IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 13, 2020, accepted January 9, 2021, date of publication January 18, 2021, date of current version January 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3052149

Boosted Whale Optimization Algorithm With Natural Selection Operators for Software Fault Prediction

YOUSEF HASSOUNEH[1], HAMZA TURABIEH[2], THAER THAHER[3,4], IYAD TUMAR[5], HAMOUDA CHANTAR[6], AND JINGWEI TOO[7]

[1]Department of Computer Science, Birzeit University, Ramallah, Palestine
[2]Department of Information Technology, College of Computers and Information Technology, Taif University, Taif 21944, Saudi Arabia
[3]Department of Engineering and Technology Sciences, Arab American University, Ramallah, Palestine
[4]Department of Computer Science, Al-Quds University, Jerusalem, Palestine
[5]Faculty of Information Technology, Sebha University, Sebha 18758, Libya
[6]Department of Electrical and Computer Engineering, Birzeit University, Ramallah, Palestine
[7]Faculty of Electrical Engineering, Universiti Teknikal Malaysia Melaka, Malacca 76100, Malaysia

Corresponding author: Hamza Turabieh (h.turabieh@tu.edu.sa)

This work was supported by Taif University Researchers Supporting, Taif University, Taif, Saudi Arabia, under Project TURSP-2020/125.

ABSTRACT Software fault prediction (SFP) is a challenging process that any successful software should go through it to make sure that all software components are free of faults. In general, soft computing and machine learning methods are useful in tackling this problem. The size of fault data is usually huge since it is obtained from mining software historical repositories. This data consists of a large number of features (metrics). Determining the most valuable features (i.e., Feature Selection (FS) is an excellent solution to reduce data dimensionality. In this paper, we proposed an enhanced version of the Whale Optimization Algorithm (WOA) by combining it with a single point crossover method. The proposed enhancement helps the WOA to escape from local optima by enhancing the exploration process. Five different selection methods are employed: Tournament, Roulette wheel, Linear rank, Stochastic universal sampling, and random-based. To evaluate the performance of the proposed enhancement, 17 available SFP datasets are adopted from the PROMISE repository. The deep analysis shows that the proposed approach outperformed the original WOA and the other six state-of-the-art methods, as well as enhanced the overall performance of the machine learning classifier.

INDEX TERMS Software fault prediction, feature selection, binary whale optimization algorithm, adaptive synthetic sampling, classification.

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
ADASYN	Adaptive synthetic sampling method
AIRS	Artificial Immune Recognition Systems
ANN	Artificial Neural Network
AOA	Antlion Optimization Algorithm
ASD	Agile Software Development
AUC	Area Under Receiver Operating Characteristics Curve
BACO	Binary Ant Colony Optimization
BALO	Binary Ant Lion Optimizer
BBA	Binary Bat Algorithm
BGA	Binary Genetic Algorithm
BGSA	Binary Gravitational Search Algorithm
BGWO	Binary Grey Wolf Optimization
BMFO	Binary Moth Flame Optimization
BN	Bayesian Networks
BPSO	Binary Particle Swarm Optimization
BQSA	Binary Queuing Search Algorithm
BSSA	Binary Salp Swarm Algorithm
BWOA	Binary Whale Optimization Algorithm
CR	Case-Based Reasoning
DE	Deferential Evolutionary

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

DOA	Dragonfly Optimization Algorithm
DT	Decision Tree
FC	Fuzzy Clustering
FIS	Fuzzy Inference System
FS	Feature Selection
GA	Genetic Algorithm
GBO	Gradient-Based Optimizer
GBRCR	Gradient Boosting Regression-based Combination Rule
GP	Genetic Programming
GWO	Gray Wolf Optimization
HHO	Harris Hawk Optimization
k-NN	K-Nearest Neighbors
LDA	Linear Discriminant Analysis
LR	Linear Rank
LRBWOA	Linear Rank Based selection method with binary WOA
LRCR	Linear Regression-based Combination Rule
LRM	Logistic Regression Method
LRNN	Layered Recurrent Neural Network
MFO	Moth-Flame Optimization
MGA	Modified Genetic Algorithm
ML	Machine Learning
MLP	Multilayer Perceptron
NB	Naive Bayes
NFIS	Neuro-Fuzzy Inference System
PB	Proportional Based
PSO	Particle Swarm Optimization
RB	Random-Based
RF	Random Forest
SA	Simulated Annealing
SBWOA	Stochastic Universal Sampling with binary WOA
SC	Soft Computing
SDLC	Software Development Life Cycle
SDP	Software Defect Prediction
SE	Software Engineering
SFP	Software Fault Prediction
SI	Swarm Intelligence
SMA	Slime Mould Algorithm
SMOTE	Synthetic Minority Oversampling Technique
SQA	Software Quality Assurance
SUS	Stochastic Universal Sampling
SVM	Support Vector Machines
TBWOA	Tournament selection method with binary WOA
TF	Transfer Function
TS	Tournament Selection
TSA	Tabu Search Algorithm
WOA	Whale Optimization Algorithm

I. INTRODUCTION

The SDLC is a methodology that clearly defines the processes of developing reliable and free of bugs software [1]. In SDLC, the software goes through a set of stages starting with requirements elicitation, where the specifications of that software are determined, and then analysis and design of the software. The implementation, testing, and documentation stages come respectively after the completion of the design stage. Many software development methodologies have been introduced to ease and improve the software development process. The most commonly used SDLC models are waterfall, Agile and Spiral models. Empirically, testing is primarily concerned with the enhancement of the software quality and reducing the total cost [2]–[4]. The task of detecting or predicting faults in software is named SFP. In the SFP process, prior to a new version of software being developed, hidden or clear fault-prone models can be detected with the help of user comments, historical fault datasets gathered from previous projects, or predefined software matrices [3], [4]. In contrast with the classic sequential waterfall model that was introduced by [5] in 1970, the appearance of ASD in 2001 [6] makes the process of SFP easier. The basis of ASD is the incremental delivery of software development. This advantage leads to rapid adaptation of the volatile requirements and also shortens both the development time and the gap between software developers and business owners [7].

Generally speaking, the cost (in terms of time, effort, and resources) or recovering fault in the early stages of the SDLC is much lower than doing so in the later stages. Thus, predicting software faults before delivering products to customers becomes essential in order to reduce the negative influence of the latest versions of the software [3].

To guarantee that the software being developed meets the end-users needs, the development process is maintained under the control of SQA. SQA has various processes such as formal code inspection, software testing, and software fault prediction [4], [8]. Conventionally, SFP models rely on various approaches include software matrices, SC, and ML [9].

In general, SFP models are constructed depending on either certain software matrices (features) or software fault datasets (gathered in advance from previously released versions of similar projects). These models are appropriate especially when the project is very large and hard to test [3], [4], [10]. The type of SFP model that is built according to pre-defined software metrics may not be accurate in predicting faults in given software. The reason is that if an SFP model is constructed based on some pre-defined software matrices and then is used to detect bugs in different software projects without rebuilding it with updated values of the used metrics, it will not be that accurate since each project has its matrices and characteristics. In such cases, only faults related to similar areas of the software may be detected [3], [11]. Alternatively, software matrices that consider the historical changes of a software project can be the solution to this problem. However, in dealing with complex software projects, these models are known as a time consuming and infeasible [3], [4]. Thus, SC techniques can be a good solution to discover faulty modules in software projects.

Different SC techniques were successfully applied to the SFP field with much success [8], [12]–[14]. For instance, as stated in [12], a distinct type of association rules called relational association rules mining was used as a classifier to predict whether a software model is faulty or not. As another example, a supervised technique named ANFIS has been utilized for the SFP problem. It integrates the advantages of two methods, a FIS and an ANN. ANSIF has shown superior performance in addressing SFP problem when compared to SVM and ANN classifiers [13]. In general, ML is known as a core branch of SC methods that have been found effective in solving a wide range of real-life, industrial, and data mining related problems. Moreover, many studies have approved the superiority of ML classifiers in improving the performance of SFP models [15]. Some of used ML algorithms for dealing with SFP include OneR (One Rule), DTs, NB, SVM and ANN classifiers [16]–[18].

With the advancement in software development, and due to the fact the computer systems are used to manage all life aspects, the amount of available data becomes large in every field. Therefore, using basic ML techniques as fault prediction techniques becomes impractical in some cases and needs more enhancement. That's to say, The need for advanced methods to predict faults in software becomes mandatory. One of the main methods used to enhance the performance of the machine learning techniques is to reduce the dimensionality of the available datasets. Different dimensionality reduction techniques have reported in the literature, and FS [19]–[22] is one of these main techniques.

FS is a well-known dimensionality reduction technique that aims to reduce the number of available features in a dataset as it may contain some irrelevant/redundant features [23]–[26]. The presence of such features misleads the learning algorithm and consequently affects its performance. Finding and removing noisy and irrelevant features can improve the performance of the learning algorithm in terms of classification accuracy and lowers the computational time [3], [4], [27]. The authors of [28] and [29] conducted systematic reviews on ML algorithms that have applied to SFP. In contrast with classic statistical methods, they concluded that FS could significantly enhance the accuracy of SFP models that utilize ML classifiers. Correspondingly, FS methods can be classified based on two criteria; subset generation and evaluation mechanisms. In terms of evaluation, FS methods can be further categorized into wrapper and filter as two main FS models. Complete, heuristic and random search mechanisms are considered as the main subset generation mechanisms in most FS approaches.

Among the FS models, the wrapper model is frequently employed due to its excellent performance. In this model, a learning algorithm (e.g., classifier) is usually used as an evaluation criterion. So, the results are associated with the selected learning algorithm. The main advantage of the wrapper model over the filter is that the former produces feature subsets that are able to maximize the performance of a specific learning technique.

Searching for the best performing feature subset is the second aspect that should be highly considered when designing a FS approach. Meta-heuristics algorithms proved their ability to tackle the FS problem with high-performance levels. The main two categories of meta-heuristics algorithms are EA (such as GA [30] and DE [31]), and the SI (such as PSO [32], HHO [33], SMA [34], GBO [35], and WOA [36]).

Since the aim of SFP is to predict the faults in the new projects based on the historical data, then using a learning algorithm in evaluating the performance of the selected features becomes mandatory. Thus, we adopted the wrapper FS model as it employs a learning algorithm as an evaluation method. In contrast, the filter FS methods do not consider any learning algorithm in the evaluation process and depend on the relations between features themselves.

The second aspect that needs to consider when designing a FS method is the subset generation mechanism. Since the datasets used in this paper are high dimensional datasets, the complete search strategy (that generates all possible feature subsets to select the best one) becomes impractical. In this work, we adopted the heuristic search strategy as it proved its ability to tackle different optimization problems in different fields [37]–[39].

WOA is a recent meta-heuristic algorithm, was proposed by Mirjalili and Liwes [40], that mimics the intelligent food foraging behavior of the humpback whales in the ocean. Even though the simple structure, ease of implementation, and the good performance of the WOA have proven through the literature in many fields, it has shortcomings when applied to combinatorial optimization problems [41]. To enrich the ability of WOA in exploitation tendency, [42] proposed an enhanced associate learning-based WOA, in which the β hill-climbing algorithm was utilized to exploit the local solutions. Besides, a modified version of WOA was introduced in [43]. The authors implemented the levy flight operator into WOA to prevent early stagnation. Also, the elite opposition based learning and information gain were adopted in WOA to enhance its search ability for FS problems [44]. Previous works propounded that different strategies can be employed for improving the WOA's performance. Moreover, No Free Lunch theorem [45] suggests that no universal optimizer can excellently solve all the problems, which motivates our attempts to propose a new variant of WOA as a wrapper FS method for software fault prediction.

This paper presents an efficient wrapper FS approach that is based on the WOA optimizer and enhanced with nature selection operators to improve the efficacy of the basic WOA in dealing with FS tasks. The key contributions in this paper are summarized as follows:

- Five natural selection schemes augmented with crossover operator are integrated with WOA to enhance the guided solution while performing the exploitation process.
- Seventeen challenging software fault projects are employed to confirm the effectiveness of the proposed approach.

- The proposed approach showed a significant improvement compared to state-of-the-art approaches.

The rest of this paper is organized as follows: Section II presents the related works for SFP and FS. Section III presents the proposed enhancement of WOA based on natural selection methods. Section IV presents a description of the datasets were used in this paper. Section V reports the obtained results and its analysis. Finally, Section VI presents the conclusion and the future works of this paper.

II. REVIEW OF RELATED WORKS

A. SOFTWARE FAULT PREDICTION

The literature reveals that ML algorithms can effectively tackle the SFP problem where several techniques have been proposed for detecting faults in software modules. Examples of ML-based SFP approaches include LRM [46], FC [47], CR [48], DT [49], NB [16], [50], ANN [51], RF [50], BN [52] and SVM [17], [53]. Moreover, for evaluating SFP techniques, various publicly available datasets are used. The most commonly applied public datasets by researchers in the area of SFP are NASA datasets, PROMISE repository, Bug prediction dataset, and Qualitas corpus [54]. For instance, Catal and Diri [55] investigated the effect of three factors comprising dataset size, feature selection, and software matrices on software fault proneness detection. Besides, various machine learning classifiers such as NB, RF, DT, and AIRS were applied. The authors also used five NASA datasets to examine the classifiers. As per the results, the values of the evaluation parameter AUC show that the RF classifier was superior for large datasets, whereas Naive Bayes was the appropriate predictor for small datasets. Moreover, Singh and Malhotra [56] studied the relationship between software design that is developed based on the object-oriented concept and fault proneness. The authors examined the performance of SVM over a public dataset obtained NASA repository (KC1). The obtained results show that the SVM classifier is able to discover the faulty classes in OO based systems.

Rathore and Kumar [57] employed a set of ensemble learning methods to predict software faults. The authors applied two methods that are LRCR and GBRCR, as an ensemble output for GP, MLP, and LRM algorithms. The proposed approach has been examined over eleven public datasets obtained from the PROMISE data repository. The obtained results show that ensemble techniques can outperform other traditional techniques in predicting software faults.

