize the time complexity of the proposed algorithm (Algorithm 9), observing that Algorithm 1 has a time complexity of $O(n)$, Algorithm 3 has a time complexity of $O(1)$, Algorithm 4 has a time complexity of $O(n)$, Algorithm 8 has a time complexity of $O(n)$, and Algorithm 10 has a time complexity of $O(n^4)$. Based on the elementary time complexities discussed earlier, we conclude that the time complexity of the improved version of the GWCA is $O(n^5)$.

V. EXPERIMENTAL STUDY AND DISCUSSION

Within this section, our focus centers on the rigorous evaluation of the proposed algorithm's efficacy in addressing the FS problem. Section V-A lays the foundation by providing a comprehensive overview of both the datasets employed in our comparative study and the parameter settings configured for optimal performance of our proposed optimizer. Subsequently, Section V-B meticulously delineates the diverse algorithms included in the comparative study, shedding light on their respective parameter configurations. The culmination of this evaluation is encapsulated in Section V-C, where a detailed presentation of the comparative study unfolds. This section systematically delves into the selected criteria, offering a nuanced exploration of the obtained numerical results.

A. USED DATASETS AND PARAMETERS SETTING

Table 3 provides a comprehensive overview of the key characteristics of the 22 datasets employed in our comparative study. Access to the datasets can be obtained through the link https://archive.ics.uci.edu/datasets. The datasets are categorized into three groups – small, medium, and large – based on the number of features, with datasets having fewer than 20 features classified as small, those with 21 to 100 features as medium, and datasets with more than 100 features categorized as large. To evaluate the impact of selected feature subsets, each dataset underwent division into training, testing, and validation sets using the cross-validation method. Subsequently, the KNN classifier was applied to calculate the objective function as defined by Equation 21 (the number of neighbours to use is 5).

In Table 4, a comprehensive overview of the parameter settings for the various variables employed in our proposed algorithm dedicated to addressing the FS problem is presented.

B. BENCHMARK ALGORITHMS

In evaluating the efficacy of the suggested algorithm, a comprehensive performance analysis was conducted through a comparative study with ten prominent state-of-the-art

Algorithm 9 Pseudocode of the Proposed Algorithm

Input: $P = \{X_1, \ldots, X_N\}$: The population of individuals.
Input: $M = \{M_1, \ldots, M_N\}$: The memory of individuals.
Input: LB: Lower boundaries of the search space.
Input: UB: Upper boundaries of the search space.
Input: N: The population size.
Input: D: The dimensionality of the search space.
Input: JR: The jumping rate.
Input: Initialize the parameters of the GWCA.
Output: X^*: The best solution.

for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** D **do**
    $X_{i,j}^{(0)}$ is initialized using Equation 1;
    $M_{i,j} \leftarrow X_{i,j}$;
  end
end
while *Termination Condition is not Satisfied* **do**
  if $rand(0, 1) \leq JR$ **then**
    Update the population P using Algorithm 10;
    Update individuals' memory using Algorithm 4;
  end
  else
    for $i \leftarrow 1$ **to** N **do**
      Generate a random integer number $I \in \{1, 2, 3\}$;
      if $I = 1$ **then**
        for $j \leftarrow 1$ **to** D **do**
          $X_{i,j}^{(t)}$ is updated using Equation 2;
        end
      else if $I = 2$ **then**
        for $j \leftarrow 1$ **to** D **do**
          $X_{i,j}^{(t)}$ is updated using Equation 3;
        end
      else
        for $j \leftarrow 1$ **to** D **do**
          $X_{i,j}^{(t)}$ is updated using Equation 4;
        end
      for $j \leftarrow 1$ **to** D **do**
        $X_{i,j}^{(t)}$ is updated using Algorithm 3;
      end
      M_i is updated using Algorithm 4;
    end
    Replace undesired individuals using Algorithm 1;
    Update individuals' memory using Algorithm 4;
  end
  Update the population P using Algorithm 8;
  Update individuals' memory using Algorithm 4;
end
$X^* \leftarrow \underset{i \in \{1,\ldots,N\}}{\operatorname{argmin}} \left\{ f\left(X_i^{(t)}\right) \right\}$;

methodologies. Seven of the algorithms are novel methods introduced between 2020 and 2023, while the remaining three are classical approaches, including particle swarm optimization, genetic algorithms, and differential evolution. The selected algorithms were scrutinized in depth, and their respective parameter configurations have been succinctly outlined in Table 5 for clarity and reference. It is worth

Algorithm 10 The Opposition-Based Learning

Input: $P = \{X_1, \ldots, X_N\}$: The population of individuals.
Input: DT: The diversity threshold.
Output: $P = \{X_1, \ldots, X_N\}$: The updated population of individuals.

Compute the population's diversity normDiv using Equation 7;
if $normDiv > DT$ **then**
  Update the individuals within P using Algorithm 5;
end
else
  Update the individuals within P using Algorithm 6;
end

pointing out that these parameters have been extracted from the original published papers.

1) Binary Arithmetic Optimization Algorithm (BAOA) [52].
2) Binary Sand Cat Swarm Optimization algorithm (BSCSO) [53].
3) Improved Bald Eagle Search algorithm (IBES) [54].
4) Chaotic Binary Reptile Search Algorithm (CBRSA) [55].
5) Chaotic Vortex Search Algorithm (CVSA) [39].
6) Chaotic Gaining Sharing Knowledge-based optimization algorithm (CBi-GSK) [56].
7) Chaotic Atom Search Optimization (CASO) [57].
8) Particle Swarm Optimization (PSO) [58].
9) Differential Evolution (DE) [59].
10) Genetic Algorithm (GA) [60].

All experiments were replicated on a laptop with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 16.0 GB RAM, using Matlab R2020b; and statistical analyses were conducted using IBM SPSS Statistics. The number of candidates solutions, the number of iterations and the number of runs were set to 10, 100 and 30, respectively, for all the algorithms.

C. NUMERICAL RESULTS AND DISCUSSION

To assess and compare the performance of the various algorithms employed in the comparative study, we utilized a set of evaluation metrics. These metrics were chosen to provide a comprehensive analysis of algorithmic effectiveness and efficiency, enabling a thorough examination of their performance across different criteria.

1) Average of Classification Accuracy (ACA): It furnishes the average of accuracy values calculated through Equation 16 over the specified number of runs.
2) Average of Fitness Values (AFV): It presents the mean of fitness values derived from Equation 21 across the designated number of runs.
3) Minimum of Fitness Values (MiFV): It gives the minimum of fitness values calculated from Equation 21 across the designated number of runs.

TABLE 3. The description of datasets used in the comparative study.

ID	Name	Number of features	Number of instances	Number of classes
	Small datasets			
d_1	Tic-Tac-Toe Endgame	9	958	2
d_2	Breast Cancer Wisconsin (Original)	10	699	2
d_3	Statlog (Heart)	13	270	2
d_4	Wine	13	178	3
d_5	Congressional Voting Records	16	435	2
d_6	Zoo	16	101	7
d_7	Lymphography	18	148	4
d_8	Hepatitis	19	155	2
d_9	German Credit Dataset Analysis	20	1000	2
	Medium datasets			
d_{10}	Waveform	21	5000	3
d_{11}	Breast Cancer Wisconsin (Diagnostic)	30	569	2
d_{12}	Ionosphere	34	351	2
d_{13}	Dermatology	34	366	6
d_{14}	Soybean (Small)	35	47	4
d_{15}	Lung Cancer	56	32	3
d_{16}	Connectionist Bench (Sonar, Mines vs. Rocks)	60	208	2
d_{17}	Hill-Valley	100	1212	2
	Large datasets			
d_{18}	Musk Version 1	166	476	2
d_{19}	Semeion Handwritten Digit	265	1593	2
d_{20}	Malware Executable Detection	531	373	2
d_{21}	Parkinson's Disease Classification	754	756	2
d_{22}	CNAE-9	856	1080	3

TABLE 4. The parameters used in the proposed algorithm.

Parameter	Value
N	30
D	The number of features present in the considered dataset.
LB	**0**
UB	**1**
$T_{\max}$	100
T	8.3
m	3
g	9.8
P	9
Q	6
$C_{\min}$	e^2
$C_{\max}$	e^3
ρ	0.25
JR	0.05
DT	10^{-6}
α	Random in the range $[0, 1]$
γ	0.05
δ	1
μ	0
σ	0.5

4) Maximum of Fitness Values (MaFV): It provides the maximums of fitness values computed from Equation 21 across the designated number of runs.
5) Average of Selected Features (ASF): It offers the average number of selected features over the specified runs.
6) Average of Completion Time (ACT): It provides the mean of completion times over the designated number of runs. The time is given in seconds.

To evaluate the impact of reducing the number of features on the performance of the preceding metrics, we additionally calculated various average values when considering the inclusion of all available features. The obtained numerical results are summarized in Table 6. Furthermore, Table 7 provides a comprehensive summary of the values for various metrics achieved through the proposed algorithm.

TABLE 5. The parameters' values of the algorithms used for the comparative study.

Parameter	Value
BAOA	
Min_{MOA}	0.2
Min_{MOA}	1
μ	0.49
α	5
BSCSO	
r_G	$[2, 0]$
R	$[-2r_G, 2r_G]$
IBES	
a	10
R	1.5
CBRSA	
α	0.1
β	0.005
CVSA	
μ_0	0.5
CBi-GSK	
ρ	0.1
K	10
$N_p(\leq 20)$	50
$N_p(\geq 20)$	100
CASO	
α	50
β	0.2
PSO	
c_1	2
c_2	2
w	$[0.2, 0.9]$
DE	
F	0.7
Cr	0.8
GA	
P_c	0.7
P_m	0.1

Tables 10, 11, 12, 13, 14, and 15 showcase diverse metric values derived from the algorithms under consideration for the comparative study. Each table serves as input for the Friedman and the Wilcoxon signed ranks tests, facilitating

TABLE 6. The values of various metrics when considering all features.

ID	ACA	AFV	MiFV	MaFV	ASF	ACT
d_1	0.8211	0.1872	0.1872	0.1872	9.0000	0.1369
d_2	0.5362	0.4691	0.4691	0.4691	10.0000	0.0095
d_3	0.6296	0.3767	0.3767	0.3767	13.0000	0.0208
d_4	0.6471	0.3594	0.3594	0.3594	13.0000	0.0068
d_5	0.9302	0.0791	0.0791	0.0791	16.0000	0.0065
d_6	0.9000	0.1090	0.1090	0.1090	16.0000	0.0109
d_7	0.5000	0.5050	0.5050	0.5050	18.0000	0.0065
d_8	0.6000	0.4060	0.4060	0.4060	19.0000	0.0065
d_9	0.6300	0.3763	0.3763	0.3763	20.0000	0.0073
d_{10}	0.8260	0.1823	0.1823	0.1823	21.0000	0.0176
d_{11}	0.9821	0.0277	0.0277	0.0277	30.0000	0.0067
d_{12}	0.9143	0.0949	0.0949	0.0949	34.0000	0.0068
d_{13}	0.8889	0.1200	0.1200	0.1200	34.0000	0.0067
d_{14}	1.0000	0.0100	0.0100	0.0100	35.0000	0.0065
d_{15}	0.6667	0.3400	0.3400	0.3400	56.0000	0.0063
d_{16}	0.8500	0.1585	0.1585	0.1585	60.0000	0.0069
d_{17}	0.5702	0.4355	0.4355	0.4355	100.0000	0.0108
d_{18}	0.8511	0.1574	0.1574	0.1574	166.0000	0.0076
d_{19}	0.9811	0.0287	0.0287	0.0287	265.0000	0.0253
d_{20}	0.9459	0.0635	0.0635	0.0635	531.0000	0.0104
d_{21}	0.7600	0.2476	0.2476	0.2476	754.0000	0.0216
d_{22}	0.9352	0.0742	0.0742	0.0742	856.0000	0.0367

TABLE 7. The values of various metrics obtained by the proposed algorithm.

ID	ACA	AFV	MiFV	MaFV	ASF	ACT
d_1	0.8842	0.1202	0.1202	0.1202	5.0000	28.7156
d_2	0.9913	0.0122	0.0060	0.0163	3.6000	25.1408
d_3	0.9185	0.0842	0.0764	0.1131	4.6000	23.3756
d_4	1.0000	0.0015	0.0015	0.0015	2.0000	22.6313
d_5	1.0000	0.0019	0.0019	0.0019	3.0000	24.1523
d_6	0.9600	0.0421	0.0025	0.1015	4.0000	22.7894
d_7	1.0000	0.0017	0.0017	0.0017	3.0000	22.5171
d_8	1.0000	0.0005	0.0005	0.0005	1.0000	22.6201
d_9	0.8560	0.1457	0.1327	0.1515	6.2000	27.0323
d_{10}	0.8584	0.1456	0.1404	0.1537	11.4000	56.0294
d_{11}	1.0000	0.0007	0.0007	0.0007	2.0000	23.4479
d_{12}	1.0000	0.0006	0.0006	0.0009	2.2000	22.3402
d_{13}	1.0000	0.0012	0.0012	0.0012	4.0000	22.0240
d_{14}	1.0000	0.0003	0.0003	0.0003	1.0000	21.5273
d_{15}	1.0000	0.0004	0.0004	0.0004	2.0000	20.7130
d_{16}	1.0000	0.0007	0.0007	0.0008	4.2000	21.7060
d_{17}	0.7256	0.2724	0.2545	0.2951	7.4000	29.1522
d_{18}	1.0000	0.0004	0.0004	0.0004	6.6000	24.5433
d_{19}	1.0000	0.0004	0.0002	0.0006	9.8000	58.7090
d_{20}	1.0000	0.0000	0.0000	0.0000	1.2000	29.0329
d_{21}	0.8640	0.1347	0.1189	0.1717	7.0000	53.4145
d_{22}	0.9648	0.0362	0.0192	0.0468	115.0000	89.7991

the examination of subtle differences among the algorithms concerning the specified evaluation criteria. It is worth noting that the best values for each metric are presented in bold font. The initial observation reveals that the accuracy values achieved by the proposed algorithm surpass those attained when utilizing all features, indicating a substantial positive impact on performance due to the reduction in the number of features.

