IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 17, 2021, accepted October 1, 2021, date of publication October 4, 2021, date of current version October 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3117853

An Improved Binary Grey-Wolf Optimizer With Simulated Annealing for Feature Selection

MOHAMED ABDEL-BASSET[1], (Senior Member, IEEE), KARAM M. SALLAM[1,2], (Member, IEEE), REDA MOHAMED[1], IBRAHIM ELGENDI[2], (Member, IEEE), KUMUDU MUNASINGHE[2], (Member, IEEE), AND OSAMA M. ELKOMY[1]

[1]Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt
[2]School of IT and Systems, University of Canberra, Canberra, ACT 2601, Australia

Corresponding author: Karam M. Sallam (karam_sallam@zu.edu.eg)

ABSTRACT This paper proposes improvements to the binary grey-wolf optimizer (BGWO) to solve the feature selection (FS) problem associated with high data dimensionality, irrelevant, noisy, and redundant data that will then allow machine learning algorithms to attain better classification/clustering accuracy in less training time. We propose three variants of BGWO in addition to the standard variant, applying different transfer functions to tackle the FS problem. Because BGWO generates continuous values and FS needs discrete values, a number of V-shaped, S-shaped, and U-shaped transfer functions were investigated for incorporation with BGWO to convert their continuous values to binary. After investigation, we note that the performance of BGWO is affected by the selection of the transfer function. Then, in the first variant, we look to reduce the local minima problem by integrating an exploration capability to update the position of the grey wolf randomly within the search space with a certain probability; this variant was abbreviated as IBGWO. Consequently, a novel mutation strategy is proposed to select a number of the worst grey wolves in the population which are updated toward the best solution and randomly within the search space based on a certain probability to determine if the update is either toward the best or randomly. The number of the worst grey wolf selected by this strategy is linearly increased with the iteration. Finally, this strategy is combined with IBGWO to produce the second variant of BGWO that was abbreviated as LIBGWO. In the last variant, simulated annealing (SA) was integrated with LIBGWO to search around the best-so-far solution at the end of each iteration in order to identify better solutions. The performance of the proposed variants was validated on 32 datasets taken from the UCI repository and compared with six wrapper feature selection methods. The experiments show the superiority of the proposed improved variants in producing better classification accuracy than the other selected wrapper feature selection algorithms.

INDEX TERMS Grey-wolf optimizer, feature selection, simulated annealing, mutation strategy.

I. INTRODUCTION

Feature selection (FS) plays a crucial role in exploring datasets to eliminate noisy, redundant, and irrelevant data that prevent machine learning algorithms (MLA) from achieving better classification or clustering accuracy. Generally, FS seeks to minimize data dimensionality to reduce training time needed by MLA, to obtain better accuracy for both classification and clustering models, to improve prediction capability, and to understand data better for different machine learning applications [1], [2]. Many problems and applications need to preprocess datasets by FS including: text categorization [3], image classification [4], genomics [5], cancer detection [6], and many others [7]–[12].

FS approaches are classified into two categories: wrapper-based approaches [13], and filter-based approaches [13]. In wrapper-based approaches, the FS tools use MLAs to check the accuracy of the selected subset of features until reaching a subset of features with the highest possible accuracy. Unfortunately, wrapper methods are not effective with datasets that have high-dimensions due to the time needed by MLAs to be trained on each selected subset of features until reaching the near-optimal subset that could achieve the highest accuracy [14]. Consequently, new approaches, namely

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Wei Tsai.

filter-based approaches, were proposed to eliminate using MLAs when selecting the near-optimal subset of features and alternatively use statistical data dependency techniques to reach the best subset faster. Note that the filter-based approaches are faster, more scalable, and less computationally expensive than the wrapper-based approaches, but unfortunately are less accurate [15]. Due to the high accuracy that can be achieved with wrapper-based approaches, this paper proposes such an approach for selecting the near-optimal subset of the features.

Due to the ability of meta-heuristic algorithms to solve many real problems [16]–[19] in less time with higher accuracy, they are widely used for FS to resolve a time complexity problem required by the traditional techniques such as mutual information, information gain, relief, depth search, and breadth search. Recently, several meta-heuristic algorithms have been proposed for tackling this problem, such as gradient descent algorithm (GDA) [20], tabu search [21], quantum-based whale optimization algorithm (QWOA) [22], improved binary sailfish optimizer (BFO) [23], novel chaotic crow search algorithm (CCSA) [24], novel chaotic selfish herd optimizer (CSHO) [25], chaotic dragonfly algorithm (CDFA) [26], and fish swarm optimization (FSO) [27], S-shaped binary whale optimization algorithm (BWOA) [28], grey wolf optimizer with a two-phase mutation strategy (GWOTM) [29], binary particle swarm optimization with time-varying inertia weight strategies [30], Gaussian mutational chaotic fruit fly-built optimization (MCFOA) [31], and a discrete binary version of the particle swarm optimization (BPSO) [32]. Major applications of those algorithms are surveyed within the rest of this section.

In [21], tabu search (TS) integrated with the binary particle swarm optimization (BPSO) has been proposed for FS, where BPSO is used as a local optimizer for each iteration. In addition, QWOA [22] has been proposed for FS, integrating the quantum concepts with the WOA. The exploration and exploitation capabilities of WOA were improved in QWOA using the quantum bit representation of the whales within the populations and the quantum rotation gate operator as a variation operator. In [23], a binary variant of the sailfish optimizer (BFO) has been developed for FS. BFO used the sigmoid function to transform the continuous values generated by the standard version into discrete/binary values until the enabling of solving the FS problem that is deemed as a binary one. In [32], a BPSO has been proposed for FS. Mafarja *et al.* [30] proposed a binary version of PSO with time-varying inertia weight for FS. This approach studied the impact of different time-varying of the inertia weight on the performance of the BPSO to balance between the exploration and the exploitation operators.

Sayed *et al.* [24] proposed the crow search algorithm improved using a chaotic map and, in [25], a binary variant of the selfish herd optimizer (BSHO) has been proposed for FS and improved using various chaotic maps to get rid of the local optima that fall into the standard selfish herd optimizer (SFO). Sayed *et al.* [26] proposed a new dragonfly algorithm improved using ten chaotic maps to control the parameters of the dragonflies' movement within the optimization phase to increase the convergence toward the best solution. Anand and Arora [25] employed fish swarm optimization (FSO) for FS.

A binary version of the whale optimization algorithm (BWOA) [28] was proposed for FS, using the S-shaped transfer function to convert the continuous values generated each iteration by the standard WOA into binary values used to solve binary FS problems. Abdel-Basset *et al.* [29] proposed a new version of the grey wolf optimizer enhanced using a novel strategy, namely a two-phase mutation (TM) strategy, for FS. TM's two phases first minimize the number of features while preserving the classification accuracy or improving it, and then work on maximizing the classification accuracy by neglecting the number of features because the main objective of the FS tools to find the optimal subset of features that maximize the accuracy.

In [31], a binary version of fruit fly optimization algorithm (BFOA) improved using the Gaussian mutation operator to reduce the early convergence and boost the exploitation capability of the classical variant, in addition to using also chaotic search as a local search strategy to enhance the searching ability locally of the agents within the swarm. In [14], both opposition-based learning and the differential evolution algorithm are combined with the binary variant of the moth-flame optimization algorithm (BIMFO) for FS. Within BIMFO, the opposition-based learning is utilized to initialize the population optimally to increase the convergence of this algorithm. In addition, it also used the differential evolution to boost the exploitation ability of the approach until increasing the convergence toward the best solution and subsequently reaching better solutions. In [33], the simulated annealing (SA) and bitwise operations are combined with the Harris hawks optimization algorithm (HHASA) to solve FS problems for classification purposes under wrapper-based methods. With experiments conducted in the literature on some recent robust wrapper feature selection algorithms, we notice that they still suffer from falling into local minima, and subsequently they cannot reach the near-optimal number of the features that could achieve better classification accuracy [34].

The whale optimization algorithm (WOA) proposed five years ago has been widely applied for tackling this problem, some of those applications will be reviewed within this paragraph. A binary version of the whale optimization algorithm (BWOA) [28] was proposed for FS, using the S-shaped transfer function to convert the continuous values generated each iteration by the standard WOA into binary values used to solve binary FS problems. In addition, QWOA [22] has been proposed for FS, integrating the quantum concepts with the WOA. The exploration and exploitation capabilities of WOA were improved in QWOA using the quantum bit representation of the whales within the populations and the quantum rotation gate operator as a variation operator. Furthermore, Integration WOA with the simulated annealing has been proposed by Mafarja to explore the regions around the best-so-far solution obtained by WOA at the end of each

iteration to improve its exploitation operator. Also, Mafarja and Mirjalili [35] improved the classical WOA using the tournament and roulette wheel selection techniques instead of randomization in the optimization process to improve the exploration, in addition to borrowing the mutation and crossover operators to promote its exploitation for reaching better outcomes. There are several other WOA variants for the FS problem like Quantum based whale optimization algorithm [22], WOA with hyperbolic tangent fitness function [36], improved WOA [37], and several else [28], [38].

There are much more metaheuristic algorithms tackled the FS problem, such as sine cosine optimization algorithm (SCA) [39]–[41], salp swarm algorithm (SSA) [42]–[44], bat algorithm (BA) [45]–[47], genetic algorithm (GA) [7], [48], [49], flower pollination algorithm (FPA) [50]–[53], cuckoo search (CS) algorithm [54]–[57], differential evolution (DE) [41], [58]–[62], marine predators algorithm (MPA) [63], equilibrium optimizer (EO) [64]–[68], slime mould algorithm (SMA) [69], [70], spotted hyena optimization algorithm [71], [72], emperor penguin optimizer [73], cat swarm optimization algorithm (CSA) [74], [75], harmony search [76]–[81], firefly optimization algorithm (FFA) [82], [83], Chaotic vortex search algorithm [84], crow search algorithm (CSA) [85]–[88], grasshopper optimization algorithm [89]–[92], bacterial foraging algorithm [93]–[95], and dragonfly algorithm [96]–[99].

With experiments conducted in the literature on some recent robust wrapper feature selection algorithms, we notice that they still suffer from falling into local minima, and subsequently they cannot reach the near-optimal subset of the features that could achieve better classification accuracy. As a result, in this paper, a strong metaheuristic algorithm known as grey wolf optimizer (GWO) is effectively improved to propose a new feature selection technique with better performance helping to alleviate the aforementioned drawbacks to the existing technique. It is worth mentioning that GWO has been proposed for tackling the FS problem within several papers, some of which are a hybrid binary grey wolf with Harris Hawks optimizer (HBGWOHHO) [100], binary optimization using hybrid grey wolf optimization (BGWOPSO) [101], binary multi-objective grey wolf optimizer for feature selection (MOGWO-S) [102], review of grey wolf optimizer-based feature selection [103], and several else [104]–[111].

Therefore, in this paper, we propose three binary variants of the grey wolf optimizer (BGWO) in addition to the standard one. The first variant improves the performance of the standard (BGWO) by integrating the exploitation capability with a certain probability within the optimization process to help BGWO to move out of the local minima, this variant is called IBGWO. The second variant, called LIBGWO, improves the performance of IBGWO by integrating it with a novel strategy, the linearly increased worst solutions mutation strategy, to find the worst n, increasing with the iteration, of solutions and to update them toward the best solution and randomly within the search space based on a certain probability. Finally, the last variant combines SA with LIBGWO on the best-so-far solution at the end of each iteration to find a better solution. the efficacy of our proposed algorithms is validated on 32 datasets taken from the UCI repository, in addition to comparing their performance with a number of robust recent wrapper-based FS meta-heuristic algorithms, such as HHASA [33], BA [112], WOA [28], PSO [32], NLPSO [30], GWOTM [29], and genetic algorithm (GA) to measure their superiority. The main contributions of this paper are as follows:

1) Proposal of three variants of BGWO: IBGWO, LIBGWO, and LIBGWO_SA in addition to the standard version under various transfer functions for FS.
2) Comparison of the proposed variants with six wrapper-based FS methods on 32 well-known datasets taken from the UCI repository.
3) Evidence that LIBGWO, LIBGWO_SA could be effective for FS problems in comparison with selected wrapper-based FS algorithms.

The rest of this paper is organized as follows. Section II describes the standard grey wolf optimizer and an SA algorithm. In section III, describes our proposed variants. Section IV determines the performance of the proposed algorithms under some experiments and their discussions. Section V gives a summarization of our experiments and provides some discussions. Section VI provides conclusions about the current work and makes some observations regarding future work.

II. THE USED OPTIMIZATION METHODS

This section discusses the details of the algorithms that have been used to build the main steps of the proposed algorithm.

A. STANDARD GREY WOLF OPTIMIZER

In [113], the authors proposed a novel meta-heuristic algorithm called the grey wolf optimizer (GWO) inspired by the nature of grey wolves when searching, encircling, and catching their prey. In GWO, grey wolves are categorized based on their dominance and leadership into four types: alpha α, beta β, delta δ, and omega ω, where α wolf is considered the best solution found so far, β is the second best one, δ is the third best one, while ω represents the rest wolves. During the hunt of the prey, the wolves can mathematically encircle them using the following model:

$$\overrightarrow{D} = |\overrightarrow{C}.\overrightarrow{X}_p - \overrightarrow{X}(t)| \quad (1)$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X}_p(t) - \overrightarrow{D}.\overrightarrow{A} \quad (2)$$

where $\overrightarrow{X}(t)$, and $\overrightarrow{X}_p$ represent the position of the prey and the grey wolf in the current iteration, t, respectively. $\overrightarrow{D}$ is a vector to contain the absolute of the difference between the prey vector $\overrightarrow{X}_p$ multiplied in the coefficient vector $\overrightarrow{C}$ to avoid local minima and the grey wolf $\overrightarrow{X}(t)$. $\overrightarrow{C}$ is generated using the following equation:

$$\overrightarrow{C} = 2 \times \overrightarrow{r}_1 \quad (3)$$

where $\overrightarrow{r}_1$ is a vector generated randomly between 0 and 1. In relative to the coefficient vector $\overrightarrow{A}$, it is a factor that controls in the exploration and exploitation capability within the optimization process and is formulated as:

$$\overrightarrow{A} = 2 \times \overrightarrow{a} \times \overrightarrow{r}_2 - \overrightarrow{a} \tag{4}$$

where $\overrightarrow{r}_2$ is a vector randomly assigned within 0 and 1, $\overrightarrow{a}$ is a distance control factor that starts with a large value equal to 2 and reduces linearly until reaching 0. Generally, $\overrightarrow{a}$ could be generated using the following equation:

$$\overrightarrow{a} = 2 - 2 \times \frac{t}{t_{max}} \tag{5}$$

where t refers to the current iteration, and t_{max} expresses the maximum iteration. The previous mathematical model simulates the encircling behavior of the grey wolves. GWO proposed that the best three grey wolves so far (α, β, δ, and ω) know the potential position of the prey and the other grey wolves will update their position according to the best three grey wolves. Mathematically, the hunting mechanism of the grey wolves is formulated as follows:

$$\overrightarrow{X}(t+1) = (\overrightarrow{x}_1 + \overrightarrow{x}_2 + \overrightarrow{x}_3)/0.3 \tag{6}$$

$$\overrightarrow{X}_1 = \overrightarrow{X}_\alpha - \overrightarrow{A}_1.\overrightarrow{D}_\alpha, \quad \overrightarrow{D}_\alpha = |\overrightarrow{C}_1.\overrightarrow{X}_\alpha - \overrightarrow{X}| \tag{7}$$

$$\overrightarrow{X}_2 = \overrightarrow{X}_\beta - \overrightarrow{A}_2.\overrightarrow{D}_\beta, \quad \overrightarrow{D}_\beta = |\overrightarrow{C}_2.\overrightarrow{X}_\beta - \overrightarrow{X}| \tag{8}$$

$$\overrightarrow{X}_3 = \overrightarrow{X}_\delta - \overrightarrow{A}_3.\overrightarrow{D}_\delta, \quad \overrightarrow{D}_\delta = |\overrightarrow{C}_3.\overrightarrow{X}_\delta - \overrightarrow{X}| \tag{9}$$

When the values of $\overrightarrow{A}$ are between 1 and -1, the next position of a grey wolf is in any position between its current position and the prey position. On the contrary, $\overrightarrow{A} > 1$ and $\overrightarrow{A} < 1$ oblige the grey wolves to diverge from the prey to explore other regions in the hope of finding a better prey. The pseudo-code of the GWO is listed in algorithm 1.

B. SIMULATED ANNEALING

Kirkpatrick *et al.* [114] proposed a single-solution optimization algorithm known as simulated annealing (SA) based on a local search method called hill-climbing. One of the most advantages of SA is that it accepts the worst solution with a specific probability until overcoming the local minima problem. At the outset, SA initially randomly spread the points in a single solution (initial solution, S_I) within the search space; after that, in each iteration, a new solution will be generated around the best-so-far solution based on the predefined neighborhood structure and evaluated using the objective function, or also known as the fitness function. If the newly generated solution is better, it is always accepted, whilst the worst one is accepted based on a probability calculated using the Boltzmann probability $P = e^{(-\theta/T)}$, where θ is the difference between the objective values of both the new generated solution (S_n) and the best-so-far one (S_b). T is a temperature that periodically reduces based on some cooling schedule. Within our proposed algorithm, the initial temperature,T_0, is set to 30 to reduce the running time needed by SA, and the cooling schedule is calculated as $T = T \times \alpha$, where $\alpha = 0.5$ also to overcome the time complexity generated using SA as a local search. The steps of SA are listed in Algorithm 2. Within our proposed algorithm, each position within the best solution with a value of 1 is converted into 0 and evaluated to see if it is better or not. However, when the number of features is extremely high this will result in long-running times, so this strategy will be applied with a probability 0.1. This strategy is called a flipping mutation.

Algorithm 1 Grey Wolf Optimizer (GWO)

```
Initialize step;
Initialize a, A, and C;
Generate an initial population of random solutions;
Compute the fitness value for each grey wolf;
Select the best three grey wolves as X_α, X_β and X_delta, respectively;
t ← 0;
while t ≤ t_max do
    Update the position of each grey wolf using Eq. 6;
    Calculate the fitness value of each grey wolf;
    Update a, A, and C;
    Compute the fitness for the grey wolf;
    Update X_α, X_β and X_delta
    t ← t + 1;
end while
Return X_α
```

Algorithm 2 Simulated Annealing (SA)

```
T_0 = 30, S_b = S_I;
S_b_old indicates the old solution;
T_F = 0.01.
while T > T_F do
    Create a new solution S_n around S_b_old using the flipping mutation;
    Compute the fitness for (S_n);
    if Fit(S_n) < Fit(S_b) then
        S_b = S_n;
    else
        θ = Fit(S_n) − Fit(S_b);
        P = e^(−θ/T)
        r_1 is a number generated randomly between 0 and 1;
        if r_1 < θ then
            S_b_old = S_n;
        end if
    end if
    T = T × 0.5;
end while
Return S_b and Fit(S_b).
```

III. THE PROPOSED ALGORITHM: IBGWO ALGORITHM

Within this section, the improved grey wolf optimizer will be adapted to FS problems. Generally, the main steps of this algorithm are abbreviated as initialization, transfer function,

evaluation, and improvement methodology. In detail, those steps are illustrated within the following subsections.

A. INITIALIZATION

GWO, like most meta-heuristic algorithms, creates at the outset a population of size n with a number of dimensions d for each member within the population, and then those dimensions are initialized according to the nature of the problems. Since FS problems are discrete, the dimensions with a size d equal to the length of the features within the datasets will be randomly initialized with 0 and 1 to mark the selected features to identify the optimal subset of the features that could reach better classification/clustering accuracy. For more illustration see Fig. 1, which shows how to initialize the solutions when solving the FS problem. This figure shows that the feature with 0 value within its corresponding position is not selected.

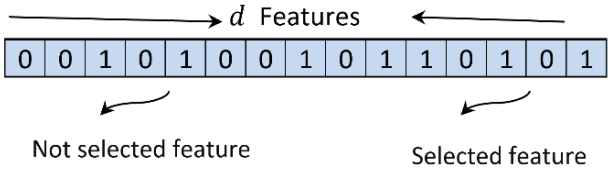


FIGURE 1. Illustration of the initialization steps.

B. TRANSFER FUNCTIONS

Any meta-heuristic algorithm that generates continuous values could not be used to solve the discrete FS problem. As a result, the transfer function, also known as a transformation function, has been used to normalize the continuous values, generated by the meta-heuristic algorithm, between 0 and 1. The normalized values are then converted into 0 and 1 according to Eq.10.

Of interest then is the type of transfer function and its utility in the normalization. Transfer functions are divided into three types: V-shaped, S-shaped, and U-shaped as illustrated in Table 1 and depicted graphically in Fig. 2. The U-shaped transfer function has recently been proposed to tackle the mission of normalizing continuous values using Eq. 11, which contains two parameters: α and β. α indicates the slope of the transfer function and β refers to the width of the basin of the transfer function.

$$V_j = \begin{cases} 1 & \text{If } V_j > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$U(x) = \alpha |x^{\beta}| \tag{11}$$

C. EVALUATION

For the FS problem, two objectives must be achieved: the first is maximizing the classification/clustering accuracy, and the second is minimizing the number of features until reaching the smallest possible number of features that could achieve better classification accuracy. Subsequently, the function for evaluating each solution to solve the FS problem has to achieve the two conflicting weighted objectives at the same time, and this function is mathematically modeled in Eq. 12.

$$f = \alpha \times \gamma_R(D) + \beta \times \frac{|S|}{|N|} \tag{12}$$

where $\gamma_R(D)$ is the rate of the classification error calculated using a k-nearest neighbor (KNN) classifier [115], $|S|$ indicates the selected features number, $|N|$ is the length of all features in the dataset, and β and α are two scalar values located between 0 and 1 to identify the weight of each objective, $\alpha \in [0, 1]$, and $\beta = 1 - \alpha$. In this paper, KNN has been used to check the classification accuracy based on the selected features due to its simplicity, low time complexity and its efficiency [116].

To train the KNN classifier, the dataset is divided into training and testing datasets according to the holdout method [34], where 80% of the original dataset is used as the training set, and the remaining 20% is used as the testing dataset. After each iteration, the number of features within each solution is evaluated using KNN on the training dataset. Then, after the training process, each record in the testing dataset is evaluated to see if the model could reach better classification accuracy under the obtained number of features.

TABLE 1. S-shaped and V-shaped families of TFs.

S-shaped family		V-shaped family	
Name	Transfer function	Name	Transfer function
S_{v1}	$TF(x) = \frac{1}{1+exp^{-x}}$	V_{v1}	$TF(x) = \lvert\tanh(x)\rvert$
S_{v1c}	$TF(x) = \frac{1}{1+exp^{x}}$	V_{v2}	$TF(x) = \left\lvert \text{erf}\left(\frac{\sqrt{\pi}}{2}x\right)\right\rvert$
S_{v2}	$TF(x) = \frac{1}{1+exp^{(-x/2)}}$	V_{v3}	$TF(x) = \left\lvert (x)/\sqrt{1+x^2}\right\rvert$
S_{v3}	$TF(x) = \frac{1}{1+exp^{(-x/3)}}$	V_{v4}	$TF(x) = \left\lvert \frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)\right\rvert$
S_{v4}	$TF(x) = \frac{1}{1+exp^{-2x}}$		

D. IMPROVED GWO (IBGWO)

Within the optimization process, the algorithms will search for a better solution. At the start it tries to explore most regions within the search space; then the exploration is converted into exploitation until the algorithm focuses on the best solution found so far in the hope of finding better solutions around it. However, if the best solution found so far is a local optimum, GWO will focus on it and subsequently won't find a better solution because the optimal solution is found in another region. As a result, we propose a strategy that helps the GWO in particular, and any meta-heuristic in general, to update the current solution that is located within a certain probability randomly within the search space area. The pseudo-code of the improved binary variant of GWO (IBGWO) for FS is shown in Algorithm 3.