As aforementioned, FS has become an essential step in data mining in general and machine learning in particular since it plays an important role in cleaning data from noisy, irrelevant, and redundant features. This step will enhance the overall quality of data and reduce its dimensionality. It has been approved in many studies that removing such features can significantly enhance the overall performance of machine learning classifiers [58], [59]. Many researchers in the field of SFP investigated the effect of applying different filters and wrappers FS approaches on the performance of ML-based SFP models. For example, using some public NASA datasets, Catal and Diri [55] applied the correlation-based FS method to remove the highly relevant matrices for the SFP problem. Balogun *et al.* [60] studied the performance of eighteen FS methods (i.e., four feature ranking and fourteen feature subset selection) to detect SDP. The authors applied those methods on five public datasets obtained from the NASA repository. The obtained results show that FS methods can enhance the performance of ML methods for SDP problems.

Dhamayanthi *et al.* [61] made use of the well-known statistical tool Principle Component Analysis as a feature reduction technique for solving the SFP problem. Using the extracted features by PCA, and the NB classifier was applied over seven projects from NASA Metrics Data Program. It was observed that the prediction accuracy of the NB classifier was increased when using the set of features selected through PCA.

Wrapper feature selection approaches were also applied in the SFP domain. Wahono *et al.* [62] proposed a combination of GA and bagging technique for enhancing the performance of the software defect prediction. GA was used to tackle the feature selection, while the bagging technique was used to solve the class imbalance problem. Several machine learning classifiers were applied over nine datasets from NASA metric data repository in order to assess the proposed method. Results of the AUC indicated that the proposed method improved the prediction performance of the most applied ML classifiers. Wahono conducted another similar work in [63], where GA and PSO algorithms were applied as feature selection techniques for the SFP problem, and bagging technique was utilized for dealing with class imbalance problem. Ten classifiers were applied over nine NASA MDP datasets to evaluate these FS approaches. Results of AUC confirmed that the proposed FS approaches yielded significant enhancement in prediction performance for most of the applied classifiers. A hybrid feature selection technique hybridizing PSO and MGA was introduced by Banga *et al.* [64] for improving SFP. Furthermore, bagging was also integrated with this approach for resolving the class imbalance problem. Empirical results using NASA MDP dataset and a pool of machine learning algorithms (i.e., KNN, SVM, least-squares twin SVMs, mean weighted least squares twin SVMs, and RF) showed that applying the proposed hybrid FS approach improves the efficiency of SFP modules in classifying software modules into defective or not defective modules.

Recently, Turabieh *et al.* [3] applied a L-RNN classifier with wrapper FS methods (i.e., BGA, BPSO, and BACO) to predict faults in software modules. The proposed approach employed an iterative method in order to remove the redundant features. The authors applied their approach over 19 public datasets obtained from PROMISE data repository. The obtained results show the ability of wapper FS of enhancing the performance of the L-RNN classifier. Thaher *et al.* [65] introduced a wrapper-based feature selection method based on BQSA for solving the SFP problem.

SMOTE was applied for re-balancing fourteen real-world datasets obtained from the PROMISE repository. in terms of AUC results, Binary QSA integrated with SMOTE technique yielded the best results compared with different FS algorithms including BBA, BWOA, BGSA, BALO and KNN classifier without feature selection. Tumar *et al.* [4] developed a wrapper FS method to predict faults in software modules using a modified version of BMFO named Enhanced BMFO (EBMFO) in conjunction with ADASYN for solving class imbalance issue. The proposed method was evaluated using a set of fifteen real projects data from the PROMISE repository, and three machine learning classifiers comprising LDA, KNN, and DT. Recorded results confirm that the proposed FS method improved the overall performance of the three ML classifiers. In [66], Thaher *et al.* proposed an enhanced binary version of the new meta-heuristic algorithm, the HHO, to search for ideal features subsets for SFP problem. In addition, the ADASYN oversampling technique was used for re-balancing the datasets. Fifteen SFP datasets from PROMISE, the same ML classifiers used by [4], and a similar set of optimization algorithms used in [65] were employed to assess the proposed approaches. The obtained results confirmed the superiority of the enhanced HHO approach when used with the ADASYN oversampling method and Linear Discriminant Analysis classifier compared to other optimization algorithms.

It is clear from the reviewed literature that machine learning-based SFP modules are highly influenced by the quality of matrices (features) and the imbalanced data. It can be observed from the previously published works that building efficient SFP modules are possible when using sophisticated FS algorithms for discarding trivial and irrelevant features and re-balancing the data to avoid the problem of class imbalance in SFP datasets. These facts motivated the authors of this work to propose an efficient feature selection approach for the SFP problem. The efficiency that the Whale Optimization algorithm has previously shown in selecting the optimal sets of features from simpler datasets (Comparing to SFP datasets) nominated it in this work to be used for dealing with feature selection in SFP domain [67], [68].

B. FEATURE SELECTION

Feature selection (FS) is treated as a binary optimization problem that aims to explore the search space to find the ideal subset of features. A subset of features can be viewed as a binary vector where each cell in the vector represents one of the attributes (features) in the dataset. The length of the vector equal to the number of features in the dataset. If a cell of the vector has a value of one, then it is retained, and if its value equals zero, it will be ignored. FS eliminates uninformative features that negatively impact the classification process while keeping the most informative ones [26], [58], [59]. Generally, depending on the search mechanism, FS follows two main wide branches: filter and wrapper. Filter FS methods usually use a statistical measure (e.g., chi-squared test, information gain, correlation coefficient score, and so forth) to calculate and assign a weight to each feature. Features are ranked by their weights, and only the features with their weights greater than a pre-specified threshold are retained for data representation, while the rest of the features are eliminated from the feature space [58], [59]. Unlike filter methods, wrapper FS typically starts a search procedure (using a search strategy) by generating a set of potential solutions (subsets of features) called population, and employs a machine learning classifier to evaluate the generated feature subsets to determine the best one.

In practice, the complete search is infeasible when the number of features is huge. Compared to complete and random approaches, Meta-heuristic algorithms have demonstrated their superiority in finding the best or near the best solutions for many simple and complex feature selection problems [69]. Based on the number of solutions (subsets of features) to be generated and assessed in each generation, meta-heuristic algorithms are classified into two types: population-based and single-based algorithms. The most popular single solution based algorithms are TSA [70] and SA [71] while the famous population-based meta-heuristic algorithms comprise GA [72], [73] and PSO [74], [75], ABC [69], ACO [76] and BBA [77]. In addition, in the last few years, many population-based meta-heuristic algorithms have been proposed and applied for FS problem (e.,g., AOA [78], GWO [59], MFO [4], [79], DOA [24], and WOA [67], [68]).

III. PROPOSED METHOD

A. WHALE OPTIMIZATION ALGORITHM

One of the largest mammals in the world that lives in groups is the whales. In deep oceans, there are seven kinds of whales names as humpback, sei, right, black whales, killer, minke, and finback. Humpback whales are the largest whale kind, which has a brilliant strategy for hunting the prays such as small fishes, seals, squid, and krill [80]. Humpback whales use a search strategy called bubble-net feeding while searching for their food, where a set of bubbles are created in an upward spiral swimming path around the food source. In nature, whales start generating bubbles-net to determine the prey's location, and then the whales start to move the target in a spiral shape before the attack. Figure 1 demonstrates the hunting process for humpback whales. Mirjalili and Lewis [36] in 2016 proposed WOA that simulates the process of humpback whales in the hunting stage. WOA is a population-based algorithm that mimics a group of whales (each whale represents a solution) when moving toward a pray location that represents the optimal location. The whales swim in a helical route that is generated by blowing a net of bubbles.

Since WOA is a population-based algorithm, the first step of WAO is to generate the initial population (i.e., group of whales). The following procedure demonstrates the process of generating the initial population.

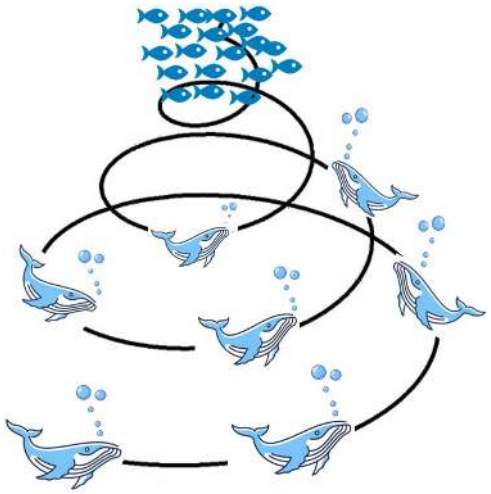


FIGURE 1. Bubble-net feeding method for whale.

procedure Generate initial population(LB, UB,npop,n)
LB=[LB_1, LB_2,..., LB_n]
UB=[UB_1, UB_2,..., UB_n]
for i=1:npop **do**
for j=1:n **do**
initial population(i,j)=$(UB(j) - LB(j)) \times$ rand + $LB(j)$
end for
end for
end procedure

where LB and UB present the lower and upper bounds of the decision variables, $npop$ presents the population size (number of whales), and n presents the number of decision variables.

The second step of WOA is to evaluate the population with respect to the fitness function. This step will determine the best solution obtained so far. To simplify the coding process of WOA, each solution is presented as a vector of the size n (i.e., number of variables). All solutions except the best one will update their locations toward the best solution using Eqs.(1) and (2).

$$D = \mid \vec{C}.\vec{X}^*(t) - \vec{X}(t) \mid \tag{1}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A}.D \tag{2}$$

where t refers to the current iteration, $\vec{X}^*$ refers to the best solution so far at iteration t. $\vec{A}$ and $\vec{C}$ present specific coefficients calculated using Eqs.(3) and (4), respectively.

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{3}$$

$$\vec{C} = 2 \cdot \vec{r} \tag{4}$$

where the variable $\vec{a}$ that is initially equals 2 and linearly decrease toward 0 after a number of iterations, and the variable $\vec{r}$ is a random between 0 and 1, that is generated using a uniform distribution function. Eqs.(1) and (2) enable the WOA to search in n-dimensional solution space in an outstanding performance as shown in Figure 3 for 2 and 3 dimensions.

Two mathematical models have been proposed by Mirjalili and Lewis [40] to simulate the hunting process itself: The shrinking encircling mechanism and spiral updating position. Shrinking encircling mechanism is used to update the whale position with respect to the best solution so far. This process happens by decreasing the value of variable $\vec{a}$ over the iterations in a linear manner, as shown in Figure 3(a). While spiral updating position demonstrates the swimming style (i.e., upward spiral path) of whales to reach their target. A logarithmic spiral function is used to mimic this swimming style, as shown in Eq.(5).

$$\vec{X}^*_{(t+1)} = \overrightarrow{|X^*_{(t)} - X_{(t)}|}.e^{bt}.cos(2\pi I) + \vec{X}^*_{(t)} \tag{5}$$

The shape of the spiral function is created by the parameter b, and I is a random number between -1 and 1. Figure 3(b) shows the spiral swimming style while hunting for whales.

Whales employed shrinking encircling and spiral swimming methods with a probability of 50% for each one. The location of the best solution obtained so far determines which operation to be executed. In WOA, a random probability p is generated to determine operation selection, as shown in Eq.(6).

$$\vec{X}_{t+1} = \begin{cases} \vec{X}^*_{(t)} - \vec{A}.\vec{D}, & p < 0.5 \\ \vec{X}^*_{t+1} = \vec{D}.e^{bt}.cos(2\pi I) + \vec{X}^*_{(t)}, & p \geq 0.5 \end{cases} \tag{6}$$

The exploration process in WOA is performed once each whale updates its location based on a randomly chosen whale. In this case, the next position of the whale will be between its current location and the location of the selected whale. This behavior occurs when the variable (A) between -1 and 1. In contrast, the exploitation process is performed when each whale updates its location with respect to the best whale (solution). This situation happens when the variable A is greater than 1. Figure 4 shows the exploration and exploitation process inside WOA. Eqs.(7) and (8) demonstrate the exploration process of WOA. Finally, the pseudo-code for WOA is presented in Algorithm 1.

$$\vec{D} = |\vec{C}.\vec{X}_{rand} - \vec{X}| \tag{7}$$

$$\vec{X}_{(t+1)} = \vec{X}_{rand} - \vec{A}.\vec{D} \tag{8}$$

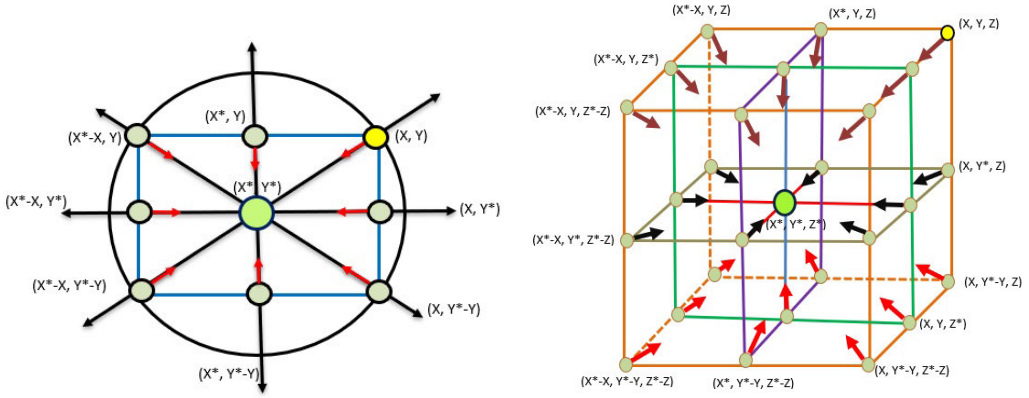
B. BINARY WOA

Adapting the WOA algorithm to deal with binary optimization problems requires employing so-called binarization rules. In this regard, different mathematical TFs along with binarization rules have been introduced to convert real input values into binary [81]. In this work, the S-shaped TF as in Eq. (9) is incorporated with the updating rule in Eq. (10) to present a BWOA.

$$T(x_j) = \frac{1}{1 + e^{-x_j}} \tag{9}$$

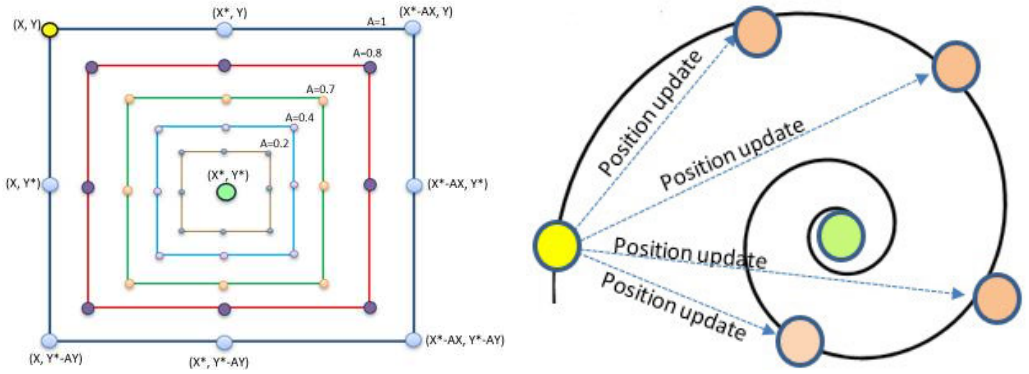
where x_j represents the real value of j^{th} dimension, and $T(x_j)$ is the probability of x_j to be 1.

$$S_j = \begin{cases} 1 & r < T(x_j) \\ 0 & Otherwise \end{cases} \tag{10}$$



(a) 2D position. (b) 3D position.