For each table, we have considered small, medium, and large datasets separately to perform the Friedman and Kruskal-Wallis tests and compute the mean ranks. First, the Friedman test is a non-parametric statistical test used to detect differences in treatments across multiple related groups. It is often employed when the data violate the assumptions of normal distribution or when the data are measured on an ordinal scale. Second, the Kruskal-Wallis test is a non-parametric statistical test used to determine whether there are any statistically significant differences between the medians of three or more independent (unrelated) groups. It is an extension of the Wilcoxon rank-sum test (Mann-Whitney U test) for two groups to multiple groups. We consider the null hypothesis (H_0), representing the statement of no effect or no difference, and the alternative hypothesis (H_1), representing the statement that contradicts the null hypothesis (i.e., suggesting the presence of an effect or difference). The significance level, denoted as α, is the probability of rejecting the null hypothesis when it is actually true. In our study, α is set to 0.05. Tables 8 and 9 summarize the p-values associated with the Friedman and Kruskal-Wallis tests, respectively, indicating the likelihood of obtaining the observed differences among the groups due to random chance. In other words, if the p-value is less than the chosen significance level (i.e., 0.05), the null hypothesis is rejected. From Table 8, it is observed that all the p-values are less than 0.05, which suggests to reject the null hypothesis. From Table 9, it is observed that all the p-values are less than 0.05, which suggests to reject the null hypothesis, except for large datasets suggesting to retain the null hypothesis. On the other side, mean ranks refer to the average ranks assigned to each treatment or group across different levels of the independent variable: Mean Rank 1 and Mean Rank 2 are computed using the Friedman and Kruskal-Wallis tests, respectively. They provide a summary measure of the average performance or rank order of each treatment under varying conditions. Higher mean ranks, signifying smaller values, indicate superior performance or a higher position in the rank order. As observed in Tables 10, 11, 12, and 13, BGWCA consistently secures the first place, reflecting the best values in terms of accuracy and fitness, as computed using Equations 16 and 21. However, according to Table 15, BGWCA attains medium ranks in the majority of cases (i.e., either the fourth or the sixth place out of the 11 algorithms). This behavior arises from the conflicting relationship between optimality and computing time.

Figures 1 and 2 represent box-and-whisker plots for small, medium and large datasets for all the optimizers. The box-and-whisker plots shown in Figure 1 reveal distinct patterns in the distribution of the ACA values. For the left figure, the majority of data is clustered around zero, with a small interquartile range and whiskers extending to a maximum value of 0.1440. Two non-zero values, 0.0087 and 0.0400, could be considered potential outliers. The middle figure shows a concentration of values around zero, with a small interquartile range and whiskers extending to a maximum value of 0.2744. The right figure consists mainly of zero values, with a larger interquartile range and whiskers extending from the minimum to the maximum values of 0 and 0.1360, respectively. The presence of a non-zero value, 0.0352, could be considered an outlier in the context of this figure. The box-and-whisker plots shown in Figure 2 divulge insights into the distribution of the AFV values. For the left figure, the majority of the data is concentrated

TABLE 8. Summary of the Friedman test results.

	ACA	AFV	MiFV	MaFV	ASF	ACT
Small datasets	7.4397e-10	6.2937e-10	7.8794e-09	7.8722e-10	3.4652e-11	8.4144e-15
Medium datasets	1.4949e-09	6.0471e-10	4.7091e-07	1.9101e-09	5.4992e-11	1.3981e-12
Large datasets	0.0029	0.0019	0.0083	7.3662e-04	2.1260e-06	7.2312e-07

TABLE 9. Summary of the Kruskal-Wallis test results.

	ACA	AFV	MiFV	MaFV	ASF	ACT
Small datasets	3.2282e-09	1.6740e-09	0.0012	4.3553e-10	9.7190e-11	7.2472e-16
Medium datasets	1.2725e-04	3.2282e-05	0.0056	1.7810e-06	1.6376e-08	2.4180e-11
Large datasets	0.1650	0.0974	0.6199	0.0478	6.9207e-04	1.9551e-05

TABLE 10. The ACA values for all algorithms.

ID	BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA	ALL
						Small datasets						
d_1	**0.8842**	0.7095	0.3979	0.7474	0.8084	0.6842	0.6779	0.7958	0.7495	0.8632	0.7368	0.8211
d_2	0.9913	0.9594	0.5507	0.5362	0.7942	0.8203	0.7623	0.9884	0.8319	**1.0000**	0.7623	0.5362
d_3	0.9185	0.8444	0.5778	0.7630	0.5630	0.7111	0.6667	0.8889	0.8074	**0.9630**	0.6593	0.6296
d_4	**1.0000**	0.8471	0.3412	0.6235	0.7765	0.8941	0.7882	0.9294	0.9059	**1.0000**	0.8000	0.6471
d_5	**1.0000**	0.9628	0.5395	0.7070	0.9442	0.8744	0.9395	0.9907	0.9953	0.9767	0.9302	0.9302
d_6	0.9600	0.8600	0.3600	0.5200	0.8000	0.8600	0.8600	0.9800	**1.0000**	**1.0000**	0.9800	0.9000
d_7	**1.0000**	0.7143	0.6286	0.5714	0.7857	0.6143	0.8286	0.9286	0.6286	0.9714	0.7143	0.5000
d_8	**1.0000**	0.9200	0.5600	0.4000	0.6667	0.7733	0.8000	0.7600	0.7467	0.9733	0.7333	0.6000
d_9	**0.8560**	0.7200	0.3920	0.6940	0.6600	0.6520	0.6360	0.8220	0.6540	0.8420	0.7620	0.6300
Mean Ranks 1	**1.7222**	5.3889	10.3889	9.0000	7.3333	7.4444	7.3889	3.5000	5.1111	1.8889	6.8333	
	1	5	11	10	7	9	8	3	4	2	6	
Mean Ranks 2	**16.2222**	43.1667	92.6111	80.0000	57.9444	56.7778	55.1111	30.5000	46.7778	17.1667	53.7222	
	1	4	11	10	9	8	7	3	5	2	6	
						Medium datasets						
d_{10}	**0.8584**	0.7488	0.2316	0.6168	0.8064	0.7320	0.7832	0.8152	0.8240	0.8416	0.7680	0.8260
d_{11}	**1.0000**	0.9321	0.7929	0.6679	0.9250	0.8643	0.9107	0.9536	0.9143	0.9893	0.8500	0.9821
d_{12}	**1.0000**	0.9371	0.6914	0.8914	0.8743	0.8914	0.8229	0.9429	0.8971	0.9829	0.8571	0.9143
d_{13}	**1.0000**	0.8944	0.4444	0.7889	0.8444	0.8278	0.9444	0.9833	0.9500	**1.0000**	0.8889	0.8889
d_{14}	**1.0000**	**1.0000**	0.2500	0.9000	**1.0000**	0.9000	0.9000	**1.0000**	0.9500	**1.0000**	0.8500	**1.0000**
d_{15}	**1.0000**	0.8000	0.4000	0.6000	0.9333	0.7333	0.4667	0.8000	0.8667	**1.0000**	0.4000	0.6667
d_{16}	**1.0000**	0.8900	0.7500	0.6500	0.7600	0.7800	0.8900	0.8900	0.8900	0.9800	0.7700	0.8500
d_{17}	**0.7256**	0.5421	0.4479	0.5074	0.5355	0.5554	0.4314	0.5190	0.6215	0.6347	0.5702	0.5702
Mean Ranks 1	**1.3750**	5.1250	10.5625	9.1875	6.0000	7.4375	7.5625	4.2500	4.4375	2.0000	8.0625	
	1	5	11	10	6	7	8	3	4	2	9	
Mean Ranks 2	**18.6250**	39.5625	76.5625	60.6875	41.8125	52.4375	48.3125	35.0625	36.6875	22.8125	56.9375	
	1	5	11	10	6	8	7	3	4	2	9	
						Large datasets						
d_{18}	**1.0000**	0.8596	0.8383	0.8936	0.8255	0.8340	0.8340	0.9319	0.8553	0.9660	0.8723	0.8511
d_{19}	**1.0000**	0.9774	0.7572	0.9371	0.9849	0.9610	0.9623	0.9836	0.9899	0.9987	0.9736	0.9811
d_{20}	**1.0000**	0.9892	0.5784	0.8324	**1.0000**	0.9946	**1.0000**	0.9459	0.9730	**1.0000**	0.9838	0.9459
d_{21}	**0.8640**	0.7653	0.7760	0.6747	0.6773	0.6800	0.8533	0.7120	0.7733	0.8347	0.6933	0.7600
d_{22}	**0.9648**	0.6167	0.2407	0.5093	0.8019	0.6222	0.6889	0.8556	0.6667	0.8093	0.7093	0.9352
Mean Ranks 1	**1.3000**	6.6000	9.0000	9.0000	6.3000	8.1000	5.6000	5.2000	6.0000	2.5000	6.4000	
	1	8	11	11	6	9	4	3	5	2	7	
Mean Ranks 2	**10.9000**	28.8000	44.0000	36.8000	27.3000	31.9000	26.4000	25.6000	28.8000	19.3000	28.2000	
	1	7	11	10	5	9	4	3	8	2	6	

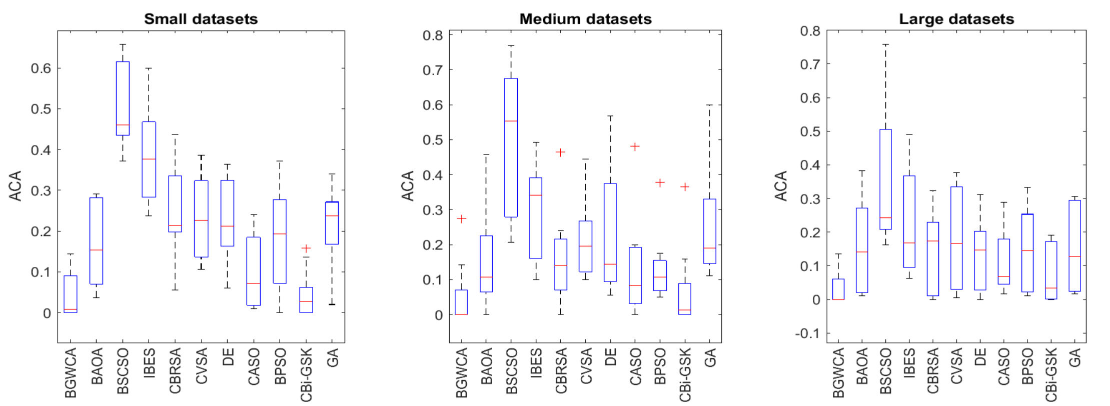


FIGURE 1. The box-and-whisker plot for all the optimizers over all the datasets for ACA values.

TABLE 11. The AFV values for all algorithms.

