Efficient Multi-Swarm Binary Harris Hawks Optimization as a Feature Selection Approach for Software Fault Prediction

Thaer Thaher
Department of Computer Science and Engineering
Palestine Polytechnic University
Hebron, Palestine
thaer.thaher@gmail.com

Nabil Arman
Department of Computer Science and Engineering
Palestine Polytechnic University
Hebron, Palestine
narman@ppu.edu

Abstract—**Designing a solution for an optimization problem requires two main aspects; the optimization technique (e.g., search strategy) and the evaluation criteria (i.e., objective function). In this paper, an enhanced binary version of a recent metaheuristic algorithm, the Harris Hawk Optimization algorithm (EBHHO), is presented to find a (near) optimal solution for the Feature Selection (FS) problem. Moreover, three different classifiers called K-nearest neighbors (kNN), Decision Trees (DT), and Linear Discriminant Analysis (LDA) were used as evaluation criteria to formulate the objective function. In addition to reducing the dimensionality of the dataset using the FS technique, the Adaptive Synthetic (ADASYN) oversampling technique was used to enhance the quality of the learning algorithm by re-balancing the dataset. A set of well-known datasets in the field of Software Fault Prediction (SFP) were used to validate the efficiency of the proposed approach. The obtained results showed that EBHHO is superior over the basic HHO as well as proved the ability of the EBHHO algorithm to produce the best result among a set of well-known optimization methods.**

Index Terms—**Optimization, Approximation Algorithms, Metaheuristics, Harris Hawks Optimization, Feature Selection, Software Fault Prediction, Imbalanced Data, Adaptive synthetic Sampling, Classification.**

I. Introduction

Most of the real-world optimization problems are NP-hard [1]. There is no effective algorithm that can solve these problems in polynomial time. Therefore, approximation algorithms are introduced as an efficient alternative to handle these kinds of problems [2]. Metaheuristics are a significant subclass of approximation algorithms that proved their ability to explore the search space for the problem being solved and find the (near) optimal solution in a reasonable time [3].

Metaheuristic algorithms can be classified into a single solution based and population-based [1]. In the population-based algorithms, a collection of solutions (called population) are generated at the beginning of the optimization process, and each solution in the population is manipulated according to a specific mechanism, depending on the behavior of the algorithm. Population-based algorithms have two phases; exploration and exploitation, which help in avoiding being stuck at local optima. At the beginning of the optimization process, the algorithm starts exploring the search space to find the promising regions, while in exploitation, the algorithm searches for the best solution in a specific area of the search space. Swarm Intelligence (SI) algorithms are population-based metaheuristics which recently exploitedto tackle different optimization problems, including Feature Selection (FS) [4] [5].

FS is a combinatorial optimization problem that aims to reduce the complexity of data mining processes, when dealing with high dimensional datasets, by eliminating the irrelevant and/or redundant features [6]. With the emergent of the advanced data collection tools, the FS step becomes crucial and mandatory to enhance the performing of the mining process. The main challenge in the FS process is searching for the best feature combination. For those datasets with a small number of features, one can use the complete (exhaustive) search that tends to test all possible combinations and then selects the best performing one, or the exact search algorithms These search strategies become impractical and time-consuming when dealing with high dimensional datasets, where the search space is exponentially increased. That's to say, for a dataset with N features; the search space consists of $2^N - 1$ feature subsets. Thus, metaheuristics algorithms are the best alternative to the previously mentioned search strategies.

As an optimization problem, in addition to adopting the right optimization technique, an evaluation criteria should be defined. In FS, there are two main models to evaluate a feature subset, filters and wrappers. In filter model, the evaluation is based on the data itself without considering a learning technique. While in wrappers, a learning algorithm (e.g., classifier) is considered to evaluate the goodness of the selected features [7].