FIGURE 2. Possible 2D and 3D locations of whales nearby the prey.



(a) Shrinking encircling. (b) Spiral updating.

FIGURE 3. Shrinking encircling and Spiral updating methods.

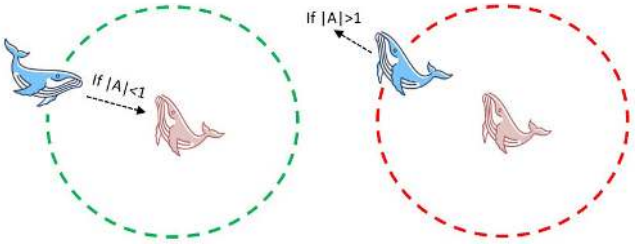


FIGURE 4. Updating whale position either toward or moving away from a randomly chosen humpback whale.

where r is a random number within [0,1] and S_j is the binary output.

In BWOA, each solution S is presented as a binary vector, where $S = (S_1, S_2, \ldots, S_n)$, $S_i \in \{0, 1\}$. BWOA can be employed to solve binary optimization problems such as FS problems. In FS, two contradictory objectives where highest classification rate and least reduction rate of selected features are considered. These two criteria are combined in a single formula using Eq. 11 [4].

$$Fitness = 0.99 \times (1 - AUC) + 0.01 \times \frac{N_F}{D} \qquad (11)$$

where N_F refers to the number of selected features, and D refers to the number of total features.

C. NATURAL SELECTION METHODS

The selection scheme for any searching algorithm is a critical component that provides a good ratio between intensification (i.e., selecting better solutions) and diversification (i.e., selecting random solutions). Many common selection schemes can be employed to maintain a better trade-off between intensification and diversification. However, only five methods are employed in this work: Linear rank-based [82], Proportional (Roulette wheel) based [83], Random-based [84], [85], Stochastic universal sampling [86], and Tournament based [87]. These selection methods are commonly used in the population-based algorithm, and the default setting of these selection methods are used [84].

1) PROPORTIONAL (ROULETTE WHEEL) BASED

PB selection is originally proposed for genetic algorithm [21]. PB is one of the most common selection methods due to its simplicity and ease of implementation. In simple, for any minimization problem, the PB starts by calculating the

Algorithm 1 Pseudo-Code of WOA

```
A population is randomly generated
All coefficients are initialized
All solutions in the population are evaluated using fitness function
The best solution so far (denoted as X*) in determined
while (t < maximum nymber of iterations) do
    for each solution (i.e., whale) do
        Update all coefficients (i.e., a, A, C, l, and p)
        if (p < 0.5) then
            if |A| < 1 then
                Update the position of the current solution by Eq.(2).
            else if |A| > 1 then
                Select a random solution from a population
                Update the position of X(t) by Eq.(8)
            end if
        else if (p > 0.5) then
            Update the position of X(t) by Eq.(6)
        end if
    end for
    Evaluate the fitness value for each solution.
    Update X*
    t = t + 1
end while
```

selection probability of each solution (p_j) in the population pool with respect to its absolute fitness value for each solution divided by the summation of the fitness value for all solutions, as shown in Equation 12.

$$p_j = \frac{|fitt(x_j|}{|\sum_{i=1}^{N} fitt(x_i)|} \tag{12}$$

where N represents the population size, $fitt(x_j)$ is the fitness of the j^{th} solution that is calculated as in Eq.(13):

$$fitt(x_j) = \frac{1}{1 + fitt(x_j)} \tag{13}$$

The PB will select the fittest solution with a higher probability to perform the diversification step. Algorithm 2 demonstrates the pseudo-code for BP selection using roulette wheel structure. Where the final value of S represents accumulative selection probabilities for all solutions in the population pool.

2) LINEAR RANK-BASED

LR based selection is proposed to overcome all the weak points for BP selection methods [82]. The basic idea of LR is to use a real ranking system for each solution based on its fitness value. Then each solution will gain a selection probability based on a linear mapping function, as shown in Equation 14.

$$p_j = \frac{1}{N} \times (\eta^+ - (\eta^+ - \eta^- \times \frac{j-1}{N-1})) \tag{14}$$

where j refers to the solution rank, η^+ refers to the expected value of the best solution in the population

Algorithm 2 Pseudo-Code of Proportional Based

```
Set r ∈ U(0, 1)
Set S = 0
Set i = 0
while i ≤ N do
    S = S + P_i
    if S ≥ r then
        best=i
        break;
    end if
    i = i + 1
end while
return(best)
```

(i.e., $\eta^+ = N \times P_1$), and η^- refers to the expected value for the worst solution (i.e., $\eta^- = N \times P_N$). Where P_1 presents the probability of the best solution, while P_N presents the probability of the worst solution. The slope of the linear function is determined based on η^+ and η^-. The selective pressure of LR depends on η^+ value; a higher one means a higher selective pressure [82].

3) STOCHASTIC UNIVERSAL SAMPLING

SUS selection is a modified version of the proportional selection method proposed in 1987 [86]. The main idea of SUS is to find a selection probability for each solution with respect to its fitness value related to the total fitness values inside the current population pool. In simple, roulette wheel N times to select N parents, while SUS spins the wheel once to select N parents. The main weakness of SUS that once the population is converged, the SUS will not work in a good manner.

4) TOURNAMENT BASED

One of the most well-known selection methods in the evolutionary and swarm algorithm is TS. In this work, we combined TS with BWOA. In simple, the TS starts by selecting a set of solutions Φ of size t from the population, where t is less than population size (N). Then determine the best solution from Φ based on the fitness function. A selection probability for each solution in Φ is evaluated based on Equation 15.

$$p_j = \frac{1}{N^t}[(N - j + 1)^t - (N - j)^t] \tag{15}$$

The main factor that plays a vital role in selection pressure is the tournament size (t). Normally, a higher value of t is used for complex and ragged search space, which increases the selection pressure, while a lower value of t will reduce the selection pressure and direct the search space toward diversification. In this paper, several preliminary experiments were employed, and the obtained results indicate that t=0.3 of population size provided the best performance.

5) RANDOM-BASED

The RB selection method selects a solution randomly from the population pool. In this method, all solutions have the

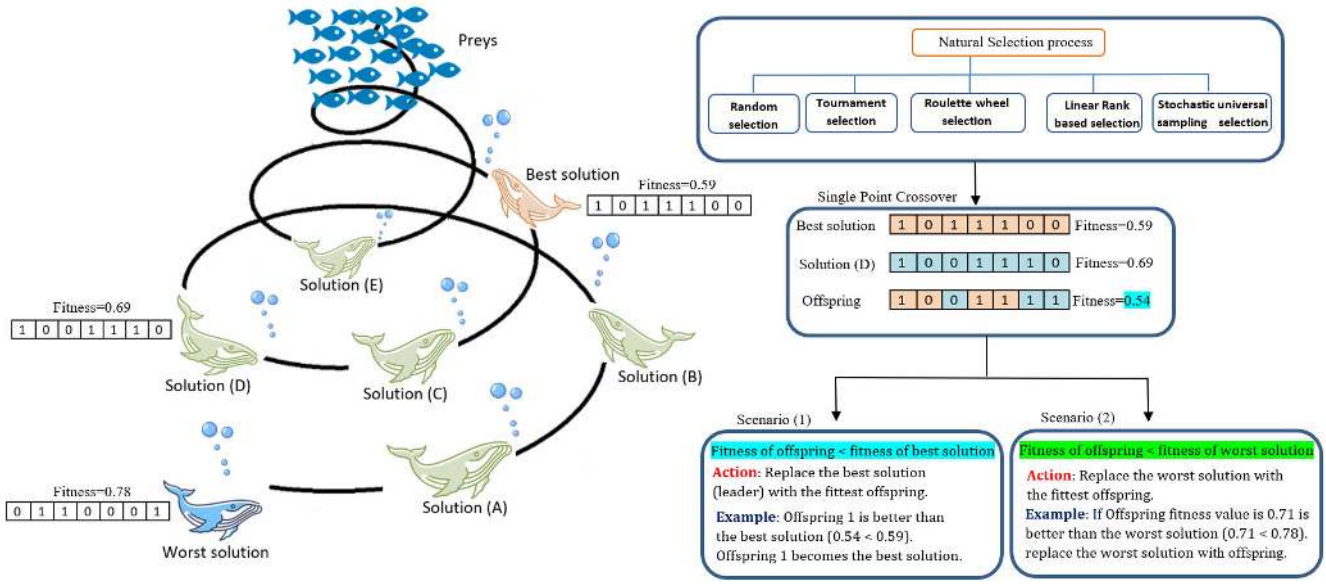


FIGURE 5. Pictorial diagram for the proposed enhancement.

same probability regardless of their fitness value. Therefore, the selection pressure is unified. Unlike the aforementioned methods, this approach may lead to slow convergence.

D. ENHANCED WHALE OPTIMIZATION ALGORITHM

In this work, an enhanced version of WOA is proposed to overcome the weakness of the basic WOA while performing the exploitation process. The exploitation process occurs once a solution moves toward the best solution. The exploitation process is meaningless if the best solution is trapped in local optima, and no chance to enhance the best solution. To overcome this problem, we combine the WOA with different types of natural selection methods along with a single point crossover method. Figure 5 demonstrates the proposed method at iteration t. In this figure, the best solution has a fitness value equals to 0.59, while the worst solution has a fitness value equals to 0.78. There are five different solutions (i.e., A, B, C, D, and E) that follow the best solution. The proposed enhancement starts by selecting a solution randomly from the population, except the best and worst solutions. Five different natural selection methods is employed: Stochastic universal sampling selection, Random selection, Tournament selection, Roulette wheel selection, and Linear Rank-based selection. A single point crossover method is employed between the best solution so far and the selected solution (i.e., solution (D)). At this step, a new solution will be generated (i.e., offspring). There are three scenarios to handle the newly generated solutions as follows:

- Scenario (1): If the fitness value of the new solution is better than the fitness value of the best solution, the fitness new solution will become the best solution.
- Scenario (2): If the fitness value of the new solution is better than the fitness value of the worst solution, the fitness new solution will replace the worst solution.
- Scenario (3): If the fitness value of the new solution is better than the fitness value of the worst solution, the new solution will be discarded.

Algorithm 3 explores the pseudo-code of the proposed enhancement. Meanwhile, the flowchart of the proposed approach is shown in Figure 6.

TABLE 1. Description of PROMISE datasets.

Dataset	version	#instances	#defective instances	%defective instances
ant	1.7	745	166	0.223
camel	1.2	608	216	0.355
camel	1.4	872	145	0.166
camel	1.6	965	188	0.195
jedit	3.2	272	90	0.331
jedit	4.0	306	75	0.245
jedit	4.1	312	79	0.253
jedit	4.2	367	48	0.131
log4j	1.0	135	34	0.252
log4j	1.1	109	37	0.339
log4j	1.2	205	189	0.922
lucene	2.0	195	91	0.467
lucene	2.2	247	144	0.583
lucene	2.4	340	203	0.597
xalan	2.4	723	110	0.152
xalan	2.5	803	387	0.482
xalan	2.6	885	411	0.464

IV. DESCRIPTION OF SFP DATASETS

In this research, 17 PROMISE datasets are utilized to assess the performance of the proposed algorithms [88], [89]. Table 1 provides the details of utilized PROMISE datasets. Each dataset contains 20 features as shown in Table 2.

Algorithm 3 Pseudo-Code of Enhanced WOA

A population is randomly generated.
All coefficients are initialized.
All solutions in the population are evaluated using fitness function.
The best solution so far (denoted as X^*) in determined.
while ($t <$ maximum number of iterations) **do**
  Identify the worst solution (X^{worst})
  Apply selection scheme to select another solution ($X^{selected}$)
  generate new solution ($X^{crossed}$) using crossover between X^* and $X^{selected}$
  if ($X^{crossed}$ is superior to X^*) **then**
    use $X^{crossed}$ as a leader
  else if ($X^{crossed}$ is superior to X^{worst}) **then**
    replace X^{worst} with $X^{crossed}$
  end if
  for each solution (i.e., whale) **do**
    Update all coefficients (i.e., a, A, C, l, and p)
    if ($p < 0.5$) **then**
      if $|A| < 1$ **then**
        Update the position of the current solution by Eq.(2).
      else if $|A| > 1$ **then**
        Select a random solution from a population
        Update the position of $X(t)$ by Eq.(8)
      end if
    else if ($p > 0.5$) **then**
      Update the position of $X(t)$ by Eq.(6)
    end if
  end for
  Evaluate the fitness value for each solution.
  Update X^*
  $t = t + 1$
end while

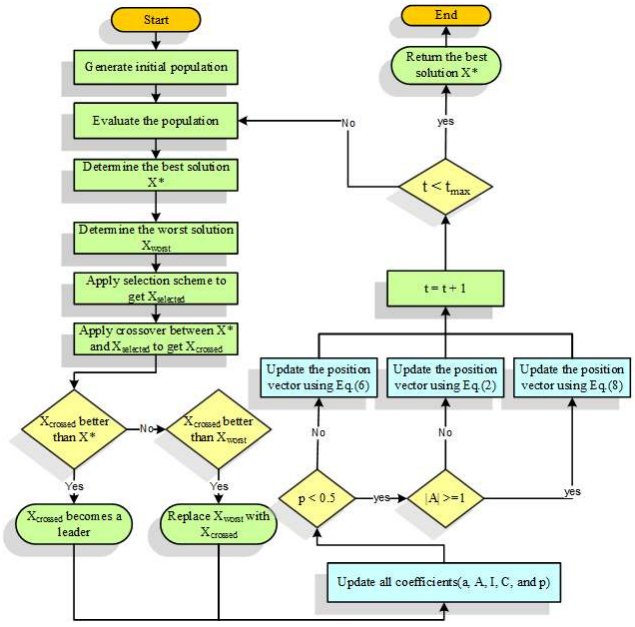


FIGURE 6. Flowchart of the enhanced WOA.