ID	BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA	ALL
						Small datasets						
d_1	**0.1202**	0.2907	40.2008	0.2561	0.1977	0.3177	0.3238	0.2071	0.2538	0.1410	0.2670	0.1872
d_2	0.0122	0.0430	40.0494	40.0667	0.2101	0.1829	0.2417	0.0165	0.1712	**0.0020**	0.2383	0.4691
d_3	0.0842	0.1555	20.2206	0.2404	0.4417	0.2906	0.3345	0.1143	0.1948	**0.0390**	0.3418	0.3767
d_4	**0.0015**	0.1536	60.0589	20.1789	0.2281	0.1094	0.2141	0.0728	0.0975	0.0023	0.2025	0.3594
d_5	**0.0019**	0.0392	20.2585	20.0978	0.0610	0.1283	0.0661	0.0125	0.0087	0.0236	0.0733	0.0791
d_6	0.0421	0.1412	40.2381	40.0841	0.2035	0.1422	0.1450	0.0236	0.0037	**0.0019**	0.0233	0.1090
d_7	**0.0017**	0.2843	0.3684	20.2320	0.2196	0.3863	0.1747	0.0742	0.3723	0.0297	0.2871	0.5050
d_8	**0.0005**	0.0810	20.2380	40.2019	0.3377	0.2283	0.2027	0.2418	0.2550	0.0286	0.2676	0.4060
d_9	**0.1457**	0.2798	40.2062	0.3075	0.3423	0.3485	0.3654	0.1810	0.3464	0.1594	0.2403	0.3763
Mean Ranks 1	**1.6667**	5.2222	10.4444	9.0000	7.2222	7.4444	7.5556	3.5556	5.2222	1.8889	6.7778	
	1	5	11	10	7	8	9	3	5	2	6	
Mean Ranks 2	**15.3889**	43.0000	92.4444	82.2222	57.6667	56.3333	55.0000	30.4444	46.7778	17.6111	53.1111	
	1	4	11	10	9	8	7	3	5	2	6	
						Medium datasets						
d_{10}	**0.1456**	0.2522	60.1670	20.1870	0.1995	0.2701	0.2203	0.1893	0.1803	0.1616	0.2349	0.1823
d_{11}	**0.0007**	0.0689	0.2055	20.1359	0.0824	0.1381	0.0939	0.0494	0.0901	0.0114	0.1537	0.0277
d_{12}	**0.0006**	0.0641	20.1078	0.1114	0.1320	0.1115	0.1808	0.0612	0.1064	0.0181	0.1460	0.0949
d_{13}	**0.0012**	0.1074	0.5506	0.2156	0.1607	0.1747	0.0599	0.0214	0.0542	0.0025	0.1152	0.1200
d_{14}	**0.0003**	0.0019	0.7428	0.1055	0.0065	0.1030	0.1041	0.0026	0.0545	0.0006	0.1525	0.0100
d_{15}	**0.0004**	0.2003	20.3961	0.4014	0.0724	0.2677	0.5333	0.2021	0.1373	0.0004	0.5985	0.3400
d_{16}	**0.0007**	0.1111	0.2479	20.1537	0.2431	0.2216	0.1140	0.1132	0.1141	0.0216	0.2320	0.1585
d_{17}	**0.2724**	0.4560	20.3487	0.4911	0.4653	0.4440	0.5681	0.4810	0.3794	0.3635	0.4304	0.4355
Mean Ranks 1	**1.0000**	4.8750	10.7500	9.2500	6.2500	7.3750	7.5000	4.3750	4.6250	2.0000	8.0000	
	1	5	11	10	6	7	8	3	4	2	9	
Mean Ranks 2	**17.3750**	38.7500	77.6250	64.8750	42.1250	52.0000	47.2500	35.2500	36.5000	22.0000	55.7500	
	1	5	11	10	6	8	7	3	4	2	9	
						Large datasets						
d_{18}	**0.0004**	0.1415	0.1604	0.1116	0.1797	0.1685	0.1694	0.0722	0.1481	0.0359	0.1315	0.1574
d_{19}	**0.0004**	0.0251	20.0425	0.0671	0.0222	0.0432	0.0423	0.0209	0.0148	0.0032	0.0309	0.0287
d_{20}	**0.0000**	0.0131	0.4175	0.1707	0.0074	0.0098	0.0049	0.0579	0.0316	0.0006	0.0209	0.0635
d_{21}	**0.1347**	0.2348	0.2218	0.3259	0.3273	0.3214	0.1502	0.2898	0.2292	0.1650	0.3083	0.2476
d_{22}	**0.0362**	0.3822	0.7519	0.4898	0.2040	0.3784	0.3129	0.1480	0.3350	0.1913	0.2928	0.0742
Mean Ranks 1	**1.0000**	6.6000	9.0000	8.8000	7.0000	8.0000	5.8000	5.0000	6.0000	2.4000	6.4000	
	1	7	11	10	8	9	4	3	5	2	6	
Mean Ranks 2	**10.0000**	28.4000	46.2000	36.2000	28.4000	31.4000	26.8000	25.2000	28.8000	19.0000	27.6000	
	1	6	11	10	7	9	4	3	8	2	5	

TABLE 12. The MiFV values for all algorithms.

ID	BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA	ALL
						Small datasets						
d_1	0.1202	0.2036	0.3346	0.2173	**0.1142**	0.2661	0.2754	0.1920	0.1663	0.1410	0.2080	0.1872
d_2	0.0060	0.0327	0.0584	0.0193	0.0347	0.0347	0.0357	0.0050	0.0183	**0.0020**	0.0460	0.4691
d_3	0.0764	0.1123	0.2574	0.1505	0.3369	0.1528	0.2597	0.0428	0.1115	**0.0390**	0.2979	0.3767
d_4	**0.0015**	**0.0015**	0.1188	0.0621	0.0621	**0.0015**	0.1211	0.0598	0.0031	0.0023	0.0621	0.3594
d_5	**0.0019**	0.0268	0.2539	0.0952	0.0268	0.0529	0.0299	**0.0019**	0.0031	0.0236	0.0044	0.0791
d_6	0.0025	0.1015	0.1993	0.1084	0.1040	0.0037	0.1052	0.0037	0.0025	**0.0019**	0.0031	0.1090
d_7	**0.0017**	0.1431	0.3541	0.2205	0.0779	0.2884	0.0757	0.0033	0.2149	**0.0017**	0.0757	0.5050
d_8	**0.0005**	0.0665	0.2645	0.2661	0.2687	0.2001	0.1357	0.0692	0.1352	0.0021	0.2012	0.4060
d_9	**0.1327**	0.2505	0.3173	0.2525	0.3253	0.2911	0.3223	0.1649	0.2698	0.1421	0.1827	0.3763
Mean Ranks 1	**1.9444**	5.2778	10.1111	8.0000	7.7778	7.2222	8.5000	3.5556	4.7222	2.2778	6.6111	
	1	5	11	9	8	7	10	3	4	2	6	
Mean Ranks 2	**24.3333**	47.6111	78.5556	62.1111	59.6667	55.5556	61.0556	36.0556	46.5556	25.9444	52.5556	
	1	5	11	10	8	7	9	3	4	2	6	
						Medium datasets						
d_{10}	**0.1404**	0.2008	0.3178	0.1834	0.1575	0.1972	0.1952	0.1442	0.1562	0.1483	0.1670	0.1823
d_{11}	**0.0007**	0.0540	0.1421	0.0617	0.0604	0.1111	0.0744	0.0384	0.0574	0.0010	0.0397	0.0277
d_{12}	**0.0006**	0.0015	0.0569	0.0884	0.0902	0.0625	0.1473	0.0330	0.0321	0.0012	0.1173	0.0949
d_{13}	**0.0012**	0.0576	0.3590	0.0872	0.0609	0.0328	0.0047	0.0044	0.0041	0.0021	0.0065	0.1200
d_{14}	**0.0003**	0.0014	0.4953	0.0031	0.0031	0.0023	0.0043	0.0014	0.0037	0.0006	0.0034	0.0100
d_{15}	**0.0004**	0.0018	0.3302	0.3305	0.0045	0.0036	0.3346	0.0045	0.0052	**0.0004**	0.3334	0.3400
d_{16}	**0.0007**	0.0515	0.1488	0.1020	0.1528	0.1510	0.0050	0.0535	0.0052	0.0013	0.1525	0.1585
d_{17}	**0.2545**	0.4117	0.4011	0.4603	0.4412	0.4208	0.5453	0.4469	0.3159	0.3451	0.4060	0.4355
Mean Ranks 1	**1.0625**	5.4375	8.7500	8.1875	7.6250	7.2500	8.5000	4.7500	4.8750	2.1875	7.3750	
	1	5	11	9	8	6	10	3	4	2	7	
Mean Ranks 2	**19.6250**	39.3125	68.7500	55.4375	49.6250	48.1250	52.5000	39.5000	39.6250	23.5000	53.5000	
	1	3	11	10	7	6	8	4	5	2	9	
						Large datasets						
d_{18}	**0.0004**	0.1076	0.1059	0.0701	0.1571	0.1315	0.1313	0.0473	0.0888	0.0231	0.1106	0.1574
d_{19}	**0.0002**	0.0210	0.0376	0.0407	0.0174	0.0300	0.0300	0.0167	0.0048	0.0015	0.0106	0.0287
d_{20}	**0.0000**	0.0020	0.0269	0.0003	0.0048	0.0037	0.0046	0.0577	0.0312	0.0003	0.0048	0.0635
d_{21}	**0.1189**	0.1479	0.1848	0.3049	0.3219	0.3202	0.1501	0.2555	0.2289	0.1334	0.3082	0.2476
d_{22}	**0.0192**	0.2960	0.5139	0.1815	0.1473	0.2340	0.2708	0.0692	0.2340	0.1581	0.2524	0.0742
Mean Ranks 1	**1.0000**	6.2000	8.2000	6.1000	7.7000	7.8000	7.4000	5.6000	6.2000	2.5000	7.3000	
	1	6	11	4	9	10	8	3	6	2	7	
Mean Ranks 2	**11.4000**	28.6000	34.0000	29.7000	31.5000	32.2000	30.4000	28.0000	30.2000	20.9000	31.1000	
	1	4	11	5	9	10	7	3	6	2	8	

TABLE 13. The MaFV values for all algorithms.

ID	BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA	ALL
Small datasets												
d_1	**0.1202**	0.3253	100.0000	0.2891	0.2869	0.3483	0.3483	0.2244	0.3055	0.1410	0.3193	0.1872
d_2	0.0163	0.0470	100.0000	100.0000	0.3247	0.3914	0.3810	0.0327	0.3800	**0.0020**	0.3627	0.4691
d_3	0.1131	0.1841	100.0000	0.3315	0.5226	0.4821	0.4821	0.1856	0.2995	**0.0390**	0.4079	0.3767
d_4	**0.0015**	0.3517	100.0000	100.0000	0.3594	0.2376	0.2958	0.1195	0.3571	0.0023	0.2414	0.3594
d_5	**0.0019**	0.0479	100.0000	100.0000	0.1425	0.2327	0.0965	0.0517	0.0286	0.0236	0.1649	0.0791
d_6	0.1015	0.1993	100.0000	100.0000	0.3014	0.4975	0.2036	0.1009	0.0056	**0.0019**	0.1009	0.1090
d_7	**0.0017**	0.4956	0.4254	100.0000	0.2929	0.4989	0.2183	0.1459	0.4994	0.0718	0.4994	0.5050
d_8	**0.0005**	0.1341	100.0000	100.0000	0.4060	0.2698	0.3368	0.3353	0.4657	0.0681	0.3347	0.4060
d_9	**0.1515**	0.3079	100.0000	0.4361	0.3525	0.3980	0.3896	0.2035	0.4198	0.1718	0.3203	0.3763
Mean Ranks 1	**1.6667**	5.1111	10.1667	9.3889	6.8889	7.6667	7.0000	3.7222	6.5000	**1.6667**	6.2222	
	1	3	10	9	6	8	7	2	5	1	4	
Mean Ranks 2	**13.2778**	41.2222	90.5556	81.3333	56.2222	63.0000	52.8889	29.8333	53.6111	14.9444	53.1111	
	1	4	11	10	8	9	5	3	7	2	6	
Medium datasets												
d_{10}	**0.1537**	0.2781	100.0000	100.0000	0.3009	0.3315	0.2523	0.2795	0.2002	0.1706	0.3573	0.1823
d_{11}	**0.0007**	0.0904	0.3362	100.0000	0.1284	0.1801	0.1284	0.0734	0.1131	0.0183	0.2165	0.0277
d_{12}	**0.0009**	0.1429	100.0000	0.1231	0.1771	0.1724	0.2307	0.0904	0.1747	0.0295	0.1744	0.0949
d_{13}	**0.0012**	0.1682	0.7428	0.6056	0.4160	0.3063	0.1688	0.0585	0.1697	0.0026	0.2794	0.1200
d_{14}	**0.0003**	0.0026	0.9903	0.2575	0.0100	0.4996	0.4993	0.0040	0.2526	0.0006	0.7468	0.0100
d_{15}	**0.0004**	0.6625	100.0000	0.6664	0.3366	0.6637	0.6659	0.3343	0.3361	0.0005	0.9955	0.3400
d_{16}	**0.0008**	0.1508	0.4952	100.0000	0.3507	0.3005	0.2532	0.2018	0.3512	0.0512	0.3020	0.1585
d_{17}	**0.2951**	0.4775	100.0000	0.5162	0.5040	0.4628	0.5952	0.5121	0.4132	0.3867	0.4717	0.4355
Mean Ranks 1	**1.0000**	4.5000	10.6875	8.9375	7.0625	7.0000	7.0625	4.2500	5.5000	2.0000	8.0000	
	1	4	11	10	8	6	8	3	5	2	9	
Mean Ranks 2	**12.7500**	35.7500	77.3750	67.5000	45.1875	52.8750	48.3125	33.6250	41.5000	18.6250	56.0000	
	1	4	11	10	6	8	7	3	5	2	9	
Large datasets												
d_{18}	**0.0004**	0.1704	0.2741	0.1706	0.2155	0.2152	0.2165	0.0891	0.2157	0.0654	0.1521	0.1574
d_{19}	**0.0006**	0.0338	100.0000	0.1309	0.0283	0.0560	0.0488	0.0298	0.0236	0.0082	0.0483	0.0287
d_{20}	**0.0000**	0.0292	0.6422	0.8295	0.0096	0.0308	0.0051	0.0581	0.0317	0.0008	0.0316	0.0635
d_{21}	0.1717	0.2664	0.2509	0.3454	0.3349	0.3230	**0.1503**	0.3216	0.2296	0.1860	0.3085	0.2476
d_{22}	**0.0468**	0.4701	0.8893	0.8710	0.2984	0.4265	0.4174	0.2617	0.4084	0.2141	0.3718	0.0742
Mean Ranks 1	**1.2000**	6.2000	9.6000	9.6000	6.0000	7.8000	5.8000	5.6000	6.0000	2.2000	6.0000	
	1	8	11	11	6	9	4	3	6	2	6	
Mean Ranks 2	**10.0000**	27.8000	47.0000	40.0000	26.4000	30.8000	25.8000	26.4000	27.4000	18.0000	28.4000	
	1	7	11	10	4	9	3	5	6	2	8	

TABLE 14. The ASF values for all algorithms.