In addition to the high dimensionality problem, some datasets may have another problem that may degrade the performance of the learning algorithms, where one class (the minority class) is much rarer than other classes (the majority class) in the dataset. This problem is known as imbalanced data [8]. To solve this problem, various approaches have been proposed, such as kernel-Based methods, Active Learning methods, Cost-Sensitive methods, and sampling methods [8]. One of the major topics that may suffer from these both

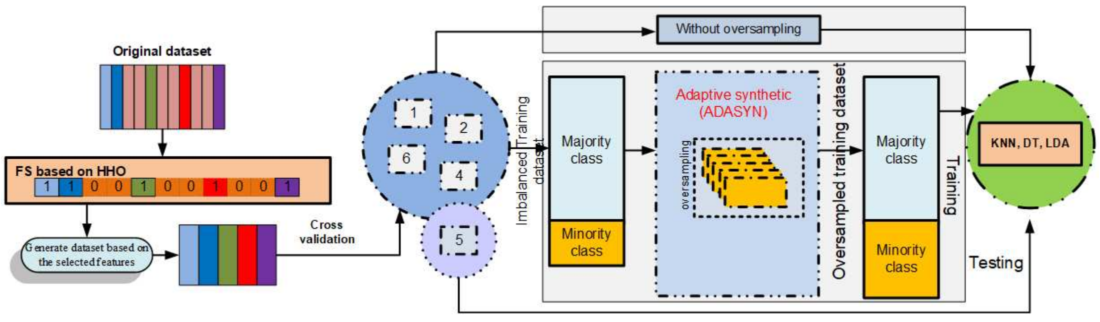


Fig. 1: The proposed SFP model

challenges is known as Software Fault Prediction (SFP), that tends to predict the faulty components in a software project. In SFP, many metrics (features) can be considered, at the same time, the faulty components are rarely occurred in software projects.

In this paper, we propose an efficient SFP model based on a wrapper FS approach augmented with the Adaptive Synthetic Oversampling (ADASYN) to increase the prediction performance in real-world software projects. The main contributions are summarized as follows:

- An enhanced binary version of HHO (EBHHO) using a multi-swarm strategy is designed as a search method in the wrapper FS approach.
- The ADASYN oversampling technique is used to re-balance the used datasets.
- The proposed model is evaluated on 15 real datasets of software projects.

The rest of this paper is organized as follows. Section II reviews the related works for SFP, including pure machine learning models, and those augmented with FS as a prepossessing technique. The proposed methodology, as well as the employed methods, are presented deeply in Section III. Section IV presents the experimental results, discussion, and analysis. Finally, Section V concludes the overall results as well as future work directions.

II. Review of related works

There are many research papers investigating the SFP problem in the literature. Most recent articles employed machine learning methods to build an acceptable model(s) for this challenging problem. Some articles applied pure machine learning methods without any pre-processing on the input datasets [9], while other researchers perform a set of pre-processing on the datasets such as feature selection to reduce the dimensionally of input datasets [10], or noise removal that comes from imbalanced datasets [11], [4].

Erturk and Sezer [12] applied an Artificial Neural Network (ANN) and Adaptive Neuro-Fuzzy Inference System for SFP problem. Erturk and Sezer [9] employed three different machine learning classifiers (i.e., ANN, SVM, and ANFIS). Manjula and Florence [13] proposed a hybrid method between genetic algorithm (GA) and deep neural network (DNN) to handle the SFP problem. Rhmann et al. [14] employed a set of hybrid machine learning algorithms to predict software fault for an android project. Jayanthi and Florence [15] employed Principle Component Analysis (PCA) with ANN as a feature reduction for SFP.

Turabieh et al.[10] employed an iterated feature selection using a layered recurrent neural network to predict object-oriented faults. The authors used a pool of population-based feature selection algorithms (i.e., binary genetic algorithm, binary particle swarm optimization, and binary ant colony optimization) to select the most valuable features. The obtained results explore the importance of feature selection for this domain. Tumar et al. [11] introduced an intelligent model for SFP problem based on a Binary Moth Flame Optimization (BMFO) as wrapper feature selection and ADASYN to handle the problem of imbalanced data. Thaher et al. [4] proposed a Wrapper-based feature selection model for SFP by employing Binary Queuing Search Algorithm (BQSA). They utilized Synthetic Minority Oversampling Technique (SMOTE) to re-balance the datasets.

It is clear that from previous literature, there are tremendous research papers that investigate the software fault prediction problem. In reality, each project has its attributes, so it is essential to develop a good model that analyzes the collected data in advance. This motivates us to employ HHO as a feature selection algorithm as the first step in order to achieve a high-quality classifier.