E. HYBRID IBGWO WITH A LINEARLY INCREASED WORST SOLUTION MUTATION STRATEGY (LIBGWO)

Improving the worst solution may accelerate the convergence towards the optimal solution so, in this paper, we propose

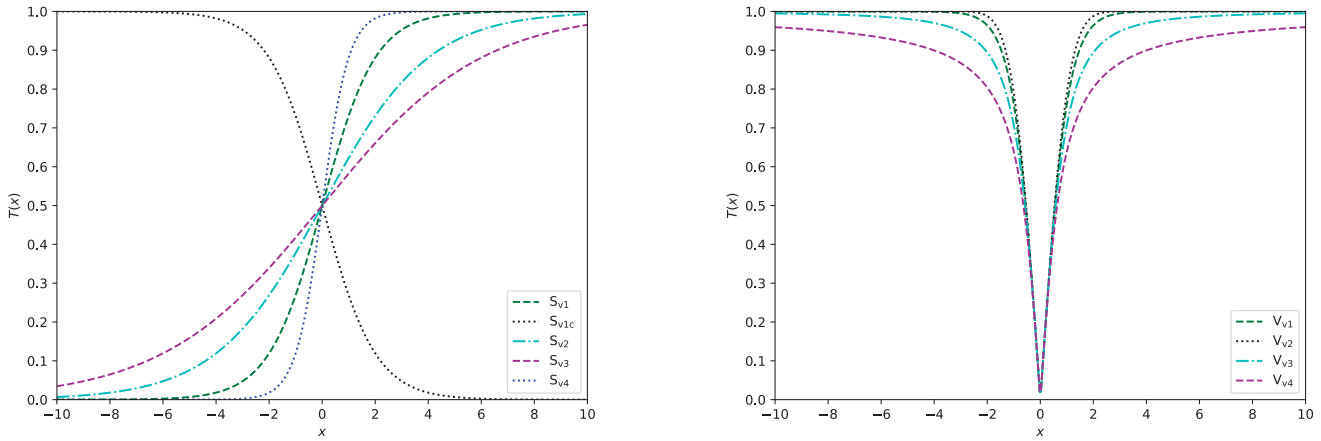


FIGURE 2. Transformation Functions (a) S-shaped; and (b) V-Shaped.

Algorithm 3 IBGWO

Initialize step;
Initialize a, A, and C;
Generate an initial population of random solutions (X);
Convert each X_i into binary one using any of the transfer functions;
Compute the fitness of each grey wolf $\overrightarrow{X}_i$ using Eq. 12, $i = 0, 1, 2, \ldots, n$;
Select the best three grey wolves as X_α, X_β and X_{delta}, respectively;
$t \leftarrow 0$;
while $t \leq t_{max}$ **do**
  for $i = 1 : n$ **do**
    Generate random number r_1 to determine the exploration rate;
    if $r_1 < ER$ **then**
      Update the position of i^{th} grey wolf randomly within the search space
    else
      Update the position of i^{th} grey wolf using Eq.6;
    end if
  end for
  Calculate the fitness value of each grey wolf, $\overrightarrow{X}_i$;
  Update a, A, and C;
  Convert each $\overrightarrow{X}_i$ into binary one using any of transfer functions;
  Compute the fitness for the grey wolf $\overrightarrow{X}_i$ using Eq. 12;
  Update X_α, X_β and X_{delta}
  $t \leftarrow t + 1$;
end while
Return X_α

a strategy that selects a number of the worst particles *wp* increased with the iteration linearly using the following equation:

$$wp = n \times \frac{t}{t_{max}} \tag{13}$$

After calculating *wp*, those worst solutions will be updated toward the best solution if r_3 is smaller than a certain mutation probability (MP), otherwise randomly within the search space (see Fig.2). This strategy is known as the linearly increased worst solutions mutation strategy. Note that *wp* increases linearly with the iteration

FIGURE 3. Worst solutions mutation strategy.

Fig. 3 depicts our strategy used to improve a number of *wp* of the worst solutions. In this figure, the green cell refers to the positions updated randomly within the search space, whilst the other positions are set with the same values within the corresponding position in the best solution. Finally, Algorithm 4 illustrates the pseudo-code of LIBGWO.

F. HYBRID LIBGWO-SA

This subsection outlines the steps of adapting LIBGWO integrated with SA. Searching around the best solution may find the best number of selected features that may maximize the classification accuracy. Therefore, SA is integrated with LIBGWO at the end of each iteration to exploit the regions around the best solutions in the hope of finding better solutions nearby. The steps of integrating LIBGWO with SA are illustrated in Fig.4.

IV. RESULTS AND DISCUSSION

In this section, the proposed algorithms are validated on a number of well-known datasets and compared with a number

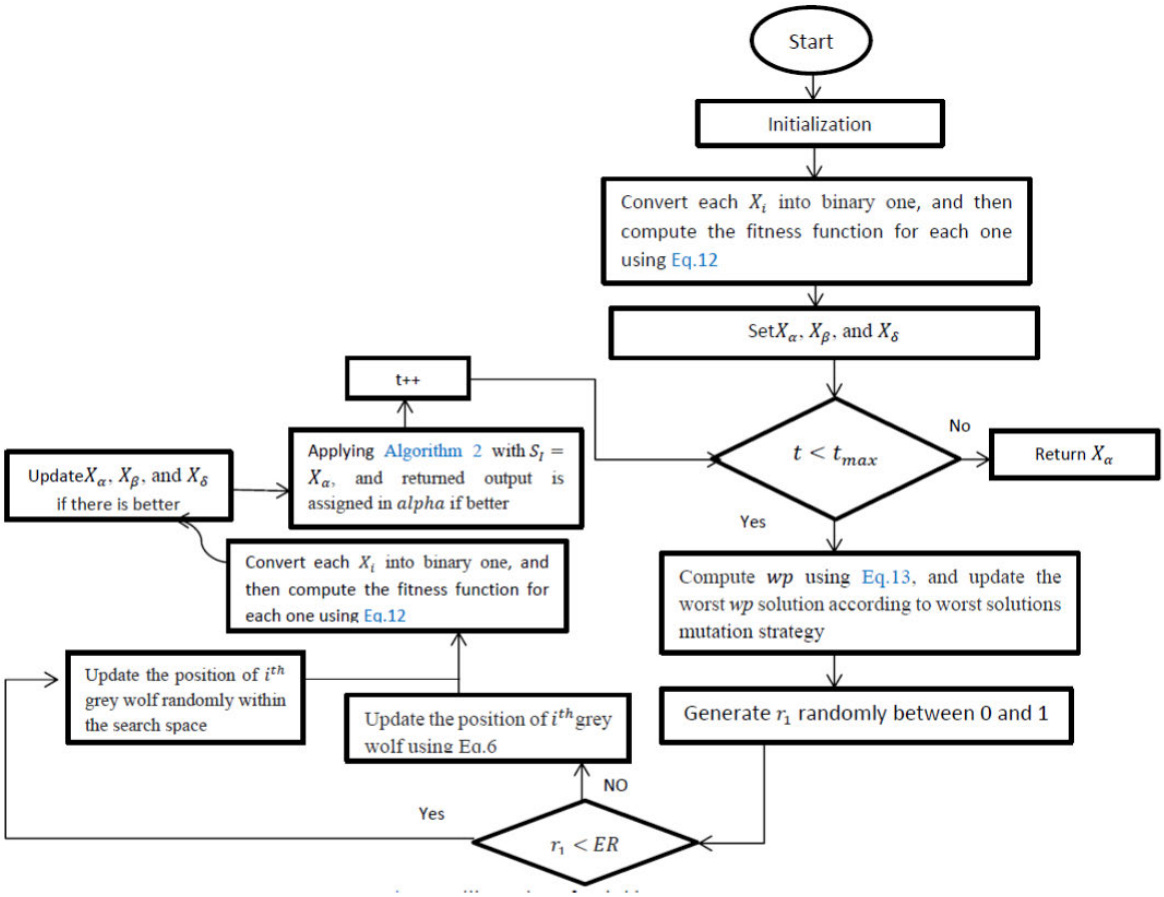


FIGURE 4. An illustration of hybrid LIBGWO-SA.

of selected state-of-the-art algorithms to determine their superiority compared with those algorithms. Our experiments and their settings are organized within this section under the subsections entitled as:

- Section 4.1: Datasets description.
- Section 4.2: Performance metrics.
- Section 4.3: Parameter settings.
- Section 4.4: Investigating the different transfer functions performance.
- Section 4.5: Comparing the different variants of GWO when solving FS.
- Section 4.6: Comparing the proposed with some state-of-the-art algorithms.
- Section 4.7: Comparison under interval plot.

A. DATASETS DESCRIPTION

Most papers discussed in the literature assessed the performance of their techniques based on a collection of well-known instances with various scales (small, medium, and large) to check their stability in addition to their ability to find better results [73], [84], [117]–[121]. After observing those papers, we found that most employed instances were taken from the UCI repository. Therefore, most of those instances in addition to others taken also from this repository with a number of up to 32 instances have been here employed to validate our proposed algorithm compared to some of the rival algorithms discussed later. As aforementioned that we employed those instances, as they have been widely used in the literature as an attempt to achieve a fair comparison within our experiments [29], [30], [33], [36], [73], [84], [117]–[121]. Generally, those employed instances are described in Table 2 to illustrate their characteristics such as the number of features (#F), number of classes (#C), and number of samples (#S). additionally, These datasets are found online at https://www.openml.org/search.

B. PERFORMANCE METRICS

The performance of the proposed algorithm is evaluated using Classification accuracy using KNN, Fitness values, the selected number of features and standard deviation metrics within 30 runs.

1) CLASSIFICATION ACCURACY USING KNN

After finishing the optimization process, the subset of features obtained are investigated under the NN algorithm to see by how much this subset improves the accuracy of classification. After calculating the classification accuracy

TABLE 2. Description of well-known datasets.

ID#	Dataset	#F	#S	#C	ID#	Dataset	#F	#S	#C
1	Australian	15	690	2	**17**	Clean1	169	476	2
2	Climate	21	540	2	**18**	Abalone	19	846	4
3	Fri_c0_500_10	11	500	2	**19**	heart-statlog	13	270	2
4	Fri_c0_1000_10	11	1000	2	**20**	Ilpd	10	583	2
5	Fri_c1_1000_10	11	1000	2	**21**	Exactly	13	1000	2
6	fri_c1_1000_25	26	1000	2	**22**	exactly2	13	1000	2
7	fri_c2_1000_25	26	1000	2	**23**	m-of-n	13	1000	2
8	Glass	10	214	7	**24**	Waveform	40	5000	3
9	Ionosphere	35	351	2	**25**	water2	39	521	3
10	Segment	20	2310	7	**26**	derm	35	366	6
11	WDBC	31	569	2	**27**	derm2	35	358	6
12	Vote	16	300	2	**28**	Satimg	37	6430	6
13	Sonar	60	208	2	**29**	Clean2	169	6598	2
14	liver_numeric2	11	583	2	**30**	Page-blocks	11	5473	2
15	Lsvt	310	126	2	**31**	Spect	45	267	2
16	Breast cancer issue	9	699	2	**32**	Vehicle	19	846	2

Algorithm 4 LIBGWO

Initialize step;
Initialize a, A, and C;
Generate an initial population of random solutions (X);
Convert each X_i into binary one using any of the transfer functions;
Compute the fitness of each grey wolf $\overrightarrow{X}_i$ using Eq. 12, $i = 0, 1, 2, \ldots, n$;
Select the best three grey wolves as X_α, X_β and X_δ, respectively;
$t \leftarrow 0$;
while $t \leq t_{max}$ **do**
  Compute wp using Eq. 13;
  Select the worst wp and update it toward the best or randomly based on MP;
  for $i = 1 : n$ **do**
    Generate random number r_1 to determine the exploration rate;
    if $r_1 < ER$ **then**
      Update the position of i^{th} grey wolf randomly within the search space
    else
      Update the position of i^{th} grey wolf using Eq.6;
    end if
  end for
  Calculate the fitness value of each grey wolf, $\overrightarrow{X}_i$;
  Update a, A, and C;
  Convert each $\overrightarrow{X}_i$ into binary one using any of the transfer functions;
  Compute the fitness for the grey wolf $\overrightarrow{X}_i$ using Eq. 12;
  Update X_α, X_β and X_{delta}
  $t \leftarrow t + 1$;
end while
Return X_α

within 30 runs using KNN, the best, average (Avg), and worst values within those 30 runs is calculated and used to validate and compare the performance of the different algorithms, including the proposed ones. In addition, among all the algorithms, the rank of each algorithm in each instance is calculated using the Avg classification accuracy value. The algorithm with the highest accuracy is the best.

2) FITNESS VALUES

Eq. 12 guides the optimization process to the optimal solution, where it is used to measure the fitness of each solution; the one with the smallest fitness is considered the best and used within the optimization process to guide the other solutions to the region which may contain the optimal solutions. The algorithms are run 30 independent runs and the fitness values obtained at the end of each run is compared to extract the best and the worst values, summed to get the Avg, and ranked under Avg value in comparison with all the other algorithms to be utilized within our experiment to evaluate the performance of the algorithms.

3) THE SELECTED NUMBER OF FEATURES

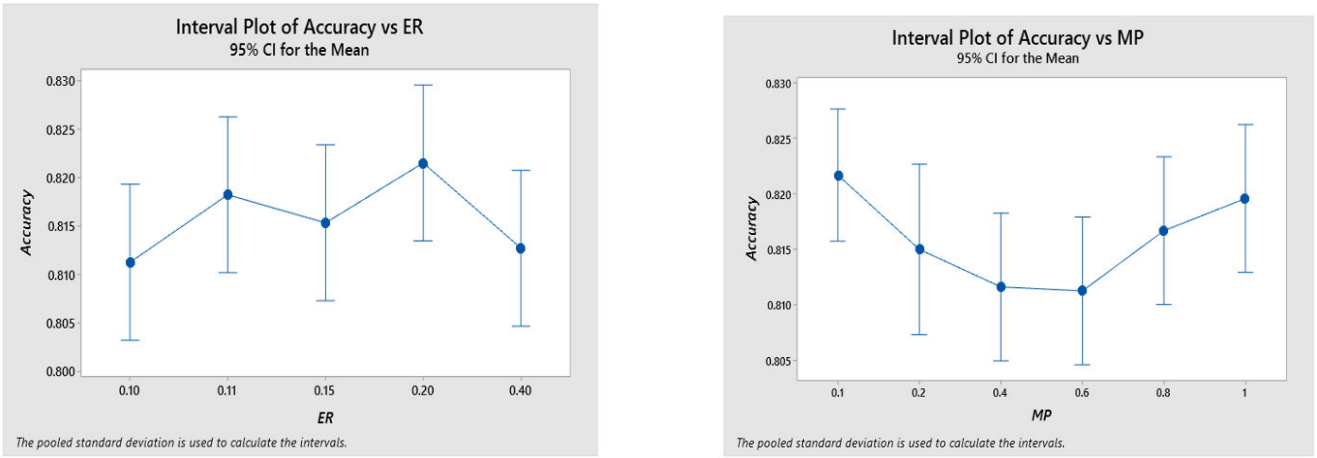
The performance of the algorithms is judged by the number of selected features, but this is not the main metric because a small number of features may still have poor accuracy. As a result, the objective when solving the FS problem is to not only minimize the number of selected features but also to maximize the classification accuracy.

4) STANDARD DEVIATION (SD)

To measure the stability of the algorithms within 30 independent runs, SD is applied to the obtained fitness values to see whether they are converged. The algorithm with the lower SD value has the highest stability. Mathematically, SD can be calculated as follows:

$$SD = \sqrt{\frac{1}{nr-1}\sum_{i=1}^{nr}(f_i - \bar{f})^2} \tag{14}$$

where nr expresses the number of independent runs (set to 30 within our experiments); f_i is the fitness value of



(a) (b)

FIGURE 5. Depiction of the experiments to pick up the best value of (a) ER; and (b) MP.

the i^{th} run; and $\bar{f}$ indicates the mean of the fitness values obtained within 30 independent runs.

C. PARAMETER SETTINGS

Within this section, our experiments to find the near-optimal parameter values are illustrated to show the effect of the choice of parameter values on the performance of the proposed algorithm. First, we describe the settings and the state-of-the-art algorithms used in our experiments. The population sizes and the numbers of iterations have been unified for all the algorithms to ensure a fair comparison–a maximum of 30 iterations, and population size of 5. All algorithms were implemented using the Java programming language under the same environments. Those algorithms are:

- Binary Harris hawks algorithm combined with SA (HHASA) [33].
- Binary bat algorithm (BA) [112].
- Binary WOA (WOA) [28].
- PSO [32].
- Non-linear PSO (NLPSO) [30].
- Binary grey wolf optimization algorithm with two-phase mutation (GWOTM) [29].
- Genetic algorithm (GA) [30].

The algorithms were run on a device with 32 GB of RAM and core i7 2.40GHZ Intel CPU using Windows 10.

The first significant issue is the selection of the best values for both exploration rate (ER), and the mutation probability (MP) within the proposed algorithms. ER is used to prevent falling into local optima by updating randomly the position of the grey wolf within the search space, enabling them to explore another region that contains more prey. This factor must be selected carefully due to its effect on the optimization process, so we conducted several experiments with values of 0.10, 0.11, 0.15, 0.2, and 0.4 for ER on ID#1. The results of the experiments under One-way ANOVA with 95% as a confidence level within 30 independent runs are shown in Fig.5 from which it can be seen that 0.2 as the best value for ER.

MP takes values between 0, and 1 and specifies the proportion of mutating the current position within the worst particle with a random value while the other positions will be assigned with the corresponding position within the best solution. To select MP, a number of values such as 0.1, 0.2, 0.4, 0.6, 0.8, and 1.0 were evaluated using D#1 within 30 independent runs. Using One-way ANOVA with a confident level of 95%, Fig.6 shows that 0.1 is the best value. As outlined earlier, the population size and the number of iterations are set to 5 and 30, respectively. K-neighbors, α, and β are adopted as in [29]. Finally, Table 3 shows the parameter values of the proposed algorithm.

TABLE 3. Parameter values of the proposed algorithm.

Parameter	Values	Parameter	Values
Number of independent runs	30	a	0.01
Number of iterations	30	b	0.99
Population Size	5	MP	0.1
K-neighbors	5	ER	0.2

D. COMPARISON OF THE PERFORMANCE OF TRANSFER FUNCTIONS

Within this section, the performance of the different transfer functions is investigated when integrating with GWO to convert its continuous values into binary values on the datasets ID#1-ID#14. In the following tables through this section the header techniques are labeled as BGWO concatenated with the name of the transfer function used; for example, BGWOS2 signifies the algorithm mapped using the S2 transfer function. The rest of this section is structured as follows:

1) Comparison under fitness values.
2) Comparison under classification accuracy.
3) Comparison under the number of selected features.

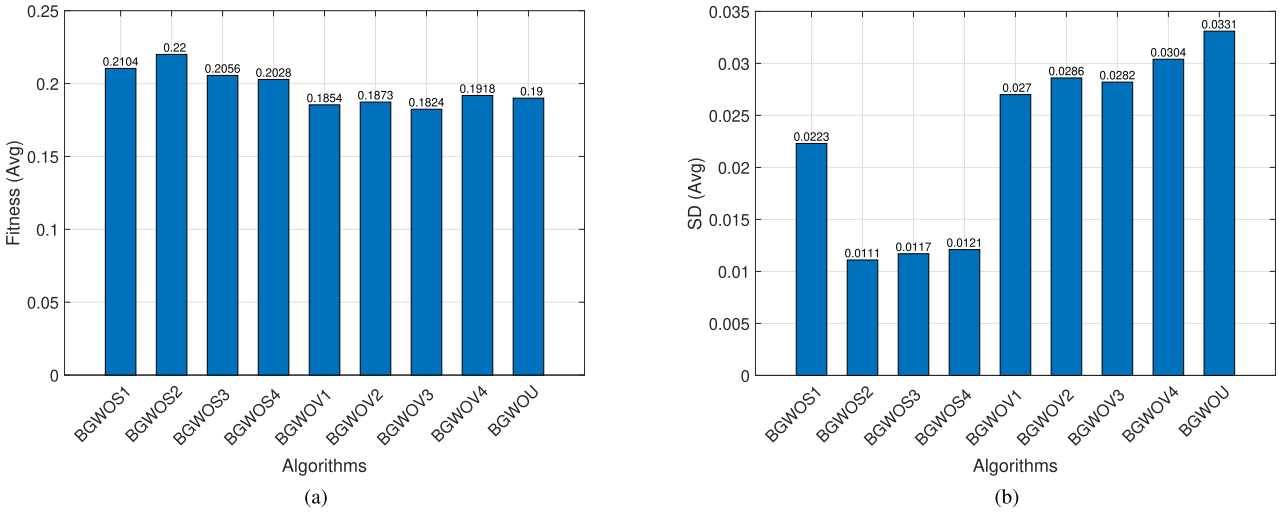


FIGURE 6. (a) Average fitness values obtained on ID#1-ID#14 and (b) Average SD values of fitness values obtained on ID#1-ID#14.

1) COMPARISON UNDER FITNESS VALUES

The fitness values obtained by integrating each transfer function with GWO have been herein presented to extract which one could fulfill better outcomes with GWO. Therefore, each one has been run 30 independent times on each instance between ID#1 and ID#14 and the average of the fitness values on all those instances have been presented in Fig.6 (a) which shows that the transfer function V_3 is the best with the lowest fitness value of 0.1824, while S_2 is the worst with a value of 0.2200. To measure a more stable transfer function, the average SD within the same runs was calculated and exposed in Fig. 6 (b), which shows the superiority of the S_2 in reaching stable outputs within the independent runs. At the end of this section, it is concluded that V_3 is the best but only in terms of fitness which is composed of two objectives: accuracy and the number of selected features, therefore there is a wonder which one of those objectives is better if the accuracy then the main target has been achieved since the machine learning techniques relies basically on accuracy as the first objective while the number of selected feature as the second one. Based on that, the accuracy obtained by various transfer functions must be compared to see if V_3 could come true better accuracy or not.

2) COMPARISON OF CLASSIFICATION ACCURACY

To determine that the transfer function could reach the near-optimal subset of features under classification accuracy, KNN, due to its simplicity and efficiency, is used to measure the classification accuracy of the selected features obtained by each transfer function within 30 independent runs, and then the average of the classification accuracy value within 30 runs was calculated and displayed in Fig. 7 (a) to find that, as with fitness results, V3 could obtain the optimal value. Fig. 7 (b) depicts the average of SD obtained by each algorithm, from which it can be seen that S2 is more stable within 30 runs. Based on that, integrating V3 with GWO could achieve the optimal subset of features that reaches better classification accuracy, therefore, V3 is the best since its performance in terms of classification accuracy is better.

3) COMPARISON UNDER THE NUMBER OF SELECTED FEATURES

To see the transfer function with the best performance in term of the selected number of features, Fig. 8 (a) is graphically presented to display the average of the selected number of features within 30 runs. According to this figure, V3 is the best, and S1 is the worst. Fig. 8 (b) shows the variant of BGWO that achieves more stable results: BGWOS3 as the best, while BGWOS1 is the worst. Ultimately, V3 could be best in terms of classification accuracy and the number of selected features, so it will be considered with GWO within the next experiments conducted to compare the performance of BGWOV3 with some of its improved variants and the rival algorithms.