ID	BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
Small datasets											
d_1	5.0000	2.8000	**0.6000**	5.4000	7.2000	4.6000	4.4000	4.4000	5.2000	5.0000	5.8000
d_2	3.6000	2.8000	**0.6000**	3.6000	6.4000	5.0000	6.4000	5.0000	4.8000	2.0000	3.0000
d_3	4.6000	2.0000	**0.8000**	7.4000	11.8000	6.0000	5.8000	5.6000	5.4000	3.0000	5.8000
d_4	2.0000	2.8000	**0.8000**	5.4000	8.8000	6.0000	5.8000	3.8000	5.6000	3.0000	5.8000
d_5	3.0000	3.8000	**1.0000**	9.2000	9.2000	6.4000	10.0000	5.2000	6.6000	**1.0000**	6.8000
d_6	4.0000	4.2000	**0.8000**	7.8000	8.8000	5.8000	10.2000	6.0000	6.0000	3.0000	5.6000
d_7	3.0000	2.6000	**1.2000**	10.2000	13.4000	8.0000	9.0000	6.2000	8.2000	2.6000	7.6000
d_8	1.0000	3.4000	**0.8000**	7.4000	14.6000	7.4000	9.0000	8.0000	8.0000	4.2000	6.8000
d_9	6.2000	5.2000	**0.6000**	9.2000	11.4000	8.0000	10.0000	9.6000	7.8000	6.0000	9.4000
Mean Ranks 1	3.7778	2.8333	**1.0556**	8.0556	10.6667	7.1111	9.0000	6.4444	7.0000	3.1667	6.8889
	4	2	1	9	11	8	10	5	7	3	6
Mean Ranks 2	29.5556	23.8333	**5.3333**	68.2778	86.8889	61.0556	73.3333	54.7222	60.5556	26.4444	60.0000
	4	2	1	9	11	8	10	5	7	3	6
Medium datasets											
d_{10}	11.4000	7.4000	**0.6000**	11.8000	16.4000	10.0000	12.0000	13.4000	12.8000	10.0000	11.0000
d_{11}	2.0000	5.2000	**1.4000**	15.2000	24.6000	11.2000	16.4000	10.4000	15.8000	2.4000	15.6000
d_{12}	2.2000	6.2000	**1.2000**	13.4000	25.6000	13.6000	18.4000	15.6000	15.4000	4.0000	15.4000
d_{13}	4.0000	9.8000	**2.2000**	22.4000	22.8000	14.4000	16.8000	16.6000	16.0000	8.4000	17.6000
d_{14}	**1.0000**	6.8000	**1.0000**	22.6000	22.6000	14.0000	17.8000	9.2000	17.4000	2.0000	14.0000
d_{15}	2.0000	12.8000	**0.8000**	30.0000	35.8000	21.0000	29.6000	22.8000	29.6000	2.2000	25.4000
d_{16}	4.2000	13.2000	**2.2000**	31.0000	33.2000	22.6000	30.6000	25.8000	31.2000	10.8000	25.6000
d_{17}	7.4000	27.4000	**1.4000**	34.2000	54.8000	38.6000	51.8000	48.2000	46.8000	18.8000	49.8000
Mean Ranks 1	2.4375	3.7500	**1.0625**	7.9375	10.9375	5.3750	8.9375	7.1250	8.1250	3.0625	7.2500
	2	4	1	8	11	5	10	6	9	3	7
Mean Ranks 2	15.5000	32.0000	**6.1875**	59.9375	71.4375	49.3750	62.2500	52.4375	59.6875	24.3125	56.3750
	2	4	1	9	11	5	10	6	8	3	7
Large datasets											
d_{18}	6.6000	41.4000	**5.6000**	103.8000	116.2000	70.0000	84.6000	79.8000	81.0000	36.4000	84.8000
d_{19}	9.8000	71.8000	**4.2000**	129.2000	192.8000	123.0000	130.2000	125.2000	129.0000	51.4000	124.8000
d_{20}	**1.2000**	129.0000	3.4000	253.0000	394.2000	234.8000	262.0000	234.0000	257.8000	29.4000	254.8000
d_{21}	7.0000	185.4000	**6.2000**	285.2000	591.8000	347.0000	379.0000	351.4000	360.6000	100.6000	354.8000
d_{22}	115.0000	234.2000	**17.0000**	341.8000	666.4000	379.6000	423.2000	427.2000	425.4000	211.4000	426.6000
Mean Ranks 1	1.8000	4.0000	**1.2000**	7.2000	11.0000	5.6000	9.0000	7.0000	8.2000	3.0000	8.0000
	2	4	1	7	11	5	10	6	9	3	8
Mean Ranks 2	8.8000	24.1000	**4.6000**	34.0000	42.6000	33.0000	37.0000	34.6000	36.1000	17.4000	35.8000
	2	4	1	6	11	5	10	7	9	3	8

TABLE 15. **The ACT values for all algorithms.**

ID	BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
Small datasets											
d_1	28.7156	38.0384	41.6302	118.2242	27.0558	**10.2790**	106.9746	52.4825	71.5778	14.6554	63.8113
d_2	25.1408	36.9417	24.9287	102.5099	17.3524	**8.2044**	86.9156	47.1840	51.9768	12.8589	44.1263
d_3	23.3756	36.3869	30.5004	91.5272	16.7273	**7.7271**	79.5666	44.7517	47.3552	12.2842	42.2124
d_4	22.6313	36.6425	21.3329	88.7779	14.9685	**7.3813**	77.1497	42.2566	46.1027	11.6944	40.5524
d_5	24.1523	38.1455	34.1675	95.6723	17.3903	**8.0237**	82.5427	46.8641	48.9759	11.6226	42.7269
d_6	22.7894	36.2016	28.9366	91.3360	15.1493	**7.3505**	75.9122	42.1123	48.0479	11.3167	33.8477
d_7	22.5171	37.2596	36.5782	94.9737	15.2269	**7.3926**	76.6874	42.2797	48.1238	11.8195	39.8507
d_8	22.6201	36.1458	31.6999	92.7664	15.1530	**7.3565**	75.3608	40.4018	46.7110	12.1871	40.7094
d_9	27.0323	43.2949	31.8707	106.6652	17.5505	**8.8556**	87.3392	47.5865	53.8650	13.9367	46.2095
Mean Ranks 1	4.2222	6.0000	4.8889	11.0000	3.0000	**1.0000**	10.0000	7.7778	9.0000	2.0000	7.1111
	4	6	5	11	3	1	10	8	9	2	7
Mean Ranks 2	32.4444	50.7778	41.4444	94.1111	24.1111	**5.0000**	86.8889	66.1111	74.4444	14.0000	60.6667
	4	6	5	11	3	1	10	8	9	2	7
Medium datasets											
d_{10}	56.0294	89.2570	46.8782	246.5173	41.3572	**19.9459**	216.2988	112.0510	127.1671	32.2413	104.1206
d_{11}	23.4479	41.5256	41.5889	101.5365	15.3877	**7.5935**	76.6837	41.9898	46.5929	12.6921	40.3735
d_{12}	22.3402	40.0553	32.6971	100.6279	17.0740	**7.3215**	75.0268	39.6506	47.1553	12.5177	39.3907
d_{13}	22.0240	40.6899	39.7109	99.4054	17.1199	**7.3724**	75.8701	39.7341	45.8135	12.1673	41.0414
d_{14}	21.5273	36.8123	35.1491	94.0244	15.7149	**6.8368**	70.7577	37.8871	42.4545	10.9314	37.0902
d_{15}	20.7130	37.7596	28.4173	96.2899	17.4421	**6.7944**	70.5405	36.9916	41.7841	10.6181	36.7558
d_{16}	21.7060	39.3899	38.4887	100.4231	17.4111	**7.1343**	74.1653	38.8706	43.8900	10.6436	41.8556
d_{17}	29.1522	47.5488	41.0740	142.6224	24.6003	**9.2286**	98.7750	51.6531	60.1765	12.7464	55.2281
Mean Ranks 1	4.1250	6.7500	5.1250	11.0000	3.0000	**1.0000**	10.0000	7.1250	9.0000	2.0000	6.8750
	4	6	5	11	3	1	10	8	9	2	7
Mean Ranks 2	30.5000	50.3750	43.2500	81.1250	23.1250	**6.1250**	74.2500	51.5000	63.6250	13.5000	52.1250
	4	6	5	11	3	1	10	7	9	2	8
Large datasets											
d_{18}	24.5433	41.4347	43.5527	115.6920	20.1663	**7.8200**	82.6446	43.0339	49.6717	11.8932	46.9283
d_{19}	58.7090	56.5792	48.1766	308.0451	51.1045	**17.1175**	199.0906	103.8325	121.9219	18.7708	110.8297
d_{20}	29.0329	41.9633	42.4373	140.5478	24.9949	**8.7386**	92.7163	48.0460	55.1904	11.8685	51.7729
d_{21}	53.4145	52.0373	49.1810	254.5666	53.1255	**16.3041**	176.7385	90.0803	104.7303	16.8776	96.5373
d_{22}	89.7991	74.9907	59.4617	438.3906	79.0751	**26.0981**	287.7076	150.7883	170.5160	30.6294	155.4842
Mean Ranks 1	5.2000	4.6000	4.4000	11.0000	4.0000	**1.0000**	10.0000	6.8000	9.0000	2.0000	8.0000
	6	5	4	11	3	1	10	7	9	2	8
Mean Ranks 2	24.2000	24.6000	22.8000	50.2000	21.6000	**5.4000**	45.8000	33.0000	38.0000	7.0000	35.4000
	5	6	4	11	3	1	10	7	9	2	8

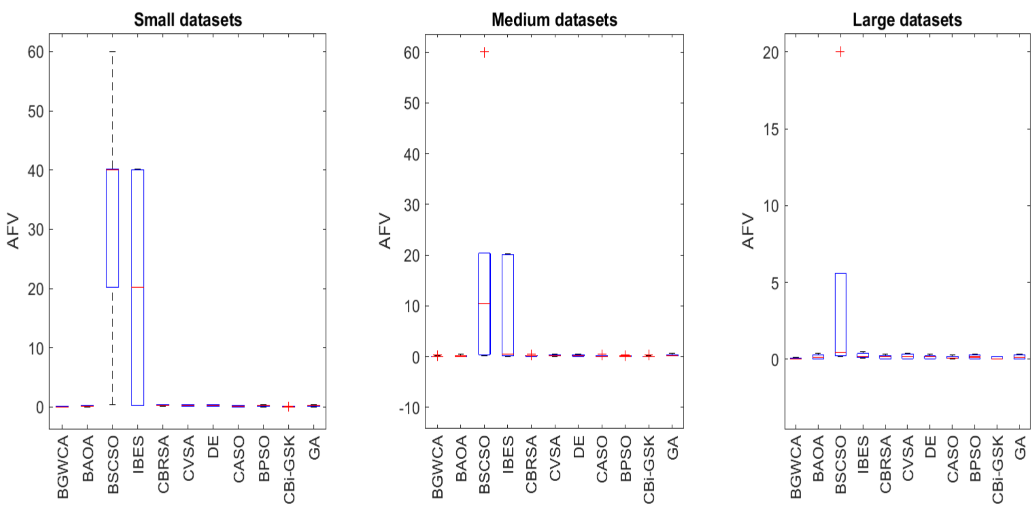


FIGURE 2. **The box-and-whisker plot for all the optimizers over all the datasets for AFV values.**

around zero, with a small interquartile range and whiskers extending to a maximum value of 0.1457. The middle figure exhibits a concentration of values near zero, with a small interquartile range and whiskers extending to a maximum value of 0.2724. In the right figure, the data is primarily composed of zero values, with a larger interquartile range and whiskers extending from the minimum to the maximum values of 0 and 0.1347, respectively. The presence of a non-zero value, 0.0362, in the right figure could be considered an outlier. In conclusion, these box-and-whisker plots provide a visual summary of the central tendency, spread, and potential outliers in each figure, aiding in the comparison of their respective distributions.

According to Table 8, it is obvious that the Friedman test indicates significant differences. Therefore, we opted to apply the Dunn's post-hoc test in order to identify specific

TABLE 16. The p-values obtained by the post hoc Dunn's test for Table 10.

		BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
	BGWCA		0.6462	**0.0000**	**0.0002**	**0.0175**	**0.0133**	**0.0153**	1.0000	0.8104	1.0000	0.0560
	BAOA			0.0714	0.6814	1.0000	1.0000	1.0000	1.0000	1.0000	0.7489	1.0000
	BSCSO				1.0000	0.9407	0.9648	0.9539	**0.0005**	**0.0385**	**0.0000**	0.7158
	IBES					1.0000	1.0000	1.0000	**0.0229**	0.5033	**0.0003**	0.9999
	CBRSA						1.0000	1.0000	0.5387	0.9999	**0.0261**	1.0000
Samll	**CVSA**							1.0000	0.4686	0.9996	**0.0200**	1.0000
	DE								0.5033	0.9998	**0.0229**	1.0000
	CASO									1.0000	1.0000	0.8381
	BPSO										0.8866	1.0000
	CBi-GSK											0.0804
	GA											
	BGWCA		0.7092	**0.0000**	**0.0001**	0.2324	**0.0119**	**0.0088**	0.9895	0.9701	1.0000	**0.0025**
	BAOA			**0.0490**	0.5194	1.0000	0.9999	0.9997	1.0000	1.0000	0.9596	0.9848
	BSCSO				1.0000	0.2575	0.9596	0.9784	**0.0065**	**0.0103**	**0.0000**	0.9994
	IBES					0.9467	1.0000	1.0000	0.1336	0.1877	**0.0006**	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.5576	1.0000
Medium	**CVSA**							1.0000	0.9467	0.9784	**0.0490**	1.0000
	DE								0.9128	0.9596	**0.0374**	1.0000
	CASO									1.0000	1.0000	0.6722
	BPSO										0.9997	0.7789
	CBi-GSK											**0.0119**
	GA											
	BGWCA		0.4589	**0.0123**	**0.0123**	0.6015	0.0599	0.8902	0.9698	0.7419	1.0000	0.5532
	BAOA			1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9386	1.0000
	BSCSO				1.0000	1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	IBES					1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.9800	1.0000
Large	**CVSA**							1.0000	0.9999	1.0000	0.3314	1.0000
	DE								1.0000	1.0000	0.9997	1.0000
	CASO									1.0000	1.0000	1.0000
	BPSO										0.9955	1.0000
	CBi-GSK											0.9698
	GA											

TABLE 17. The p-values obtained by the post hoc Dunn's test for Table 11.

		BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
	BGWCA		0.6462	**0.0000**	**0.0002**	**0.0175**	**0.0133**	**0.0153**	1.0000	0.8104	1.0000	0.0560
	BAOA			0.0714	0.6814	1.0000	1.0000	1.0000	1.0000	1.0000	0.7489	1.0000
	BSCSO				1.0000	0.9407	0.9648	0.9539	**0.0005**	**0.0385**	**0.0000**	0.7158
	IBES					1.0000	1.0000	1.0000	**0.0229**	0.5033	**0.0003**	0.9999
	CBRSA						1.0000	1.0000	0.5387	0.9999	**0.0261**	1.0000
Samll	**CVSA**							1.0000	0.4686	0.9996	**0.0200**	1.0000
	DE								0.5033	0.9998	**0.0229**	1.0000
	CASO									1.0000	1.0000	0.8381
	BPSO										0.8866	1.0000
	CBi-GSK											0.0804
	GA											
	BGWCA		0.7092	**0.0000**	**0.0001**	0.2324	**0.0119**	**0.0088**	0.9895	0.9701	1.0000	**0.0025**
	BAOA			**0.0490**	0.5194	1.0000	0.9999	0.9997	1.0000	1.0000	0.9596	0.9848
	BSCSO				1.0000	0.2575	0.9596	0.9784	**0.0065**	**0.0103**	**0.0000**	0.9994
	IBES					0.9467	1.0000	1.0000	0.1336	0.1877	**0.0006**	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.5576	1.0000
Medium	**CVSA**							1.0000	0.9467	0.9784	**0.0490**	1.0000
	DE								0.9128	0.9596	**0.0374**	1.0000
	CASO									1.0000	1.0000	0.6722
	BPSO										0.9997	0.7789
	CBi-GSK											**0.0119**
	GA											
	BGWCA		0.4589	**0.0123**	**0.0123**	0.6015	0.0599	0.8902	0.9698	0.7419	1.0000	0.5532
	BAOA			1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9386	1.0000
	BSCSO				1.0000	1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	IBES					1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.9800	1.0000
Large	**CVSA**							1.0000	0.9999	1.0000	0.3314	1.0000
	DE								1.0000	1.0000	0.9997	1.0000
	CASO									1.0000	1.0000	1.0000
	BPSO										0.9955	1.0000
	CBi-GSK											0.9698
	GA											

TABLE 18. The p-values obtained by the post hoc Dunn's test for Table 12.

		BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
	BGWCA		0.6462	**0.0000**	**0.0002**	**0.0175**	**0.0133**	**0.0153**	1.0000	0.8104	1.0000	0.0560
	BAOA			0.0714	0.6814	1.0000	1.0000	1.0000	1.0000	1.0000	0.7489	1.0000
	BSCSO				1.0000	0.9407	0.9648	0.9539	**0.0005**	**0.0385**	**0.0000**	0.7158
	IBES					1.0000	1.0000	1.0000	**0.0229**	0.5033	**0.0003**	0.9999
	CBRSA						1.0000	1.0000	0.5387	0.9999	**0.0261**	1.0000
Samll	**CVSA**							1.0000	0.4686	0.9996	**0.0200**	1.0000
	DE								0.5033	0.9998	**0.0229**	1.0000
	CASO									1.0000	1.0000	0.8381
	BPSO										0.8866	1.0000
	CBi-GSK											0.0804
	GA											
	BGWCA		0.7092	**0.0000**	**0.0001**	0.2324	**0.0119**	**0.0088**	0.9895	0.9701	1.0000	**0.0025**
	BAOA			**0.0490**	0.5194	1.0000	0.9999	0.9997	1.0000	1.0000	0.9596	0.9848
	BSCSO				1.0000	0.2575	0.9596	0.9784	**0.0065**	**0.0103**	**0.0000**	0.9994
	IBES					0.9467	1.0000	1.0000	0.1336	0.1877	**0.0006**	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.5576	1.0000
Medium	**CVSA**							1.0000	0.9467	0.9784	**0.0490**	1.0000
	DE								0.9128	0.9596	**0.0374**	1.0000
	CASO									1.0000	1.0000	0.6722
	BPSO										0.9997	0.7789
	CBi-GSK											**0.0119**
	GA											
	BGWCA		0.4589	**0.0123**	**0.0123**	0.6015	0.0599	0.8902	0.9698	0.7419	1.0000	0.5532
	BAOA			1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9386	1.0000
	BSCSO				1.0000	1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	IBES					1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.9800	1.0000
Large	**CVSA**							1.0000	0.9999	1.0000	0.3314	1.0000
	DE								1.0000	1.0000	0.9997	1.0000
	CASO									1.0000	1.0000	1.0000
	BPSO										0.9955	1.0000
	CBi-GSK											0.9698
	GA											

TABLE 19. The p-values obtained by the post hoc Dunn's test for Table 13.

		BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
	BGWCA		0.6462	**0.0000**	**0.0002**	**0.0175**	**0.0133**	**0.0153**	1.0000	0.8104	1.0000	0.0560
	BAOA			0.0714	0.6814	1.0000	1.0000	1.0000	1.0000	1.0000	0.7489	1.0000
	BSCSO				1.0000	0.9407	0.9648	0.9539	**0.0005**	**0.0385**	**0.0000**	0.7158
	IBES					1.0000	1.0000	1.0000	**0.0229**	0.5033	**0.0003**	0.9999
	CBRSA						1.0000	1.0000	0.5387	0.9999	**0.0261**	1.0000
Samll	**CVSA**							1.0000	0.4686	0.9996	**0.0200**	1.0000
	DE								0.5033	0.9998	**0.0229**	1.0000
	CASO									1.0000	1.0000	0.8381
	BPSO										0.8866	1.0000
	CBi-GSK											0.0804
	GA											
	BGWCA		0.7092	**0.0000**	**0.0001**	0.2324	**0.0119**	**0.0088**	0.9895	0.9701	1.0000	**0.0025**
	BAOA			**0.0490**	0.5194	1.0000	0.9999	0.9997	1.0000	1.0000	0.9596	0.9848
	BSCSO				1.0000	0.2575	0.9596	0.9784	**0.0065**	**0.0103**	**0.0000**	0.9994
	IBES					0.9467	1.0000	1.0000	0.1336	0.1877	**0.0006**	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.5576	1.0000
Medium	**CVSA**							1.0000	0.9467	0.9784	**0.0490**	1.0000
	DE								0.9128	0.9596	**0.0374**	1.0000
	CASO									1.0000	1.0000	0.6722
	BPSO										0.9997	0.7789
	CBi-GSK											**0.0119**
	GA											
	BGWCA		0.4589	**0.0123**	**0.0123**	0.6015	0.0599	0.8902	0.9698	0.7419	1.0000	0.5532
	BAOA			1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9386	1.0000
	BSCSO				1.0000	1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	IBES					1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.9800	1.0000
Large	**CVSA**							1.0000	0.9999	1.0000	0.3314	1.0000
	DE								1.0000	1.0000	0.9997	1.0000
	CASO									1.0000	1.0000	1.0000
	BPSO										0.9955	1.0000
	CBi-GSK											0.9698
	GA											

TABLE 20. The p-values obtained by the post hoc Dunn's test for Table 14.

		BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
	BGWCA		0.6462	**0.0000**	**0.0002**	**0.0175**	**0.0133**	**0.0153**	1.0000	0.8104	1.0000	0.0560
	BAOA			0.0714	0.6814	1.0000	1.0000	1.0000	1.0000	1.0000	0.7489	1.0000
	BSCSO				1.0000	0.9407	0.9648	0.9539	**0.0005**	**0.0385**	**0.0000**	0.7158
	IBES					1.0000	1.0000	1.0000	**0.0229**	0.5033	**0.0003**	0.9999
	CBRSA						1.0000	1.0000	0.5387	0.9999	**0.0261**	1.0000
Samll	**CVSA**							1.0000	0.4686	0.9996	**0.0200**	1.0000
	DE								0.5033	0.9998	**0.0229**	1.0000
	CASO									1.0000	1.0000	0.8381
	BPSO										0.8866	1.0000
	CBi-GSK											0.0804
	GA											
	BGWCA		0.7092	**0.0000**	**0.0001**	0.2324	**0.0119**	**0.0088**	0.9895	0.9701	1.0000	**0.0025**
	BAOA			**0.0490**	0.5194	1.0000	0.9999	0.9997	1.0000	1.0000	0.9596	0.9848
	BSCSO				1.0000	0.2575	0.9596	0.9784	**0.0065**	**0.0103**	**0.0000**	0.9994
	IBES					0.9467	1.0000	1.0000	0.1336	0.1877	**0.0006**	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.5576	1.0000
Medium	**CVSA**							1.0000	0.9467	0.9784	**0.0490**	1.0000
	DE								0.9128	0.9596	**0.0374**	1.0000
	CASO									1.0000	1.0000	0.6722
	BPSO										0.9997	0.7789
	CBi-GSK											**0.0119**
	GA											
	BGWCA		0.4589	**0.0123**	**0.0123**	0.6015	0.0599	0.8902	0.9698	0.7419	1.0000	0.5532
	BAOA			1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9386	1.0000
	BSCSO				1.0000	1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	IBES					1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.9800	1.0000
Large	**CVSA**							1.0000	0.9999	1.0000	0.3314	1.0000
	DE								1.0000	1.0000	0.9997	1.0000
	CASO									1.0000	1.0000	1.0000
	BPSO										0.9955	1.0000
	CBi-GSK											0.9698
	GA											

TABLE 21. The p-values obtained by the post hoc Dunn's test for Table 15.

		BGWCA	BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
	BGWCA		0.6462	**0.0000**	**0.0002**	**0.0175**	**0.0133**	**0.0153**	1.0000	0.8104	1.0000	0.0560
	BAOA			0.0714	0.6814	1.0000	1.0000	1.0000	1.0000	1.0000	0.7489	1.0000
	BSCSO				1.0000	0.9407	0.9648	0.9539	**0.0005**	**0.0385**	**0.0000**	0.7158
	IBES					1.0000	1.0000	1.0000	**0.0229**	0.5033	**0.0003**	0.9999
	CBRSA						1.0000	1.0000	0.5387	0.9999	**0.0261**	1.0000
Samll	**CVSA**							1.0000	0.4686	0.9996	**0.0200**	1.0000
	DE								0.5033	0.9998	**0.0229**	1.0000
	CASO									1.0000	1.0000	0.8381
	BPSO										0.8866	1.0000
	CBi-GSK											0.0804
	GA											
	BGWCA		0.7092	**0.0000**	**0.0001**	0.2324	**0.0119**	**0.0088**	0.9895	0.9701	1.0000	**0.0025**
	BAOA			**0.0490**	0.5194	1.0000	0.9999	0.9997	1.0000	1.0000	0.9596	0.9848
	BSCSO				1.0000	0.2575	0.9596	0.9784	**0.0065**	**0.0103**	**0.0000**	0.9994
	IBES					0.9467	1.0000	1.0000	0.1336	0.1877	**0.0006**	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.5576	1.0000
Medium	**CVSA**							1.0000	0.9467	0.9784	**0.0490**	1.0000
	DE								0.9128	0.9596	**0.0374**	1.0000
	CASO									1.0000	1.0000	0.6722
	BPSO										0.9997	0.7789
	CBi-GSK											**0.0119**
	GA											
	BGWCA		0.4589	**0.0123**	**0.0123**	0.6015	0.0599	0.8902	0.9698	0.7419	1.0000	0.5532
	BAOA			1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9386	1.0000
	BSCSO				1.0000	1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	IBES					1.0000	1.0000	0.9975	0.9800	0.9999	0.0965	1.0000
	CBRSA						1.0000	1.0000	1.0000	1.0000	0.9800	1.0000
Large	**CVSA**							1.0000	0.9999	1.0000	0.3314	1.0000
	DE								1.0000	1.0000	0.9997	1.0000
	CASO									1.0000	1.0000	1.0000
	BPSO										0.9955	1.0000
	CBi-GSK											0.9698
	GA											

TABLE 22. The p-values obtained by the Wilcoxon signed ranks test for Table 10.

		BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
BGWCA	Small	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0209**	**0.0152**	0.7794	**0.0109**
	Medium	**0.0180**	**0.0117**	**0.0117**	**0.0180**	**0.0117**	**0.0117**	**0.0180**	**0.0116**	**0.0431**	**0.0116**
	Medium	**0.0431**	**0.0431**	**0.0431**	0.0679	**0.0431**	0.0679	**0.0431**	**0.0431**	0.0679	**0.0431**

TABLE 23. The p-values obtained by the Wilcoxon signed ranks test for Table 11.

		BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
BGWCA	Small	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0209**	**0.0152**	0.6784	**0.0109**
	Medium	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**
	Medium	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**

TABLE 24. The p-values obtained by the Wilcoxon signed ranks test for Table 12.

		BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
BGWCA	Small	**0.0117**	**0.0077**	**0.0077**	**0.0109**	**0.0117**	**0.0077**	0.0929	**0.0117**	0.4838	**0.0077**
	Medium	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0180**	**0.0117**
	Medium	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**

TABLE 25. The p-values obtained by the Wilcoxon signed ranks test for Table 13.

		BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
BGWCA	Small	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0109**	**0.0152**	0.6784	**0.0109**
	Medium	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**
	Medium	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	0.1380	**0.0431**	**0.0431**	**0.0431**	**0.0431**

TABLE 26. The p-values obtained by the Wilcoxon signed ranks test for Table 14.

		BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
BGWCA	Small	0.5132	**0.0077**	**0.0117**	**0.0077**	**0.0107**	**0.0108**	**0.0109**	**0.0076**	0.3615	**0.0108**
	Medium	**0.0296**	**0.0180**	**0.0116**	**0.0117**	**0.0173**	**0.0117**	**0.0117**	**0.0117**	**0.0499**	**0.0172**
	Medium	**0.0431**	0.2249	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**

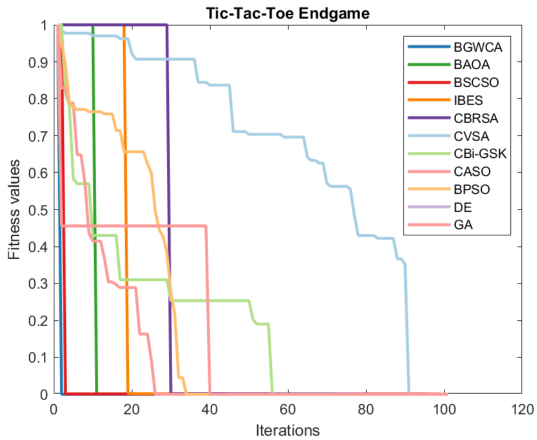
TABLE 27. The p-values obtained by the Wilcoxon signed ranks test for Table 15.

		BAOA	BSCSO	IBES	CBRSA	CVSA	DE	CASO	BPSO	CBi-GSK	GA
BGWCA	Small	**0.0077**	**0.0209**	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0077**	**0.0077**
	Medium	**0.0117**	**0.0251**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**	**0.0117**
	Medium	0.8927	0.8927	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**	**0.0431**

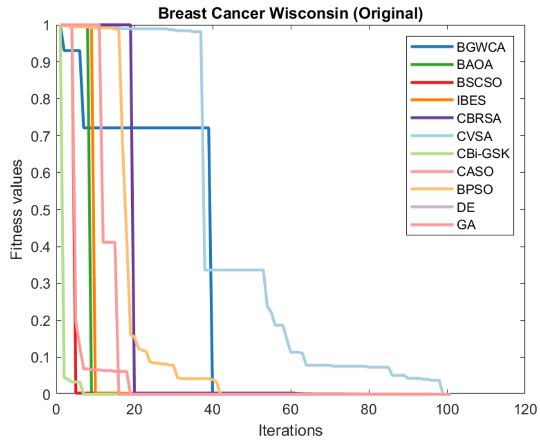
pairs of treatments that are significantly different from each other after finding a significant result in the Friedman test. Tables 16, 17, 18, 19, 20, and 21 summarize the p-values computed by the Dunn's post-hoc test. The p-values obtained from the Dunn's test provide information about the significance of the differences between specific pairs of groups. P-values below the significance threshold of 0.05 are emphasized in bold font. To interpret a particular value at the intersection of a row (representing an algorithm, e.g., BGWCA) and a column (representing another algorithm, e.g., IBES), if the associated p-value is less than 0.05, it signifies a significant difference between these algorithms, suggesting that the algorithm denoted by the row label outperforms the one denoted by the column label. Conversely, if the p-value is greater than or equal to 0.05, we infer nearly similar performance between the two algorithms.

Tables 22, 23, 24, 25, 26, and 27 summarize the p-values obtained by the Wilcoxon signed ranks test. As evident from the results, the algorithm put forward demonstrates superior performance in addressing the FS problem compared to all other contenders across datasets of varying sizes, considering a predetermined threshold of $\alpha = 0.05$.

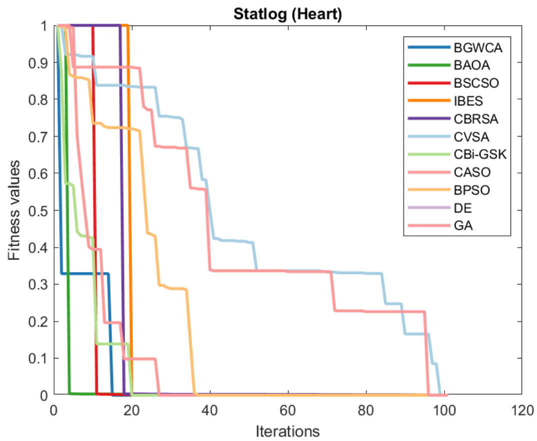
In Figures 3, 4, and 5, the convergence curves of fitness values over 100 iterations, calculated using Equation 21, are depicted for each dataset across the range of considered optimizers. These visualizations offer a comprehensive view of the optimization process, showcasing how the fitness values evolve over iterations. The comparison across multiple optimizers provides insights into their respective convergence behaviors and performance on diverse datasets. Figures 3, 4, and 5 clearly illustrate the high convergence rates achieved across various datasets, demonstrating the efficacy of the proposed algorithm. Importantly, the algorithm maintains optimal solutions throughout the convergence process, underscoring its reliability. Notably, the algorithm successfully avoids premature convergence in the majority



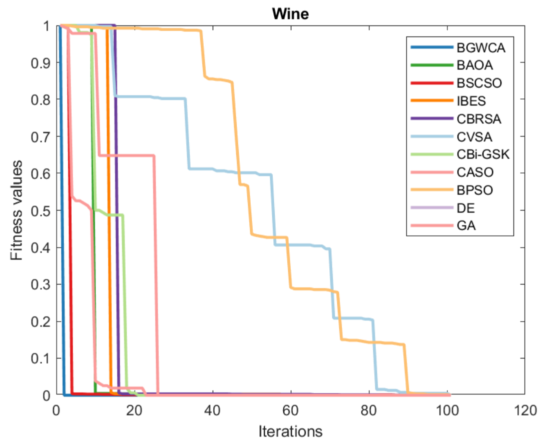
(a) Convergence analysis of fitness values for dataset d_1.



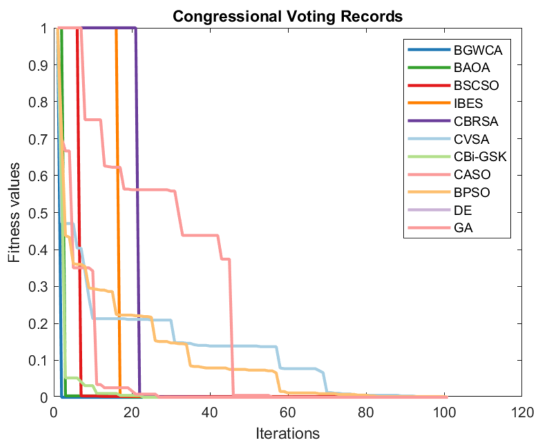
(b) Convergence analysis of fitness values for dataset d_2.



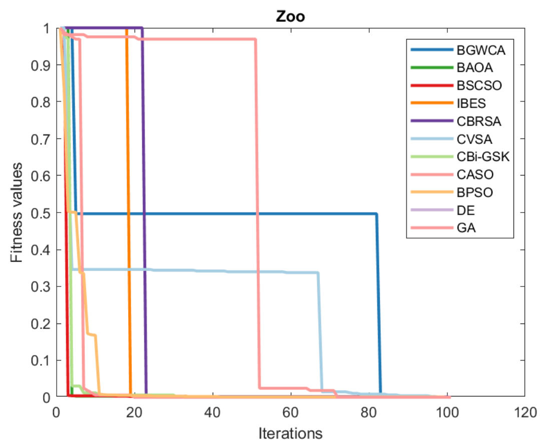
(c) Convergence analysis of fitness values for dataset d_3.



(d) Convergence analysis of fitness values for dataset d_4.



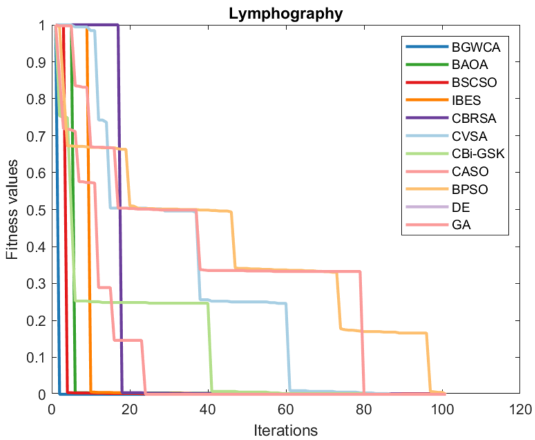
(e) Convergence analysis of fitness values for dataset d_5.



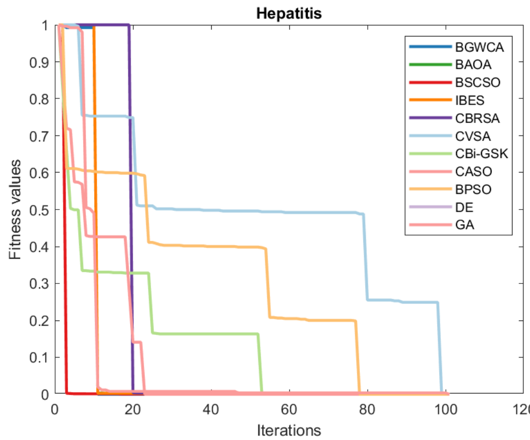
(f) Convergence analysis of fitness values for dataset d_6.

FIGURE 3. **Convergence analysis of fitness values across optimizers for small dataset (d_1 to d_6).**

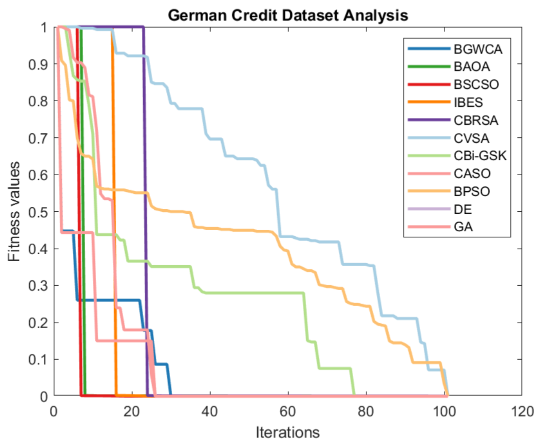
of cases, a critical aspect in ensuring robust optimization. The incorporation of opposition-based learning and Gaussian mutation emerges as a key contributing factor to the enhanced performance of the GWCA. This conclusion highlights the significance of these innovative techniques in improving the algorithm's convergence behavior and overall effectiveness in solving the FS problem across diverse datasets.



(g) Convergence analysis of fitness values for dataset d_7.



(h) Convergence analysis of fitness values for dataset d_8.



(i) Convergence analysis of fitness values for dataset d_9.

FIGURE 3. ***(Continued.)*** **Convergence analysis of fitness values across optimizers for small dataset (d_7 to d_9).**

TABLE 28. **The list of symbols used in the paper.**

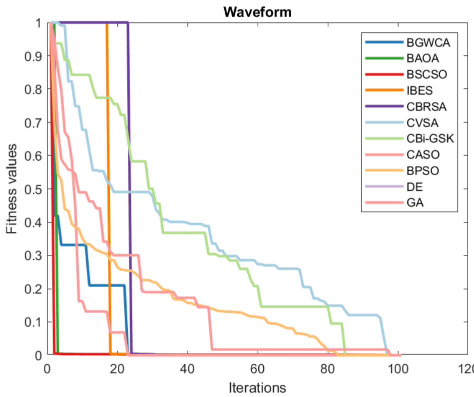
Symbol	Explanation
ML	Machine Learning
FS	Feature Selection
MA	Metaheuristic Algorithm
GWCA	Great Wall Construction Algorithm
OBL	Opposition-Based Learning
GM	Gaussian Mutation
KNN	K-Nearest Neighbors
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives
CER	Classification Error Rate

VI. CONCLUSION

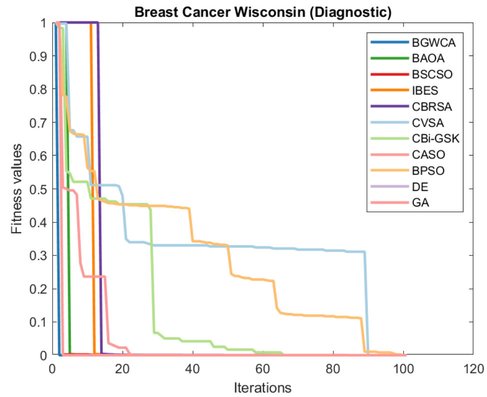
In conclusion, we presented a comprehensive exploration of the feature selection problem, emphasizing the critical role of selecting relevant features for enhancing machine learning model performance. The inherent complexity of this problem, stemming from a vast search space, was tackled through the utilization of the Great Wall Construction Algorithm, a recently proposed metaheuristic approach. To further augment the algorithm's effectiveness, opposition-based learning and Gaussian mutation techniques were integrated, addressing challenges related to exploration, exploitation, and local optima avoidance.