III. The proposed methodology

In this research, we propose a novel SFP model based on wrapper FS method augmented with ADASYN oversampling technique. An enhanced binary version of recent HHO algorithm is employed for searching the most valuable features. Figure 1 illustrates the abstract of the proposed framework in details. For each candidate subset of features (i.e., candidate solution) generated by the BHHO algorithm, the evaluation process is done using the k-folds cross-validation method (where k=5). The method starts by dividing up the dataset into five blocks, such that four blocks (80%) are used for training, and the remaining block (20%) is used for testing. For each training and testing procedure, the ADASYN technique is used to over-sample the imbalanced training part, while the testing part remains the same. This procedure is repeated k times, and the prediction results are summarized at the end. The overall approach is repeated until reaching the maximum number of allowable iterations. In the following subsections, we will clarify the framework components deeply.

A. Harris Hawks Optimization (HHO)

HHO is a recent nature-inspired metaheuristic algorithm classified under the umbrella of swarm intelligence algorithms. It was proposed by Heidari et al. in 2019 [16] to mimic the natural behavior of Harris hawks, which employ distinctive chasing styles to trap the prey.

HHO is classified as a population-based algorithm in which a group of hawks (each represents a candidate solution) cooperates using various chasing styles to follow the location of prey (the fittest candidate solution). The proposed mathematical model in the original paper of HHO showed a compelling performance in handling various constrained and unconstrained problems. It can balance between exploration and exploitation through employing six updating phases; two phases of exploration and four phases of exploitation.

1) Exploration phases: In this stage, the search agents are updated using two strategies. This achieved by assuming a probability of 50% to select between the two updating methods. So agents are updated based on another randomly selected agent or based on the best solution obtained so far.

2) Transformation from exploration to exploitation: HHO model employs an adaptive parameter called escaping energy (E) which decreases linearly over time to smoothly transfer from exploration to exploitation. The idea of this mechanism comes from that the energy of the rabbit decreases through the escaping behavior.

3) Exploitation phases: HHO utilizes various time-varying mechanisms with a greedy scheme to model the exploitation phase. The choice between the four updating mechanisms depends on the energy (E) and the escaping behaviors of the rabbit. The employed phases are: soft besiege, hard besiege, soft besiege with progressive rapid dives, and hard besiege with progressive rapid dives. More details about the mathematical model can be found in the original paper [16].

B. The enhanced multi-swarm HHO

Despite the effectiveness of the HHO algorithm, it suffers like the most of metaheuristics from the premature convergence problem (i.e., the earlier convergence into local optima). To maintain the population diversity through the search process and to overcome the premature convergence problem, a multi-swarm approach has been introduced. The idea of this approach is based on dividing up the swarm into three smaller groups. Each group is guided by a high-quality solution called the leader. To maintain a balance between exploration and exploitation potentials, the top three agents (i.e., the fittest) obtained so far are selected as leaders.

Based on the idea that the fittest leader has the ability to guide more agents, the number of search agents in each swarm is calculated as in Eq.(1)

$$swarm_i = N \times \frac{1/f_i}{1/f_1 + 1/f_2 + 1/f_3} \tag{1}$$

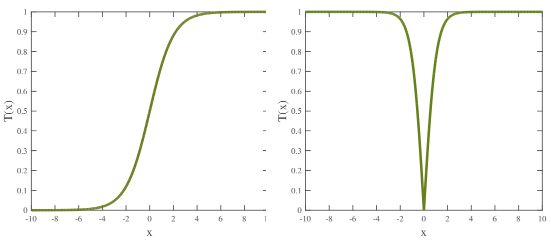
where $swarm_i$ (i=1,2,3) denotes the number of search agents in the i^{th} swarm, N represents the total number of search agents, f_i (i=1,2,3) represents the fitness value for the selected leaders (which are the fittest three agents obtained so far).

The process of distributing search agents into groups is done randomly. Then the agents in each swarm are manipulated according to the updating mechanisms in the original HHO. The multi-swarm with multi-leader strategy allows the search agents to explore various regions at once and thus avoid trapping early into sub-optimal solutions.

C. The proposed Binary HHO (BHHO)

Most of the well-known meta-heuristic algorithms including HHO have been originally designed to work on continuous search spaces. However, some real-world problems such as FS has a binary search space. For this purpose, these algorithms should be reformulated efficiently to work on binary spaces [17].

In this work, the Two-step binarization method was employed to introduce a binary version of HHO called (BHHO). In this technique, The Transfer Function (TF) is utilized in the first step to convert the solution from R^n to an intermediate solution, represents the probability of turning the solution to 1 or 0. In the second step, a binarization rule is used to map the output of TF into binary form.