V. EXPERIMENTAL RESULTS AND SIMULATIONS

A. EXPERIMENTAL SETUP

This sub-section briefly describes the setup and parameter settings of the proposed algorithms. There are two common parameters: population size and the number of iterations that have to be set for all wrapper methods. To ensure a fair environment, we set the population size and the number of iterations at 10 and 100. Moreover, we set the parameters of LRBWOA (η^+) and TBWOA (T) to 1.1 and 3, respectively. The sensitivity analysis of these parameters is presented in the next sub-section. In this study, six state-of-the-art methods include BGWO [90], BGSA [91], BPSO [92], BALO [93], BBA [20], and BSSA [94], are used to verify the efficacy of the proposed algorithm. The detailed parameter settings of the algorithms are tabulated in Table 3.

Many classification methods can be employed when dealing with FS problems. In this article, four popular classifiers, namely, SVM, DT, LDA, and kNN are adopted to evaluate the AUC performance. These classifiers are commonly used in the area of SE [59], [95]. Table 4 presents the AUC results of different classification algorithms. Judging from Table 4, the highest AUC values were mostly perceived by kNN and DT classifiers (6 datasets). The results of mean rank affirm the supremacy of DT and kNN in SFP. Hence, in the analysis of the proposed variants, only two methods are used: kNN and DT classifiers. For performance validation, both classifiers are trained and tested using the hold-out method, 80% for training and 20% for testing.

The performances of the BWOA with natural selection methods are investigated through the validation method. All experiments in this paper have been implemented using MATLAB 2018. Additionally, all selection methods are evaluated using the same criteria (i.e., AUC value) and the same hardware infrastructure (CPU, memory, operating system, etc.) for a fair comparison.

B. PRELIMINARY EXPERIMENTS

A set of preliminary experiments have been performed to find the best BWOA settings. Initially, different combinations of common parameters (e.g., number of iterations [50, 100, and 150], and population size [10, 20, 30, and 40]) are examined on the overall performance of BWOA. Its worth mentioning that these values are the most used values in the main FS papers in the highly ranked journals [20], [90], [96]. For the sake of simplicity, the internal classifier at this step is kNN with K=5 [20], [21], [90]. Table 5 presents the initial results of BWOA with a different number of iterations and population size. The findings show that the best AUC performance was achieved when the population size equal to 10 with 100 iterations, where the algorithm achieved the best results in 12 out of 17 datasets (i.e., 71% of the datasets).

TABLE 2. Description of object-oriented metrics.

Metrics		Descriotion
f1	wmc	Number of methods defined in a class.
f2	dit	Depth of a class within the class hierarchy from the root of inheritance.
f3	noc	Number of immediate descendants of a class.
f4	cbo	Count the number of classes coupled to class.
f5	rfc	Count the number of distinct methods invoked by a class in response to a received message.
f6	lcom	Count the number of methods that do not share a field to the method pairs that do.
f7	ca	Count the number of dependent classes for a given class.
f8	ce	Count the number of classes on which a class depends.
f9	npm	Number of public methods defined in a class.
f10	lcom3	Count the number of connected components in a method graph.
f11	loc	Count the total number of lines of code of a class.
f12	dam	Computes the ratio of private attributes in a class.
f13	moa	Count the number of data members declared as class type.
f14	mfa	Shows the fraction of the methods inherited by a class to the methods accessible by the functions defined in the class.
f15	cam	Computes the cohesion among methods of a class based on the parameters list.
f16	ic	Count the number of coupled ancestor classes of a class.
f17	cbm	Count the number of new or redefined methods that are coupled with the inherited methods.
f18	amc	Measures the average method size for each class.
f19	max_cc	Maximum counts of the number of logically independent paths in a method.
f20	avg_cc	Average counts of the number of logically independent paths in a method.

TABLE 3. The parameter settings of comparative algorithms.

common parameters		
Number of runs		30
population size		10
Number of iterations		100
Dimension		#features
Internal parameters		
Algorithm	parameter	value
BWOA	convergence constant a	[2 0]
	Spiral factor b	1
LRBWOA	η^+	1.1
TBWOA	T	3
BGSA	initial gravitational constant G_0	10
	Rpower	1
	Alpha	20
BBA	Q_{min} Frequency minimum	0
	Q_{max} Frequency maximum	2
	A loudness	0.5
	r Pulse rate	0.5
BGWO	convergence constant a	[2 0]
BPSO	Inertai weight w	[0.9 0.2]
	cognitive constant c1	2
	social constant c2	2
BALO	w	2,3,4,5,6
BSSA	convergence constant c1	[2 0]

TABLE 4. AUC results of different classification algorithms.

Benchmark	KNN	DT	LDA	SVM
ant-1.7	**0.728**	0.682	0.660	0.644
camel-1.2	0.535	**0.593**	0.549	0.531
camel-1.4	0.553	**0.560**	0.490	0.497
camel-1.6	0.535	**0.546**	0.536	0.497
jedit-3.2	0.676	0.604	0.699	**0.731**
jedit-4.0	0.593	0.688	0.670	**0.690**
jedit-4.1	**0.740**	0.617	0.719	0.658
jedit-4.2	0.600	**0.719**	0.592	0.592
log4j-1.0	0.536	**0.734**	0.609	0.661
log4j-1.1	0.714	0.732	0.777	**0.813**
log4j-1.2	**0.737**	0.712	0.500	0.500
lucene-2.0	**0.638**	0.589	0.487	0.616
lucene-2.2	0.567	0.606	0.529	**0.668**
lucene-2.4	0.647	0.706	0.750	**0.765**
xalan-2.4	**0.685**	0.615	0.580	0.576
xalan-2.5	**0.638**	0.614	0.621	0.634
xalan-2.6	0.698	**0.748**	0.719	0.738
Mean Rank	2.47	2.18	2.88	2.47

While the worst performances were achieved when the population size equals to 40, 20, and 30, at 50, 100, and 100 iterations, respectively. So, it is important to carefully tune parameters for the population-based algorithm. The result of F-test from Table 5 supports the clarification.

Table 6 shows the performance of LRBWOA using kNN classifier. Here, we examine different numbers of η^+ (i.e., 1.1, 1.3, 1.5, 1.7, 1.9). Based on the result obtained, the LRBWOA with η^+ equals 1.1 outperformed other settings on average AUC values and F-Test analysis. On the one hand, the worst performance was perceived when η^+ equals 1.7 with f-test score equals 3.47. At this step, we conclude that excellent tuning of the parameter η^+ will control the selection pressure that can best fit with the solution distribution in the search space.

In the next stage, we study the performance of tournament selection over all the datasets. We simulate the performance of BWOA with tournament selection using different tournament size t (i.e., 0.3, 0.5, 0.7, 0.9). Table 7 shows the obtained results based on average AUC and standard deviations. From the analysis, it is clear that the tournament size played a vital role in the performance of BWOA. For example, the performance of BWOA at $t = 0.3$ or $t = 0.7$ outperform other sizes. Moreover, increase the tournament size (i.e., $t = 0.9$) will reduce the performance of BWOA. The results suggest that the analysis of tournament size was an essential step before the evaluation process.

C. RESULTS OF DIFFERENT BWOA VARIANTS

This subsection presents a comparative study of BWOA variants using DT and KNN classifiers.

TABLE 5. Average AUC results of BWOA for different combinations of population sizes and number of iterations.

#iterations	50				100				150			
population size	10	20	30	40	10	20	30	40	10	20	30	40
ant-1.7	0.684	0.635	**0.701**	0.679	**0.786**	0.660	0.694	0.626	0.668	0.634	0.665	**0.677**
camel-1.2	0.582	0.555	**0.599**	0.566	**0.634**	0.590	0.577	0.597	0.538	0.519	0.524	**0.598**
camel-1.4	**0.625**	0.590	0.570	0.609	0.616	0.593	0.571	**0.621**	0.566	0.552	0.534	**0.641**
camel-1.6	0.584	0.581	**0.589**	0.588	0.604	**0.610**	0.584	0.578	0.574	**0.628**	0.582	0.597
jedit-3.2	0.637	**0.742**	0.636	0.627	**0.828**	0.687	0.660	0.728	0.714	0.691	**0.745**	0.712
jedit-4.0	**0.705**	0.684	0.676	0.627	**0.743**	0.628	0.634	0.721	**0.678**	0.635	0.654	0.676
jedit-4.1	0.654	0.629	**0.728**	0.637	**0.740**	0.624	0.655	0.669	**0.739**	0.652	0.685	0.697
jedit-4.2	**0.633**	0.610	0.622	0.593	**0.661**	0.595	0.596	0.617	0.623	**0.703**	0.617	0.579
log4j-1.0	0.700	0.598	**0.723**	0.653	**0.845**	0.664	0.742	0.582	0.647	0.693	**0.737**	0.685
log4j-1.1	0.678	**0.855**	0.795	0.719	0.800	0.685	0.803	**0.814**	0.612	**0.830**	0.743	0.728
log4j-1.2	0.576	0.464	**0.739**	0.625	0.750	0.617	0.518	**0.838**	0.646	0.637	0.635	**0.855**
lucene-2.0	0.625	0.631	0.644	**0.702**	**0.803**	0.732	0.690	0.667	0.556	**0.681**	0.674	0.575
lucene-2.2	0.620	**0.636**	0.541	0.550	**0.737**	0.579	0.531	0.616	0.552	**0.611**	0.579	0.560
lucene-2.4	0.603	0.651	**0.702**	0.631	**0.721**	0.640	0.667	0.663	0.660	**0.698**	0.628	0.643
xalan-2.4	0.599	0.600	**0.639**	0.587	0.616	0.613	**0.691**	0.578	0.580	0.586	**0.603**	0.587
xalan-2.5	**0.662**	0.625	0.640	0.646	**0.715**	0.625	0.624	0.655	0.647	0.661	0.603	**0.670**
xalan-2.6	0.714	0.714	**0.734**	0.705	**0.793**	0.726	0.734	0.712	**0.739**	0.719	0.681	0.732
Ranking (W)	4	3	9	1	12	1	1	3	3	6	3	5
F-Test	6.65	7.53	5.41	8.5	**1.82**	7.47	6.65	6.38	7.53	6.41	7.71	5.94

TABLE 6. Impact of parameter n^+ on the performance of LRBWOA in terms of average AUC results.

Dataset	$n^+ = 1.1$	$n^+ = 1.3$	$n^+ = 1.5$	$n^+ = 1.7$	$n^+ = 1.9$
ant-1.7	**0.7540**	0.7180	0.7122	0.7125	0.7072
camel-1.2	0.6291	0.6159	**0.6564**	0.6461	0.6071
camel-1.4	0.5970	0.5944	0.6204	**0.6258**	0.6097
camel-1.6	**0.5972**	0.5914	0.5884	0.5715	0.5708
jedit-3.2	0.7855	0.7808	0.6327	**0.8412**	0.8108
jedit-4.0	**0.8086**	0.7563	0.7735	0.6683	0.7677
jedit-4.1	0.7485	**0.7626**	0.7334	0.7520	0.7526
jedit-4.2	0.6741	0.6250	0.6634	0.5991	**0.7146**
log4j-1.0	0.7603	0.8021	**0.8028**	0.7389	0.7102
log4j-1.1	0.5843	0.6426	**0.9376**	0.8376	0.9161
log4j-1.2	**0.8912**	0.6634	0.8278	0.7000	0.5000
lucene-2.0	**0.8339**	0.8224	0.8174	0.7828	0.8106
lucene-2.2	0.7489	0.7802	**0.7850**	0.6534	0.7448
lucene-2.4	**0.8163**	0.8076	0.7569	0.7599	0.7810
xalan-2.4	0.6377	0.6296	0.6189	0.7022	**0.7587**
xalan-2.5	0.7223	0.7307	**0.7318**	0.7183	0.7129
xalan-2.6	0.7765	0.7746	**0.8146**	0.7716	0.7791
Ranking (F_test)	**2.47**	3.06	2.65	3.47	3.35

TABLE 7. Impact of tournament size (T) on the performance of BWOA-T in terms of average AUC results.

Dataset	T=0.3	T=0.5	T=0.7	T=0.9
ant-1.7	0.7234	0.6946	**0.7494**	0.7248
camel-1.2	0.6158	**0.6450**	0.6339	0.6130
camel-1.4	0.5682	**0.6009**	0.5961	0.5899
camel-1.6	**0.6244**	0.5704	0.5987	0.5849
jedit-3.2	**0.8485**	0.8156	0.7817	0.7667
jedit-4.0	**0.8079**	0.7021	0.7967	0.7507
jedit-4.1	**0.8346**	0.7296	0.7491	0.7614
jedit-4.2	0.6617	0.6272	**0.6826**	0.6208
log4j-1.0	**0.8503**	0.8167	0.7556	0.8102
log4j-1.1	**0.9833**	0.8500	0.5926	0.6551
log4j-1.2	0.5583	0.5556	**0.8908**	0.6685
lucene-2.0	0.8225	**0.8669**	0.8270	0.8147
lucene-2.2	0.7573	0.7598	0.7497	**0.7800**
lucene-2.4	0.7899	0.7339	**0.8165**	0.8035
xalan-2.4	0.6543	**0.7001**	0.6371	0.6242
xalan-2.5	0.6570	0.7021	0.7215	**0.7304**
xalan-2.6	0.7486	0.7746	0.7736	**0.7765**
Ranking(F-test)	**2.35**	2.59	**2.35**	2.71

1) RESULTS OF BWOA VARIANTS WITH KNN CLASSIFIER

Firstly, we inspect the performance of proposed approaches using the kNN classifier. Table 8 presents the obtained AUC results. Inspecting the result, the performance of TBWOA dominated other versions of BWOA with respect to the F-test value (2.88), while the performance of SBWOA was the worst (4.41). The reason for the low performance of SBWOA is because it preserves a higher selection probability for the solution with better fitness, thereby increasing the chance of the whale being trapped in the local regions.