E. COMPARISON OF THE PROPOSED VARIANTS

Variants of the binary GWO (BGWO) are proposed in this paper to tackle the FS problem. To determine the best, extensive experiments are performed within this section on datasets ID#1-ID#14. In the first experiment performed under fitness function and shown in Table 4, the superiority of LIBGWO_SAV3 appears in most datasets, but it fails to outperform LIBGWOV3 on ID#1, and ID#12. In addition, we calculate the rank of each algorithm's performance on each dataset, as shown in Table 4 and depicted in Fig. 9 (a) for the average of the ranks on all datasets. Fig. 9 (b) shows that LIBGWO_SAV3 performs best with a value of 1.28, LIBGWOV3 is second best with 2.000, while BGWOV3 is worst with a value of 3.71. Fig. 9 (b) shows the average of

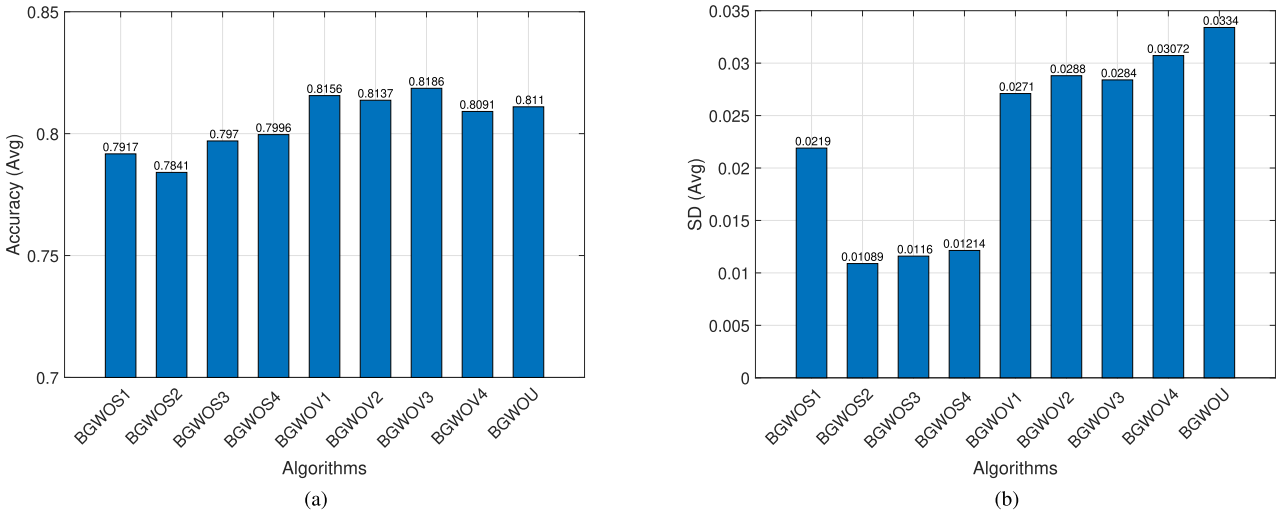


FIGURE 7. (a) Average classification accuracy obtained on ID#1-ID#14 and (b)Average SD values of classification accuracy on ID#1-ID#14.

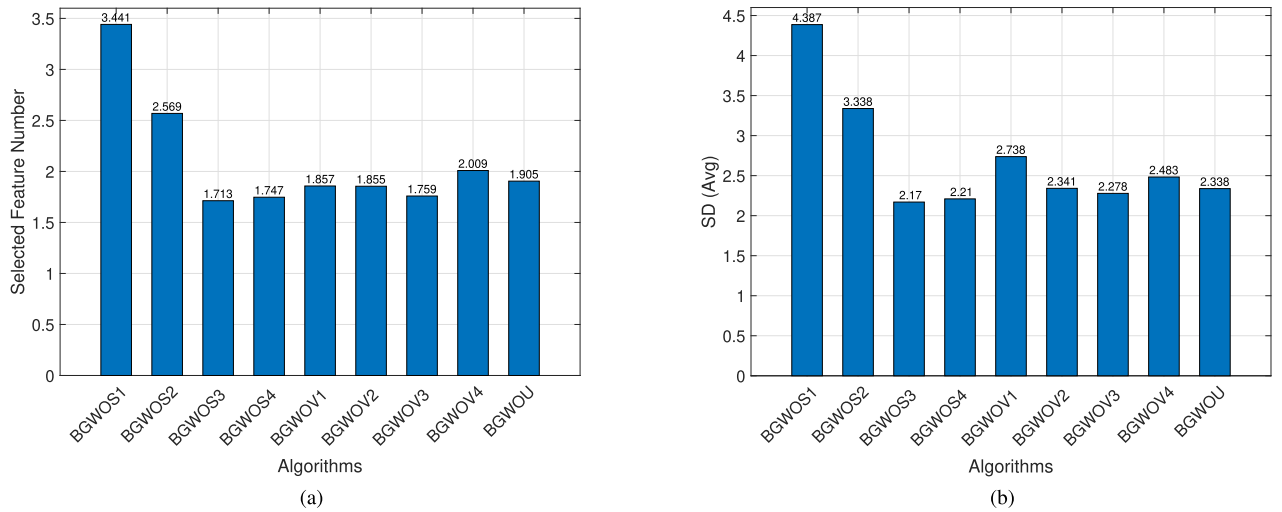


FIGURE 8. (a) Average classification accuracy obtained on ID#1-ID#14 and (b)Average SD values of classification accuracy on ID#1-ID#14.

the SD values obtained on all instances using each proposed variant; LIBGWO_SAV3 is the best while BGWOV3 is the worst. From an analysis from the perspective of fitness value, it is clear that all the proposed improved algorithms perform better than the standard one when solving the FS problem. From an analysis from the perspective of fitness value, it is clear that all the proposed improved algorithms, especially LIBGWO_SAV3 which outperforms all for most instances, perform better than the standard one when solving the FS problem.

After analysis of the performance of the proposed variants of BGWO under the fitness values, here, its performance is evaluated under the classification accuracy. Table 5 shows that LIBGWO_SAV3 outperforms all the proposed variants in most instances except for ID#1 and ID#12 for which LIBGWOV3 performs better. Fig. 10 (a) shows the average of the ranks obtained using each algorithm on each dataset, from which it can be seen that the insights drawn under fitness values are the same under classification accuracy such that as LIBGWO_SAV3 come in the first rank with a value of 1.28, and LIBGWOV3 occupied the second rank, while both IBGWOV3 and BGWOV3 came in the last two ranks, respectively. Fig. 10 (b) shows the average of the SD obtained by each proposed variant in all instances, which shows the superiority of LIBGWO_SAV3 that could reach more stable results with an average SD value of 0.0166, while BGWOV3 performs the worst with an amount of 0.0322.

Finally, the comparison of algorithms is completed under the selected number of features to determine which algorithm achieves the best results under both classification accuracy and fitness and also identify the smallest number of selected features. According to Table 6, under Avg, LIBGWO_SAV3 reaches a better number of selected

TABLE 4. Fitness value obtained using different proposed variants of BGWO.

ID#		BGWOV3	IBGWOV3	LIBGWOV3	LIBGWO_SAV3	ID#	BGWOV3	IBGWOV3	LIBGWOV3	LIBGWO_SAV3
1	**Worst**	0.2826	0.2188	0.1987	0.2174	**8**	0.9451	0.8782	0.8782	0.8782
	Avg	0.2048	0.1867	0.1810	0.1832		0.8913	0.8775	0.8775	0.8775
	Best	0.1549	0.1679	0.1549	0.1679		0.8771	0.8771	0.8771	0.8771
	SD	0.0286	0.0139	0.0110	0.0129		0.0219	0.0005	0.0005	0.0005
	Rank	4	3	**1**	2		2	**1**	**1**	**1**
2	**Worst**	0.0830	0.0763	0.0763	0.0753	**9**	0.0291	0.0157	0.0154	0.0151
	Avg	0.0724	0.0702	0.0647	0.0623		0.0120	0.0034	0.0055	0.0023
	Best	0.0565	0.0493	0.0565	0.0483		0.0012	0.0009	0.0012	0.0009
	SD	0.0062	0.0065	0.0059	0.0082		0.0080	0.0045	0.0061	0.0037
	Rank	4	3	2	**1**		4	2	3	**1**
3	**Worst**	0.2802	0.1624	0.1624	0.1545	**10**	0.0958	0.0391	0.0374	0.0326
	Avg	0.1667	0.1404	0.1353	0.1354		0.0423	0.0289	0.0273	0.0253
	Best	0.1327	0.1327	0.1327	0.1327		0.0262	0.0230	0.0214	0.0208
	SD	0.0305	0.0101	0.0073	0.0064		0.0185	0.0047	0.0036	0.0030
	Rank	4	3	**1**	2		4	3	2	**1**
4	**Worst**	0.2891	0.1694	0.1783	0.1773	**11**	0.0554	0.0564	0.0554	0.0531
	Avg	0.1824	0.1348	0.1285	0.1262		0.0449	0.0406	0.0421	0.0391
	Best	0.1139	0.1099	0.1099	0.1099		0.0281	0.0274	0.0281	0.0271
	SD	0.0403	0.0207	0.0199	0.0227		0.0092	0.0099	0.0089	0.0083
	Rank	4	3	2	**1**		4	2	3	**1**
5	**Worst**	0.1773	0.1397	0.1268	0.1268	**12**	0.1186	0.0196	0.0349	0.0342
	Avg	0.1239	0.1087	0.1055	0.1063		0.0274	0.0179	0.0186	0.0185
	Best	0.0980	0.0980	0.0980	0.0980		0.0178	0.0178	0.0178	0.0178
	SD	0.0252	0.0110	0.0081	0.0071		0.0233	0.0004	0.0033	0.0032
	Rank	4	3	**1**	2		4	**1**	3	2
6	**Worst**	0.2866	0.2164	0.1901	0.1646	**13**	0.4020	0.4970	0.4497	0.3549
	Avg	0.1601	0.1544	0.1362	0.1231		0.2383	0.2672	0.2484	0.1856
	Best	0.0857	0.1105	0.0857	0.0857		0.0483	0.0715	0.0951	0.0714
	SD	0.0531	0.0292	0.0264	0.0180		0.0981	0.0950	0.0927	0.0686
	Rank	4	3	2	**1**		2	4	3	**1**
7	**Worst**	0.2458	0.1818	0.1662	0.1781	**14**	0.2409	0.2325	0.2315	0.2220
	Avg	0.1469	0.1292	0.1250	0.0941		0.2315	0.2182	0.2133	0.2102
	Best	0.0804	0.0804	0.0804	0.0804		0.2071	0.1986	0.1986	0.1986
	SD	0.0433	0.0251	0.0293	0.0273		0.0090	0.0099	0.0098	0.0092
	Rank	4	3	2	**1**		4	3	2	**1**

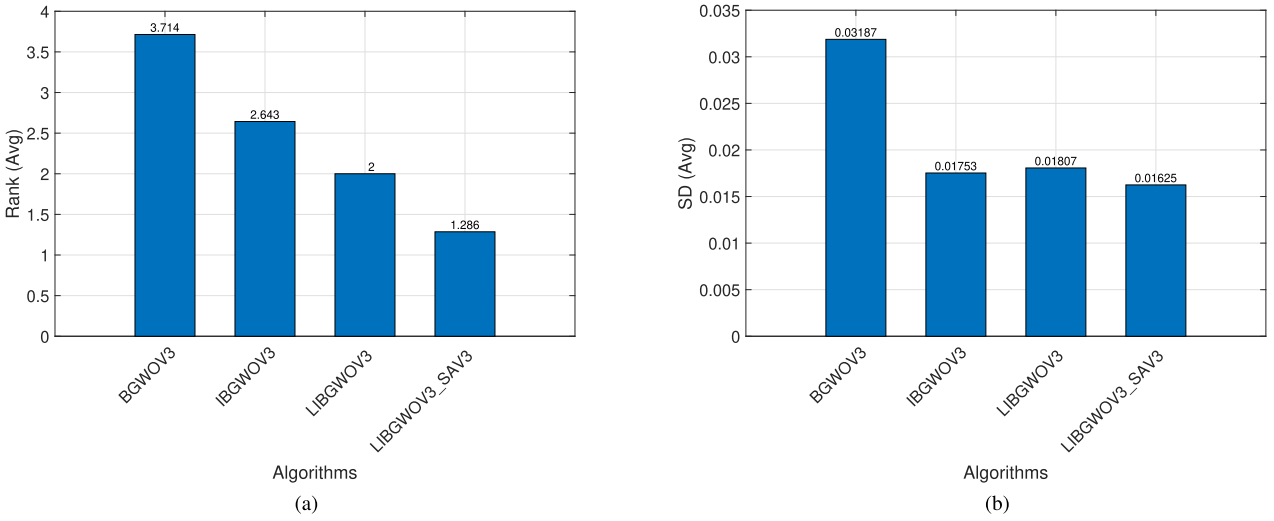


FIGURE 9. (a) Average classification accuracy obtained on ID#1-ID#14 and (b)Average SD values of classification accuracy on ID#1-ID#14.

features on 11 out of 14 datasets, while LIBGWO_SAV3 works better on ID#3and BGWOV3 on ID#4 and ID#13. Fig.11 (a) and (b) show the average of the ranks obtained under the selected features number on all the datasets. LIBGWO_SAV3 is better, where it achieved values of 1.2 and 1.05 for both the average of the ranks and the average of SD, respectively, while both IBGWOV3 and BGWOV3 were the worst for those two terms (average of the ranks and average of SD)), respectively. Finally, LIBGWO_SAV3 could be the best in terms of all employed performance metrics (fitness, accuracy, length of selected features, and standard deviation).

TABLE 5. Number of selected features selected by the different proposed variants.

ID#		BGWOV3	IBGWOV3	LIBGWOV3	LIBGWO_SAV3	ID#	BGWOV3	IBGWOV3	LIBGWOV3	LIBGWO_SAV3
1	**Worst**	8.0000	7.0000	7.0000	6.0000	8	5.0000	3.0000	3.0000	3.0000
	Avg	4.6400	4.6800	4.3600	3.7200		2.3200	2.2800	2.2800	2.2800
	Best	2.0000	3.0000	3.0000	3.0000		1.0000	2.0000	2.0000	2.0000
	SD	1.6942	0.9261	1.1960	0.9600		0.8818	0.4490	0.4490	0.4490
	Rank	3	4	2	**1**	**9**	2	**1**	**1**	**1**
2	**Worst**	11.0000	10.0000	9.0000	12.0000		8.0000	10.0000	9.0000	6.0000
	Avg	4.7600	4.7200	5.4400	4.3600		4.6400	5.8000	5.4000	4.1600
	Best	1.0000	2.0000	3.0000	2.0000		2.0000	3.0000	4.0000	3.0000
	SD	2.3880	1.8659	1.6989	2.1887		1.3230	1.7436	1.2649	0.7310
	Rank	4	3	2	**1**	**10**	2	4	3	**1**
3	**Worst**	8.0000	7.0000	6.0000	7.0000		10.0000	10.0000	10.0000	7.0000
	Avg	5.1200	4.5600	4.2000	4.3200		6.8400	6.9600	7.1600	5.2800
	Best	3.0000	4.0000	4.0000	4.0000		4.0000	4.0000	4.0000	4.0000
	SD	1.2432	0.8523	0.5657	0.7859		1.7817	1.3705	1.4610	0.8727
	Rank	4	3	**1**	2	**11**	2	3	4	**1**
4	**Worst**	8.0000	8.0000	7.0000	7.0000		16.0000	17.0000	14.0000	8.0000
	Avg	5.0000	5.8800	5.5200	5.1600		7.6400	9.1600	7.4800	3.7200
	Best	2.0000	5.0000	5.0000	4.0000		2.0000	4.0000	3.0000	2.0000
	SD	1.6248	0.8158	0.6400	0.6741		3.9888	3.4021	2.6999	1.2172
	Rank	**1**	4	3	2	**12**	3	4	2	**1**
5	**Worst**	6.0000	6.0000	5.0000	4.0000		5.0000	5.0000	3.0000	3.0000
	Avg	4.0800	3.8000	3.7200	3.3200		2.6000	2.2800	2.2400	2.0800
	Best	3.0000	3.0000	3.0000	3.0000		2.0000	2.0000	2.0000	2.0000
	SD	0.7960	0.7483	0.6645	0.4665		0.9381	0.6645	0.4271	0.2713
	Rank	4	3	2	**1**	**13**	3	4	2	**1**
6	**Worst**	11.0000	9.0000	7.0000	4.0000		12.0000	14.0000	16.0000	8.0000
	Avg	3.8000	4.2400	4.3200	3.3200		4.2800	7.9200	8.4000	4.9200
	Best	1.0000	2.0000	3.0000	3.0000		2.0000	3.0000	3.0000	3.0000
	SD	2.1726	1.3351	1.0852	0.4665		2.5222	3.3457	3.0199	1.4400
	Rank	2	3	4	**1**	**14**	**1**	3	4	2
7	**Worst**	11.0000	9.0000	7.0000	12.0000		8.0000	9.0000	5.0000	4.0000
	Avg	4.4400	5.3200	4.6800	4.0000		3.7200	4.3200	3.4400	3.4400
	Best	2.0000	3.0000	3.0000	3.0000		2.0000	2.0000	2.0000	2.0000
	SD	1.9612	1.3482	1.3182	2.1354		1.3422	1.5419	0.8980	0.8980
	Rank	1	4	3	**1**		2	3	**1**	**1**

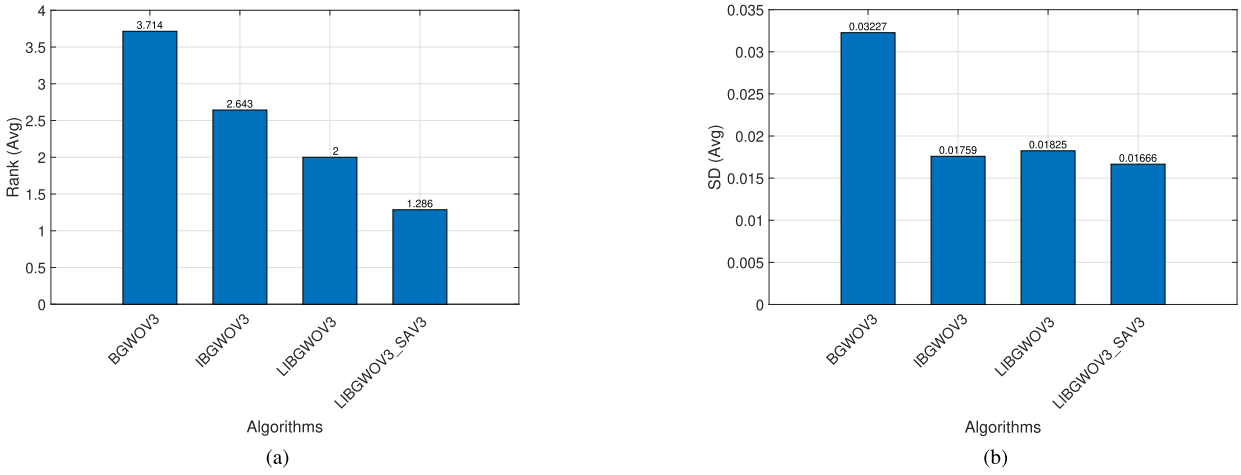


FIGURE 10. (a) Average Rank values obtained under accuracy on datasets from ID#1-ID#14 and (b)Average SD values under accuracy on datasets from ID#1-ID#14.

From those experiments, it can be noted that the improvements of the standard GWO have a significant effect on its performance for solving FS problems. LIBGWO_SAV3 performs best and LIBGWO is second best, while IBGWO is third-best so the two best of those two algorithms (LIBGWO_SAV3 and LIBGWO) are compared in the next experiments with selected state-of-the-art feature selection algorithms.

F. COMPARISON OF THE PROPOSED ALGORITHMS WITH SELECTED OTHER ALGORITHMS

In this section, the proposed algorithms are compared with some state-of-the-art algorithms named earlier. This section is organized as follows:

1) Section A: compares the algorithms on datasets ID#1-ID#14.

TABLE 6. Classification accuracy of the different proposed variants.

ID#		BGWOV3	IBGWOV3	LIBGWOV3	LIBGWO_SAV3	ID#	BGWOV3	IBGWOV3	LIBGWOV3	LIBGWO_SAV3
1	**Best**	0.8478	0.8333	0.8478	0.8333	8	0.1163	0.1163	0.1163	0.1163
	Avg	0.7965	0.8148	0.8203	0.8177		0.1023	0.1163	0.1163	0.1163
	Worst	0.7174	0.7826	0.8043	0.7826		0.0465	0.1163	0.1163	0.1163
	SD	0.0288	0.0141	0.0110	0.0132		0.0228	0.00	0.00	0.0000
	Rank	4	3	1	2	9	4	1	1	1
2	**Best**	0.9444	0.9537	0.9444	0.9537		1.0000	1.0000	1.0000	1.0000
	Avg	0.9293	0.9315	0.9374	0.9393		0.9893	0.9983	0.9961	0.9989
	Worst	0.9167	0.9259	0.9259	0.9259		0.9718	0.9859	0.9859	0.9859
	SD	0.0064	0.0069	0.0060	0.0087		0.0082	0.0046	0.0063	0.0038
	Rank	4	3	2	1	10	4	2	3	1
3	**Best**	0.8700	0.8700	0.8700	0.8700		0.9762	0.9805	0.9827	0.9827
	Avg	0.8368	0.8628	0.8676	0.8676		0.9610	0.9745	0.9762	0.9772
	Worst	0.7200	0.8400	0.8400	0.8500		0.9069	0.9632	0.9675	0.9697
	SD	0.0311	0.0096	0.0071	0.0059		0.0187	0.0047	0.0035	0.0034
	Rank	3	2	1	1	11	4	3	2	1
4	**Best**	0.8900	0.8950	0.8950	0.8950		0.9737	0.9737	0.9737	0.9737
	Avg	0.8208	0.8698	0.8758	0.8784		0.9572	0.9621	0.9600	0.9618
	Worst	0.7100	0.8350	0.8250	0.8250		0.9474	0.9474	0.9474	0.9474
	SD	0.0417	0.0204	0.0199	0.0233		0.0097	0.0101	0.0093	0.0086
	Rank	4	3	1	2	12	4	3	2	1
5	**Best**	0.9050	0.9050	0.9050	0.9050		0.9833	0.9833	0.9833	0.9833
	Avg	0.8790	0.8940	0.8972	0.8960		0.9740	0.9833	0.9827	0.9827
	Worst	0.8250	0.8650	0.8750	0.8750		0.8833	0.9833	0.9667	0.9667
	SD	0.0253	0.0109	0.0083	0.0075		0.0231	0.0000	0.0033	0.0033
	Rank	4	3	1	2	13	4	1	3	2
6	**Best**	0.9150	0.8900	0.9150	0.9150		0.9524	0.9286	0.9048	0.9286
	Avg	0.8398	0.8458	0.8642	0.8770		0.7600	0.7314	0.7505	0.8133
	Worst	0.7150	0.7850	0.8100	0.8350		0.5952	0.5000	0.5476	0.6429
	SD	0.0531	0.0291	0.0265	0.0183		0.0992	0.0958	0.0936	0.0692
	Rank	4	3	2	1	14	2	4	3	1
7	**Best**	0.9200	0.9200	0.9200	0.9200		0.7949	0.8034	0.8034	0.8034
	Avg	0.8534	0.8716	0.8756	0.9066		0.7699	0.7839	0.7880	0.7911
	Worst	0.7550	0.8200	0.8350	0.8250		0.7607	0.7692	0.7692	0.7778
	SD	0.0435	0.0251	0.0293	0.0268		0.0090	0.0101	0.0105	0.0100
	Rank	4	3	2	1		4	3	2	1

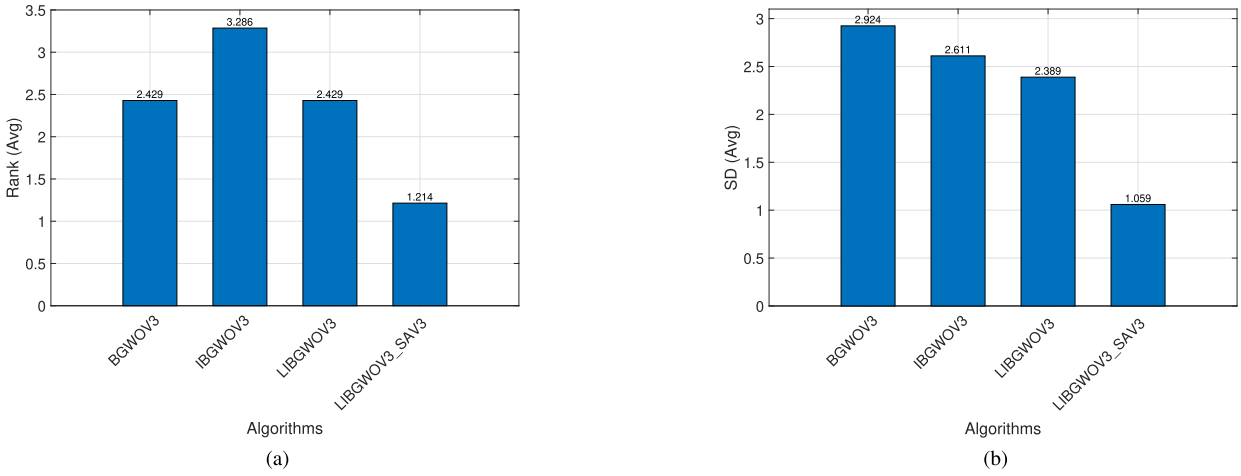


FIGURE 11. (a) Average Rank values number selected features on datasets ID#1-ID#14 and (b)Average SD values under selected features on datasets ID#1-ID#14.