The empirical evaluation of the proposed algorithm involved a thorough comparative analysis against ten state-of-the-art methodologies, encompassing both contemporary and classical algorithms. The assessment spanned 22 datasets of varying sizes, providing a diverse testing ground ranging from 9 to 856 features. Six distinct evaluation metrics, covering aspects such as accuracy, classification error rate, number of selected features, and completion time, were employed to ensure a comprehensive understanding of the algorithm's performance.

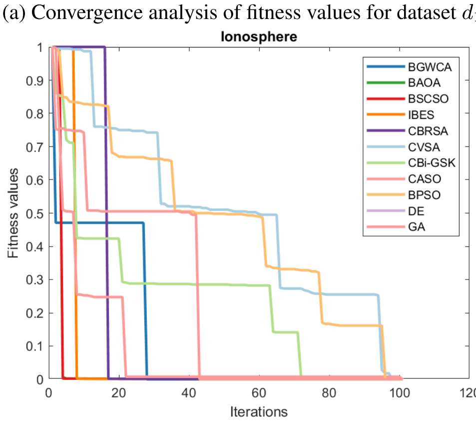
The rigorous validation of results was conducted through non-parametric statistical tests, including the Friedman test, post hoc Dunn's test, and the Wilcoxon signed ranks test. The obtained mean ranks and p-values conclusively demonstrated the superior efficacy of the proposed algorithm in addressing the feature selection problem. The algorithm showcased



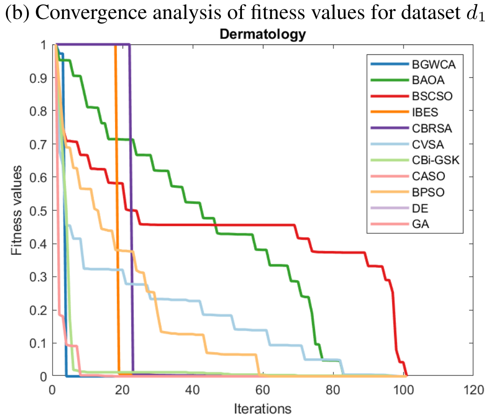
(a) Convergence analysis of fitness values for dataset d_{10}.



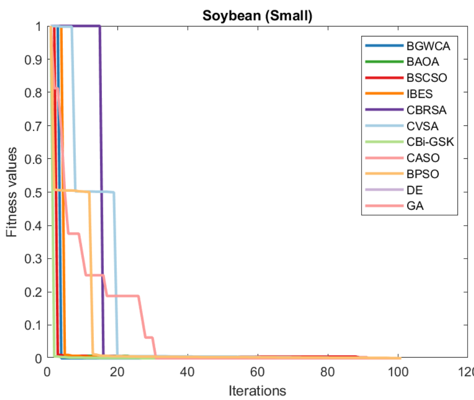
(b) Convergence analysis of fitness values for dataset d_{11}.



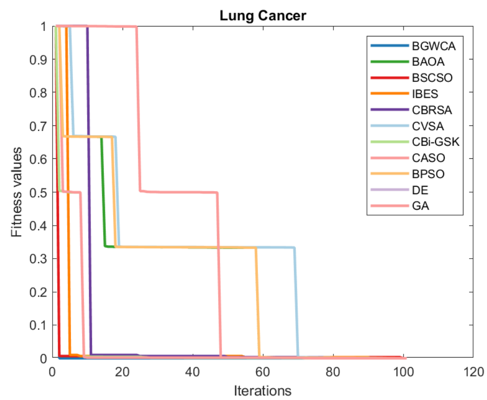
(c) Convergence analysis of fitness values for dataset d_{12}.



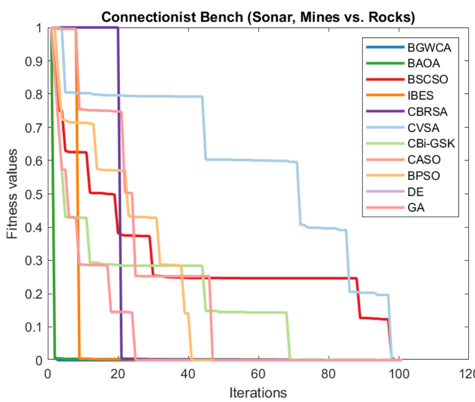
(d) Convergence analysis of fitness values for dataset d_{13}.



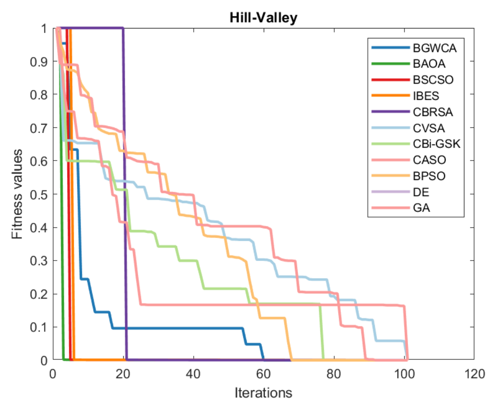
(e) Convergence analysis of fitness values for dataset d_{14}.



(f) Convergence analysis of fitness values for dataset d_{15}.

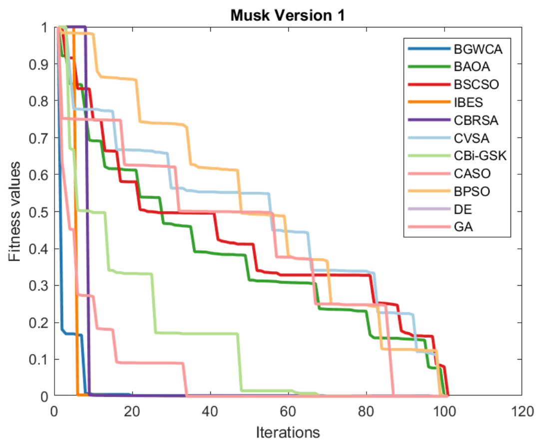


(g) Convergence analysis of fitness values for dataset d_{16}.

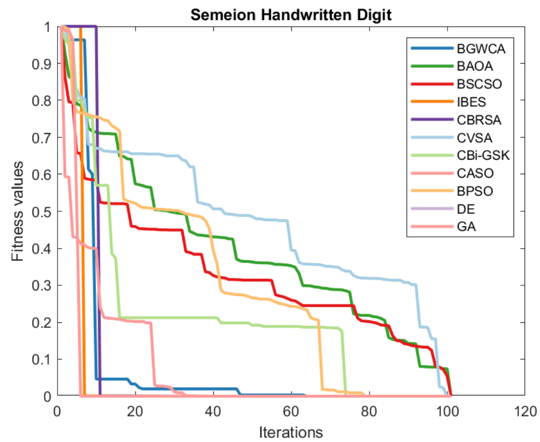


(h) Convergence analysis of fitness values for dataset d_{17}.

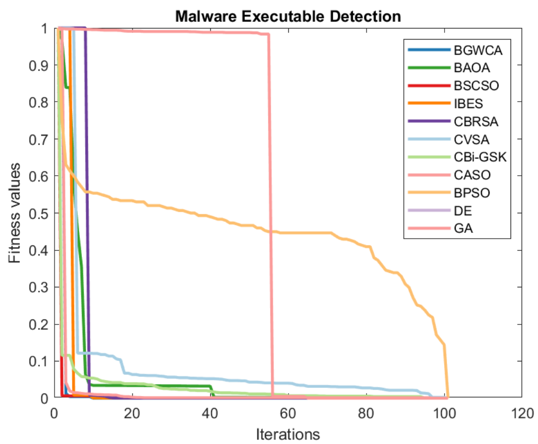
FIGURE 4. **Convergence analysis of fitness values across optimizers for medium datasets (d_{10} to d_{17}).**



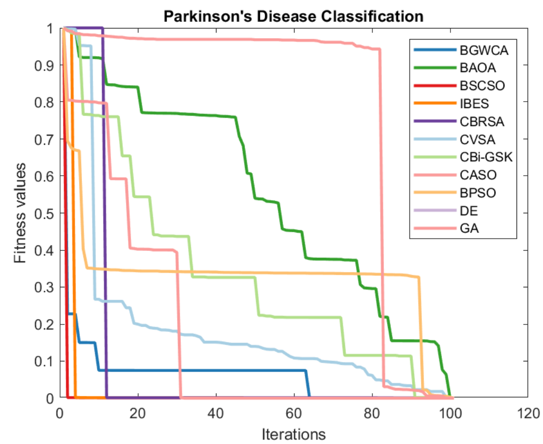
(a) Convergence analysis of fitness values for dataset d_{18}.



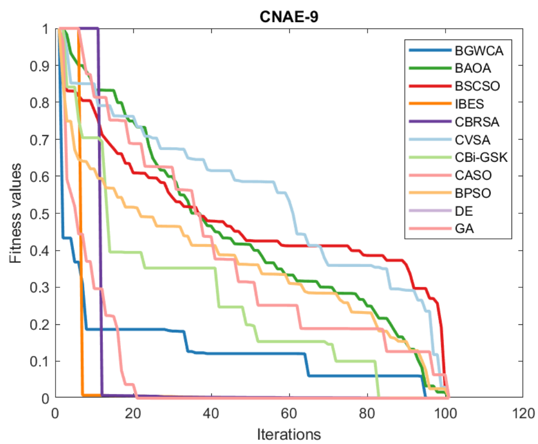
(b) Convergence analysis of fitness values for dataset d_{19}.



(c) Convergence analysis of fitness values for dataset d_{20}.



(d) Convergence analysis of fitness values for dataset d_{21}.



(e) Convergence analysis of fitness values for dataset d_{22}.

FIGURE 5. Convergence analysis of fitness values across optimizers for large datasets d_{18} to d_{22}.

its prowess by outperforming competing methodologies across multiple metrics, establishing itself as a robust and promising solution for enhancing the efficiency and accuracy of feature selection in machine learning models. The findings of this research contribute valuable insights to the field, offering a compelling approach to addressing one of the fundamental challenges in machine learning model optimization.

APPENDIX. TABLE OF USED SYMBOLS

Table 28 serves to summarize and elucidate the list of symbols utilized in the paper, aiming to enhance the clarity and ease of comprehension for readers.

REFERENCES

[1] F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, and M. Tornatore, "An overview on application of machine learning techniques in optical networks," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1383–1408, 2nd Quart., 2019.

[2] M. D. Lal and R. Varadarajan, "A review of machine learning approaches in synchrophasor technology," *IEEE Access*, vol. 11, pp. 33520–33541, 2023.

[3] A. N. Wilson, K. A. Gupta, B. H. Koduru, A. Kumar, A. Jha, and L. R. Cenkeramaddi, "Recent advances in thermal imaging and its applications using machine learning: A review," *IEEE Sensors J.*, vol. 23, no. 4, pp. 3395–3407, Feb. 2023.

[4] H. Mosaffa, M. Sadeghi, I. Mallakpour, M. N. Jahromi, and H. R. Pourghasemi, "Application of machine learning algorithms in hydrology," in *Computers in Earth and Environmental Sciences*. Amsterdam, The Netherlands: Elsevier, 2022, pp. 585–591.

[5] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *Computational Intelligence and Bioinspired Systems*, J. Cabestany, A. Prieto, and F. Sandoval, Eds. Berlin, Germany: Springer, 2005, pp. 758–770.

[6] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Elect. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014.

[7] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed, "Metaheuristic algorithms on feature selection: A survey of one decade of research (2009–2019)," *IEEE Access*, vol. 9, pp. 26766–26791, 2021.

[8] F. Khan, I. Tarimer, H. S. Alwageed, B. C. Karadağ, M. Fayaz, A. B. Abdusalomov, and Y.-I. Cho, "Effect of feature selection on the accuracy of music popularity classification using machine learning algorithms," *Electronics*, vol. 11, no. 21, p. 3518, Oct. 2022.

[9] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *Proc. Sci. Inf. Conf.*, 2014, pp. 372–378.

[10] S. Kotsiantis, "Feature selection for machine learning classification problems: A recent overview," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 157–176, 2011.

[11] A. Jovic, K. Brkic, and N. Bogunovic, "A review of feature selection methods with applications," in *Proc. 38th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2015, pp. 1200–1205.

[12] U. Stańczyk, "Feature evaluation by filter, wrapper, and embedded approaches," in *Feature Selection for Data and Pattern Recognition*, U. Stańczyk and L. C. Jain, Eds. Berlin, Germany: Springer, 2015, pp. 29–44, doi: 10.1007/978-3-662-45620-0_3.

[13] M. Nssibi, G. Manita, and O. Korbaa, "Advances in nature-inspired metaheuristic optimization for feature selection problem: A comprehensive survey," *Comput. Sci. Rev.*, vol. 49, Aug. 2023, Art. no. 100559.

[14] Z. Sadeghian, E. Akbari, H. Nematzadeh, and H. Motameni, "A review of feature selection methods based on meta-heuristic algorithms," *J. Experim. Theor. Artif. Intell.*, pp. 1–51, Feb. 2023.

[15] M. N. M. Salleh, K. Hussain, S. Cheng, Y. Shi, A. Muhammad, G. Ullah, and R. Naseem, "Exploration and exploitation measurement in swarm-based metaheuristic algorithms: An empirical analysis," R. Ghazali, M. M. Deris, N. M. Nawi, and J. H. Abawajy, Eds. Cham, Switzerland: Springer, Jan. 2018, pp. 24–32. [Online]. Available: https://heronet.epa.gov/heroneUindex.cfm/reference/download/reference_id/7171675, doi: 10.1007/978-3-319-72550-5_3.