Furthermore, a Wilcoxon test is performed to determine if the performance of BWOA has a significant difference or not. A threshold value 0.05 is used. If the p-value is less than 0.05, that means there is a significant difference

TABLE 8. Comparison between different variants of BWOA with KNN in terms of Average AUC results.

Dataset	BWOA	LRBWOA	PBWOA	RBWOA	SBWOA	TBWOA
ant-1.7	**0.7862**	0.7362	0.7229	0.7391	0.6556	0.7196
camel-1.2	0.6344	**0.6502**	0.6025	0.6242	0.5936	0.6317
camel-1.4	0.6162	0.6431	0.6055	0.6079	**0.6497**	0.5897
camel-1.6	0.6035	0.5729	0.5895	0.5918	0.6089	**0.6127**
jedit-3.2	0.8285	0.7218	**0.8287**	0.7855	0.7723	0.7549
jedit-4.0	0.7428	0.7329	**0.7606**	0.6782	0.7043	0.7126
jedit-4.1	0.7397	**0.8707**	0.8352	0.7807	0.7478	0.7694
jedit-4.2	0.6610	0.7473	**0.7967**	0.6410	0.6586	0.7272
log4j-1.0	0.8454	0.8431	0.7829	0.8002	0.8198	**0.8595**
log4j-1.1	0.8000	0.8571	0.7833	0.8766	0.8403	**0.9444**
log4j-1.2	0.7500	**0.9444**	0.7500	0.6667	0.5444	0.6792
lucene-2.0	0.8032	0.7531	0.8299	0.7410	0.7881	**0.9137**
lucene-2.2	0.7372	0.7380	0.6934	0.6901	0.6900	**0.7788**
lucene-2.4	0.7210	0.7306	0.7517	0.7863	0.7538	**0.8474**
xalan-2.4	0.6158	0.6197	0.6425	0.6080	0.6308	**0.6506**
xalan-2.5	0.7148	**0.7298**	0.7251	0.7283	0.7019	0.7164
xalan-2.6	0.7927	0.7964	**0.8043**	0.7600	0.7708	0.7923
Ranking(F_test)	3.38	2.94	3.21	4.18	4.41	**2.88**

TABLE 9. P-values obtained from Wilcoxon test for the results of TBWOA and other variants in Table 8 ($p \leq 0.05$ are significant and are bolded).

Dataset	BWOA	LRBWOA	PBWOA	RBWOA	SBWOA
ant-1.7	**2.16E-11**	**2.06E-08**	9.82E-01	**2.84E-09**	**3.27E-11**
camel-1.2	5.68E-01	**8.30E-08**	**7.74E-09**	7.17E-02	**2.69E-11**
camel-1.4	**1.71E-06**	**3.04E-09**	**2.63E-03**	**1.06E-02**	**4.55E-11**
camel-1.6	**5.34E-04**	**4.67E-10**	**2.69E-08**	**7.19E-05**	0.210792
jedit-3.2	**5.03E-11**	**3.30E-06**	**2.19E-11**	**9.23E-09**	**7.02E-05**
jedit-4.0	**0.00674**	8.72E-02	**1.24E-05**	**3.68E-07**	4.45E-01
jedit-4.1	**6.61E-11**	**1.70E-11**	**5.97E-12**	**4.84E-03**	**8.83E-08**
jedit-4.2	**2.43E-12**	**0.000105**	**2.90E-12**	**1.24E-12**	**3.06E-12**
log4j-1.0	**7.97E-08**	**4.58E-04**	**1.12E-12**	**2.69E-11**	**3.96E-08**
log4j-1.1	**1.69E-14**	**1.69E-14**	**2.71E-14**	**4.76E-10**	**8.64E-14**
log4j-1.2	**1.43E-06**	**4.94E-12**	**1.43E-06**	3.38E-01	**4.15E-11**
lucene-2.0	**1.46E-11**	**1.81E-11**	**1.07E-11**	**1.06E-11**	**1.32E-11**
lucene-2.2	**5.48E-06**	**3.96E-08**	**4.39E-11**	**3.83E-11**	**7.53E-11**
lucene-2.4	**2.61E-11**	**2.89E-11**	**4.33E-11**	**5.57E-10**	**3.29E-11**
xalan-2.4	**1.07E-05**	**1.54E-10**	3.16E-01	**3.26E-12**	**7.87E-03**
xalan-2.5	7.62E-01	**2.39E-05**	**1.08E-03**	**6.03E-05**	**6.42E-06**
xalan-2.6	0.911593	**0.045051**	**1.85E-07**	**2.94E-11**	**1.89E-10**

TABLE 10. Comparison between different variants of BWOA with DT classifier in terms of Average AUC results.

Dataset	BWOA	LRBWOA	PBWOA	RBWOA	SBWOA	TBWOA
ant-1.7	0.7069	0.7216	**0.7402**	0.6974	0.7375	0.7369
camel-1.2	0.6591	0.6933	0.6860	0.6595	0.6885	**0.7029**
camel-1.4	0.6493	**0.7207**	0.6083	0.6581	0.6714	0.6864
camel-1.6	0.6426	0.6881	**0.6999**	0.6347	0.6803	0.6767
jedit-3.2	0.7959	**0.8734**	0.8647	0.7942	0.8054	0.7899
jedit-4.0	0.8024	**0.8450**	0.8242	0.8067	0.7625	0.8229
jedit-4.1	0.8173	0.7786	**0.8256**	0.8173	0.6909	0.7245
jedit-4.2	0.7085	**0.8615**	0.7003	0.7107	0.8466	0.8472
log4j-1.0	0.7565	0.8104	0.8500	0.7667	**0.9421**	0.9036
log4j-1.1	0.8908	0.8559	0.8606	0.8920	0.9073	**0.9822**
log4j-1.2	0.8167	0.7598	0.6995	0.8167	0.7875	**1.0000**
lucene-2.0	0.7530	0.7171	0.8353	0.7485	**0.8545**	0.8180
lucene-2.2	0.7212	0.7538	0.7482	0.7236	0.7634	**0.7653**
lucene-2.4	0.7675	0.7807	0.7980	0.7633	0.7609	**0.8108**
xalan-2.4	0.6573	0.6887	0.7023	0.6539	0.7482	**0.7735**
xalan-2.5	0.7260	0.7255	0.7485	0.7261	0.7051	**0.7884**
xalan-2.6	0.7911	0.7997	0.7958	0.7870	**0.8149**	0.7957
Ranking(F_test)	4.53	3.19	3.14	4.47	3.19	**1.47**

TABLE 11. Comparison between different variants of BWOA with DT classifier in terms of Average running time.

Dataset	BWOA	LRBWOA	PBWOA	RBWOA	SBWOA	TBWOA
ant-1.7	**36.22**	42.50	41.65	38.16	41.35	40.90
camel-1.2	**38.63**	41.84	44.01	40.68	43.93	42.80
camel-1.4	**40.32**	43.88	45.85	42.95	44.46	44.56
camel-1.6	**47.77**	48.56	49.38	49.72	49.20	49.08
jedit-3.2	**25.63**	27.51	28.14	26.59	27.33	27.51
jedit-4.0	26.01	**23.64**	28.16	27.21	28.03	27.99
jedit-4.1	**26.72**	27.86	28.58	27.30	27.68	27.69
jedit-4.2	**25.75**	27.61	26.75	27.17	27.19	25.80
log4j-1.0	**21.42**	23.61	23.85	21.91	23.67	24.01
log4j-1.1	**20.93**	22.95	23.21	22.14	23.51	22.85
log4j-1.2	**21.59**	23.63	23.78	22.87	24.34	23.69
lucene-2.0	**24.24**	26.15	26.47	25.03	26.52	26.34
lucene-2.2	**26.08**	28.96	29.04	27.20	28.40	28.32
lucene-2.4	**27.35**	31.15	30.50	28.94	30.73	29.72
xalan-2.4	**35.51**	38.10	38.49	38.30	38.64	38.68
xalan-2.5	**44.30**	49.73	48.84	47.27	50.13	49.26
xalan-2.6	**43.18**	47.90	47.82	45.71	47.10	46.84
Ranking (F_test)	**1.06**	3.97	5	2.47	4.65	3.85

between algorithms. Table 9 presents the Wilcoxon test result (p-value) between TBWOA and other methods. From the reported results, the AUC performance of TBWOA was significantly better than other methods in most cases (p-value <0.05). The findings suggest that the implementation of a natural selection strategy can greatly enhance the performance of BWOA in SFP. That is, the enhanced BWOA constructs a potential solution based on natural selection to exploit the best and worst solution, which can significantly improve the diversity of the population.

2) RESULTS OF BWOA VARIANTS WITH DT CLASSIFIER

Secondly, the performance of BWOA with five different nature selection methods is examined using the DT classifier. From Table 10, the TBWOA outperformed the standard BWOA and all other versions with respect to F-test value (1.47). In a dataset like log4j-1.1 and log4j-1.2, the TBWOA has yielded the highest AUC values of 0.9822 and 1, respectively. On the other side, the standard BWOA has the worst performance with F-test value equal to 4.53. The superiority of BWOA variants can be interpreted by the formidable characteristic of natural selection strategies, and the enrichment that has been promoted in the proposed approaches. Therefore, the BWOA variants can usually overtake the BWOA in FS problems. Furthermore, the computational complexity of BWOA and its variants is tabulated in Table 11. From Table 11, it is seen that the BWOA variants were more time consuming compared to BWOA. The increment in the complexity of the proposed variants is mainly due to the additional computational cost to compute the natural selection and the enhancement process.

Table 12 shows the results of the Wilcoxon test between TBWOA and all other versions. Judging from Table 12, it is obvious that there was a significant difference between

TABLE 12. P-values obtained from Wilcoxon test for the results of TBWOA and other variants in Table 10 ($p \leq 0.05$ are significant and are bolded).

Dataset	BWOA	LRBWOA	PBWOA	RBWOA	SBWOA
ant-1.7	**2.30E-06**	**3.63E-03**	7.50E-01	**9.15E-09**	0.923393
camel-1.2	**4.31E-11**	9.59E-02	**2.95E-04**	**2.90E-11**	**7.57E-03**
camel-1.4	**3.00E-05**	**5.98E-06**	**1.70E-10**	**4.48E-03**	2.11E-01
camel-1.6	**5.89E-06**	**0.041636**	**0.000321**	**4.03E-09**	0.558355
jedit-3.2	5.39E-02	**1.84E-10**	**3.33E-10**	1.10E-01	**1.45E-02**
jedit-4.0	0.128582	**2.75E-02**	6.93E-01	2.69E-01	**2.73E-06**
jedit-4.1	**5.34E-11**	**2.20E-08**	**2.12E-11**	**2.77E-11**	**4.15E-07**
jedit-4.2	**1.36E-11**	**0.033834**	**1.06E-12**	**1.19E-11**	0.96972
log4j-1.0	**9.28E-13**	**6.13E-14**	**1.69E-14**	**9.95E-13**	**9.21E-08**
log4j-1.1	**3.16E-12**	**3.87E-12**	**7.28E-13**	**3.38E-12**	**6.84E-09**
log4j-1.2	**6.12E-14**	**2.57E-13**	**3.80E-13**	**6.12E-14**	**1.59E-13**
lucene-2.0	**4.85E-10**	**1.55E-11**	**7.00E-03**	**3.97E-10**	**5.97E-10**
lucene-2.2	**3.11E-09**	5.15E-02	**1.31E-02**	**7.82E-08**	5.89E-01
lucene-2.4	**9.29E-10**	**9.49E-06**	**2.29E-03**	**1.86E-10**	**9.52E-11**
xalan-2.4	**2.80E-11**	**4.18E-11**	**3.87E-11**	**2.73E-11**	**6.56E-03**
xalan-2.5	**4.01E-11**	**2.98E-11**	**2.94E-11**	**2.98E-11**	**2.94E-11**
xalan-2.6	**0.030223**	**0.00334**	0.923398	**1.15E-05**	**1.28E-10**

TBWOA and other algorithms in most of the datasets. These findings support our claim that natural selection methods have a great influence on the performance of BWOA. With the Tournament based natural selection method, the whale can be able to exploit the promising local region. This creative motion assists the whale to evade the local optimal. Besides, the proposed approach carefully considers the scenarios while handling the newly generated solution, which is beneficial in improving the current best and worst solution.

Figures 7 and 8 show the convergence curves for different versions of BWOA with DT classifiers. From these Figures, one can see that the proposed TBWOA expressed a good convergence speed in some cases (i.e., ant-1.7, camel-1.2, camel-1.4, camel-1.6, jedit-3.2, and jedit-4.0) and an excellent convergence speed in others (i.e., log4j-1.1, log4j-1.2, lucene-2.2, lucene-2.4, xalan-2.4, and xalan-2.5). Among the competitors, the TBWOA can accelerate to reach the global optimum. In an alternative word, the proposed TBWOA was able to track the global minimum quickly for finding the preferable features. In a nutshell, it can be inferred that the convergence behavior of TBWOA is competitive compared to other variants.

From the empirical analysis in Table 8 and 10, the proposed BWOA variants have shown victorious performances. As compared to BWOA, the TBWOA with DT was superior and robust over all datasets. Our results imply that the TBWOA cannot only show excellent performance in the kNN classifier, it is also worked properly in the DT model. Hence, TBWOA can be considered a robust algorithm. Intuitively, the performance of the DT classifier was much better than kNN when applied to SFP. The TBWOA not only gives higher AUC but also maintains a lower feature selection ratio. As an instant conclusion, the TBWOA is known as the best BWOA variant in this work.