2) Section B: compares the algorithms on datasets ID#15-ID#32.

1) COMPARISON UNDER DATASETS ID#1-ID#14

To confirm the performance of the proposed algorithms, they are compared with a number of selected wrapper-based FS algorithms on the datasets from ID#1-ID#14 under different performance metrics: fitness values, classification accuracy, length of selected features, and standard deviation (SD). Tables 7, and 8 introduce the fitness values obtained by each algorithm on the datasets, ID#1-ID#14, and ID#15-ID#32, respectively. From Tables 7 and 8 it can be seen that both LIBGWO_SAV3 and LIBGWOV3 perform best and second best except for ID#13, where LIBGWOV3 does not outperform GA. Broadly speaking, Table 7 shows that LIBGWO_SAV3 could fulfill the best

TABLE 7. Fitness values obtained under different algorithms on datasets ID#1-ID#7.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
1	**Worst**	0.2174	0.2245	0.2439	0.2382	0.2690	0.2754	0.2704	0.2891	0.2690
	Avg	0.1832	0.1822	0.2105	0.1970	0.2159	0.2440	0.2559	0.2258	0.2263
	Best	0.1679	0.1549	0.1628	0.1679	0.1549	0.1908	0.2059	0.1743	0.1750
	SD	0.0129	0.0140	0.0234	0.0180	0.0324	0.0266	0.0172	0.0298	0.0256
	Rank	2	**1**	4	3	5	8	9	6	7
2	**Worst**	0.0753	0.0758	0.0840	0.0835	0.0840	0.0885	0.0850	0.0840	0.0927
	Avg	0.0623	0.0650	0.0723	0.0715	0.0732	0.0813	0.0697	0.0749	0.0803
	Best	0.0483	0.0483	0.0565	0.0493	0.0483	0.0672	0.0565	0.0565	0.0565
	SD	0.0082	0.0075	0.0076	0.0077	0.0077	0.0052	0.0099	0.0082	0.0077
	Rank	**1**	2	5	4	6	9	3	7	8
3	**Worst**	0.1545	0.1535	0.1951	0.1842	0.1931	0.2278	0.2060	0.2703	0.2109
	Avg	0.1354	0.1367	0.1551	0.1562	0.1530	0.1954	0.1657	0.1854	0.1551
	Best	0.1327	0.1327	0.1327	0.1327	0.1327	0.1337	0.1327	0.1327	0.1327
	SD	0.0064	0.0061	0.0179	0.0155	0.0177	0.0309	0.0207	0.0392	0.0239
	Rank	**1**	2	4	5	3	8	6	7	4
4	**Worst**	0.1773	0.1832	0.1664	0.1872	0.1961	0.2427	0.1951	0.3099	0.2109
	Avg	0.1262	0.1285	0.1302	0.1373	0.1374	0.1819	0.1285	0.1950	0.1469
	Best	0.1099	0.1099	0.1099	0.1099	0.1099	0.1099	0.1099	0.1099	0.1099
	SD	0.0227	0.0199	0.0184	0.0208	0.0247	0.0387	0.0294	0.0456	0.0292
	Rank	**1**	2	3	4	5	7	2	8	6
5	**Worst**	0.1268	0.1179	0.1615	0.1615	0.1763	0.1882	0.1882	0.1763	0.1763
	Avg	0.1063	0.1042	0.1239	0.1193	0.1232	0.1630	0.1244	0.1309	0.1197
	Best	0.0980	0.0980	0.0980	0.0980	0.0980	0.1189	0.0980	0.0980	0.0980
	SD	0.0071	0.0068	0.0184	0.0146	0.0230	0.0222	0.0369	0.0274	0.0211
	Rank	2	**1**	6	3	5	9	7	8	4
6	**Worst**	0.1646	0.2008	0.2784	0.2981	0.2727	0.3497	0.3228	0.2392	0.2727
	Avg	0.1231	0.1368	0.1736	0.1991	0.1805	0.2717	0.2062	0.1593	0.1921
	Best	0.0857	0.1105	0.1200	0.1208	0.0857	0.1856	0.1162	0.1155	0.0857
	SD	0.0180	0.0209	0.0380	0.0421	0.0580	0.0369	0.0660	0.0381	0.0586
	Rank	**1**	2	4	7	5	9	8	3	6
7	**Worst**	0.1781	0.1662	0.2115	0.2045	0.2252	0.3120	0.2539	0.2186	0.2430
	Avg	0.0941	0.1234	0.1629	0.1621	0.1621	0.2305	0.1572	0.1446	0.1684
	Best	0.0804	0.0713	0.1059	0.1109	0.0713	0.1402	0.0618	0.0804	0.0804
	SD	0.0273	0.0263	0.0278	0.0247	0.0424	0.0432	0.0533	0.0365	0.0529
	Rank	**1**	2	6	5	5	8	4	3	7

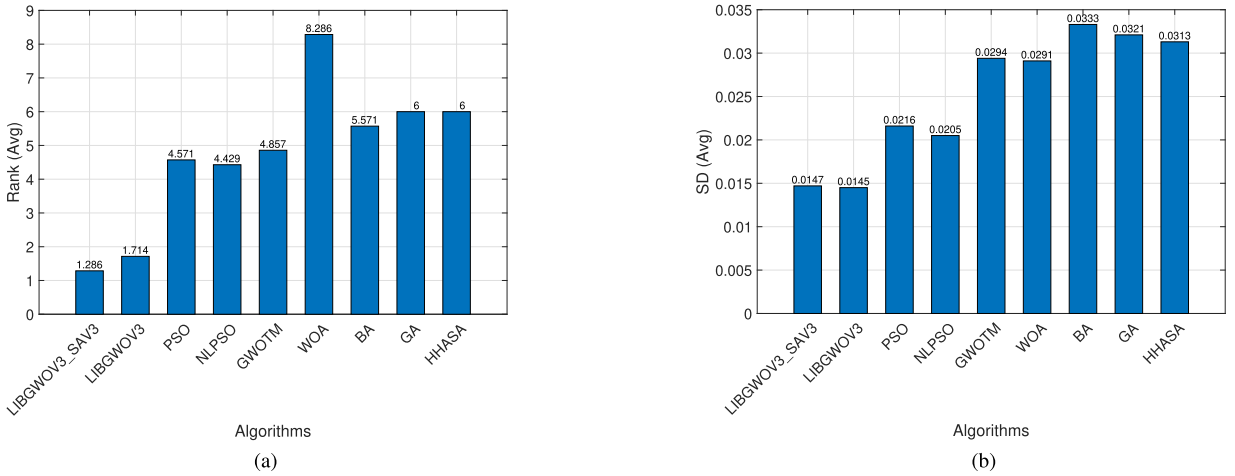


FIGURE 12. (a) Average Rank values under fitness values on datasets ID#1-ID#7 and (b)Average SD values under fitness values on datasets ID#1-ID#7.

for two instances: ID#1 AND ID#5 in all independent runs, and so is LIBGWOV3 which performed better in all independent runs for ID#2 and ID#6. For the other instances, both of which (LIBGWO_SAV3 and LIBGWOV3) are approximately competitive in terms of Best, Avg, Worst, and SD cases. On the contrary, Table 8 show the effectiveness of LIBGWO_SAV3 compared to LIBGWOV3 and the others for six out of seven instances within all independent runs.

Graphically, Fig. 12 (a) and Fig. 13 (b) show the average of rank values obtained by each algorithm under fitness values on the datasets from ID#1-ID#7, respectively. According to Fig. 13 (a), LIBGWO_SAV3 performs best with values of 1.2857 and LIBGWOV3 is second best with 1.71, and WOA

TABLE 8. Fitness values obtained under different algorithms on datasets ID#8-ID#14.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
8	**Worst**	0.8782	0.8782	0.9451	0.9001	0.9243	0.9473	0.9451	0.9462	0.9462
	Avg	0.8775	0.8776	0.8890	0.8804	0.8877	0.9068	0.8943	0.8997	0.9170
	Best	0.8771	0.8771	0.8771	0.8771	0.8771	0.8782	0.8771	0.8771	0.8771
	SD	0.0005	0.0006	0.0186	0.0066	0.0153	0.0229	0.0172	0.0238	0.0261
	Rank	**1**	2	5	4	3	8	6	7	9
9	**Worst**	0.0151	0.0157	0.0166	0.0166	0.0157	0.0299	0.0157	0.0291	0.0291
	Avg	0.0023	0.0022	0.0091	0.0110	0.0087	0.0160	0.0125	0.0114	0.0151
	Best	0.0009	0.0009	0.0012	0.0009	0.0012	0.0018	0.0009	0.0009	0.0012
	SD	0.0037	0.0028	0.0068	0.0064	0.0069	0.0054	0.0057	0.0088	0.0088
	Rank	2	**1**	4	5	3	9	7	6	8
10	**Worst**	0.0326	0.0364	0.0535	0.0438	0.0449	0.0614	0.0465	0.0792	0.0411
	Avg	0.0253	0.0278	0.0342	0.0326	0.0297	0.0481	0.0355	0.0382	0.0333
	Best	0.0208	0.0219	0.0240	0.0245	0.0192	0.0273	0.0224	0.0224	0.0246
	SD	0.0030	0.0034	0.0064	0.0055	0.0069	0.0114	0.0075	0.0114	0.0042
	Rank	**1**	2	5	4	3	9	7	8	6
11	**Worst**	0.0531	0.0568	0.0621	0.0588	0.0618	0.0691	0.0615	0.0618	0.0621
	Avg	0.0391	0.0400	0.0496	0.0494	0.0508	0.0555	0.0544	0.0496	0.0491
	Best	0.0271	0.0277	0.0277	0.0277	0.0277	0.0284	0.0531	0.0271	0.0274
	SD	0.0083	0.0094	0.0085	0.0092	0.0066	0.0099	0.0020	0.0085	0.0100
	Rank	**1**	2	5	4	6	8	7	5	3
12	**Worst**	0.0342	0.0349	0.0386	0.0367	0.0501	0.0754	0.0349	0.1339	0.0501
	Avg	0.0185	0.0186	0.0221	0.0239	0.0239	0.0507	0.0236	0.0308	0.0309
	Best	0.0178	0.0178	0.0178	0.0178	0.0178	0.0178	0.0178	0.0178	0.0178
	SD	0.0032	0.0033	0.0072	0.0079	0.0085	0.0160	0.0079	0.0269	0.0134
	Rank	**1**	2	3	5	5	8	4	6	7
13	**Worst**	0.3549	0.4965	0.4251	0.4748	0.5187	0.7106	0.5207	0.4251	0.6603
	Avg	0.1856	0.2361	0.2711	0.3027	0.2875	0.5268	0.3902	0.2191	0.3657
	Best	0.0714	0.0953	0.0720	0.1663	0.0246	0.3302	0.1438	0.0725	0.2125
	SD	0.0686	0.1099	0.0711	0.0818	0.1226	0.1010	0.1022	0.0866	0.1021
	Rank	**1**	3	4	6	5	9	8	2	7
14	**Worst**	0.2220	0.2315	0.2429	0.2409	0.2429	0.2608	0.2409	0.2623	0.2623
	Avg	0.2102	0.2154	0.2215	0.2208	0.2272	0.2349	0.2298	0.2319	0.2376
	Best	0.1986	0.1986	0.1986	0.1986	0.1986	0.2155	0.2071	0.1986	0.1986
	SD	0.0092	0.0101	0.0111	0.0097	0.0115	0.0105	0.0091	0.0155	0.0159
	Rank	**1**	2	4	3	5	8	6	7	9

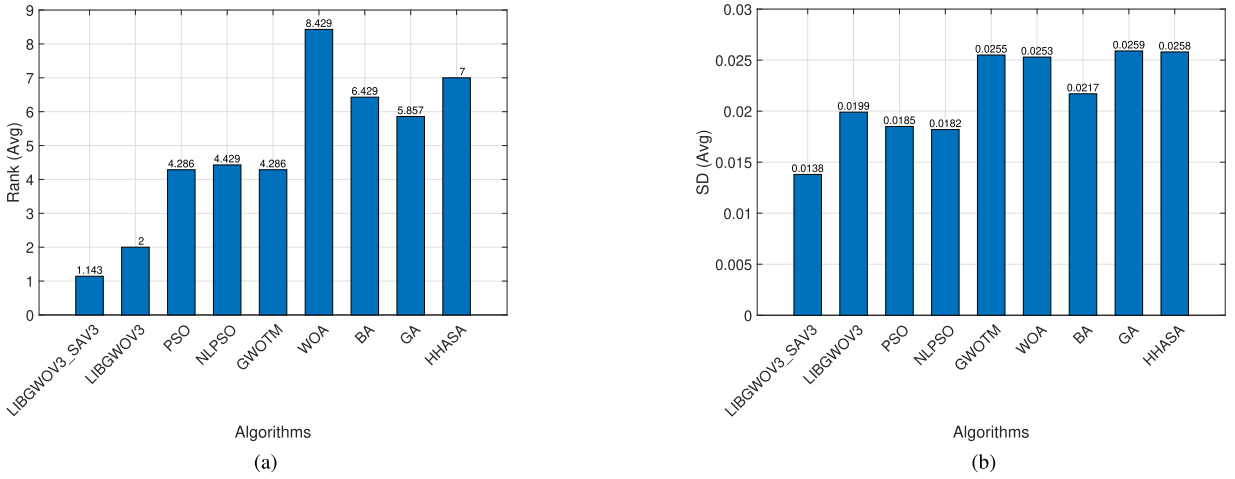


FIGURE 13. (a) Average Rank values under fitness values on datasets ID#8-ID#14 and (b)Average SD values under fitness values on datasets ID#8-ID#14.

performs worst. Fig. 13 (b) shows that LIBGWO_SAV3 is more stable than the others. Similarly, for the datasets, ID#8-ID#14, Fig. 13 (a), and Fig. 13 (b) show the average of rank values obtained using each algorithm, and the average SD values, respectively. According to Fig. 13 (a), LIBGWO_SAV3 is best with 1.14, and LIBGWOV3 is worse than LIBGWO_SAV3 but better than the others with values of 2.000. Fig. 13 (b) shows that LIBGWO_SAV3 is more stable. Although LIBGWO_SAV3 could perform better for fitness, it doesn't consider even now a strong alternative to the existing feature selection techniques. Broadly speaking, the superiority of the proposed algorithm: LIBGWO_SAV3

TABLE 9. Classification accuracy values obtained under different algorithms on datasets ID#1-ID#7.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
1	**Worst**	0.8333	0.8478	0.8406	0.8333	0.8478	0.8116	0.7971	0.8261	0.8261
	Avg	0.8177	0.8188	0.7918	0.8046	0.7848	0.7594	0.7459	0.7744	0.7739
	Best	0.7826	0.7754	0.7609	0.7609	0.7319	0.7319	0.7319	0.7101	0.7319
	SD	0.0132	0.0143	0.0238	0.0182	0.0330	0.0251	0.0172	0.0302	0.0260
	Rank	2	**1**	4	3	5	8	9	6	7
2	**Worst**	0.9537	0.9537	0.9444	0.9537	0.9537	0.9352	0.9444	0.9444	0.9444
	Avg	0.9393	0.9370	0.9296	0.9302	0.9281	0.9210	0.9321	0.9259	0.9204
	Best	0.9259	0.9259	0.9167	0.9167	0.9167	0.9167	0.9167	0.9167	0.9074
	SD	0.0087	0.0079	0.0078	0.0082	0.0082	0.0057	0.0100	0.0086	0.0081
	Rank	**1**	2	5	4	6	9	3	7	8
3	**Worst**	0.8700	0.8700	0.8700	0.8700	0.8700	0.8700	0.8700	0.8700	0.8700
	Avg	0.8676	0.8664	0.8497	0.8477	0.8510	0.8100	0.8387	0.8170	0.8477
	Best	0.8500	0.8500	0.8100	0.8200	0.8100	0.7800	0.8000	0.7300	0.7900
	SD	0.0059	0.0056	0.0172	0.0154	0.0180	0.0296	0.0203	0.0397	0.0243
	Rank	**1**	2	4	6	3	9	7	8	5
4	**Worst**	0.8950	0.8950	0.8950	0.8950	0.8950	0.8950	0.8950	0.8950	0.8950
	Avg	0.8784	0.8682	0.8748	0.8678	0.8670	0.8238	0.8760	0.8070	0.8562
	Best	0.8250	0.8200	0.8400	0.8200	0.8100	0.7650	0.8100	0.6900	0.7900
	SD	0.0233	0.0204	0.0179	0.0201	0.0246	0.0384	0.0292	0.0466	0.0302
	Rank	**1**	4	3	5	6	7	2	9	7
5	**Worst**	0.9050	0.9050	0.9050	0.9050	0.9050	0.8850	0.9050	0.9050	0.9050
	Avg	0.8960	0.8986	0.8797	0.8837	0.8798	0.8418	0.8797	0.8710	0.8823
	Best	0.8750	0.8850	0.8450	0.8450	0.8250	0.8200	0.8200	0.8250	0.8250
	SD	0.0075	0.0070	0.0180	0.0142	0.0232	0.0206	0.0352	0.0277	0.0214
	Rank	2	**1**	6	3	5	9	7	8	4
6	**Worst**	0.9150	0.8900	0.8800	0.8800	0.9150	0.8150	0.8850	0.8850	0.9150
	Avg	0.8770	0.8634	0.8263	0.8010	0.8192	0.7288	0.7953	0.8403	0.8068
	Best	0.8350	0.8000	0.7200	0.7050	0.7250	0.6500	0.6800	0.7600	0.7250
	SD	0.0183	0.0208	0.0381	0.0420	0.0587	0.0364	0.0649	0.0386	0.0596
	Rank	**1**	2	4	6	5	8	6	3	9
7	**Worst**	0.9200	0.9300	0.8950	0.8900	0.9300	0.8600	0.9400	0.9200	0.9200
	Avg	0.9066	0.8774	0.8380	0.8387	0.8390	0.7718	0.8447	0.8552	0.8310
	Best	0.8250	0.8350	0.7900	0.7950	0.7750	0.6950	0.7500	0.7800	0.7550
	SD	0.0268	0.0264	0.0279	0.0247	0.0418	0.0422	0.0526	0.0369	0.0537
	Rank	**1**	2	4	5	3	9	7	6	8

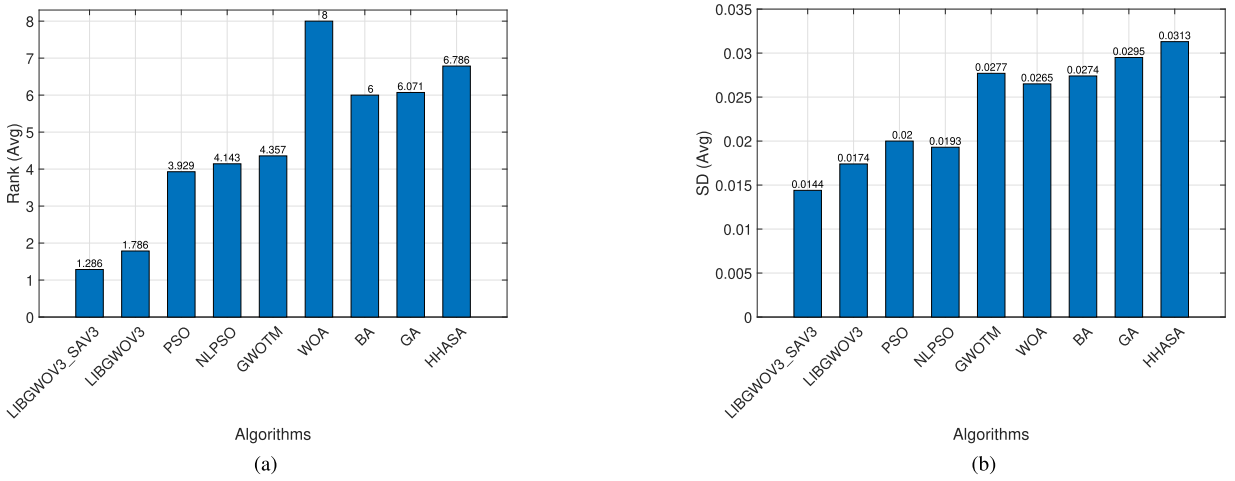


FIGURE 14. (a) Average Rank values under Classification accuracy on ID#1-ID#14 and (b)Average SD values under classification accuracy on ID#1-ID#14.

in terms of fitness value is not enough to affirm that it is the best because the fitness function, as mentioned in section 3.3, is compounded of two objectives: accuracy and length of selected features, and since the main objective of the machine learning techniques is achieving better accuracy even if on the account of running time, LIBGWO_SAV3 must be better regarding this objective to become a strong alternative to all the existing FS techniques. Therefore, our judgment for its superiority will be delayed to the next paragraph which discusses the performance of various algorithms under classification accuracy.