[16] B. Morales-Castañeda, D. Zaldívar, E. Cuevas, F. Fausto, and A. Rodríguez, "A better balance in metaheuristic algorithms: Does it exist?" *Swarm Evol. Comput.*, vol. 54, May 2020, Art. no. 100671.

[17] J. Xu and J. Zhang, "Exploration-exploitation tradeoffs in metaheuristics: Survey and analysis," in *Proc. 33rd Chin. Control Conf.*, 2014, pp. 8633–8638.

[18] S. Afreen, A. K. Bhurjee, and R. M. Aziz, "Gene selection with game Shapley Harris hawks optimizer for cancer classification," *Chemometric Intell. Lab. Syst.*, vol. 242, Nov. 2023, Art. no. 104989.

[19] R. Mahto, S. U. Ahmed, R. U. Rahman, R. M. Aziz, P. Roy, S. Mallik, A. Li, and M. A. Shah, "A novel and innovative cancer classification framework through a consecutive utilization of hybrid feature selection," *BMC Bioinf.*, vol. 24, no. 1, p. 479, Dec. 2023.

[20] R. M. Aziz, R. Mahto, A. Das, S. U. Ahmed, P. Roy, S. Mallik, and A. Li, "CO-WOA: Novel optimization approach for deep learning classification of fish image," *Chem. Biodiversity*, vol. 20, no. 8, Aug. 2023, Art. no. e202201123.

[21] A. A. Joshi and R. M. Aziz, "Deep learning approach for brain tumor classification using metaheuristic optimization with gene expression data," *Int. J. Imag. Syst. Technol.*, Dec. 2023, Art. no. e23007.

[22] Z. Guan, C. Ren, J. Niu, P. Wang, and Y. Shang, "Great wall construction algorithm: A novel meta-heuristic algorithm for engineer problems," *Exp. Syst. Appl.*, vol. 233, Dec. 2023, Art. no. 120905.

[23] T. J. Choi, "A rotationally invariant stochastic opposition-based learning using a beta distribution in differential evolution," *Exp. Syst. Appl.*, vol. 231, Nov. 2023, Art. no. 120658.

[24] K.-T. Lan and C.-H. Lan, "Notes on the distinction of Gaussian and Cauchy mutations," in *Proc. 8th Int. Conf. Intell. Syst. Design Appl.*, Nov. 2008, pp. 272–277.

[25] O. S. Qasim and Z. Y. Algamal, "Feature selection using different transfer functions for binary bat algorithm," *Int. J. Math., Eng. Manag. Sci.*, vol. 5, no. 4, pp. 697–706, Aug. 2020.

[26] M. Mafarja, I. Aljarah, H. Faris, A. I. Hammouri, A. M. Al-Zoubi, and S. Mirjalili, "Binary grasshopper optimisation algorithm approaches for feature selection problems," *Exp. Syst. Appl.*, vol. 117, pp. 267–286, Mar. 2019.

[27] H. Chantar, M. Mafarja, H. Alsawalqah, A. A. Heidari, I. Aljarah, and H. Faris, "Feature selection using binary grey wolf optimizer with elite-based crossover for Arabic text classification," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12201–12220, Aug. 2020.

[28] S. Maza and D. Zouache, "Binary firefly algorithm for feature selection in classification," in *Proc. Int. Conf. Theor. Applicative Aspects Comput. Sci. (ICTAACS)*, vol. 1, Dec. 2019, pp. 1–6.

[29] L. Cervante, B. Xue, M. Zhang, and L. Shang, "Binary particle swarm optimisation for feature selection: A filter based approach," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.

[30] P. Agrawal et al., "S-shaped and V-shaped gaining-sharing knowledge-based algorithm for feature selection," *Appl. Intell.*, pp. 1–32, vol. 152, pp. 81–112, 2022, doi: 10.1007/s10489-021-02233-5.

[31] A. I. Hafez, H. M. Zawbaa, E. Emary, and A. E. Hassanien, "Sine cosine optimization algorithm for feature selection," in *Proc. Int. Symp. Innov. Intell. Syst. Appl. (INISTA)*, Aug. 2016, pp. 1–5.

[32] M. Nssibi, G. Manita, and O. Korbaa, "Binary giza pyramids construction for feature selection," *Proc. Comput. Sci.*, vol. 192, pp. 676–687, Jan. 2021.

[33] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary ant lion approaches for feature selection," *Neurocomputing*, vol. 213, pp. 54–65, Nov. 2016.

[34] H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A. M. Al-Zoubi, S. Mirjalili, and H. Fujita, "An efficient binary salp swarm algorithm with crossover scheme for feature selection problems," *Knowl.-Based Syst.*, vol. 154, pp. 43–67, Aug. 2018.

[35] S. Salesi and G. Cosma, "A novel extended binary cuckoo search algorithm for feature selection," in *Proc. 2nd Int. Conf. Knowl. Eng. Appl. (ICKEA)*, Oct. 2017, pp. 6–12.

[36] Y. Gao, Y. Zhou, and Q. Luo, "An efficient binary equilibrium optimizer algorithm for feature selection," *IEEE Access*, vol. 8, pp. 140936–140963, 2020.

[37] Y. Li, X. Cui, J. Fan, and T. Wang, "Global chaotic bat algorithm for feature selection," *J. Supercomput.*, vol. 78, no. 17, pp. 18754–18776, Nov. 2022.

[38] G. I. Sayed, A. Tharwat, and A. E. Hassanien, "Chaotic dragonfly algorithm: An improved metaheuristic algorithm for feature selection," *Appl. Intell.*, vol. 49, no. 1, pp. 188–205, Jan. 2019.

[39] F. S. Gharehchopogh, I. Maleki, and Z. A. Dizaji, "Chaotic vortex search algorithm: Metaheuristic algorithm for feature selection," *Evol. Intell.*, vol. 15, no. 3, pp. 1777–1808, Sep. 2022.

[40] R. A. Khurma, I. Aljarah, and A. Sharieh, "An efficient moth flame optimization algorithm using chaotic maps for feature selection in the medical applications," in *Proc. ICPRAM*, 2020, pp. 175–182.

[41] D. Yousri, M. Abd Elaziz, D. Oliva, A. Abraham, M. A. Alotaibi, and M. A. Hossain, "Fractional-order comprehensive learning marine predators algorithm for global optimization and feature selection," *Knowl.-Based Syst.*, vol. 235, Jan. 2022, Art. no. 107603.

[42] Y. Liu, A. Ghandar, and G. Theodoropoulos, "Island model genetic algorithm for feature selection in non-traditional credit risk evaluation," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 2771–2778.

[43] R. K. Agrawal, B. Kaur, and S. Sharma, "Quantum based whale optimization algorithm for wrapper feature selection," *Appl. Soft Comput.*, vol. 89, Apr. 2020, Art. no. 106092.

[44] A. Chaudhuri and T. P. Sahu, "A hybrid feature selection method based on binary Jaya algorithm for micro-array data classification," *Comput. Electr. Eng.*, vol. 90, Mar. 2021, Art. no. 106963.

[45] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[46] H. Jia, X. Peng, and C. Lang, "Remora optimization algorithm," *Exp. Syst. Appl.*, vol. 185, Dec. 2021, Art. no. 115665.
[47] H. M. Jia, C. Lang, D. Oliva, and W. Song, "Dynamic Harris hawks optimization with mutation mechanism for satellite image segmentation," *Remote Sens.*, vol. 11, no. 12, p. 1421, 2019.
[48] S. Mahdavi, S. Rahnamayan, and K. Deb, "Opposition based learning: A literature review," *Swarm Evol. Comput.*, vol. 39, pp. 1–23, Apr. 2018.
[49] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.
[50] J. Laaksonen and E. Oja, "Classification with learning K-nearest neighbors," in *Proc. Int. Conf. Neural Netw.*, vol. 3, 1996, pp. 1480–1483.
[51] J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger, "Evaluating the quality of machine learning explanations: A survey on methods and metrics," *Electronics*, vol. 10, no. 5, p. 593, Mar. 2021.
[52] N. Khodadadi, E. Khodadadi, Q. Al-Tashi, E.-S. M. El-Kenawy, L. Abualigah, S. J. Abdulkadir, A. Alqushaibi, and S. Mirjalili, "BAOA: Binary arithmetic optimization algorithm with K-nearest neighbor classifier for feature selection," *IEEE Access*, vol. 11, pp. 94094–94115, 2023.
[53] A. Seyyedabbasi, "Binary sand cat swarm optimization algorithm for wrapper feature selection on biological data," *Biomimetics*, vol. 8, no. 3, p. 310, Jul. 2023.
[54] A. Chhabra, A. G. Hussien, and F. A. Hashim, "Improved bald eagle search algorithm for global optimization and feature selection," *Alexandria Eng. J.*, vol. 68, pp. 141–180, Apr. 2023.
[55] L. Abualigah and A. Diabat, "Chaotic binary reptile search algorithm and its feature selection applications," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 10, pp. 13931–13947, Oct. 2023.
[56] P. Agrawal, T. Ganesh, and A. W. Mohamed, "Chaotic gaining sharing knowledge-based optimization algorithm: An improved metaheuristic algorithm for feature selection," *Soft Comput.*, vol. 25, no. 14, pp. 9505–9528, Jul. 2021.
[57] J. Too and A. R. Abdullah, "Chaotic atom search optimization for feature selection," *Arabian J. Sci. Eng.*, vol. 45, no. 8, pp. 6063–6079, Aug. 2020.
[58] M. A. Khanesar, M. Teshnehlab, and M. A. Shoorehdeli, "A novel binary particle swarm optimization," in *Proc. Medit. Conf. Control Autom.*, 2007, pp. 1–6.
[59] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Sep. 2007, pp. 3523–3530.
[60] R. L. Haupt, "An introduction to genetic algorithms for electromagnetics," *IEEE Antennas Propag. Mag.*, vol. 37, no. 2, pp. 7–15, Apr. 1995.

FAROUQ ZITOUNI received the Ph.D. degree in computer science from Abdelhamid Mehri University, Constantine, Algeria, in 2020. He is currently an Associate Professor with the Computer Science Department, Kasdi Merbah University, Ouargla, Algeria. He has contributed to the academic community by publishing several papers in international conferences and journals. His research interests include machine learning, continuous and combinatorial optimization, and metaheuristic algorithms.

ABDULAZIZ S. ALMAZYAD received the Ph.D. degree in computer engineering from Syracuse University, Syracuse, NY, USA. He is currently a Professor with the College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. His research interests include the Internet of Things, cloud computing, artificial intelligence, mobile and wireless networks, and information security.

GUOJIANG XIONG received the B.Sc. degree in automation from Zhejiang University, Hangzhou, China, in 2009, and the M.Sc. and Ph.D. degrees in power system and its automation from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2011 and 2014, respectively. From August 2014 to August 2017, he was an Engineer with Guizhou Electric Power Grid Dispatching and Control Center, Guiyang, China. After that, he joined the College of Electrical Engineering, Guizhou University (GZU), as an Associate Professor. Since January 2019, he has been a Distinguished Professor with GZU. He has published over 70 research articles in journals and has been a reviewer of over 30 journal articles and conference papers. His main research interests include renewable energy, power system operation, fault diagnosis of power systems, and application of artificial intelligence in power systems.

ALI WAGDY MOHAMED received the B.Sc., M.Sc., and Ph.D. degrees from Cairo University, Egypt, in 2000, 2004, and 2010, respectively. He was an Associate Professor of statistics with the Wireless Intelligent Networks Center (WINC), Faculty of Engineering and Applied Sciences, Nile University, from 2019 to 2021. He is currently a Professor and the Chair of the Operations Research Department, Faculty of Graduate Studies for Statistical Research, Cairo University. He is also a Professor with the Mathematics and Actuarial Science Department, School of Sciences and Engineering, The American University in Cairo, Cairo, Egypt. Recently, he has been recognized among the top 2% of scientists according to Stanford University reports for 2020, 2021, and 2022, respectively. He has presented and participated in more than ten international conferences. He published more than 140 articles in reputed and high-impact journals. He participated as a member of the reviewer committee for 35 different conferences sponsored by Springer and IEEE. Recently, he has been appointed as a member of the Education and Scientific Research Policy Council, Academy of Scientific Research, from 2021 to 2024. He serves as a reviewer for more than 100 internationally accredited top-tier journals and has been awarded the Publons Peer Review Awards 2018, for placing in the top 1% of reviewers worldwide in the assorted field. He is the Chair of the Egyptian Chapter of the African Federation of Operations Research Societies (AFROS). He is an Associate Editor of *Swarm and Evolutionary Computation* (Elsevier). He is an editor of more than ten journals of *Information Sciences*, *Applied Mathematics*, *Engineering*, *System Science*, and *Operations Research*.

SAAD HAROUS received the Ph.D. degree in computer science from Case Western Reserve University, Cleveland, OH, USA, in 1991. He has more than 30 years of experience in teaching and research in three different countries, including USA, Oman, and United Arab Emirates. He is currently a Professor with the College of Computing and Informatics, University of Sharjah, United Arab Emirates. His teaching interests include programming, data structures, design and analysis of algorithms, operating systems, and networks. He has published more than 200 journal articles and conference papers. His research interests include parallel and distributed computing, P2P delivery architectures, wireless networks, and the use of computers in education and processing Arabic language.