D. COMPARISON OF TBWOA WITH OTHER WELL-KNOWN OPTIMIZERS

To measure the efficacy of the proposed TBWOA in SFP, six well-known optimizers, namely BGWO [90], BGSA [91], BPSO [92], BALO [93], BBA [20], and BSSA [94], are used in this comparison section. Table 13 compares the AUC performance of the TBWOA with other optimizers. According to findings, the TBWOA scored the highest AUC value in 7 cases, followed by BGWO, BPSO, and BALO (3 cases). Besides, the TBWOA perceived the optimal rank with F-test equal to 2.47. The result indicates that the exploitative tendency of the TBWOA has been substantially improved. In the case of stagnation, the TBWOA can effectively escape the local region by exploiting the current best solution. This, in turn, will increase the capability of the algorithm in local optima avoidance. The result of the F-test in Table 13 supports this clarification.

TABLE 13. Comparison between TBWOA and other optimizers based on AUC values.

Dataset	TBWOA	BGWO	BGSA	BPSO	BALO	BBA	BSSA
ant-1.7	0.7369	0.7472	0.7385	0.7619	**0.7706**	0.6855	0.7465
camel-1.2	**0.7029**	0.6723	0.6321	0.6763	0.7022	0.5697	0.6949
camel-1.4	0.6864	**0.7263**	0.6535	0.6973	0.6390	0.5942	0.6454
camel-1.6	0.6767	0.5739	0.6250	0.6388	**0.6850**	0.5634	0.6843
jedit-3.2	0.7899	0.8308	0.7828	0.8443	0.8042	0.7059	**0.8675**
jedit-4.0	0.8229	**0.8727**	0.6731	0.8257	0.7826	0.7106	0.7680
jedit-4.1	0.7245	0.8039	0.7089	**0.8414**	0.8392	0.6553	0.8202
jedit-4.2	0.8472	**0.8839**	0.7527	0.7617	0.8216	0.6104	0.7812
log4j-1.0	**0.9036**	0.7810	0.8330	0.7986	0.7859	0.5051	0.8571
log4j-1.1	**0.9822**	0.8681	0.7417	0.8339	0.9025	0.6924	0.9140
log4j-1.2	**1.0000**	0.8393	0.6452	0.6787	0.8645	0.6219	0.6531
lucene-2.0	0.8180	0.7964	0.7490	**0.8908**	0.7594	0.7036	0.8046
lucene-2.2	**0.7653**	0.7391	0.7401	0.7575	0.7137	0.6053	0.7297
lucene-2.4	0.8108	0.7584	0.7294	0.8144	**0.8248**	0.6800	0.7344
xalan-2.4	**0.7735**	0.6651	0.6696	0.7507	0.6884	0.6590	0.6583
xalan-2.5	**0.7884**	0.7397	0.6791	0.7810	0.7123	0.6510	0.7388
xalan-2.6	0.7957	0.7803	0.7564	**0.8076**	0.8052	0.7548	0.7874
Ranking(F_test)	**2.47**	3.65	5.35	2.65	3.24	6.88	3.76

On the other side, the result of computational time is presented in Table 14. According to findings in Table 14, it is clear that the BGWO was the fastest algorithm while the TBWOA was the slowest method. Even though the computational cost consumed by the TBWOA was the highest, however, it can usually select the top attributes that can best explain the SFP, in which an optimal AUC performance can be ensured.

E. RELEVANT FEATURES SELECTED BY TBWOA

In this subsection, we demonstrate the potential features that have been selected by the TBWOA during the FS process. Figure 9 illustrates the importance of feature as well as their selection rates. As can be seen in Figure 9, the top five most frequently selected features were 'ce' (68.82%), 'moa' (67.84%), 'dit' (67.06%), 'cbo' (67.06%), and "rfc' (64.9%). The result signifies that these features had high discriminative power, and they can best describe the software fault.

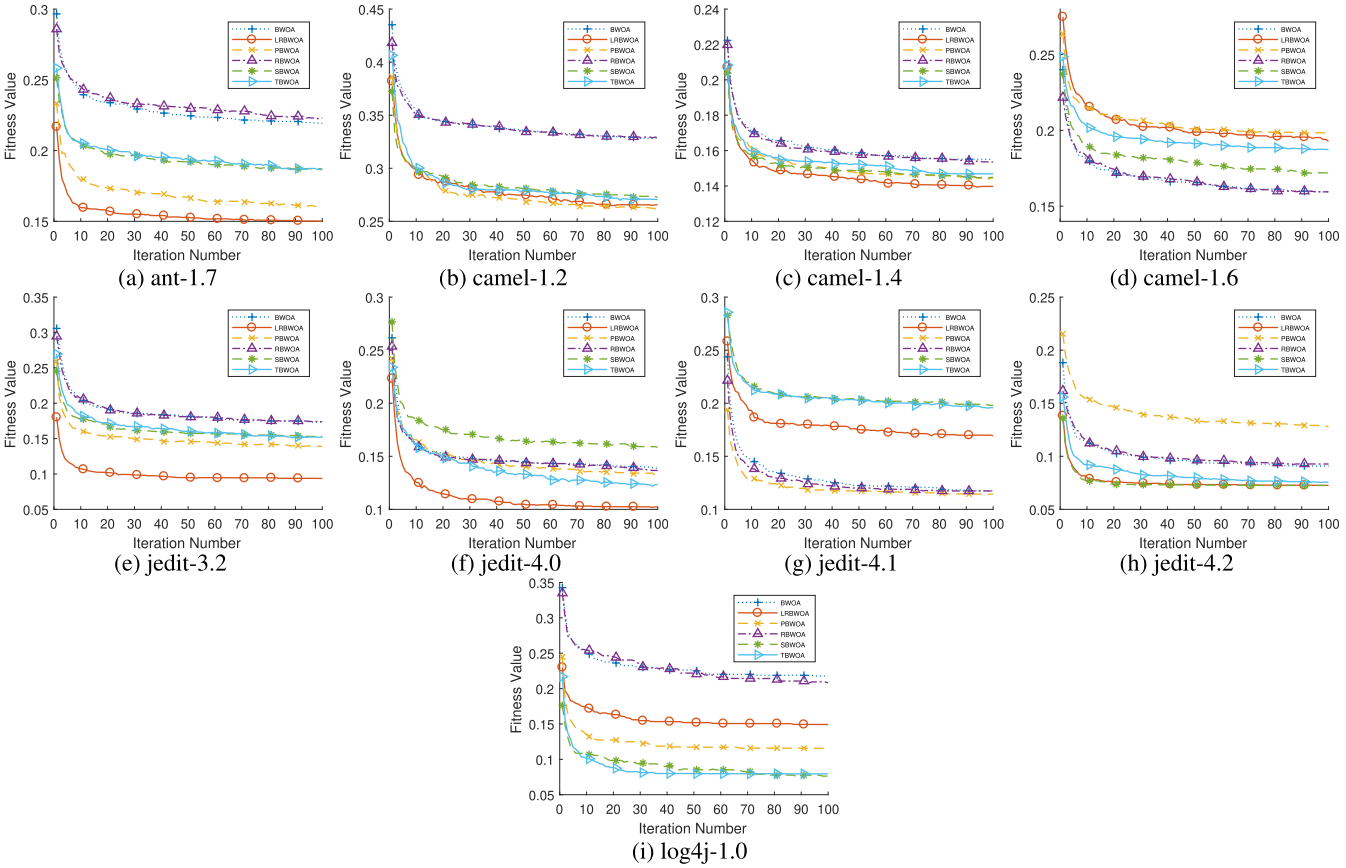


FIGURE 7. Convergence curves for different variants of BWOA with DT classifier on ant-1.7, camel-1.2, camel-1.4, camel-1.6, jedit-3.2, jedit-4.0, jedit-4.1, jedit-4.2, and log4j-1.0 datasets.

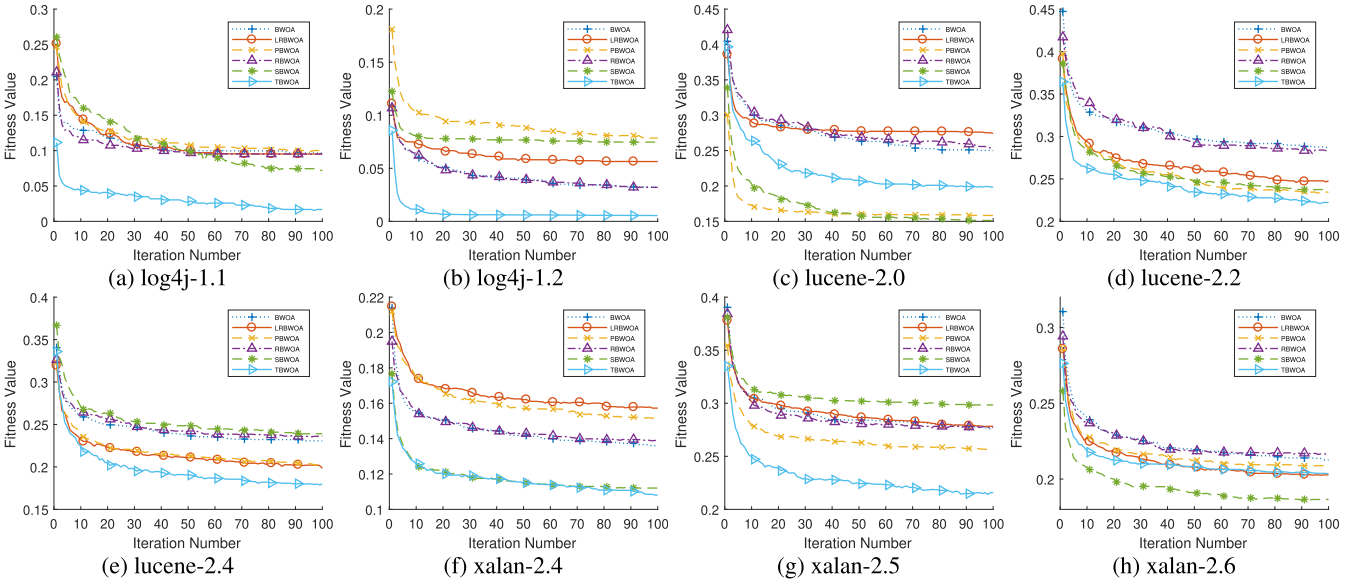


FIGURE 8. Convergence curves for different variants of BWOA with DT classifier on log4j-1.1, log4j-1.2, lucene-2.0, lucene-2.2, lucene-2.4, xalan-2.4, xalan-2.5, AND xalan-2.6 datasets.

F. PERFORMANCE OF THE PROPOSED TBWOA USING DIFFERENT CLASSIFIERS

From the previous sections, we have shown the efficacy of the TBWOA in SFP. Nevertheless, it is worth to investigate whether the performance of the TBWOA can be further improved when different classifiers are implemented. The TBWOA with four different learning algorithms (SVM, kNN, LDA, and DT) are used and compared in this

TABLE 14. Comparison between TBWOA and other optimizers in terms of average running time.

Dataset	TBWOA	BGWO	BGSA	BPSO	BALO	BBA	BSSA
ant-1.7	40.90	**27.15**	33.82	31.01	39.14	30.23	40.13
camel-1.2	42.80	**27.98**	34.52	32.46	39.60	31.14	39.77
camel-1.4	44.56	**29.32**	34.93	33.03	42.08	31.47	41.86
camel-1.6	49.08	**26.41**	38.11	33.40	45.74	34.86	44.87
jedit-3.2	27.51	**21.44**	23.30	22.17	24.76	22.77	24.25
jedit-4.0	27.99	22.62	23.23	22.48	24.96	**22.03**	24.29
jedit-4.1	27.69	**21.80**	23.78	22.86	25.48	22.26	24.77
jedit-4.2	25.80	**22.03**	22.81	22.11	24.11	22.49	24.74
log4j-1.0	24.01	**18.94**	19.44	20.13	20.92	19.37	20.55
log4j-1.1	22.85	19.18	19.07	19.87	20.01	**18.86**	19.54
log4j-1.2	23.69	19.76	20.73	19.75	20.48	**19.18**	20.14
lucene-2.0	26.34	21.37	21.76	21.75	22.51	**20.90**	22.94
lucene-2.2	28.32	22.60	23.86	24.11	24.91	**22.21**	25.90
lucene-2.4	29.72	**22.58**	24.39	24.83	28.52	24.00	26.25
xalan-2.4	38.68	**25.07**	29.05	29.87	35.00	28.53	35.20
xalan-2.5	49.26	**34.45**	36.56	38.76	48.05	36.00	45.90
xalan-2.6	46.84	**31.80**	36.18	36.28	44.35	33.15	44.44
Ranking(F_test)	7.00	**1.47**	3.65	3.18	5.53	1.88	5.29

section. Table 15 depicts the results of the TBWOA with four different learning algorithms. As can be seen, TBWOA with DT achieved the highest AUC performance (overall rank of 1.53). The result indicates that DT was the best learning algorithm for SFP. The second best learning algorithm was SVM, which was much better than kNN and LDA. However, the complexity results obtained in Table 15 turn out that the computational time of the SVM was the highest.

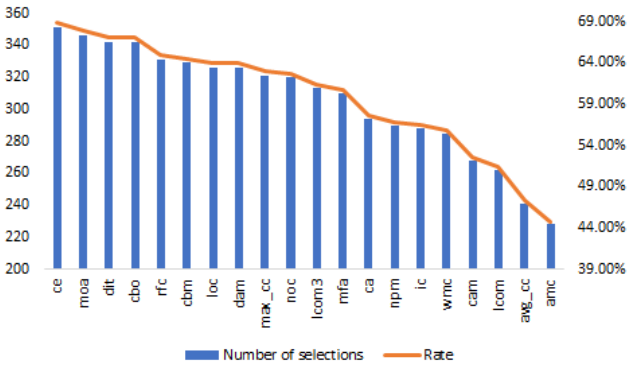


FIGURE 9. Importance of features in terms of the number of selections by TBWOA.