(a) S-shaped (b) V-shaped

Fig. 2: The types of used TFs

There are two leading families of TFs used in literature: S-shaped (Sigmoid) and V-shaped (tanh) as shown in Figure 2. S-shaped TF was initially introduced by Kennedy and Eberhart [18] to propose a binary variant of Particle Swarm Optimization (PSO) based on Eq. 2 as a first step, and Eq. 3 as a binarization rule.

$$T(x_i^j(t)) = \frac{1}{1 + e^{-x_i^j(t)}} \tag{2}$$

where x_i^j is the real value corresponds to the j^{th} dimension of the i^{th} solution at iteration t , and $T(x_i^j(t))$ is the probability value resulted from TF.

$$x_i^j(t+1) = \begin{cases} 0 & \text{If } r < T(x_i^j(t)) \\ 1 & otherwise \end{cases} \tag{3}$$

where $X_i^j(t+1)$ is the binary value corresponds to the j^{th} dimension of the i^{th} solution, $T(x_i^j(t))$ is the probability value calculated by TF in Eq. (2), and r is a uniformly distributed random number inside (0,1) .

Rashedi et al. [19] was firstly introduced the V-shaped TF to propose a binary version of Gravitational Search Algorithm (BGSA). The two-step method using V-shaped was modeled using Eqs. 4 and 5.

$$T(x_i^j(t)) = |\tanh(x_i^j(t))| \tag{4}$$

$$x_i^j(t+1) = \begin{cases} \neg x_i^k(t) & r < T(x_i^j(t)) \\ x_i^j(t) & otherwise \end{cases} \tag{5}$$

where $\neg$ indicates the complement of current dimension, and $T(x_i)$ is the probability value provided by Eq. (4).

In this paper, two variants of Binary HHO denoted by (SBHHO) and (VBHHO) were presented by exploiting the aforementioned S-shaped and V-shaped TFs, respectively. These TFs showed a superior results in several previous works for combinatorial problems [20] [17].

D. BHHO-Based Feature Selection

In this research, we propose a wrapper-based FS that exploits the BHHO as a search strategy to enhance the prediction accuracy for SFP. Two major aspects should be considered when adapting metaheuristic algorithms to deal with an optimization problem: the representation (or encoding) of solutions and the guiding objective function.

- **Representation of solutions**: In FS problem, each feature is either selected or not, so the candidate solution is encoded as a one-dimensional vector of zeros and ones such that $X = \{x_1, x_2, x_3, ..., x_N\}$, where N is the total number of features.
- **Objective function**: It is useful to guide the search process by assigning each candidate solution with a real value represents its quality. The major desired objectives of FS method are to minimize the number of selected features that maximize the classification performance. In this work we used a single objective HHO algorithm, so the two contradictory objectives are combined in one fitness function as in Eq. 6

$$\downarrow Fitness = \alpha \times E + \beta \times \frac{|R|}{|N|} \tag{6}$$

where E is the classification error rate, $|R|$ is the number of selected features, $|N|$ is the total number of features in the handled dataset. α and β (the complement of α) are two user-defined parameters $\in [0,1]$ which are used to balance between the importance of both criteria [5].

IV. Experimental Results and Discussion

Fifteen well-regarded SFP datasets from PROMISE Software Engineering Repository are employed to study the effectiveness of the proposed model. These datasets are free of noise and missing values and having various sizes of 109–965 instances. Moreover, each dataset has 20 different object-oriented metrics (features) as input and a single fault value as an output variable. Table I presents a brief explanation for each dataset. By inspecting the table, it can be easily observed that all datasets are highly imbalanced, where the occurrences of the defective instances are very low when compared to the normal ones.The proposed model is evaluated using three common metrics; specificity, sensitivity, and area under the curve (AUC) [4]. The experimental work in this research was performed in four stages. In the first stage, we evaluated the sensitivity of BHHO to the number of employed search agents as well as the impact of oversampling ratio, with ADASYN, on the overall prediction quality. In this stage, the basic SBHHO (i.e., HHO with S-shaped TF) and KNN classifier were used. In the second stage, the SBHHO augmented with the ADASYN technique was assessed on three different classifiers (KNN, DT, and LDA) to select the suitable one that provides better prediction performance. In the third stage, The BHHO with V-shaped TF (VBHHO) is implemented and compared with SBHHO to investigate the impact of TF on the efficiency of BHHO. In the last stage, the best settings obtained in the previous stages were used, and the enhanced versions (ESBHHO and EVBHHO) were compared with the basic versions (SBHHO and VBHHO). In addition, their performance was confirmed by comparing with other state-of-the-art approaches. The experimental design conducted in this research is demonstrated in Table II. In all experiments, the classification model was trained and tested using the k-folds cross-validation method(where k=5). to perform a fair comparison, all algorithms in this research were evaluated using the same common parameters (150 iterations and 40 search agents). The other specific parameters were selected based on recommended settings in the original papers and related works on FS. The proposed approach is coded in