After showing the superiority of the proposed algorithms over all the compared algorithms under fitness values, Tables 9 and 10 show the results obtained under classification

TABLE 10. Classification accuracy values obtained under different algorithms on datasets ID#8-ID#14.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
8	**Worst**	0.1163	0.1163	0.1163	0.1163	0.1163	0.1163	0.1163	0.1163	0.1163
	Avg	0.1163	0.1163	0.1054	0.1140	0.1062	0.0884	0.0992	0.0938	0.0760
	Best	0.1163	0.1163	0.0465	0.0930	0.0698	0.0465	0.0465	0.0465	0.0465
	SD	0.0000	0.0000	0.0187	0.0070	0.0155	0.0228	0.0179	0.0244	0.0268
	Rank	**1**	2	5	3	4	8	6	7	9
9	**Worst**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	Avg	0.9989	0.9994	0.9925	0.9906	0.9925	0.9869	0.9887	0.9897	0.9859
	Best	0.9859	0.9859	0.9859	0.9859	0.9859	0.9718	0.9859	0.9718	0.9718
	SD	0.0038	0.0028	0.0070	0.0066	0.0070	0.0051	0.0056	0.0089	0.0089
	Rank	2	**1**	3	4	3	7	6	5	8
10	**Worst**	0.9827	0.9827	0.9805	0.9805	0.9848	0.9762	0.9805	0.9805	0.9784
	Avg	0.9772	0.9757	0.9707	0.9718	0.9734	0.9582	0.9672	0.9641	0.9690
	Best	0.9697	0.9675	0.9524	0.9632	0.9589	0.9481	0.9567	0.9221	0.9632
	SD	0.0034	0.0036	0.0059	0.0049	0.0068	0.0095	0.0075	0.0117	0.0041
	Rank	**1**	2	5	4	3	9	7	8	6
11	**Worst**	0.9737	0.9737	0.9737	0.9737	0.9737	0.9737	0.9474	0.9737	0.9737
	Avg	0.9618	0.9621	0.9526	0.9532	0.9500	0.9482	0.9468	0.9509	0.9518
	Best	0.9474	0.9474	0.9386	0.9474	0.9386	0.9386	0.9386	0.9386	0.9386
	SD	0.0086	0.0095	0.0087	0.0092	0.0069	0.0094	0.0022	0.0087	0.0101
	Rank	**2**	**1**	4	3	7	8	9	6	5
12	**Worst**	0.9833	0.9833	0.9833	0.9833	0.9833	0.9833	0.9833	0.9833	0.9833
	Avg	0.9827	0.9827	0.9800	0.9778	0.9783	0.9533	0.9778	0.9706	0.9700
	Best	0.9667	0.9667	0.9667	0.9667	0.9500	0.9333	0.9667	0.8667	0.9500
	SD	0.0033	0.0033	0.0067	0.0079	0.0088	0.0152	0.0079	0.0268	0.0139
	Rank	**1**	**1**	2	4	3	7	4	5	6
13	**Worst**	0.9286	0.9048	0.9286	0.8333	0.9762	0.6667	0.8571	0.9286	0.7857
	Avg	0.8133	0.7629	0.7270	0.6952	0.7103	0.4706	0.6079	0.7794	0.6310
	Best	0.6429	0.5000	0.5714	0.5238	0.4762	0.2857	0.4762	0.5714	0.3333
	SD	0.0692	0.1110	0.0719	0.0828	0.1240	0.1008	0.1032	0.0876	0.1032
	Rank	**1**	2	3	5	4	8	7	2	6
14	**Worst**	0.8034	0.8034	0.8034	0.8034	0.8034	0.7863	0.7949	0.8034	0.8034
	Avg	0.7911	0.7863	0.7821	0.7821	0.7755	0.7684	0.7721	0.7689	0.7630
	Best	0.7778	0.7692	0.7607	0.7607	0.7607	0.7436	0.7607	0.7350	0.7350
	SD	0.0100	0.0105	0.0103	0.0093	0.0115	0.0107	0.0092	0.0160	0.0175
	Rank	**1**	2	3	3	4	6	4	5	7

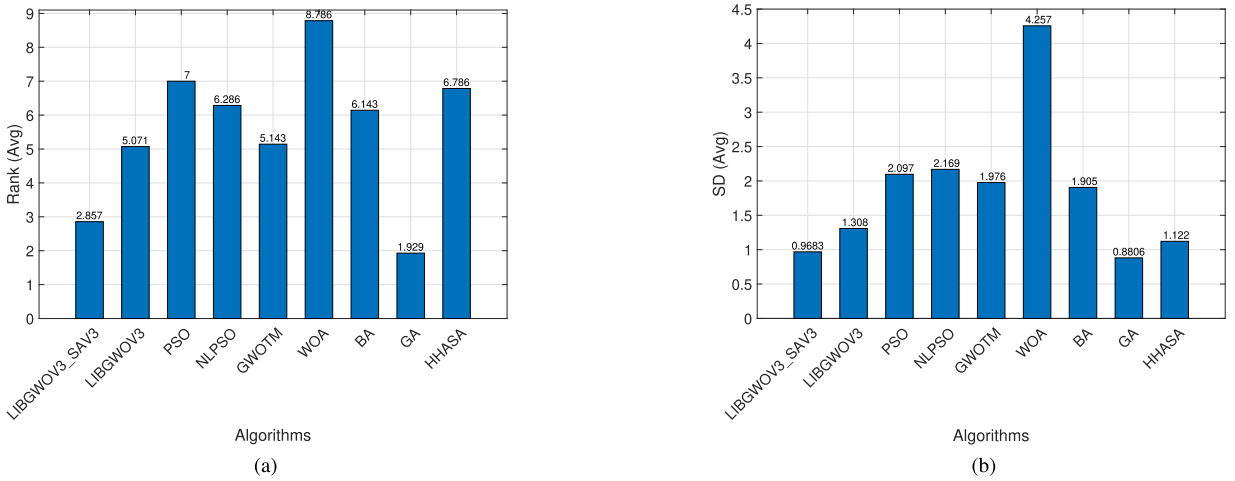


FIGURE 15. (a)Average Rank values under selected features on ID#1-ID#14 and (b)Average SD values under selected features on ID#1-ID#14.

accuracy metric on datasets ID#1-ID#7, and ID#8-ID#14, illustrating the superiority of our proposed algorithms. From those tables, the number of instances with bold values (Avg), indicating the best outcomes, for LIBGWO_SAV3 got to 10 out of 14 instances, while LIBGWOV3 was the best for the other four instances.

The descriptive values within those tables are shown graphically in Fig. 14 (a) and Fig. 14 (b), which show the average of SD values obtained by each algorithm on all instances, ID#1-ID#14, within 30 independent runs. After observing those figures, it is observed that we found that LIBGWO_SAV3 came in the first rank for both average

TABLE 11. Selected features lengths obtained by different algorithms on datasets ID#1-ID#7.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
1	**Worst**	6.0000	6.0000	10.0000	9.0000	7.0000	14.0000	10.0000	5.0000	5.0000
	Avg	3.7200	4.0400	6.0667	5.0000	4.0333	8.1000	6.1333	3.4333	3.4333
	Best	3.0000	3.0000	2.0000	2.0000	2.0000	4.0000	3.0000	2.0000	2.0000
	SD	0.9600	0.9156	1.6918	1.5275	1.3287	3.6364	1.8209	0.8439	0.8825
	Rank	3	5	7	6	4	9	8	2	**1**
2	**Worst**	12.0000	11.0000	12.0000	11.0000	10.0000	13.0000	8.0000	5.0000	9.0000
	Avg	4.3600	5.3600	5.2333	4.8333	4.0000	6.0667	4.9333	3.1000	2.9333
	Best	2.0000	3.0000	2.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	SD	2.1887	2.1887	2.7530	2.2961	1.8797	3.1404	1.9137	1.1060	1.5691
	Rank	3	8	7	5	4	9	6	2	**1**
3	**Worst**	7.0000	7.0000	8.0000	7.0000	8.0000	10.0000	8.0000	6.0000	7.0000
	Avg	4.3200	4.4800	6.3000	5.4000	5.4667	7.3333	5.9333	4.2000	4.3333
	Best	4.0000	4.0000	4.0000	4.0000	3.0000	3.0000	4.0000	3.0000	3.0000
	SD	0.7859	0.7547	1.2949	0.9866	1.4996	2.0385	1.4360	0.9092	0.9428
	Rank	2	4	8	5	6	9	7	**1**	3
4	**Worst**	7.0000	7.0000	8.0000	9.0000	8.0000	10.0000	7.0000	6.0000	6.0000
	Avg	5.1600	5.5200	6.3000	6.4667	5.7000	7.4667	5.7333	3.9333	4.5333
	Best	4.0000	4.0000	5.0000	5.0000	4.0000	4.0000	5.0000	3.0000	3.0000
	SD	0.6741	0.7547	0.8226	0.9911	1.1299	1.4996	0.7717	0.9638	0.8844
	Rank	3	4	7	8	5	9	6	**1**	2
5	**Worst**	4.0000	4.0000	8.0000	8.0000	7.0000	10.0000	10.0000	4.0000	4.0000
	Avg	3.3200	3.8000	4.7667	4.1667	4.2000	6.4000	5.2667	3.2000	3.2333
	Best	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	SD	0.4665	0.4000	1.1160	1.0980	1.1944	2.6407	2.4074	0.4000	0.4230
	Rank	3	**4**	7	5	6	9	8	**1**	2
6	**Worst**	4.0000	7.0000	8.0000	15.0000	11.0000	18.0000	19.0000	5.0000	6.0000
	Avg	3.3200	3.8400	4.0667	5.3333	3.7000	8.0000	8.8667	3.0333	2.2333
	Best	3.0000	3.0000	2.0000	3.0000	1.0000	3.0000	3.0000	2.0000	1.0000
	SD	0.4665	1.0837	1.6720	2.8441	2.4786	3.8902	5.1234	0.7063	1.1743
	Rank	3	4	6	7	5	8	9	2	**1**
7	**Worst**	12.0000	8.0000	11.0000	11.0000	18.0000	25.0000	16.0000	4.0000	4.0000
	Avg	4.0000	5.0800	6.3333	5.8333	6.6667	11.5333	8.4667	3.0667	2.6000
	Best	3.0000	3.0000	3.0000	3.0000	2.0000	1.0000	3.0000	2.0000	1.0000
	SD	2.1354	1.3242	2.1029	2.0669	4.4222	6.9269	3.8793	0.4422	0.8000
	Rank	3	4	6	5	7	9	8	2	**1**

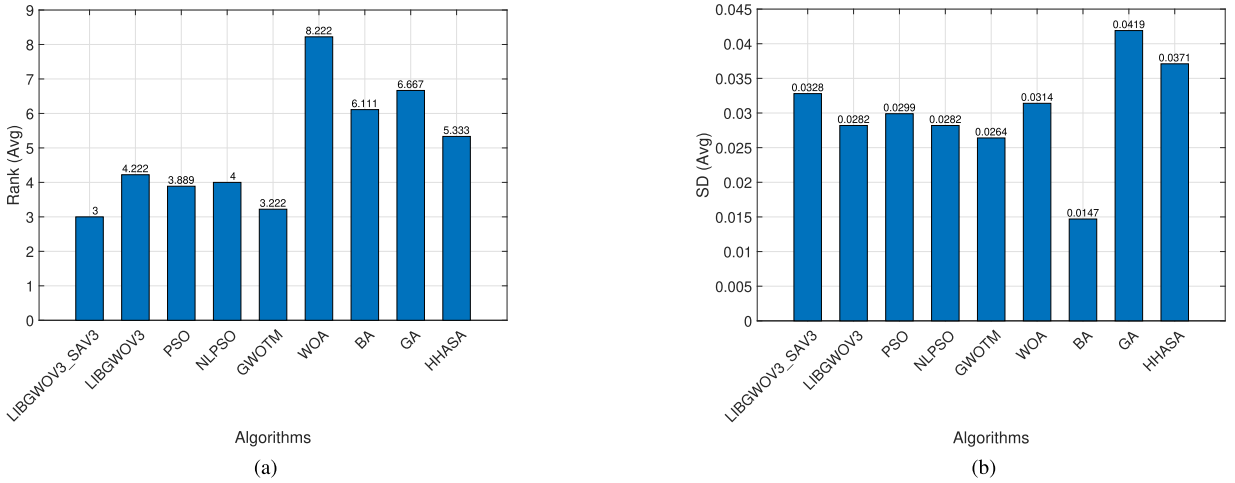


FIGURE 16. (a)Average Rank values under fitness values on datasets ID#15-ID#23 and (b)Average SD values under fitness values on datasets ID#15-ID#23.

ranks and SD respectively with values of 1.2 and 0.014, and LIBGWOV3 is the second best one with values of 1.7 and 0.018, while both WOA and HHASA are the worst in terms of average ranks and SD, respectively. Consequently, our proposed algorithm, LIBGWO_SAV3, is more efficient and stable than all rival algorithms.

After comparing algorithms under both fitness values and classification accuracy, the number of features obtained by each algorithm are compared within Tables 11, and 12 on the datasets ID#1-ID#7, and ID#8-ID#14. It can be seen that GA, HHASA performs better under the features length selected and illustrated in Fig. 15 (a) and Fig. 15 (b) shows the SD

TABLE 12. Selected features lengths obtained by different algorithms on datasets ID#8-ID#14.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
8	**Worst**	3.0000	3.0000	6.0000	5.0000	3.0000	8.0000	3.0000	3.0000	3.0000
	Avg	2.2800	2.4400	3.0333	2.9333	2.6000	3.8667	2.2667	2.2667	1.9667
	Best	2.0000	2.0000	1.0000	2.0000	2.0000	1.0000	1.0000	1.0000	1.0000
	SD	0.4490	0.4964	0.9826	0.8138	0.4899	1.5434	0.5735	0.6289	0.7520
	Rank	4	5	8	7	6	9	3	2	**1**
9	**Worst**	6.0000	9.0000	10.0000	10.0000	6.0000	24.0000	6.0000	6.0000	5.0000
	Avg	4.1600	5.6800	5.6333	5.6333	4.4333	10.0667	4.5333	3.9667	4.1000
	Best	3.0000	3.0000	3.0000	2.0000	3.0000	4.0000	3.0000	3.0000	3.0000
	SD	0.7310	1.3775	1.7792	1.9576	0.8035	5.7209	0.8055	0.6574	0.5972
	Rank	3	7	6	6	4	8	5	**1**	2
10	**Worst**	7.0000	10.0000	16.0000	18.0000	13.0000	19.0000	8.0000	7.0000	13.0000
	Avg	5.2800	7.1200	9.8333	8.9667	6.4000	12.8000	5.8000	4.9333	4.8333
	Best	4.0000	4.0000	6.0000	4.0000	4.0000	6.0000	4.0000	3.0000	4.0000
	SD	0.8727	1.4784	2.4911	3.3713	2.0591	4.7497	1.0456	0.9978	1.6948
	Rank	3	6	8	7	5	9	4	2	**1**
11	**Worst**	8.0000	14.0000	20.0000	20.0000	11.0000	25.0000	11.0000	6.0000	15.0000
	Avg	3.7200	7.5200	8.2000	9.1333	3.8333	12.8667	5.1333	3.0333	3.9333
	Best	2.0000	4.0000	2.0000	3.0000	2.0000	4.0000	2.0000	2.0000	2.0000
	SD	1.2172	2.6851	4.8813	4.6814	2.0344	5.4328	2.1561	0.8360	2.2647
	Rank	**1**	6	7	8	3	9	5	2	4
12	**Worst**	3.0000	4.0000	12.0000	7.0000	12.0000	15.0000	4.0000	6.0000	5.0000
	Avg	2.0800	2.3600	3.6667	3.0000	3.8667	7.2667	2.5333	2.6000	1.9333
	Best	2.0000	2.0000	2.0000	2.0000	1.0000	1.0000	2.0000	2.0000	1.0000
	SD	0.2713	0.6248	2.7366	1.3416	3.3639	3.8291	0.6182	1.0832	0.8138
	Rank	2	3	7	6	8	9	4	5	**1**
13	**Worst**	8.0000	16.0000	18.0000	20.0000	13.0000	56.0000	21.0000	11.0000	5.0000
	Avg	4.9200	7.8800	5.0333	6.0333	4.4333	16.1333	12.2667	3.7667	2.0000
	Best	3.0000	2.0000	2.0000	1.0000	1.0000	1.0000	6.0000	1.0000	1.0000
	SD	1.4400	3.2658	2.9381	4.3778	3.1057	12.4519	3.3757	1.8562	0.8563
	Rank	4	7	5	6	3	9	8	2	**1**
14	**Worst**	4.0000	6.0000	9.0000	9.0000	7.0000	9.0000	6.0000	4.0000	7.0000
	Avg	3.4400	3.8400	5.7667	5.0333	4.9333	5.6000	4.2000	3.1667	2.9333
	Best	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	3.0000	1.0000	1.0000
	SD	0.8980	0.9666	2.0926	2.0080	1.8785	2.0913	0.7483	0.8975	2.0483
	Rank	3	4	9	7	6	8	5	2	**1**

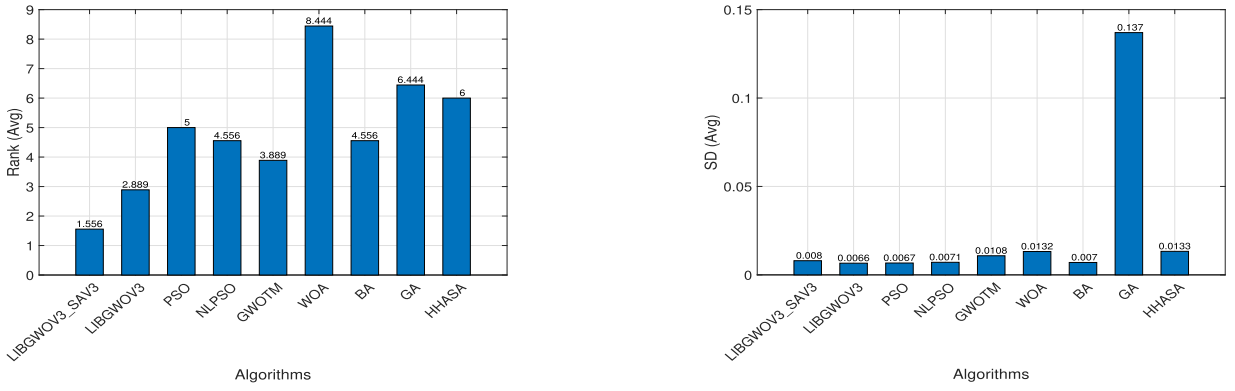


FIGURE 17. (a)Average Rank values under fitness values on datasets ID#24-ID#323 and (b)average SD values under fitness values on datasets ID#24-ID#32.

of the selected features length obtained by each algorithm, confirming that our proposed algorithm, LIBGWO_SAV3, has less diversification within 30 independent runs.

Although both GA and HHAS perform better under selected features length, their performance under classification accuracy is not good. Further, since the main objective of solving FS is to find the subset of features that lead to the higher accuracy with the smaller number of features as the second objective, our proposed algorithm is better because it achieves the best accuracy with an average of the ranks of the selected features length equal to 2.8, and GA has an average of 1.92, and HHAGA with 1.5. Based on the previous analysis, the number of features extracted by LIBGWO_SAV3 is so close to that obtained by the others, and the accuracy of LIBGWO_SAV3 is higher. Since accuracy is considered the main objective for FS, LIBGWO_SAV3 is better.

TABLE 13. Fitness values obtained under different algorithms on datasets ID#15-ID#23.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
15	**Worst**	0.3428	0.3455	0.2667	0.3055	0.3478	0.4224	0.3846	0.4189	0.4189
	Avg	0.3108	0.3338	0.1957	0.2339	0.2173	0.3359	0.3568	0.3073	0.2729
	Best	0.1524	0.2681	0.1523	0.0762	0.1143	0.2286	0.3463	0.0382	0.1143
	SD	0.0513	0.0227	0.0377	0.0510	0.0657	0.0384	0.0167	0.0834	0.0912
	Rank	6	7	**1**	3	2	8	9	5	4
16	**Worst**	0.0056	0.0067	0.0104	0.0126	0.0115	0.0171	0.0126	0.0175	0.0115
	Avg	0.0037	0.0042	0.0051	0.0055	0.0054	0.0110	0.0069	0.0093	0.0066
	Best	0.0033	0.0033	0.0033	0.0033	0.0033	0.0033	0.0033	0.0033	0.0033
	SD	0.0006	0.0009	0.0015	0.0017	0.0025	0.0045	0.0032	0.0042	0.0034
	Rank	**1**	2	3	5	4	9	7	8	6
17	**Worst**	0.1359	0.1784	0.1938	0.1976	0.1931	0.2369	0.1835	0.1449	0.1964
	Avg	0.0715	0.1352	0.1114	0.1238	0.1042	0.1846	0.1632	0.0737	0.0816
	Best	0.0001	0.0629	0.0001	0.0002	0.0001	0.0002	0.1165	0.0001	0.0001
	SD	0.0165	0.0288	0.0656	0.0636	0.0715	0.0540	0.0204	0.0584	0.0751
	Rank	**1**	7	5	6	4	9	8	2	**3**
18	**Worst**	0.7344	0.7345	0.7382	0.7370	0.7381	0.7476	0.7463	0.7593	0.7569
	Avg	0.7255	0.7263	0.7283	0.7269	0.7273	0.7373	0.7336	0.7372	0.7363
	Best	0.7203	0.7203	0.7203	0.7203	0.7203	0.7263	0.7203	0.7203	0.7203
	SD	0.0047	0.0055	0.0045	0.0050	0.0070	0.0062	0.0058	0.0095	0.0088
	Rank	**1**	2	5	3	4	9	6	8	7
19	**Worst**	0.1856	0.1872	0.2796	0.2231	0.2773	0.3033	0.3003	0.2956	0.2956
	Avg	0.1302	0.1406	0.1531	0.1545	0.1706	0.2133	0.1866	0.1803	0.1846
	Best	0.0955	0.0955	0.0963	0.0955	0.0955	0.1169	0.1131	0.1131	0.1131
	SD	0.0247	0.0265	0.0414	0.0299	0.0379	0.0538	0.0567	0.0419	0.0472
	Rank	**1**	2	3	4	5	9	8	6	**7**
20	**Worst**	0.2335	0.2325	0.2568	0.2325	0.2409	0.2623	0.2623	0.2623	0.2623
	Avg	0.2180	0.2152	0.2249	0.2228	0.2256	0.2396	0.2378	0.2344	0.2352
	Best	0.1986	0.1986	0.1986	0.2071	0.1986	0.2250	0.2071	0.2071	0.1986
	SD	0.0101	0.0106	0.0102	0.0072	0.0112	0.0106	0.0137	0.0133	0.0169
	Rank	2	**1**	4	3	5	9	8	6	**7**
21	**Worst**	0.3206	0.3264	0.2213	0.1958	0.0046	0.2823	0.0161	0.3267	0.2784
	Avg	0.2165	0.2295	0.1196	0.0901	0.0046	0.2499	0.0066	0.2889	0.0137
	Best	0.0046	0.0054	0.0054	0.0054	0.0046	0.0408	0.0046	0.0046	0.0046
	SD	0.1311	0.1141	0.0692	0.0629	0.0000	0.0656	0.0036	0.0952	0.0491
	Rank	6	7	5	4	**1**	8	2	9	**3**
22	**Worst**	0.2921	0.2921	0.2921	0.2921	0.2921	0.2921	0.2921	0.2933	0.2921
	Avg	0.2856	0.2896	0.2783	0.2729	0.2772	0.2886	0.2896	0.2900	0.2903
	Best	0.2711	0.2776	0.2666	0.2651	0.2666	0.2651	0.2735	0.2711	0.2719
	SD	0.0073	0.0035	0.0102	0.0083	0.0101	0.0067	0.0056	0.0055	0.0050
	Rank	4	6	3	**1**	2	5	6	7	8
23	**Worst**	0.1234	0.1307	0.1124	0.0877	0.1489	0.1536	0.0317	0.3043	0.1226
	Avg	0.0363	0.0334	0.0372	0.0442	0.0129	0.1291	0.0065	0.1813	0.0191
	Best	0.0046	0.0046	0.0046	0.0046	0.0046	0.0054	0.0046	0.0046	0.0046
	SD	0.0488	0.0413	0.0289	0.0238	0.0314	0.0432	0.0067	0.0744	0.0372
	Rank	5	4	6	7	2	8	**1**	9	3

2) COMPARISON OF PERFORMANCE ON DATASETS ID#15-ID#32

In this section, a number of state-of-the-art FS algorithms are compared with the proposed algorithms under the following:

1) Section A: comparison under fitness values.
2) Section B: comparison under classification accuracy.
3) Section C: comparison under selected features number.

a: COMPARISON UNDER FITNESS VALUES

Tables 13, and 14 show the performance of the algorithms under fitness values on the datasets ID#15-ID#32, showing the superiority of the proposed algorithms for 13 out of 18 datasets; more specific, table 13 that shows outcomes for the instances from ID#15 to ID#23 elaborates that LIBGWO_SAV3 is more superior for 4 out of 6 instances in all independent runs, while LIBGWOV3 is the best for only the instance: ID#21. As a result, the proposed algorithms on the instances shown in this table could perform better in all independent runs in five out of seven instances, while the other two instances: ID#15 and ID#21 are better solved using PSO and GWOTM, respectively. In addition, Table 14 which displays outcomes for the instances from ID#24 to ID#32 shows the efficacy of LIBGWO_SAV3 for seven instances and LIBGWOV3 for only one instance in all independent runs rather than ID#31. Consequently, from those two tables which include a total of 18 instances, the proposed algorithm LIBGWO_SAV3 can be better in 11 out of 18 instances and LIBGWOV3 is better in only two, while the optimality of the other instances is distributed among the rival algorithms, so LIBGWO_SAV3 considers the best to solve the feature selection problem.