G. COMPARISON OF TBWOA WITH OTHER APPROACHES FROM THE LITERATURE

Finally, we compare the performance of TBWOA to other approaches from the literature. The comparison result is shown in Table 16, where '-' means that the authors did not evaluate their algorithm on that dataset. It is clear that the AUC performance of TBWOA was much higher than other methods in most datasets. By observing the result in dataset log4j-1.2, an increment of more than 20% AUC can be achieved with TBWOA. On the whole, the results support our research objective that the natural selection methods can greatly enhance the overall performance of machine learning classifiers. Ultimately, it can be concluded that TBWOA is a useful tool for SFP problems.

TABLE 15. Comparison of the proposed TBWOA using different learning algorithms in terms of AUC rates and running time.

Benchmark	AUC				Time			
	DT	KNN	SVM	LDA	DT	KNN	SVM	LDA
ant-1.7	**0.7369**	0.7196	0.7362	0.6865	40.90	**27.98**	114.41	40.62
camel-1.2	**0.7029**	0.6317	0.6476	0.5825	42.80	26.14	159.65	**26.14**
camel-1.4	**0.6864**	0.5897	0.5916	0.5000	44.56	29.72	959.77	**26.69**
camel-1.6	**0.6767**	0.6127	0.6686	0.5000	49.08	31.17	1573.85	**26.84**
jedit-3.2	0.7899	0.7549	**0.8305**	0.7697	27.51	**23.22**	35.42	25.29
jedit-4.0	**0.8229**	0.7126	0.7571	0.7193	27.99	**23.54**	39.03	25.51
jedit-4.1	0.7245	0.7694	**0.7769**	0.7088	27.69	**23.57**	36.43	25.40
jedit-4.2	**0.8472**	0.7272	0.7781	0.6611	25.80	**24.25**	52.55	25.49
log4j-1.0	**0.9036**	0.8595	0.8104	0.8125	24.01	**22.70**	27.50	25.24
log4j-1.1	**0.9822**	0.9444	0.9592	0.8771	22.85	**22.80**	27.10	24.83
log4j-1.2	**1.0000**	0.6792	0.5000	0.5000	23.69	**23.06**	34.32	25.04
lucene-2.0	0.8180	**0.9137**	0.7975	0.7214	26.34	**22.97**	31.01	24.96
lucene-2.2	0.7653	**0.7788**	0.7085	0.6877	28.32	**23.25**	39.14	25.02
lucene-2.4	0.8108	0.8474	**0.8566**	0.8299	29.72	**23.96**	41.56	25.40
xalan-2.4	**0.7735**	0.6506	0.6650	0.5856	38.68	27.58	126.94	**26.39**
xalan-2.5	**0.7884**	0.7164	0.7265	0.6728	49.26	28.64	191.40	**26.62**
xalan-2.6	0.7957	0.7923	0.7750	**0.8075**	46.84	29.99	120.03	**26.74**
Overall Rank	**1.53**	2.65	2.26	3.56	2.82	**1.32**	4.00	1.85

TABLE 16. Comparison of TBWOA with DT classifier to other approaches from the literature based on AUC results.

dataset	TBWOA	Shatnawi [97]				Okutan and Yildiz [98]	Erturk and Sezer [13]		Turabieh et. al. [3]
	DT	LR	NB	5NN	C4.5	Bayesian	ANN	ANFIS	LRNN
ant-1.7	0.737	0.830	0.790	0.760	0.740	0.820	**0.847**	0.818	0.523
camel-1.2	**0.703**	0.570	0.560	0.640	0.520	-	0.601	0.601	0.443
camel-1.4	0.686	0.700	0.670	0.670	0.600	-	0.791	**0.813**	0.521
camel-1.6	0.677	0.650	0.590	0.660	0.540	-	0.681	**0.714**	0.505
jedit-3.2	0.790	-	-	-	-	-	0.880	**0.900**	0.518
jedit-4.0	0.823	0.770	0.700	0.810	0.720	-	**0.825**	0.783	0.549
jedit-4.1	0.724	0.820	0.750	0.800	0.690	-	0.868	**0.914**	-
jedit-4.2	0.847	0.840	0.750	0.770	0.640	-	0.875	**0.976**	0.519
log4j-1.0	**0.904**	-	-	-	-	-	0.893	0.886	0.535
log4j-1.1	**0.982**	-	-	-	-	-	0.902	0.902	0.522
log4j-1.2	**1.000**	-	-	-	-	-	0.780	0.772	0.361
lucene-2.0	0.818	0.770	0.750	0.700	0.670	-	0.849	**0.865**	0.553
lucene-2.2	**0.765**	0.620	0.610	0.700	0.580	-	0.763	0.746	0.463
lucene-2.4	0.811	0.750	0.690	0.730	0.680	0.633	**0.825**	0.815	0.485
xalan-2.4	0.773	-	-	-	-	-	0.819	**0.820**	0.536
xalan-2.5	**0.788**	-	-	-	-	0.624	0.675	0.663	0.510
xalan-2.6	**0.796**	-	-	-	-	-	0.682	0.678	0.506

VI. CONCLUSION AND FUTURE WORKS

In this paper, new variants of WOA were proposed as wrapper algorithms to handle the feature selection problems in SFP applications. Five different natural selection schemes (random selection, tournament selection, roulette wheel selection, linear rank-based selection, and stochastic universal sampling selection) were employed and integrated into the WOA algorithm. The main idea is to enhance the quality of the leader as well as the worst solution in the population, which assisted the whales (solution) to escape from the local optima scenario while examining the search space. The proposed approaches were simulated and validated over 17 public SFP data from the PROMISE repository. Among the proposed WOA variants, the TBWOA has retained the best performance in most datasets. The results obtained affirm the ability of the natural selection method not only at enhancing the overall performance of WOA but also in removing the unwanted attributes. Furthermore, the experimental results revealed that the proposed TBWOA overwhelmed the other six state-of-the-art methods with the highest AUC performance. Our findings showed that TBWOA is a useful and reliable tool for SFP applications.

In future work, the performance of the proposed approach can be investigated over different optimization problems from real-world applications, such as travel salesman problems, educational timetabling problems, and students' performance prediction problems.

ACKNOWLEDGMENT

This work was supported by Taif University Researchers Supporting, Taif University, Taif, Saudi Arabia, under Project TURSP-2020/125.

REFERENCES

[1] I. Sommerville, *Software Engineering*, 9th ed. London, U.K.: Pearson, 2011, p. 18.

[2] A. A. Porter and R. W. Selby, "Empirically guided software development using metric-based classification trees," *IEEE Softw.*, vol. 7, no. 2, pp. 46–54, Mar. 1990.

[3] H. Turabieh, M. Mafarja, and X. Li, "Iterated feature selection algorithms with layered recurrent neural network for software fault prediction," *Expert Syst. Appl.*, vol. 122, pp. 27–42, May 2019.

[4] I. Tumar, Y. Hassouneh, H. Turabieh, and T. Thaher, "Enhanced binary moth flame optimization as a feature selection algorithm to predict software fault prediction," *IEEE Access*, vol. 8, pp. 8041–8055, 2020.

[5] W. W. Royce, "Managing the development of large software systems: Concepts and techniques," in *Proc. 9th Int. Conf. Softw. Eng.*, Mar. 1987, pp. 328–338.

[6] M. Fowler and J. Highsmith, "The agile manifesto," *Softw. Develop.*, vol. 9, no. 8, pp. 28–35, 2001.

[7] R. Hoda, N. Salleh, J. Grundy, and H. M. Tee, "Systematic literature reviews in agile software development: A tertiary study," *Inf. Softw. Technol.*, vol. 85, pp. 60–70, May 2017.

[8] A. Kalsoom, M. Maqsood, M. A. Ghazanfar, F. Aadil, and S. Rho, "A dimensionality reduction-based efficient software fault prediction using Fisher linear discriminant analysis (FLDA)," *J. Supercomput.*, vol. 74, no. 9, pp. 4568–4602, Sep. 2018.

[9] S. S. Rathore and S. Kumar, "A decision tree logic based recommendation system to select software fault prediction techniques," *Computing*, vol. 99, no. 3, pp. 255–285, Mar. 2017.

[10] Y. Xia, G. Yan, and Q. Si, "A study on the significance of software metrics in defect prediction," in *Proc. 6th Int. Symp. Comput. Intell. Design*, vol. 2, Oct. 2013, pp. 343–346.

[11] T. Zimmermann, N. Nagappan, and A. Zeller, "Predicting bugs from history," in *Software Evolution*. Berlin, Germany: Springer, 2008.

[12] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Inf. Sci.*, vol. 264, pp. 260–278, Apr. 2014.

[13] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," *Expert Syst. Appl.*, vol. 42, no. 4, pp. 1872–1879, Mar. 2015.

[14] G. K. Rajbahadur, S. Wang, Y. Kamei, and A. E. Hassan, "The impact of using regression models to build defect classifiers," in *Proc. IEEE/ACM 14th Int. Conf. Mining Softw. Repositories (MSR)*, May 2017, pp. 135–145, doi: 10.1109/MSR.2017.4.

[15] P. Deep Singh and A. Chug, "Software defect prediction analysis using machine learning algorithms," in *Proc. 7th Int. Conf. Cloud Comput., Data Sci. Eng. Confluence*, Jan. 2017, pp. 775–781.
[16] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007.
[17] F. Xing, P. Guo, and M. R. Lyu, "A novel method for early software quality prediction based on support vector machine," in *Proc. 16th IEEE Int. Symp. Softw. Rel. Eng. (ISSRE)*, Nov. 2005, p. 10.
[18] J. C. M. Oliveira, K. V. Pontes, I. Sartori, and M. Embiruçu, "Fault detection and diagnosis in dynamic systems using weightless neural networks," *Expert Syst. Appl.*, vol. 84, pp. 200–219, Oct. 2017.
[19] M. Mafarja and S. Mirjalili, "Whale optimization approaches for wrapper feature selection," *Appl. Soft Comput.*, vol. 62, pp. 441–453, Jan. 2018.
[20] M. Mafarja, I. Aljarah, A. A. Heidari, A. I. Hammouri, H. Faris, A. M. Al-Zoubi, and S. Mirjalili, "Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems," *Knowl.-Based Syst.*, vol. 145, pp. 25–45, Apr. 2018.
[21] M. Mafarja, A. A. Heidari, M. Habib, H. Faris, T. Thaher, and I. Aljarah, "Augmented whale feature selection for IoT attacks: Structure, analysis and applications," *Future Gener. Comput. Syst.*, vol. 112, pp. 18–40, Nov. 2020.
[22] M. Mafarja, D. Eleyan, S. Abdullah, and S. Mirjalili, "S-shaped vs. V-Shaped transfer functions for ant lion optimization algorithm in feature selection problem," in *Proc. Int. Conf. Future Netw. Distrib. Syst.*, Jul. 2017, pp. 1–7.
[23] T. W. Rauber, F. de Assis Boldt, and F. M. Varejao, "Heterogeneous feature models and feature selection applied to bearing fault diagnosis," *IEEE Trans. Ind. Electron.*, vol. 62, no. 1, pp. 637–646, Jan. 2015.
[24] 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.
[25] M. Mafarja, I. Aljarah, H. Faris, A. I. Hammouri, A. M. Al-Zoubi, and S. Mirjalili, "Binary grasshopper optimisation algorithm approaches for feature selection problems," *Expert Syst. Appl.*, vol. 117, pp. 267–286, Mar. 2019.
[26] F. Luo, H. Huang, Y. Duan, J. Liu, and Y. Liao, "Local geometric structure feature for dimensionality reduction of hyperspectral imagery," *Remote Sens.*, vol. 9, no. 8, p. 790, Aug. 2017.
[27] M. Taradeh, M. Mafarja, A. A. Heidari, H. Faris, I. Aljarah, S. Mirjalili, and H. Fujita, "An evolutionary gravitational search-based feature selection," *Inf. Sci.*, vol. 497, pp. 219–239, Sep. 2019.
[28] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1276–1304, Nov. 2012.
[29] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Appl. Soft Comput.*, vol. 27, pp. 504–518, Feb. 2015, doi: 10.1016/j.asoc.2014.11.023.
[30] H. Mühlenbein, "Genetic algorithms," Citeseer, Princeton, NJ, USA, Tech. Rep., 1997.
[31] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[32] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE ICNN*, vol. 4. Nov./Dec. 1995, pp. 1942–1948.
[33] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.
[34] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: A new method for stochastic optimization," *Future Gener. Comput. Syst.*, vol. 111, pp. 300–323, Oct. 2020.
[35] I. Ahmadianfar, O. Bozorg-Haddad, and X. Chu, "Gradient-based optimizer: A new Metaheuristic optimization algorithm," *Inf. Sci.*, vol. 540, pp. 131–159, Nov. 2020.
[36] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.
[37] D. Oliva, M. Abd El Aziz, and A. Ella Hassanien, "Parameter estimation of photovoltaic cells using an improved chaotic whale optimization algorithm," *Appl. Energy*, vol. 200, pp. 141–154, Aug. 2017.
[38] M. A. E. Aziz, A. A. Ewees, and A. E. Hassanien, "Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation," *Expert Syst. Appl.*, vol. 83, pp. 242–256, Oct. 2017.
[39] I. Aljarah, H. Faris, and S. Mirjalili, "Optimizing connection weights in neural networks using the whale optimization algorithm," *Soft Comput.*, vol. 22, no. 1, pp. 1–15, Jan. 2018.
[40] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.
[41] M. Abdel-Basset, L. Abdle-Fatah, and A. K. Sangaiah, "An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment," *Cluster Comput.*, vol. 22, no. S4, pp. 8319–8334, Jul. 2019.
[42] A. A. Heidari, I. Aljarah, H. Faris, H. Chen, J. Luo, and S. Mirjalili, "An enhanced associative learning-based exploratory whale optimizer for global optimization," *Neural Comput. Appl.*, vol. 32, pp. 1–27, Jan. 2019.
[43] Y. Sun, X. Wang, Y. Chen, and Z. Liu, "A modified whale optimization algorithm for large-scale global optimization problems," *Expert Syst. Appl.*, vol. 114, pp. 563–577, Dec. 2018.
[44] 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.
[45] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[46] X. Yuan, T. M. Khoshgoftaar, E. B. Allen, and K. Ganesan, "An application of fuzzy clustering to software quality prediction," in *Proc. 3rd IEEE Symp. Appl.-Specific Syst. Softw. Eng. Technol.*, Mar. 2000, pp. 85–90.
[47] T. M. Khoshgoftaar and E. B. Allen, "Logistic regression modeling of software quality," *Int. J. Rel., Qual. Saf. Eng.*, vol. 6, no. 4, pp. 303–317, Dec. 1999.
[48] K. El Emam, S. Benlarbi, N. Goel, and S. N. Rai, "Comparing case-based reasoning classifiers for predicting high risk software components," *J. Syst. Softw.*, vol. 55, no. 3, pp. 301–320, Jan. 2001.
[49] T. M. Khoshgoftaar and N. Seliya, "Software quality classification modeling using the SPRINT decision tree algorithm," in *Proc. 14th IEEE Int. Conf. Tools Artif. Intell., (ICTAI)*, Nov. 2002, pp. 365–374.
[50] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: Do different classifiers find the same defects?" *Softw. Qual. J.*, vol. 26, no. 2, pp. 525–552, Jun. 2018.
[51] T.-S. Quah and M. M. T. Thwin, "Application of neural networks for software quality prediction using object-oriented metrics," in *Proc. Int. Conf. Softw. Maintenance, ICSM*, Sep. 2003, pp. 116–125.
[52] S. K. Pandey, R. B. Mishra, and A. K. Triphathi, "Software bug prediction prototype using Bayesian network classifier: A comprehensive model," *Procedia Comput. Sci.*, vol. 132, pp. 1412–1421, Jan. 2018.
[53] B. Shuai, H. Li, M. Li, Q. Zhang, and C. Tang, "Software defect prediction using dynamic support vector machine," in *Proc. 9th Int. Conf. Comput. Intell. Secur.*, Dec. 2013, pp. 260–263.
[54] A. Singh, R. Bhatia, and A. Singhrova, "Taxonomy of machine learning algorithms in software fault prediction using object oriented metrics," *Procedia Comput. Sci.*, vol. 132, pp. 993–1001, Jan. 2018.
[55] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Inf. Sci.*, vol. 179, no. 8, pp. 1040–1058, Mar. 2009.
[56] S. Yogesh, K. Arvinder, and R. Malhotra, "Software fault proneness prediction using support vector machines," in *Proc. World Congr. Eng. (WCE)*, in Lecture Notes in Computer Science, London, U.K., vol. 2176, Jul. 2009.
[57] S. S. Rathore and S. Kumar, "Towards an ensemble based system for predicting the number of software faults," *Expert Syst. Appl.*, vol. 82, pp. 357–382, Oct. 2017.
[58] M. Dash and H. Liu, "Feature selection for classification," *Intell. Data Anal.*, vol. 1, nos. 1–4, pp. 131–156, 1997.
[59] H. Chantar, M. Mafarja, H. Alsawalqah, A. A. Heidari, I. Aljarah, and H. Faris, "Feature selection using binary grey wolf optimizer with elite-based crossover for arabic text classification," *Neural Comput. Appl.*, vol. 32, pp. 12201–12220, Jul. 2019.
[60] A. O. Balogun, S. Basri, S. J. Abdulkadir, and A. S. Hashim, "Performance analysis of feature selection methods in software defect prediction: A search method approach," *Appl. Sci.*, vol. 9, no. 13, p. 2764, Jul. 2019.
[61] N. Dhamayanthi and B. Lavanya, "Software defect prediction using principal component analysis and naïve Bayes algorithm," in *Proc. Int. Conf. Comput. Intell. Data Eng.* Singapore: Springer, 2019, pp. 241–248.
[62] R. S. Wahono and N. S. Herman, "Genetic feature selection for software defect prediction," *Adv. Sci. Lett.*, vol. 20, no. 1, pp. 239–244, Jan. 2014.
[63] R. S. Wahono, N. Suryana, and S. Ahmad, "Metaheuristic optimization based feature selection for software defect prediction," *J. Softw.*, vol. 9, no. 5, pp. 1324–1333, May 2014.
[64] M. Banga and A. Bansal, "Proposed software faults detection using hybrid approach," *Secur. Privacy*, Jan. 2020, Art. no. e103.