TABLE I: List of SFP datasets

Dataset	version	#features	#instances	#defective instances	%defective instances
ant	1.7	20	745	166	0.223
camel	1.0	20	339	13	0.038
	1.2	20	608	216	0.355
	1.4	20	872	145	0.166
	1.6	20	965	188	0.195
jedit	3.2	20	272	90	0.331
	4.0	20	306	75	0.245
	4.1	20	312	79	0.253
	4.2	20	367	48	0.131
	4.3	20	492	11	0.022
log4j	1.0	20	135	34	0.252
	1.1	20	109	37	0.339
	1.2	20	205	189	0.922
xalan	2.4	20	723	110	0.152
	2.7	20	909	898	0.988

TABLE II: The experimental design conducted to evaluate the performance of BHHO

Experiment	No. search agents	TF	Classifier	ADASYN ratio
exp1	**5, 10, 20, 30, 40, 50**	S_shaped	KNN	0
exp2	40	S_shaped	KNN	**0.2, 0.5, 0.8, 1**
exp3	40	S_shaped	**KNN, DT, LDA**	0.8
exp4	40	**S_shape V_shape**	LDA	0.8

MATLAB-R2017b, and tested on a PC with Intel Core i7-8550U, 2.2 GHz CPU, 8 GB RAM.

A. Sensitivity of BHHO to the number of search agents

In this part, Extensive experiments with different values of search agents were performed to identify the suitable number of agents for handling our problem. We concluded that SBHHO with 40 agents had obtained the best AUC results. Using a small number of agents causes the premature convergence problem due to the loss of diversity. Accordingly, the subsequent experiments will be conducted using 40 search agents.

B. Handling Imbalanced data using ADASYN

To confirm the effectiveness of the ADASYN technique, we conducted two experiments. In the beginning, the performance of SBHHO combined with ADASYN was investigated by employing different values of oversampling ratios After determining the suitable oversampling ratio, we deeply compared the overall performance of SBHHO on the original datasets (i.e., without ADASYN) and the re-balanced datasets (i.e., after applying ADASYN).

After conducting several experiments on all datasets, the obtained results showed that the oversampling ratio of 0.8 had achieved the best rank. This can be justified by the fact that the data is highly imbalanced, as the number of defective instances is much less than the number of the normal ones. Therefore, using a low oversampling ratio is not enough to re-balance the data. Accordingly, the balance ratio of 0.8 is adopted in the subsequent experiments.

Table III explores the comparison between SBHHO over imbalanced and balanced data in terms of specificity, sensitivity, and AUC rates. It is observed that applying ADASYN improves the ability to predict the instances with minority class (i.e., sensitivity), while it is decreased for those with majority class (i.e., specificity) for all datasets. This is because the learning model is biased to the majority class when using unbalanced data. To balance between the two different behaviors, we focus on the AUC as a performance evaluation metric. By Inspecting the results, it can be seen that the SBHHO with ADASYN is superior in 80% of cases. This proved the significant impact of ADASYN in improving prediction accuracy. Kindly note that all subsequent experiments are based on balanced data.

C. Impact of selected classifiers

Different types of classification algorithms are available in the literature. However, it is difficult to identify which one is the best. It depends on the nature of the used datasets. For this purpose, extensive experiments are performed in this part to select a suitable classifier. Three classifiers from different categories were used: KNN, DT, and LDA.

The obtained results by SBHHO with different classifiers show that LDA classifier declares superior results in 66.7 % of the datasets. As per F-test, LDA attained the best rank, followed by LNN and DT, respectively.