Additionally, to display the outcomes presented in tables 13 and 14 in a more clear way, Fig 16 (a), and 16 (b)

TABLE 14. Fitness values obtained under different algorithms on datasets ID#24-ID#32.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
24	**Worst**	0.1879	0.1921	0.1788	0.1808	0.1681	0.1912	0.1756	0.2171	0.1832
	Avg	0.1736	0.1802	0.1712	0.1713	0.1570	0.1878	0.1628	0.1922	0.1605
	Best	0.1604	0.1604	0.1570	0.1612	0.1471	0.1667	0.1530	0.1560	0.1394
	SD	0.0081	0.0074	0.0046	0.0049	0.0047	0.0055	0.0073	0.0144	0.0091
	Rank	6	7	4	5	**1**	8	3	9	2
25	**Worst**	0.1619	0.1634	0.1791	0.1791	0.1791	0.1791	0.1791	0.1791	0.1791
	Avg	0.1322	0.1449	0.1561	0.1548	0.1571	0.1763	0.1732	0.1497	0.1661
	Best	0.0961	0.1249	0.1341	0.1142	0.1053	0.1605	0.1420	0.1139	0.1420
	SD	0.0171	0.0105	0.0108	0.0130	0.0195	0.0053	0.0126	0.0184	0.0142
	Rank	**1**	2	5	4	6	9	8	3	7
26	**Worst**	0.0297	0.0300	0.0350	0.0332	0.0431	0.1572	0.0306	0.1620	0.0692
	Avg	0.0136	0.0167	0.0242	0.0210	0.0179	0.0410	0.0209	0.0377	0.0284
	Best	0.0018	0.0024	0.0169	0.0041	0.0018	0.0175	0.0038	0.0018	0.0021
	SD	0.0099	0.0085	0.0068	0.0078	0.0132	0.0328	0.0091	0.0278	0.0148
	Rank	**1**	2	6	5	3	9	4	8	**7**
27	**Worst**	0.0304	0.0319	0.0349	0.0334	0.0576	0.0875	0.0439	0.0851	0.0851
	Avg	0.0136	0.0166	0.0231	0.0218	0.0197	0.0373	0.0150	0.0404	0.0382
	Best	0.0018	0.0029	0.0041	0.0026	0.0018	0.0170	0.0029	0.0161	0.0018
	SD	0.0096	0.0077	0.0088	0.0085	0.0132	0.0158	0.0138	0.0189	0.0215
	Rank	**1**	3	6	5	4	7	2	9	8
28	**Worst**	0.0912	0.0912	0.0923	0.0915	0.0930	0.1001	0.0907	0.0975	0.0931
	Avg	0.0840	0.0879	0.0876	0.0871	0.0862	0.0924	0.0885	0.0908	0.0855
	Best	0.0790	0.0845	0.0790	0.0791	0.0790	0.0837	0.0863	0.0837	0.0783
	SD	0.0032	0.0017	0.0025	0.0028	0.0032	0.0038	0.0014	0.0037	0.0040
	Rank	**1**	3	6	5	4	9	7	8	**2**
29	**Worst**	0.0023	0.0035	0.0059	0.0073	0.0078	0.0204	0.0088	0.0060	0.0100
	Avg	0.0004	0.0011	0.0019	0.0026	0.0023	0.0096	0.0075	0.0011	0.0032
	Best	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0060	0.0001	0.0001
	SD	0.0005	0.0010	0.0020	0.0019	0.0024	0.0047	0.0011	0.0018	0.0032
	Rank	**1**	2	3	5	4	8	7	2	6
30	**Worst**	0.0646	0.0646	0.0685	0.0683	0.0708	0.0777	0.0681	0.0717	0.0727
	Avg	0.0584	0.0590	0.0640	0.0638	0.0646	0.0685	0.0619	0.0653	0.0672
	Best	0.0563	0.0563	0.0563	0.0563	0.0563	0.0563	0.0563	0.0563	0.0563
	SD	0.0027	0.0029	0.0032	0.0040	0.0044	0.0050	0.0053	0.0039	0.0037
	Rank	**1**	2	5	4	6	9	3	7	8
31	**Worst**	0.1680	0.1858	0.2028	0.2033	0.2028	0.2257	0.1498	0.2023	0.2388
	Avg	0.1359	0.1585	0.1725	0.1698	0.1466	0.1904	0.1460	0.1593	0.1795
	Best	0.1118	0.1136	0.1473	0.1347	0.0951	0.1127	0.1320	0.1118	0.1109
	SD	0.0166	0.0154	0.0152	0.0145	0.0281	0.0293	0.0070	0.0201	0.0353
	Rank	**1**	4	7	6	3	9	2	5	8
32	**Worst**	0.0255	0.0208	0.0336	0.0352	0.0441	0.0799	0.0277	0.0621	0.0679
	Avg	0.0122	0.0122	0.0190	0.0187	0.0215	0.0389	0.0220	0.0282	0.0279
	Best	0.0044	0.0044	0.0103	0.0103	0.0092	0.0056	0.0139	0.0044	0.0092
	SD	0.0046	0.0046	0.0068	0.0062	0.0083	0.0162	0.0050	0.0140	0.0141
	Rank	**1**	**1**	3	2	4	8	5	7	6

display the average of the rank values and the SD values, respectively. 16 (a) shows that LIBGWO_SAV3 performs best with a value of 3.000, while WOA performs worse with a value of 8.222. Fig. 16 (b) shows that BA is more stable and GA is the worst. Similarly, for the datasets, ID#24-ID32, Fig. 17 (a), and Fig. 17 (b) illustrate that LIBGWO_SAV3 is the best under the average of rank values with a value of 1.55, and LIBGWOV3 is the best under the average of SD values.

b: COMPARISON UNDER CLASSIFICATION ACCURACY

Tables 15 and 16 show the classification accuracy under the subset of features obtained by each algorithm on the datasets ID#15-ID23 and ID#24-ID#32. After inspecting those tables, we conclude that most bold values are within the columns of the proposed algorithms and this is proved in Fig. 18 (a) that told us that the proposed algorithm, LIBGWO_SAV3 could achieve the first rank with a value of 2.5556 for the average of the rank value obtained under each instance by each algorithm and LIBGWOV3 as the second proposed algorithm came in the second rank. Fig. 18 (b) that shows the average of SD obtained by each algorithm on the instances ID#15-ID#32 shows that BA is more stable, followed by NLPSO as the second most stable and LIBGWOV3 is third.

c: COMPARISON UNDER A SELECTED NUMBER OF FEATURES

After comparing our proposed algorithms under both fitness values and classification accuracy, we compare the number of features obtained by each algorithm. To do that, we experiment using each algorithm on each instance within 30 independent runs then the best, worst, Avg, SD, and rank under Avg are extracted and shown in Tables 17, and 18

TABLE 15. Classification accuracy values obtained for different algorithms on datasets ID#15-ID#23.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
15	**Worst**	0.8462	0.7308	0.8462	0.9231	0.8846	0.7692	0.6538	0.9615	0.8846
	Avg	0.6862	0.6646	0.8026	0.7641	0.7808	0.6628	0.6436	0.6897	0.7244
	Best	0.6538	0.6538	0.7308	0.6923	0.6538	0.5769	0.6154	0.5769	0.5769
	SD	0.0518	0.0231	0.0381	0.0513	0.0660	0.0381	0.0170	0.0842	0.0921
	Rank	6	7	**1**	3	2	8	9	5	4
16	**Worst**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	Avg	1.0000	1.0000	0.9998	0.9998	0.9990	0.9952	0.9976	0.9948	0.9976
	Best	1.0000	1.0000	0.9929	0.9929	0.9929	0.9929	0.9929	0.9857	0.9929
	SD	0.0000	0.0000	0.0013	0.0013	0.0024	0.0034	0.0034	0.0045	0.0034
	Rank	**1**	**1**	2	2	3	5	4	6	4
17	**Worst**	1.0000	0.9375	1.0000	1.0000	1.0000	1.0000	0.8854	1.0000	1.0000
	Avg	0.9208	0.8654	0.8889	0.8764	0.8972	0.8181	0.8396	0.9200	0.9181
	Best	0.8646	0.8229	0.8125	0.8021	0.8125	0.7708	0.8229	0.8542	0.8021
	SD	0.0367	0.0287	0.0653	0.0635	0.0705	0.0528	0.0197	0.0587	0.0755
	Rank	**1**	7	5	6	4	9	8	2	3
18	**Worst**	0.2787	0.2787	0.2787	0.2787	0.2787	0.2727	0.2787	0.2787	0.2787
	Avg	0.2733	0.2725	0.2705	0.2724	0.2712	0.2624	0.2644	0.2602	0.2614
	Best	0.2620	0.2620	0.2620	0.2632	0.2608	0.2500	0.2512	0.2368	0.2392
	SD	0.0051	0.0058	0.0049	0.0046	0.0075	0.0062	0.0059	0.0103	0.0098
	Rank	**1**	2	5	3	4	7	6	9	8
19	**Worst**	0.9074	0.9074	0.9074	0.9074	0.9074	0.8889	0.8889	0.8889	0.8889
	Avg	0.8719	0.8622	0.8506	0.8488	0.8315	0.7895	0.8160	0.8210	0.8167
	Best	0.8148	0.8148	0.7222	0.7778	0.7222	0.7037	0.7037	0.7037	0.7037
	SD	0.0251	0.0267	0.0419	0.0306	0.0389	0.0536	0.0572	0.0429	0.0484
	Rank	**1**	2	3	4	5	9	8	6	**7**
20	**Worst**	0.8034	0.8034	0.8034	0.7949	0.8034	0.7778	0.7949	0.7949	0.8034
	Avg	0.7832	0.7863	0.7783	0.7803	0.7769	0.7638	0.7630	0.7664	0.7652
	Best	0.7692	0.7692	0.7436	0.7692	0.7607	0.7350	0.7350	0.7350	0.7350
	SD	0.0105	0.0111	0.0101	0.0067	0.0113	0.0116	0.0148	0.0137	0.0176
	Rank	2	**1**	4	3	5	8	9	6	**7**
21	**Worst**	1.0000	1.0000	1.0000	1.0000	1.0000	0.9650	1.0000	1.0000	1.0000
	Avg	0.7872	0.7742	0.8863	0.9158	1.0000	0.7570	0.9987	0.7113	0.9908
	Best	0.6850	0.6750	0.7850	0.8100	1.0000	0.7250	0.9900	0.6700	0.7250
	SD	0.1318	0.1156	0.0691	0.0627	0.0000	0.0651	0.0034	0.0969	0.0494
	Rank	6	7	5	4	**1**	8	2	9	**3**
22	**Worst**	0.7300	0.7250	0.7400	0.7400	0.7400	0.7400	0.7300	0.7300	0.7300
	Avg	0.7144	0.7100	0.7262	0.7322	0.7265	0.7103	0.7087	0.7087	0.7077
	Best	0.7050	0.7050	0.7050	0.7050	0.7050	0.7050	0.7050	0.7050	0.7050
	SD	0.0094	0.0060	0.0121	0.0099	0.0134	0.0092	0.0078	0.0068	0.0067
	Rank	4	6	3	**1**	2	5	7	7	8
23	**Worst**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	Avg	0.9692	0.9724	0.9693	0.9625	0.9917	0.8788	0.9983	0.8215	0.9853
	Best	0.8800	0.8750	0.8950	0.9200	0.8550	0.8550	0.9750	0.6950	0.8800
	SD	0.0496	0.0413	0.0282	0.0233	0.0316	0.0425	0.0062	0.0755	0.0376
	Rank	5	4	6	7	2	8	**1**	9	3

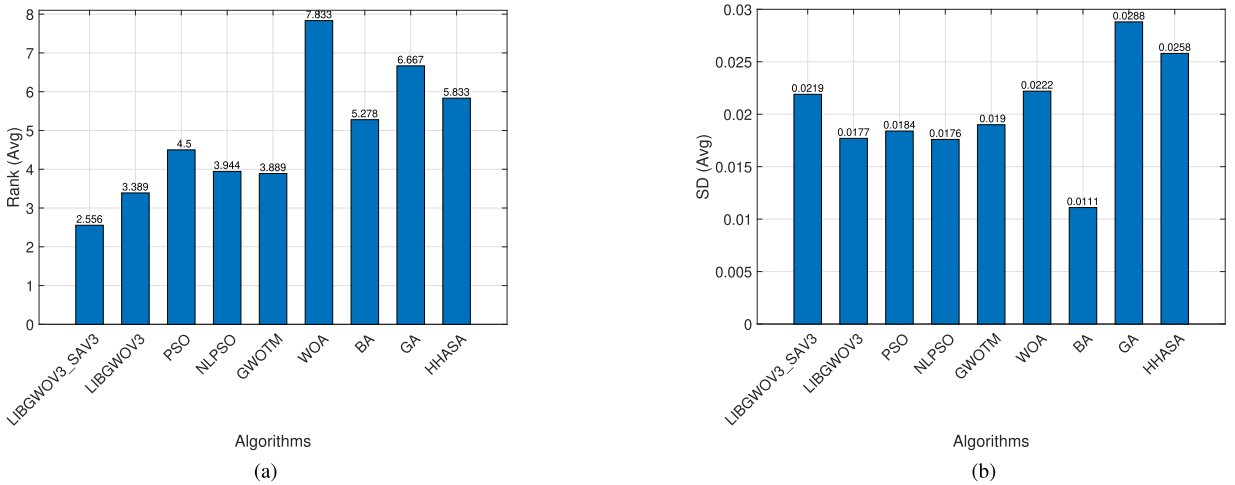


FIGURE 18. (a)Average Rank values under classification accuracy on ID#15-ID#32 and (b)Average SD values under classification accuracy on ID#15-ID#32.

TABLE 16. Classification accuracy values obtained for different algorithms on datasets ID#24-ID#32.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
24	**Worst**	0.8420	0.8420	0.8500	0.8450	0.8580	0.8400	0.8520	0.8460	0.8660
	Avg	0.8287	0.8225	0.8358	0.8357	0.8489	0.8198	0.8435	0.8091	0.8450
	Best	0.8130	0.8110	0.8280	0.8270	0.8350	0.8170	0.8300	0.7850	0.8230
	SD	0.0085	0.0077	0.0045	0.0047	0.0052	0.0051	0.0074	0.0146	0.0096
	Rank	6	7	4	5	**1**	8	3	9	2
25	**Worst**	0.9048	0.8762	0.8667	0.8857	0.8952	0.8381	0.8571	0.8857	0.8571
	Avg	0.8678	0.8564	0.8444	0.8460	0.8425	0.8229	0.8254	0.8498	0.8327
	Best	0.8381	0.8381	0.8190	0.8190	0.8190	0.8190	0.8190	0.8190	0.8190
	SD	0.0173	0.0107	0.0114	0.0133	0.0204	0.0068	0.0133	0.0187	0.0147
	Rank	**1**	2	5	4	6	9	8	3	7
26	**Worst**	1.0000	1.0000	0.9865	1.0000	1.0000	0.9865	1.0000	1.0000	1.0000
	Avg	0.9892	0.9865	0.9811	0.9842	0.9847	0.9644	0.9820	0.9644	0.9739
	Best	0.9730	0.9730	0.9730	0.9730	0.9595	0.8514	0.9730	0.8378	0.9324
	SD	0.0101	0.0085	0.0066	0.0079	0.0134	0.0321	0.0094	0.0282	0.0152
	Rank	**1**	2	6	4	3	8	5	8	**7**
27	**Worst**	1.0000	1.0000	1.0000	1.0000	1.0000	0.9861	1.0000	0.9861	1.0000
	Avg	0.9889	0.9872	0.9819	0.9829	0.9829	0.9676	0.9880	0.9616	0.9639
	Best	0.9722	0.9722	0.9722	0.9722	0.9444	0.9167	0.9583	0.9167	0.9167
	SD	0.0096	0.0078	0.0089	0.0085	0.0133	0.0158	0.0142	0.0192	0.0217
	Rank	**1**	3	6	5	4	7	2	9	8
28	**Worst**	0.9238	0.9199	0.9269	0.9277	0.9261	0.9230	0.9176	0.9191	0.9285
	Avg	0.9174	0.9161	0.9183	0.9190	0.9187	0.9136	0.9154	0.9120	0.9189
	Best	0.9121	0.9121	0.9145	0.9152	0.9106	0.9090	0.9129	0.9051	0.9121
	SD	0.0033	0.0019	0.0024	0.0027	0.0037	0.0033	0.0017	0.0038	0.0048
	Rank	**3**	4	2	1	6	8	7	9	**5**
29	**Worst**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9970	1.0000	1.0000
	Avg	0.9999	0.9995	0.9990	0.9986	0.9990	0.9928	0.9961	0.9992	0.9972
	Best	0.9985	0.9977	0.9955	0.9955	0.9924	0.9848	0.9955	0.9947	0.9902
	SD	0.0003	0.0007	0.0014	0.0013	0.0019	0.0038	0.0007	0.0015	0.0032
	Rank	**1**	2	4	5	6	9	8	3	7
30	**Worst**	0.9461	0.9461	0.9461	0.9461	0.9461	0.9461	0.9461	0.9461	0.9461
	Avg	0.9440	0.9438	0.9394	0.9401	0.9381	0.9363	0.9405	0.9374	0.9353
	Best	0.9388	0.9388	0.9352	0.9352	0.9315	0.9288	0.9342	0.9306	0.9306
	SD	0.0026	0.0028	0.0029	0.0035	0.0044	0.0044	0.0053	0.0040	0.0039
	Rank	**1**	2	5	4	6	8	3	7	9
31	**Worst**	0.8889	0.8889	0.8519	0.8704	0.9074	0.8889	0.8704	0.8889	0.8889
	Avg	0.8652	0.8430	0.8290	0.8321	0.8549	0.8117	0.8556	0.8407	0.8198
	Best	0.8333	0.8148	0.7963	0.7963	0.7963	0.7778	0.8519	0.7963	0.7593
	SD	0.0170	0.0158	0.0149	0.0151	0.0291	0.0295	0.0074	0.0206	0.0361
	Rank	**1**	4	7	6	3	9	2	5	8
32	**Worst**	1.0000	1.0000	0.9941	0.9941	0.9941	1.0000	0.9882	1.0000	0.9941
	Avg	0.9908	0.9920	0.9865	0.9871	0.9824	0.9669	0.9812	0.9745	0.9749
	Best	0.9765	0.9824	0.9706	0.9706	0.9588	0.9294	0.9765	0.9412	0.9353
	SD	0.0050	0.0047	0.0071	0.0062	0.0089	0.0158	0.0044	0.0144	0.0143
	Rank	**2**	**1**	4	3	5	9	6	8	7

on the datasets ID#15-ID#23 and ID#24-ID#32. The tables show that GA, HHASA perform better under features length in Fig. 19 (a). Fig. 19 (b) shows the SD of the selected features length obtained by each algorithm, confirming that our proposed algorithm, LIBGWO_SAV3, is more stable within 30 independent runs.

Although both GA and HHAGA perform better under selected features length, their performance is bad under classification accuracy. Since the main objective of solving FS is to find the subset of features that lead to higher accuracy with fewer features as the second objective, our proposed algorithm is better because it achieves the best accuracy with an average of the rank values under the average of selected features length equal to 3, while GA has an average of 1.94, and HHAGA has an average of 2.16. According to the previous analysis, the number of features extracted by the proposed is close to that obtained by the others. Consequently, our proposed algorithm is best.

G. COMPARISON AMONG ALGORITHMS UNDER INTERVAL PLOT

Within this section, the algorithms are compared using Interval Plot with 95% as a confidence interval under the largest instance/dataset D#29, clean2, in addition to D#1 instance. Fig. 20 represents the interval plot under both accuracy and fitness metrics for the algorithms on D#1 and D#29. Observing this figure shows that our proposed algorithms outperform the others under both fitness and accuracy for D#1. For D#29, the LIBGWO_SAV3 outperforms all the others, while LIBGWOV3 was competitive with GA on this instance. Generally, we conclude that LIBGWO_SAV3 performs well and outperforms the others for datasets with a significant number of features.