[65] T. Thaher, M. Mafarja, B. Abdalhaq, and H. Chantar, "Wrapper-based feature selection for imbalanced data using binary queuing search algorithm," in *Proc. 2nd Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2019, pp. 1–6.
[66] T. Thaher and N. Arman, "Efficient multi-swarm binary harris hawks optimization as a feature selection approach for software fault prediction," in *Proc. 11th Int. Conf. Inf. Commun. Syst. (ICICS)*, Apr. 2020, pp. 249–254.
[67] M. Sharawi, H. M. Zawbaa, E. Emary, H. M. Zawbaa, and E. Emary, "Feature selection approach based on whale optimization algorithm," in *Proc. 9th Int. Conf. Adv. Comput. Intell. (ICACI)*, Feb. 2017, pp. 163–168.
[68] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Neurocomputing*, vol. 260, pp. 302–312, Oct. 2017.
[69] E. Zorarpacı and S. A. Özel, "A hybrid approach of differential evolution and artificial bee colony for feature selection," *Expert Syst. Appl.*, vol. 62, pp. 91–103, Nov. 2016.
[70] H. Zhang and G. Sun, "Feature selection using tabu search method," *Pattern Recognit.*, vol. 35, no. 3, pp. 701–711, Mar. 2002.
[71] R. Meiri and J. Zahavi, "Using simulated annealing to optimize the feature selection problem in marketing applications," *Eur. J. Oper. Res.*, vol. 171, no. 3, pp. 842–858, Jun. 2006.
[72] F. J. Ferri, V. Kadirkamanathan, and J. Kittler, "Feature subset search using genetic algorithms," in *Proc. IEE/IEEE Workshop Natural Algorithms Signal Process.*, 1993.
[73] R. Leardi, R. Boggia, and M. Terrile, "Genetic algorithms as a strategy for feature selection," *J. Chemometrics*, vol. 6, no. 5, pp. 267–281, Sep. 1992.
[74] H. K. Chantar and D. W. Corne, "Feature subset selection for arabic document categorization using BPSO-KNN," in *Proc. 3rd World Congr. Nature Biologically Inspired Comput.*, Oct. 2011, pp. 546–551.
[75] L.-Y. Chuang, C.-H. Yang, and J.-C. Li, "Chaotic maps based on binary particle swarm optimization for feature selection," *Appl. Soft Comput.*, vol. 11, no. 1, pp. 239–248, Jan. 2011.
[76] S. Kashef and H. Nezamabadi-pour, "An advanced ACO algorithm for feature subset selection," *Neurocomputing*, vol. 147, pp. 271–279, Jan. 2015.
[77] R. Y. M. Nakamura, L. A. M. 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*, Aug. 2012, pp. 291–297.
[78] H. M. Zawbaa, E. Emary, and B. Parv, "Feature selection based on antlion optimization algorithm," in *Proc. 3rd World Conf. Complex Syst. (WCCS)*, Nov. 2015, pp. 1–7.
[79] H. M. Zawbaa, E. Emary, B. Parv, and M. Sharawi, "Feature selection approach based on moth-flame optimization algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 4612–4617.
[80] W. A. Watkins and W. E. Schevill, "Aerial observation of feeding behavior in four baleen whales: Eubalaena glacialis, balaenoptera borealis, megaptera novaeangliae, and balaenoptera physalus," *J. Mammalogy*, vol. 60, no. 1, pp. 155–163, Feb. 1979.
[81] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary particle swarm optimization," *Swarm Evol. Comput.*, vol. 9, pp. 1–14, Apr. 2013.
[82] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 101–111. [Online]. Available: http://dl.acm.org/citation.cfm?id=645511.657075
[83] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. Michigan Press, Jan. 1975.
[84] M. A. Al-Betar, M. A. Awadallah, H. Faris, I. Aljarah, and A. I. Hammouri, "Natural selection methods for grey wolf optimizer," *Expert Syst. Appl.*, vol. 113, pp. 481–498, Dec. 2018.
[85] M. A. Al-Betar, M. A. Awadallah, H. Faris, X.-S. Yang, A. Tajudin Khader, and O. A. Alomari, "Bat-inspired algorithms with natural selection mechanisms for global optimization," *Neurocomputing*, vol. 273, pp. 448–465, Jan. 2018.
[86] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proc. 2nd Int. Conf. Genetic Algorithms Genetic Algorithms Their Appl.*, Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1987, pp. 14–21. [Online]. Available: http://dl.acm.org/citation.cfm?id=42512.42515
[87] D. Goldberg, K. Deb, and B. Korb, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex Syst.*, vol. 3, no. 5, pp. 493–530, 1989.
[88] (2017). *Tera-Promise*. Accessed: Nov. 24, 2017. [Online]. Available: http://openscience.us/repo
[89] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proc. 6th Int. Conf. Predictive Models Softw. Eng. - PROMISE*, 2010, pp. 9:1–9:10, doi: 10.1145/1868328.1868342.
[90] 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.
[91] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "BGSA: Binary gravitational search algorithm," *Natural Comput.*, vol. 9, no. 3, pp. 727–745, Sep. 2010.
[92] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. IEEE Int. Conf. Syst., Man, Cybern., Comput. Cybern. Simulation*, Oct. 1997, pp. 4104–4108.
[93] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary ant lion approaches for feature selection," *Neurocomputing*, vol. 213, pp. 54–65, Nov. 2016.
[94] E. Kaya, A. Çinar, O. Uymaz, S. Korkmaz, and M. Kıran, "The binary salp swarm algorithm with using transfer functions," in *Proc. Int. Conf. Adv. Technol., Comput. Eng. Sci. (ICATCES)*, Safranbolu, Turkey, May 2018.
[95] L. Jiang, Z. Cai, D. Wang, and S. Jiang, "Survey of improving K-nearest-neighbor for classification," in *Proc. 4th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, Aug. 2007, pp. 679–683.
[96] H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A. M. Al-Zoubi, S. Mirjalili, and H. Fujita, "An efficient binary salp swarm algorithm with crossover scheme for feature selection problems," *Knowl.-Based Syst.*, vol. 154, pp. 43–67, Aug. 2018.
[97] R. Shatnawi, "The application of ROC analysis in threshold identification, data imbalance and metrics selection for software fault prediction," *Innov. Syst. Softw. Eng.*, vol. 13, nos. 2–3, pp. 201–217, Sep. 2017.
[98] A. Okutan and O. T. Yıldiz, "Software defect prediction using Bayesian networks," *Empirical Softw. Eng.*, vol. 19, no. 1, pp. 154–181, Feb. 2014.
[99] E. Erturk and E. A. Sezer, "Iterative software fault prediction with a hybrid approach," *Appl. Soft Comput.*, vol. 49, pp. 1020–1033, Dec. 2016.

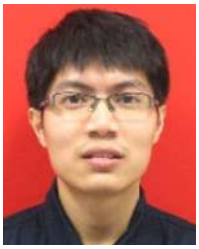
YOUSEF HASSOUNEH is currently an Assistant Professor with the Department of Computer Science, Birzeit University, teaching courses in software engineering, internet programming, and programming languages. He has an academic administration experience, as he served as the Department Chair and the Director of the Computing Master Program. He has a profound experience in human–computer interaction. He designed a collaboration framework and groupware tool to enable Requirements Engineering Team collaboration. He participated in several EU funded projects. His research interests include software architecture, virtual software engineering teams, Software risk assessment and metrics, and mining software repositories.

HAMZA TURABIEH received the B.A. and M.Sc. degrees in computer science from Balqa Applied University, Jordan, in 2004 and 2006, respectively, and the Ph.D. degree from National University of Malaysia (UKM), in 2010. He is currently an Associate Professor with the Department of Computer Science, Faculty of Science and Information Technology, Taif University. His research interests include interface of computer science and operational research, intelligent decision support systems, search and optimization (combinatorial optimization, constraint optimization, multi-modal optimization, and multi-objective optimization) using heuristics, local search, meta-heuristics (in particular memetic algorithms, particle swarm optimization), and hybrid approaches and their theoretical foundations.

THAER THAHER received the B.Sc. degree in computer engineering and the M.Sc. degree in advanced computing from An-Najah National University, Palestine, in 2007 and 2018, respectively. He is currently pursuing the Ph.D. degree in information technology engineering with Arab American University, Palestine Polytechnic University, and Al-Quds University, Palestine. His research interests include evolutionary computation, meta-heuristics, data mining, and machine learning.

IYAD TUMAR received the bachelor's degree in electrical engineer (communication systems) and the master's degree in computational science from Birzeit University, Palestine, in 2002 and 2006, respectively, and the Ph.D. degree in smart systems from Jacobs University Bremen, Germany, in 2010. He was a Research Associate and a member with the Computer Networks and Distributed Systems Group, Jacobs University Bremen, and the European Network of Excellence for the Management of Internet Technologies and Complex Services (EMANICS), from 2007 to 2010. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Birzeit University (BZU). He is also a Project Manager for two research projects funded by the Qatar Foundation and BMBF in Germany. His research interests include wireless networks, resource management of disruption tolerant networks, wireless communication, wireless sensor networks, underwater networks, network management, feature selection, software engineering, and machine learning. He was a Reviewer of IEEE/ACM journals and conferences.

HAMOUDA CHANTAR received the B.Sc. degree in computer science from Sebha University, Libya, the M.Sc. degree in advanced computer science from Sheffield University, and the Ph.D. degree in computer science from Heriot-Watt University, in 2013. He is currently a Lecturer with the Faculty of Information Technology, Sebha University. His research interests include text mining, meta-heuristics, and data mining.



JINGWEI TOO received the bachelor's degree in mechatronics engineering, in 2017, and the Ph.D. degree in electrical and computer engineering from Universiti Teknikal Malaysia Melaka. His research interests include signal processing, meta-heuristics, machine learning, and data mining.