TABLE 17. Selected features length under different algorithms on datasets ID#15-ID#23.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
15	**Worst**	19.0000	107.0000	31.0000	64.0000	158.0000	157.0000	139.0000	7.0000	5.0000
	Avg	3.4800	54.4800	8.1333	11.4000	8.1000	64.9333	121.8000	3.1667	1.8333
	Best	1.0000	28.0000	1.0000	1.0000	1.0000	1.0000	110.0000	2.0000	1.0000
	SD	3.2264	20.2270	7.4866	12.9192	28.1263	48.0152	8.1584	0.9339	0.8975
	Rank	3	7	5	6	4	8	9	2	**1**
16	**Worst**	5.0000	6.0000	6.0000	7.0000	5.0000	9.0000	5.0000	5.0000	5.0000
	Avg	3.3200	3.8000	4.3667	4.7333	4.0333	5.7000	4.1333	3.7333	3.8000
	Best	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	SD	0.5455	0.8000	1.0483	0.9978	0.7063	1.8466	0.8055	0.7272	0.7483
	Rank	**1**	3	6	7	4	8	5	2	3
17	**Worst**	32.0000	52.0000	138.0000	94.0000	137.0000	168.0000	138.0000	32.0000	35.0000
	Avg	18.5200	33.4000	23.5000	23.3333	40.7333	74.4333	73.8667	8.3333	7.4000
	Best	1.0000	17.0000	1.0000	3.0000	1.0000	3.0000	45.0000	1.0000	1.0000
	SD	6.7000	10.0200	28.1280	24.3219	35.6622	46.0193	25.0463	7.0538	7.6009
	Rank	3	6	5	4	7	9	8	2	**1**
18	**Worst**	5.0000	6.0000	7.0000	7.0000	5.0000	7.0000	5.0000	5.0000	5.0000
	Avg	4.8400	4.8800	4.9000	5.3667	4.6667	5.6667	4.2667	3.8667	4.1000
	Best	3.0000	3.0000	3.0000	4.0000	3.0000	3.0000	4.0000	3.0000	3.0000
	SD	0.4630	0.5879	0.9074	0.7063	0.6498	1.0435	0.4422	0.8055	0.9434
	Rank	5	6	7	8	4	9	3	**1**	2
19	**Worst**	6.0000	8.0000	9.0000	9.0000	9.0000	13.0000	9.0000	7.0000	9.0000
	Avg	4.3600	5.4000	6.7667	6.1667	4.8333	6.3333	5.8667	3.9667	3.9667
	Best	3.0000	3.0000	4.0000	3.0000	2.0000	2.0000	3.0000	2.0000	2.0000
	SD	0.8429	1.5232	1.3585	1.7528	1.5934	2.9814	1.7075	1.2243	1.5808
	Rank	2	5	8	6	3	7	4	**1**	**1**
20	**Worst**	7.0000	5.0000	9.0000	9.0000	7.0000	9.0000	7.0000	5.0000	7.0000
	Avg	3.4400	3.6800	5.4667	5.3000	4.7333	5.7333	3.1333	3.1333	2.7667
	Best	2.0000	2.0000	2.0000	2.0000	2.0000	1.0000	1.0000	1.0000	1.0000
	SD	1.1689	0.8818	2.0934	2.2679	1.8427	2.0320	1.6680	0.9911	1.2565
	Rank	3	4	7	6	5	8	2	2	**1**
21	**Worst**	9.0000	10.0000	11.0000	11.0000	6.0000	13.0000	8.0000	8.0000	8.0000
	Avg	7.2000	7.7600	9.2333	8.8667	6.0000	12.1333	6.8000	4.0000	6.0667
	Best	4.0000	5.0000	7.0000	7.0000	6.0000	8.0000	6.0000	1.0000	6.0000
	SD	1.2961	1.4773	1.0858	1.1175	0.0000	1.5434	0.5416	3.3166	0.3590
	Rank	5	6	8	7	2	9	4	**1**	3
22	**Worst**	8.0000	10.0000	12.0000	12.0000	12.0000	10.0000	8.0000	8.0000	7.0000
	Avg	3.6800	3.2000	9.4000	10.1333	8.3667	2.3333	1.5333	2.3000	1.1333
	Best	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	SD	3.3790	3.6770	3.5553	2.8016	4.5128	3.6086	3.0739	3.1000	2.5395
	Rank	6	5	8	9	7	4	2	3	**1**
23	**Worst**	10.0000	11.0000	11.0000	11.0000	7.0000	13.0000	9.0000	8.0000	8.0000
	Avg	6.1200	7.8400	8.8333	9.1667	6.0333	11.9000	6.3333	5.9667	6.0000
	Best	5.0000	6.0000	6.0000	6.0000	6.0000	7.0000	6.0000	3.0000	5.0000
	SD	0.9516	1.3170	1.3683	1.0980	0.1795	1.7767	0.7888	1.5808	0.4472
	Rank	4	6	7	8	3	9	5	**1**	2

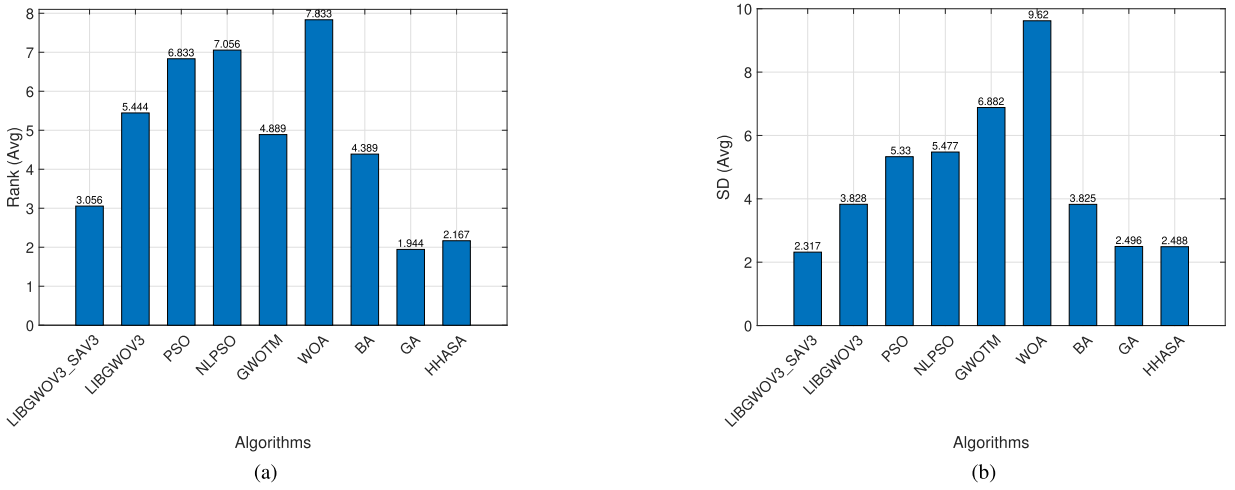


FIGURE 19. (a)Average Rank values under selected features on ID#15-ID#32 and (b)Average Avg values under selected features on ID#15-ID#32.

TABLE 18. Selected features length under different algorithms on datasets ID#24-ID#32.

ID#		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
24	**Worst**	26.0000	26.0000	38.0000	38.0000	37.0000	40.0000	37.0000	20.0000	36.0000
	Avg	15.8000	17.8000	34.6333	34.6000	29.7333	37.7000	31.1333	12.9333	28.4000
	Best	8.0000	13.0000	26.0000	28.0000	19.0000	24.0000	19.0000	6.0000	14.0000
	SD	4.0497	2.9933	3.2709	2.8119	3.7677	4.0179	4.6457	3.4345	6.2960
	Rank	2	3	8	7	5	9	6	**1**	4
25	**Worst**	8.0000	16.0000	27.0000	26.0000	22.0000	24.0000	9.0000	8.0000	7.0000
	Avg	5.1600	10.4800	7.9667	9.1000	4.8000	3.5000	1.2667	3.8667	1.7667
	Best	2.0000	3.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	SD	1.4333	3.3837	7.2318	6.8865	5.1666	7.0226	2.8860	1.9956	2.0605
	Rank	6	9	7	8	5	3	**1**	4	2
26	**Worst**	11.0000	14.0000	28.0000	29.0000	13.0000	34.0000	14.0000	11.0000	15.0000
	Avg	8.0400	11.2000	18.5000	18.4333	9.3667	19.6000	10.5333	8.3333	8.6000
	Best	5.0000	8.0000	12.0000	11.0000	5.0000	12.0000	7.0000	5.0000	5.0000
	SD	1.6120	1.6971	3.7126	4.8489	2.2433	5.4748	1.8209	1.7951	2.4304
	Rank	**1**	6	7	6	4	9	5	2	3
27	**Worst**	13.0000	18.0000	28.0000	26.0000	14.0000	31.0000	17.0000	15.0000	14.0000
	Avg	8.9200	13.2800	17.8333	16.3333	9.1667	17.9000	10.4000	7.9667	8.4333
	Best	5.0000	10.0000	9.0000	8.0000	6.0000	8.0000	7.0000	5.0000	5.0000
	SD	2.2257	2.3752	4.4802	4.3538	2.3106	5.4611	2.3889	2.1677	2.2610
	Rank	3	5	8	7	9	6	4	**1**	2
28	**Worst**	19.0000	24.0000	33.0000	32.0000	32.0000	36.0000	26.0000	18.0000	29.0000
	Avg	14.8800	17.5200	24.4333	25.1000	20.6000	24.8000	17.2000	13.2667	18.6667
	Best	11.0000	14.0000	16.0000	16.0000	12.0000	14.0000	14.0000	9.0000	11.0000
	SD	2.1600	2.3172	4.1528	4.0853	5.1549	5.3066	4.4900	2.5940	4.7140
	Rank	2	4	7	9	6	8	3	**1**	5
29	**Worst**	17.0000	40.0000	48.0000	59.0000	64.0000	101.0000	72.0000	34.0000	19.0000
	Avg	5.2800	19.6000	15.2667	19.9333	21.6000	42.5333	60.6000	5.8667	6.3000
	Best	1.0000	4.0000	1.0000	1.0000	1.0000	2.0000	50.0000	1.0000	1.0000
	SD	5.8139	9.2000	13.6331	14.6058	22.2555	24.5204	7.0880	8.2289	5.1585
	Rank	**1**	6	4	5	7	8	9	2	3
30	**Worst**	4.0000	5.0000	7.0000	6.0000	5.0000	9.0000	3.0000	5.0000	5.0000
	Avg	3.0400	3.3600	4.0000	4.4333	3.3000	5.5000	3.0000	3.3667	3.1333
	Best	3.0000	3.0000	3.0000	3.0000	2.0000	3.0000	3.0000	2.0000	2.0000
	SD	0.1960	0.5571	0.9661	0.9195	0.5859	1.5438	0.0000	0.7063	0.7180
	Rank	2	6	7	8	4	9	**1**	5	3
31	**Worst**	19.0000	22.0000	36.0000	36.0000	35.0000	35.0000	16.0000	15.0000	15.0000
	Avg	10.8000	13.4000	14.2000	15.8000	13.0667	17.8000	13.2000	7.0333	4.6667
	Best	3.0000	6.0000	3.0000	2.0000	2.0000	4.0000	10.0000	2.0000	2.0000
	SD	4.2143	3.8471	8.7231	9.7345	6.8602	7.9095	2.0396	3.0820	3.4960
	Rank	3	5	7	8	4	9	4	2	**1**
32	**Worst**	8.0000	13.0000	16.0000	16.0000	12.0000	18.0000	8.0000	8.0000	9.0000
	Avg	5.6800	7.6800	10.1667	10.6667	7.3000	10.9000	6.0000	5.3000	5.4333
	Best	4.0000	4.0000	5.0000	7.0000	4.0000	5.0000	4.0000	3.0000	4.0000
	SD	1.4344	2.0143	2.7335	2.3570	2.2531	3.0370	1.2649	1.1874	1.2828
	Rank	**3**	6	7	8	5	9	4	2	**1**

TABLE 19. Summary of experiments under Fitness, accuracy, selected Features lengths.

Criteria		LIBGWO_SAV3	LIBGWOV3	PSO [32]	NLPSO [30]	GWOTM [29]	WOA [28]	BA [112]	GA	HHASA [33]
Average fitness value	**Fitness (Avg)**	**0.150347**	0.158731	0.159322	0.1609	0.155588	0.201191	0.169791	0.176991	0.166697
	Rank (Avg)	**1.8125**	2.8125	4.4375	4.34375	4	8.34375	5.625	6.28125	6.03125
	SD (Avg)	**0.017703**	0.017331	0.019097	0.018359	0.022453	0.024444	0.018125	0.028597	0.026666
Average classification accuracy	**Accuracy (Avg)**	**0.850688**	0.842806	0.843378	0.841763	0.846313	0.802016	0.832178	0.823453	0.834181
	Rank (Avg)	**1.9375**	2.78125	4.125	4	4.03125	7.9375	5.6875	6.40625	6.3125
	SD (Avg)	**0.018616**	0.017563	0.019094	0.018363	0.022809	0.024081	0.018191	0.029091	0.027244
Average number of features	**Selected (Avg)**	5.8325	9.61625	9.619791	9.894784	8.459372	15.40312	14.47291	**4.815628**	5.233325
	Rank (Avg)	2.96875	5.28125	6.90625	6.71875	5	8.25	5.15625	1.9375	**1.90625**
	SD (Avg)	**1.727156**	2.725388	3.915325	4.029625	4.735591	7.273506	2.985359	1.789159	1.890422

V. DISCUSSION AND RESULT SUMMARY

This section summarizes all previous experiments for better understandings. Table 19 reports the averages of the fitness values, accuracy values, and selected feature lengths based on the results obtained after solving instances ID#1-ID#32 within 30 independent runs. Averages of Ranks and the SD values are also reported in Table 19. According to Table 19, LIBGWO_SAV3 obtained the first rank under both fitness and accuracy metrics with average values of 0.15 and 0.85, respectively. However, it (LIBGWO_SAV3) comes third based on the selected feature lengths (after the GA and HHASA) with a value of 5.8. Despite this, as shown in Table 19, the difference of fitness values and accuracies compared to those competing algorithms are significantly larger by the proposed algorithm, and thus, we can claim that the proposed LIBGWO_SAV3 is the better candidate for solving

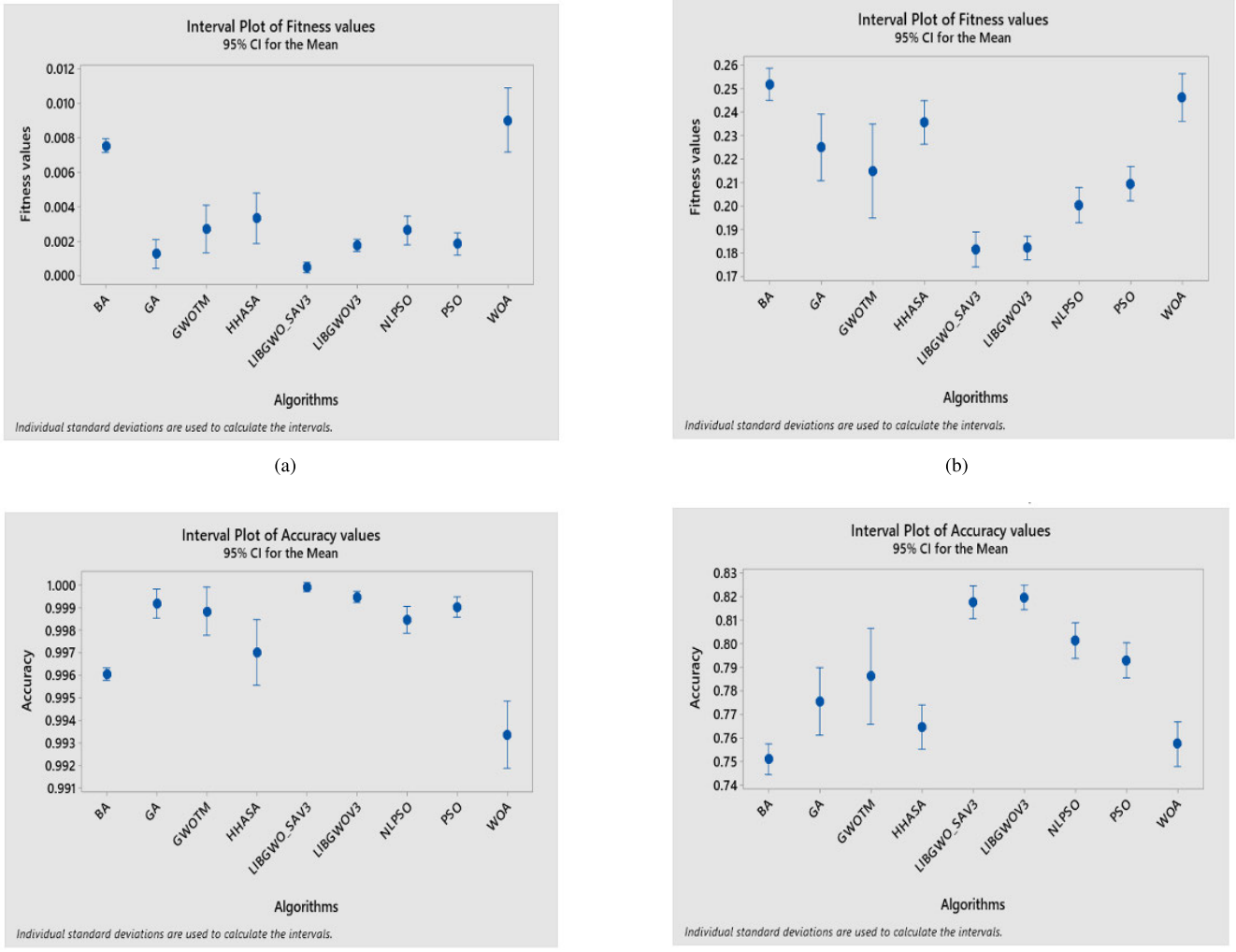


FIGURE 20. Comparison under interval plot for (a)Classification accuracy on D#29; (b) Classification accuracy on D#1; (c) Fitness values on D#1; and (d)Fitness values on D#29.

FS problems. Also, it is worth to be mentioned that our second proposed algorithm, LIBGWOV3, is second best under both fitness and accuracy metrics after LIBGWO_SAV3.

VI. CONCLUSION AND FUTURE WORK

FS is an indispensable preprocessing step to remove from datasets the irrelevant, noisy, and redundant features, in addition to reducing the data dimensionality to enable classification algorithms, or machine learning algorithms, to find better classification or clustering accuracy within less training time. So, this paper proposes three wrapper-based FS variants of the binary grey wolf optimizer (BGWO), in addition to the standard version using different S-shaped, U-shaped, and V-shaped transfer functions. The first variant integrates the standard BGWO with an exploration capability within the different phases in the optimization process by updating a specific solution randomly within the search space based on a certain probability; this variant was called IBGWO. The second variant picks the worst n, increasing with iteration, of solutions and updates them toward the best solution and randomly within the search space according to a certain probability; this variant is called LIBGWO. Finally, the last variant, abbreviated LIBGWO_SA, integrates SA with LIBGWO at the end of each iteration on the best so far solution to exploit around the best solution to identify a better solution. The proposed variants were validated on 32 datasets from the UCI repository and compared with six well-known wrapper-based FS methods to evaluate their performance. Both LIBGWO_SA and LIBGWO produce superior results compared with the other six algorithms. Our future work includes applying this improved algorithm to solving more recent applications that aims to promote and communicate advances in industrial information integration in order to provide insights into challenges, issues, and solutions related to industrial integration.

CONFLICT OF INTEREST

Authors declare that there is no conflict of interest about the research.

FUNDING

This research has no funding source.

ETHICAL APPROVAL

This article does not contain any studies with human participants or animals performed by any of the authors.

REFERENCES

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

[2] M. A. El Aziz and A. E. Hassanien, "Modified cuckoo search algorithm with rough sets for feature selection," *Neural Comput. Appl.*, vol. 29, pp. 925–934, Feb. 2018.

[3] Y. Yang, S. Slattery, and R. Ghani, "A study of approaches to hypertext categorization," *J. Intell. Inf. Syst.*, vol. 18, no. 2, pp. 219–241, 2002.

[4] R. J. Sassi, L. A. Da Silva, and E. D. M. Hernandez, "Neural networks and rough sets: A comparative study on data classification," in *Proc. IC-AI*, 2006, pp. 175–180.

[5] I. S. Kohane, A. J. Butte, and A. Kho, *Microarrays for Integrative Genomics*. Cambridge, MA, USA: MIT Press, 2002.

[6] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, nos. 1–3, pp. 389–422, 2002.

[7] A. Dacal-Nieto, E. Vazquez-Fernandez, A. Formella, F. Martìn, S. Torres-Guijarro, and H. Gonzalez-Jorge, "A genetic algorithm approach for feature selection in potatoes classification by computer vision," in *Proc. 35th Annu. Conf. Ind. Electron.*, Nov. 2009, pp. 1955–1960.

[8] W. Awada, T. M. Khoshgoftaar, D. Dittman, R. Wald, and A. Napolitano, "A review of the stability of feature selection techniques for bioinformatics data," in *Proc. IEEE 13th Int. Conf. Inf. Reuse Integr. (IRI)*, Aug. 2012, pp. 356–363.

[9] P.-C. Chang, J.-J. Lin, and C.-H. Liu, "An attribute weight assignment and particle swarm optimization algorithm for medical database classifications," *Comput. Methods Programs Biomed.*, vol. 107, no. 3, pp. 382–392, Sep. 2012.

[10] M. Goodarzi, M. P. Freitas, and R. Jensen, "Feature selection and linear/nonlinear regression methods for the accurate prediction of glycogen synthase kinase-3β inhibitory activities," *J. Chem. Inf. Model.*, vol. 49, no. 4, pp. 824–832, Apr. 2009.

[11] S. Y. Kung, Y. Luo, and M.-W. Mak, "Feature selection for genomic signal processing: Unsupervised, supervised, and self-supervised scenarios," *J. Signal Process. Syst.*, vol. 61, no. 1, pp. 3–20, Oct. 2010.

[12] W. Jiang, G. Er, Q. Dai, and J. Gu, "Similarity-based online feature selection in content-based image retrieval," *IEEE Trans. Image Process.*, vol. 15, no. 3, pp. 702–712, Mar. 2006.

[13] V. R. Balasaraswathi, M. Sugumaran, and Y. Hamid, "Feature selection techniques for intrusion detection using non-bio-inspired and bio-inspired optimization algorithms," *J. Commun. Inf. Netw.*, vol. 2, no. 4, pp. 107–119, Dec. 2017.

[14] M. A. Elaziz, A. A. Ewees, R. A. Ibrahim, and S. Lu, "Opposition-based moth-flame optimization improved by differential evolution for feature selection," *Math. Comput. Simul.*, vol. 168, pp. 48–75, Feb. 2020.

[15] I. Kojadinovic and T. Wottka, "Comparison between a filter and a wrapper approach to variable subset selection in regression problems," in *Proc. Eur. Symp. Intell. Techn. (ESIT)*, 2000, pp. 1–11.

[16] M. Abdel-Basset, V. Chang, and R. Mohamed, "A novel equilibrium optimization algorithm for multi-thresholding image segmentation problems," *Neural Comput. Appl.*, vol. 33, no. 17, pp. 10685–10718, Mar. 2020.

[17] M. Allaoui, B. Ahiod, and M. El Yafrani, "A hybrid crow search algorithm for solving the DNA fragment assembly problem," *Expert Syst. Appl.*, vol. 102, pp. 44–56, Jul. 2018.

[18] M. Abdel-Basset, R. Mohamed, S. Mirjalili, R. K. Chakrabortty, and M. J. Ryan, "Solar photovoltaic parameter estimation using an improved equilibrium optimizer," *Sol. Energy*, vol. 209, pp. 694–708, Oct. 2020.

[19] M. Abdel-Basset, R. Mohamed, K. M. Sallam, R. K. Chakrabortty, and M. J. Ryan, "An efficient-assembler whale optimization algorithm for DNA fragment assembly problem: Analysis and validations," *IEEE Access*, vol. 8, pp. 222144–222167, 2020.

[20] S. Gadat and L. Younes, "A stochastic algorithm for feature selection in pattern recognition," *J. Mach. Learn. Res.*, vol. 8, pp. 509–547, May 2007.

[21] L.-Y. Chuang, C.-H. Yang, and C.-H. Yang, "Tabu search and binary particle swarm optimization for feature selection using microarray data," *J. Comput. Biol.*, vol. 16, no. 12, pp. 1689–1703, Dec. 2009.

[22] 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.

[23] K. K. Ghosh, S. Ahmed, P. K. Singh, Z. W. Geem, and R. Sarkar, "Improved binary sailfish optimizer based on adaptive β-hill climbing for feature selection," *IEEE Access*, vol. 8, pp. 83548–83560, 2020.

[24] G. I. Sayed, A. E. Hassanien, and A. T. Azar, "Feature selection via a novel chaotic crow search algorithm," *Neural Comput. Appl.*, vol. 31, no. 1, pp. 171–188, 2017.

[25] P. Anand and S. Arora, "A novel chaotic selfish herd optimizer for global optimization and feature selection," *Artif. Intell. Rev.*, vol. 53, no. 2, pp. 1441–1486, Feb. 2020.

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

[27] R. P. S. Manikandan and A. M. Kalpana, "Feature selection using fish swarm optimization in big data," *Cluster Comput.*, vol. 22, no. 5, pp. 10825–10837, Sep. 2019.

[28] A. G. Hussien, A. E. Hassanien, E. H. Houssein, S. Bhattacharyya, and M. Amin, "S-shaped binary whale optimization algorithm for feature selection," in *Recent Trends in Signal and Image Processing: ISSIP 2017*, vol. 79. Springer, 2018.

[29] M. Abdel-Basset, D. El-Shahat, I. El-henawy, V. H. C. de Albuquerque, and S. Mirjalili, "A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection," *Expert Syst. Appl.*, vol. 139, Jan. 2020, Art. no. 112824.

[30] M. Mafarja, R. Jarrar, S. Ahmad, and A. A. Abusnaina, "Feature selection using binary particle swarm optimization with time varying inertia weight strategies," in *Proc. 2nd Int. Conf. Future Netw. Distrib. Syst.*, Jun. 2018, pp. 1–9.

[31] X. Zhang, Y. Xu, C. Yu, A. A. Heidari, S. Li, H. Chen, and C. Li, "Gaussian mutational chaotic fruit fly-built optimization and feature selection," *Expert Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112976.

[32] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. Comput. Simulation*, vol. 5, Oct. 1997, pp. 4104–4108.

[33] M. Abdel-Basset, W. Ding, and D. El-Shahat, "A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection," *Artif. Intell. Rev.*, vol. 54, no. 1, pp. 593–637, Jan. 2021.

[34] S. Yadav and S. Shukla, "Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification," in *Proc. IEEE 6th Int. Conf. Adv. Comput. (IACC)*, Feb. 2016, pp. 78–83.

[35] M. Mafarja and S. Mirjalili, "Whale optimization approaches for wrapper feature selection," *Appl. Soft Comput.*, vol. 62, pp. 441–453, Jan. 2018.

[36] A. G. Hussien, E. H. Houssein, and A. E. Hassanien, "A binary whale optimization algorithm with hyperbolic tangent fitness function for feature selection," in *Proc. 8th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2017, pp. 166–172.

[37] M. Tubishat, M. A. M. Abushariah, N. Idris, and I. Aljarah, "Improved whale optimization algorithm for feature selection in Arabic sentiment analysis," *Int. J. Speech Technol.*, vol. 49, no. 5, pp. 1688–1707, May 2019.

[38] H. Zamani and M.-H. Nadimi-Shahraki, "Feature selection based on whale optimization algorithm for diseases diagnosis," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 9, p. 1243, 2016.

[39] 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.

[40] N. Neggaz, A. A. Ewees, M. A. Elaziz, and M. Mafarja, "Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection," *Expert Syst. Appl.*, vol. 145, May 2020, Art. no. 113103.

[41] M. E. A. Elaziz, A. A. Ewees, D. Oliva, P. Duan, and S. Xiong, "A hybrid method of sine cosine algorithm and differential evolution for feature selection," in *Proc. Int. Conf. Neural Inf. Process.* Springer, 2017, pp. 145–155.

[42] R. A. Ibrahim, A. A. Ewees, D. Oliva, M. Abd Elaziz, and S. Lu, "Improved salp swarm algorithm based on particle swarm optimization for feature selection," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 8, pp. 3155–3169, Aug. 2019.
[43] M. Tubishat, S. Ja'afar, M. Alswaitti, S. Mirjalili, N. Idris, M. A. Ismail, and M. S. Omar, "Dynamic salp swarm algorithm for feature selection," *Expert Syst. Appl.*, vol. 164, Feb. 2021, Art. no. 113873.
[44] A. E. Hegazy, M. Makhlouf, and G. S. El-Tawel, "Improved salp swarm algorithm for feature selection," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 32, no. 2, pp. 335–344, 2018.
[45] R. Y. Nakamura, L. A. Pereira, K. A. Costa, D. Rodrigues, J. P. Papa, and X.-S. Yang, "BBA: A binary bat algorithm for feature selection," in *Proc. 25th SIBGRAPI Conf. Graph., Patterns Images*, 2012, pp. 291–297.
[46] A. M. Taha, A. Mustapha, and S.-D. Chen, "Naive Bayes-guided bat algorithm for feature selection," *Sci. World J.*, vol. 2013, Dec. 2013, Art. no. 325973.
[47] R. Y. M. Nakamura, L. A. M. Pereira, D. Rodrigues, K. A. P. Costa, J. P. Papa, and X.-S. Yang, "Binary bat algorithm for feature selection," in *Swarm Intelligence and Bio-Inspired Computation*. Amsterdam, The Netherlands: Elsevier, 2013, pp. 225–237.
[48] B. Oluleye, L. Armstrong, J. Leng, and D. Diepeveen, "A genetic algorithm-based feature selection," *Int. J. Electron. Commun. Comput. Eng.*, vol. 5, no. 4, pp. 899–905, 2014.
[49] F. Tan, X. Fu, Y. Zhang, and A. G. Bourgeois, "A genetic algorithm-based method for feature subset selection," *Soft Comput.*, vol. 12, no. 2, pp. 111–120, Sep. 2007.
[50] S. A.-F. Sayed, E. Nabil, and A. Badr, "A binary clonal flower pollination algorithm for feature selection," *Pattern Recognit. Lett.*, vol. 77, pp. 21–27, Jul. 2016.
[51] D. Rodrigues, X.-S. Yang, A. N. De Souza, and J. P. Papa, "Binary flower pollination algorithm and its application to feature selection," in *Recent Advances in Swarm Intelligence and Evolutionary Computation*. Springer, 2015, pp. 85–100.
[52] H. Majidpour and F. S. Gharehchopogh, "An improved flower pollination algorithm with AdaBoost algorithm for feature selection in text documents classification," *J. Adv. Comput. Res.*, vol. 9, no. 1, pp. 29–40, 2018.
[53] C. Yan, J. Ma, H. Luo, G. Zhang, and J. Luo, "A novel feature selection method for high-dimensional biomedical data based on an improved binary clonal flower pollination algorithm," *Hum. Heredity*, vol. 84, no. 1, pp. 34–46, 2019.
[54] D. Rodrigues, L. A. Pereira, T. Almeida, J. P. Papa, A. Souza, C. C. Ramos, and Y. ang.-S. andX, "BCS: A binary cuckoo search algorithm for feature selection," in *Proc. IEEE Int. Symp. circuits Syst. (ISCAS)*, May 2013, pp. 465–468.
[55] L. Pereira, D. Rodrigues, T. Almeida, C. Ramos, A. Souza, X.-S. Yang, and J. Papa, "A binary cuckoo search and its application for feature selection," in *Cuckoo Search and Firefly Algorithm*. Springer, 2014, pp. 141–154.
[56] C. Gunavathi and K. Premalatha, "Cuckoo search optimisation for feature selection in cancer classification: A new approach," *Int. J. Data Mining Bioinf.*, vol. 13, no. 3, pp. 248–265, 2015.
[57] 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.
[58] R. N. Khushaba, A. Al-Ani, A. AlSukker, and A. Al-Jumaily, "A combined ant colony and differential evolution feature selection algorithm," in *Proc. Int. Conf. Colony Optim. Swarm Intell.* Springer, 2008, pp. 1–12.
[59] R. N. Khushaba, A. Al-Ani, and A. Al-Jumaily, "Differential evolution based feature subset selection," in *Proc. 19th Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.
[60] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Landscape-assisted multi-operator differential evolution for solving constrained optimization problems," *Expert Syst. Appl.*, vol. 162, Dec. 2020, Art. no. 113033.
[61] K. Sallam, S. Elsayed, R. Sarker, and D. Essam, "Landscape-based differential evolution for constrained optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Dec. 2018, pp. 1–8.
[62] K. M. Sallam, R. K. Chakrabortty, and M. J. Ryan, "A hybrid differential evolution with cuckoo search for solving resource constrained project scheduling problems," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage. (IEEM)*, Dec. 2019, pp. 1344–1348.
[63] D. S. A. Elminaam, A. Nabil, S. A. Ibraheem, and E. H. Houssein, "An efficient marine predators algorithm for feature selection," *IEEE Access*, vol. 9, pp. 60136–60153, 2021.
[64] Y. Gao, Y. Zhou, and Q. Luo, "An efficient binary equilibrium optimizer algorithm for feature selection," *IEEE Access*, vol. 8, pp. 140936–140963, 2020.
[65] G. Tikhe, T. Joshi, A. Lahorkar, A. Sane, and J. Valadi, "Feature selection using equilibrium optimizer," in *Data Engineering and Intelligent Computing*. Springer, 2021, pp. 307–315.
[66] J. Too and S. Mirjalili, "General learning equilibrium optimizer: A new feature selection method for biological data classification," *Appl. Artif. Intell.*, vol. 35, no. 3, pp. 247–263, Feb. 2021.
[67] Z. M. Elgamal, N. M. Yasin, A. Q. M. Sabri, R. Sihwail, M. Tubishat, and H. Jarrah, "Improved equilibrium optimization algorithm using elite opposition-based learning and new local search strategy for feature selection in medical datasets," *Computation*, vol. 9, no. 6, p. 68, Jun. 2021.
[68] G. I. Sayed, G. Khoriba, and M. H. Haggag, "A novel Chaotic Equilibrium Optimizer Algorithm with S-shaped and V-shaped transfer functions for feature selection," *J. Ambient Intell. Hum. Comput.*, vol. 2021, pp. 1–26, Mar. 2021.
[69] M. Abdel-Basset, R. Mohamed, R. K. Chakrabortty, M. J. Ryan, and S. Mirjalili, "An efficient binary slime mould algorithm integrated with a novel attacking-feeding strategy for feature selection," *Comput. Ind. Eng.*, vol. 153, Mar. 2021, Art. no. 107078.
[70] A. A. Ewees, L. Abualigah, D. Yousri, Z. Y. Algamal, M. A. A. Al-qaness, R. A. Ibrahim, and M. A. Elaziz, "Improved slime mould algorithm based on firefly algorithm for feature selection: A case study on QSAR model," *Eng. Comput.*, vol. 4, pp. 1–15, Mar. 2021.
[71] V. Kumar and A. Kaur, "Binary spotted hyena optimizer and its application to feature selection," *J. Ambient Intell. Hum. Comput.*, vol. 11, no. 7, pp. 2625–2645, Jul. 2020.
[72] H. Jia, J. Li, W. Song, X. Peng, C. Lang, and Y. Li, "Spotted hyena optimization algorithm with simulated annealing for feature selection," *IEEE Access*, vol. 7, pp. 71943–71962, 2019.
[73] G. Dhiman, D. Oliva, A. Kaur, K. K. Singh, S. Vimal, A. Sharma, and K. Cengiz, "BEPO: A novel binary emperor penguin optimizer for automatic feature selection," *Knowl.-Based Syst.*, vol. 211, Jan. 2021, Art. no. 106560.
[74] K. C. Lin, K. Y. Zhang, Y. H. Huang, and J. C. Hung, "Feature selection based on an improved cat swarm optimization algorithm for big data classification," *J. Supercomput.*, vol. 72, no. 8, pp. 3210–3221, 2016.
[75] K. C. Lin, K. Y. Zhang, and J. C. Hung, "Feature selection of support vector machine based on harmonious cat swarm optimization," in *Proc. 7th Int. Conf. Comput. Workshops*, 2014, pp. 205–208.
[76] R. Diao and Q. Shen, "Feature selection with harmony search," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 42, no. 6, pp. 1509–1523, Dec. 2012.
[77] C. C. O. Ramos, A. N. Souza, G. Chiachia, A. X. Falcão, and J. P. Papa, "A novel algorithm for feature selection using harmony search and its application for non-technical losses detection," *Comput. Electr. Eng.*, vol. 37, no. 6, pp. 886–894, Nov. 2011.
[78] Y. Wang, Y. Liu, L. Feng, and X. Zhu, "Novel feature selection method based on harmony search for email classification," *Knowl.-Based Syst.*, vol. 73, no. 1, pp. 311–323, Jan. 2015.
[79] H. H. Inbarani, M. Bagyamathi, and A. T. Azar, "A novel hybrid feature selection method based on rough set and improved harmony search," *Neural Comput. Appl.*, vol. 26, no. 8, pp. 1859–1880, 2015.
[80] L. Zheng, R. Diao, and Q. Shen, "Self-adjusting harmony search-based feature selection," *Soft Comput.*, vol. 19, no. 6, pp. 1567–1579, Jun. 2015.
[81] R. Diao and Q. Shen, "Two new approaches to feature selection with harmony search," in *Proc. Int. Conf. Fuzzy Syst.*, Jul. 2010, pp. 1–7.
[82] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," *Comput. Secur.*, vol. 81, pp. 148–155, Mar. 2019.
[83] A. Kumar and R. Khorwal, "Firefly algorithm for feature selection in sentiment analysis," in *Computational Intelligence in Data Mining*. Springer, 2017, pp. 693–703.
[84] F. S. Gharehchopogh, I. Maleki, and Z. A. Dizaji, "Chaotic vortex search algorithm: Metaheuristic algorithm for feature selection," *Evol. Intell.*, vol. 2021, pp. 1–32, Mar. 2021.
[85] Y. Meraihi, A. B. Gabis, A. Ramdane-Cherif, and D. Acheli, "A comprehensive survey of crow search algorithm and its applications," *Artif. Intell. Rev.*, vol. 54, no. 4, pp. 2669–2716, Apr. 2021.
[86] N. A. Al-Thanoon, Z. Y. Algamal, and O. S. Qasim, "Feature selection based on a crow search algorithm for big data classification," *Chemometric Intell. Lab. Syst.*, vol. 212, May 2021, Art. no. 104288.

[87] A. Chaudhuri and T. P. Sahu, "Feature selection using binary crow search algorithm with time varying flight length," *Expert Syst. Appl.*, vol. 168, Apr. 2021, Art. no. 114288.
[88] A. Adamu, M. Abdullahi, S. B. Junaidu, and I. H. Hassan, "An hybrid particle swarm optimization with crow search algorithm for feature selection," *Mach. Learn. with Appl.*, vol. 6, Dec. 2021, Art. no. 100108.
[89] A. Zakeri and A. Hokmabadi, "Efficient feature selection method using real-valued grasshopper optimization algorithm," *Expert Syst. Appl.*, vol. 119, pp. 61–72, Apr. 2019.
[90] Z. Y. Algamal, M. K. Qasim, M. H. Lee, and H. T. M. Ali, "Improving grasshopper optimization algorithm for hyperparameters estimation and feature selection in support vector regression," *Chemometric Intell. Lab. Syst.*, vol. 208, Jan. 2021, Art. no. 104196.
[91] C. Dey, R. Bose, K. K. Ghosh, S. Malakar, and R. Sarkar, "LAGOA: Learning automata based grasshopper optimization algorithm for feature selection in disease datasets," *J. Ambient Intell. Hum. Comput.*, vol. 2021, pp. 1–20, Mar. 2021.
[92] H. Hichem, M. Elkamel, M. Rafik, M. T. Mesaaoud, and C. Ouahiba, "A new binary grasshopper optimization algorithm for feature selection problem," *J. King Saud Univ.-Comput. Inf. Sci.*, Nov. 2019.
[93] A. Oleynik and S. Subbotin, "Feature selection based on bacteria foraging intelligence," in *Proc.10th Int. Conf. Exper. Designing Appl. CAD Syst. Microelectron.*, 2009, pp. 332–334.
[94] D. Mittal and M. Bala, "Hybrid feature selection approach using bacterial foraging algorithm guided by naive Bayes classification," in *Proc. 8th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2017, pp. 1–7.
[95] Y. P. Chen, Y. Li, G. Wang, Y. F. Zheng, Q. Xu, and J. H. Fan, "A novel bacterial foraging optimization algorithm for feature selection," *Expert Syst. Appl.*, vol. 83, pp. 1–17, Oct. 2017.
[96] M. M. Mafarja, D. Eleyan, I. Jaber, A. Hammouri, and S. Mirjalili, "Binary dragonfly algorithm for feature selection," in *Proc. Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2017, pp. 12–17.
[97] M. Mafarja, I. Aljarah, A. A. Heidari, and H. Faris, "Binary dragonfly optimization for feature selection using time-varying transfer functions," *Knowl.-Based Syst.*, vol. 161, pp. 185–204, Dec. 2018.
[98] X. Cui, Y. Li, J. Fan, T. Wang, and Y. Zheng, "A hybrid improved dragonfly algorithm for feature selection," *IEEE Access*, vol. 8, pp. 155619–155629, 2020.
[99] A. I. Hammouri, M. Mafarja, M. A. Al-Betar, M. A. Awadallah, and I. Abu-Doush, "An improved dragonfly algorithm for feature selection," *Knowl.-Based Syst.*, vol. 203, Sep. 2020, Art. no. 106131.
[100] R. Al-Wajih, S. J. Abdulkadir, N. Aziz, Q. Al-Tashi, and N. Talpur, "Hybrid binary grey wolf with Harris hawks optimizer for feature selection," *IEEE Access*, vol. 9, pp. 31662–31677, 2021.
[101] Q. Al-Tashi, S. J. A. Kadir, H. M. Rais, S. Mirjalili, and H. Alhussian, "Binary optimization using hybrid grey wolf optimization for feature selection," *IEEE Access*, vol. 7, pp. 39496–39508, 2019.
[102] Q. Al-Tashi, S. J. Abdulkadir, H. M. Rais, S. Mirjalili, H. Alhussian, M. G. Ragab, and A. Alqushaibi, "Binary multi-objective grey wolf optimizer for feature selection in classification," *IEEE Access*, vol. 8, pp. 106247–106263, 2020.
[103] Q. Al-Tashi, H. M. Rais, S. J. Abdulkadir, S. Mirjalili, and H. Alhussian, "A review of grey wolf optimizer-based feature selection methods for classification," in *Proc. Evol. Mach. Learn. Techn.*, 2020, pp. 273–286.
[104] Q. Al-Tashi, H. Rais, and S. Jadid, "Feature selection method based on grey wolf optimization for coronary artery disease classification," in *Proc. Int. Conf. Inf. Commun. Technol.* Springer, 2018, pp. 257–266.
[105] Q. Al-Tashi, H. M. Rais, S. J. Abdulkadir, and S. Mirjalili, "Feature selection based on grey wolf optimizer for oil & gas reservoir classification," in *Proc. Int. Conf. Comput. Intell. (ICCI)*, 2020, pp. 211–216.
[106] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," *Neurocomputing*, vol. 172, pp. 371–381, Jan. 2016.
[107] P. Hu, J.-S. Pan, and S.-C. Chu, "Improved binary grey wolf optimizer and its application for feature selection," *Knowl.-Based Syst.*, vol. 195, May 2020, Art. no. 105746.
[108] J. Too, A. Abdullah, N. M. Saad, N. M. Ali, and W. Tee, "A new competitive binary grey wolf optimizer to solve the feature selection problem in EMG signals classification," *Computers*, vol. 7, no. 4, p. 58, Nov. 2018.
[109] Q. Tu, X. Chen, and X. Liu, "Hierarchy strengthened grey wolf optimizer for numerical optimization and feature selection," *IEEE Access*, vol. 7, pp. 78012–78028, 2019.
[110] B. Sathiyabhama, S. U. Kumar, J. Jayanthi, T. Sathiya, A. K. Ilavarasi, V. Yuvarajan, and K. Gopikrishna, "A novel feature selection framework based on grey wolf optimizer for mammogram image analysis," *Neural Comput. Appl.*, vol. 2021, pp. 1–20, May 2021.
[111] E. Devi and R. Suganthe, "Feature selection in intrusion detection grey wolf optimizer," *Asian J. Res. Social Sci. Hum.*, vol. 7, no. 3, pp. 671–682, 2017.
[112] S. Mirjalili, S. M. Mirjalili, and X.-S. Yang, "Binary bat algorithm," *Neural Comput. Appl.*, vol. 25, nos. 3–4, pp. 663–681, Sep. 2014.
[113] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
[114] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[115] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.
[116] A. Wang, N. An, G. Chen, L. Li, and G. Alterovitz, "Accelerating wrapper-based feature selection with K-nearest-neighbor," *Knowl.-Based Syst.*, vol. 83, pp. 81–91, Jul. 2015.
[117] S. Taghian and M. H. Nadimi-Shahraki, "A binary metaheuristic algorithm for wrapper feature selection," *Int. J. Comput. Sci. Eng.*, vol. 8, no. 5, pp. 168–172, 2019.
[118] K. H. Sheikh, S. Ahmed, K. Mukhopadhyay, P. K. Singh, J. H. Yoon, Z. W. Geem, and R. Sarkar, "EHHM: Electrical harmony based hybrid meta-heuristic for feature selection," *IEEE Access*, vol. 8, pp. 158125–158141, 2020.
[119] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Neuro Comput.*, vol. 260, pp. 302–312, Oct. 2017.
[120] T. Bhattacharyya, B. Chatterjee, P. K. Singh, J. H. Yoon, Z. W. Geem, and R. Sarkar, "Mayfly in harmony: A new hybrid meta-heuristic feature selection algorithm," *IEEE Access*, vol. 8, pp. 195929–195945, 2020.
[121] M. Alweshah, S. A. Khalaileh, B. B. Gupta, A. Almomani, A. I. Hammouri, and M. A. Al-Betar, "The monarch butterfly optimization algorithm for solving feature selection problems," *Neural Comput. Appl.*, vol. 2020, pp. 1–15, Jul. 2020.

MOHAMED ABDEL-BASSET (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in operations research from the Faculty of Computers and Informatics, Zagazig University, Egypt. He is currently an Associate Professor with the Faculty of Computers and Informatics, Zagazig University. He has published more than 200 articles in international journals and conference proceedings. He is working on the application of multiobjective and robust meta-heuristic optimization techniques. His current research interests include optimization, operations research, data mining, computational intelligence, applied statistics, decision support systems, robust optimization, engineering optimization, multiobjective optimization, swarm intelligence, evolutionary algorithms, and artificial neural networks. He is an editor and a reviewer of different international journals and conferences.

KARAM M. SALLAM (Member, IEEE) received the Ph.D. degree in computer science from the University of New South Wales at Canberra, Australian Force Academy, Canberra, Australia, in 2018. He is currently a Lecturer with Zagazig University, Zagazig, Egypt. His current research interests include evolutionary algorithms and optimization, constrained-handling techniques for evolutionary algorithms, operation research, machine learning, deep learning, cybersecurity, and the IoT. He was the winner of IEEE-CEC2020 Competition. He serves as an organizing committee member of different conferences in the evolutionary computation field and a reviewer for several international journals.

REDA MOHAMED received the B.Sc. degree from the Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Egypt. He is working on the application of multiobjective and robust metaheuristic optimization techniques in computational intelligence. His research interests include robust optimization, multiobjective optimization, swarm intelligence, evolutionary algorithms, and artificial neural networks.

IBRAHIM ELGENDI (Member, IEEE) received the Ph.D. degree in information technology from the University of Canberra, Australia. He has 17 years' experience from industry in the field of automation and AI. He is currently a Lecturer in networking and cybersecurity. He is a member of the Institution of Engineers Australia. His research interests include mobile and wireless networks, the Internet of Things, machine learning, and cyber-physical-security. He has over 14 refereed publications with over 63 citations (five H-index) in highly prestigious journals and conference proceedings. He has served as a Reviewer for a number of journals, such as *IEEE Wireless Communications Magazine*, IEEE TRANSACTION ON MOBILE COMPUTING, and IEEE/ACM TRANSACTIONS ON NETWORKING.

KUMUDU MUNASINGHE (Member, IEEE) received the Ph.D. degree in telecommunications engineering from the University of Sydney. He is currently an Associate Professor in network engineering and the Leader of the IoT Research Group, Human Centred Research Centre, University of Canberra. His research interests include next generation mobile and wireless networks, Internet of Things, green communication, smart grid communications, and cyber-physical-security. He has over 100 refereed publications with over 900 citations (H-index 17) in highly prestigious journals, conference proceedings and two books to his credit. He has secured over $ 1.6 Million dollars in competitive research funding by winning grants from the Australian Research Council (ARC), the Commonwealth and State Governments, Department of Defence, and the industry. He is currently a Charted Professional Engineer, an Engineering Executive, and a Companion (Fellow Status) of Engineers Australia. He has also won the highly prestigious ARC Australian Postdoctoral Fellowship. He served as the co-chair for many international conferences. He served as an editorial board member for a number of journals. His research has been highly commended through many research awards, including two VC's Research Awards and three IEEE Best Paper Awards.

OSAMA M. ELKOMY received the B.Sc. and M.Sc. degrees from the Faculty of Engineering, Zagazig University, in 2001 and 2009, respectively, and the Ph.D. degree in information technology from Zagazig University, in 2010. He is currently a Lecturer on information technology at Zagazig University. His research interests include computer building systems, embedded systems, robots, and signal processing.