TABLE III: comparison of SBHHO before and after using ADASYN in terms of sensitivity, specificity, and AUC metrics

Dataset	Sensitivity		Specificity		AUC	
	original	ADASYN	original	ADASYN	original	ADASYN
ant-1.7	0.4877	**0.7386**	**0.8983**	0.7138	0.6930	**0.7262**
camel-1.0	0.0462	**0.5154**	**0.9957**	0.7957	0.5209	**0.6555**
camel-1.2	0.3602	**0.5755**	**0.8056**	0.6148	0.5829	**0.5951**
camel-1.4	0.2448	**0.5917**	**0.9530**	0.6873	0.5989	**0.6395**
camel-1.6	0.2128	**0.6011**	**0.9263**	0.6753	0.5695	**0.6382**
jedit-3.2	0.6533	**0.7500**	**0.8258**	0.7110	**0.7396**	0.7305
jedit-4.0	0.4920	**0.7120**	**0.8944**	0.7229	0.6932	**0.7175**
jedit-4.1	0.5418	**0.7646**	**0.8884**	0.7558	0.7151	**0.7602**
jedit-4.2	0.3646	**0.7750**	**0.9433**	0.7643	0.6539	**0.7696**
jedit-4.3	0.0545	**0.5091**	**0.9850**	0.8191	0.5198	**0.6641**
log4j-1.0	0.5912	**0.7500**	**0.9485**	0.7446	**0.7698**	0.7473
log4j-1.1	0.7054	**0.7514**	**0.9194**	0.8236	**0.8124**	0.7875
log4j-1.2	**0.9857**	0.7265	0.3875	**0.7063**	0.6866	**0.7164**
xalan-2.4	0.2509	**0.7409**	**0.9519**	0.7039	0.6014	**0.7224**
xalan-2.7	**0.9987**	0.9855	0.7091	**0.8091**	0.8539	**0.8973**
W—L	2—13	**13—2**	**13—2**	2—13	3—12	**12—3**

D. Comparison of BHHO variants

In the current part, the efficiency of the proposed multi-swarm variants of HHO (ESBHHO, and EVBHHO) are appraised and compared with the basic variants (SBHHO AND VBHHO) by inspecting the AUC rates, and size of the selected feature.

By observing the reported results in Table IV, it can bee seen that the EVBHHO achieves the higher values of the prediction accuracy in 80 % of the datasets, whereas ESBHHO obtained the best results on 20 % of datasets. As per the number of selected features, EVBHHO scores the best results on 46% of cases followed by VBHHO, which gives the best values on 40% of cases, while both show no superiority on each other on 14% of cases. Besides, the integration between the BHHO and V-shaped TF outperforms the S-shaped TF in terms of AUC and the number of selected features.

TABLE IV: Comparison between BHHO and the enhanced variant EBHHO using S-shaped and V-shaped TFs in terms of AUC and number of selected features.

Dataset	AUC				No. of features			
	SBHHO	VBHHO	ESBHHO	EVBHHO	SBHHO	VBHHO	ESBHHO	EVBHHO
ant-1.7	0.7471	0.7541	0.7683	**0.7727**	11.8	7.7	11.5	**6.9**
camel-1.0	0.7044	0.7549	0.8013	**0.8107**	8.8	**5.9**	7	**5.9**
camel-1.2	0.6174	0.6217	0.6464	**0.6467**	14.1	9.7	13.8	**9.4**
camel-1.4	0.6745	0.6776	**0.7051**	0.7029	12.2	**9.7**	11.9	9.8
camel-1.6	0.6549	0.6532	0.6731	**0.6762**	14.1	10.3	12.8	**9.8**
jedit-3.2	0.7958	0.7994	**0.8271**	0.8270	13.5	**9.3**	13.6	10.4
jedit-4.0	0.7149	0.7308	0.7602	**0.7661**	11.4	7.1	10	**6.4**
jedit-4.1	0.7715	0.7775	0.8071	**0.8133**	11.8	**7.8**	11.1	8.1
jedit-4.2	0.7808	0.7901	0.8256	**0.8290**	10.4	7	12.5	**6.5**
jedit-4.3	0.5907	0.6407	0.8053	**0.8081**	11.1	**7.1**	12.9	8.2
log4j-1.0	0.7786	0.7766	0.8214	**0.8297**	7.8	5.1	8.1	**4.9**
log4j-1.1	0.7495	0.7762	0.8337	**0.8395**	12.6	**4**	10.9	5.2
log4j-1.2	0.6606	0.6837	**0.7767**	0.7761	11.8	7.3	11.1	**5.9**
xalan-2.4	0.7149	0.7220	0.7467	**0.7521**	10.2	**6.4**	10.2	6.7
xalan-2.7	0.8699	0.8633	0.9006	**0.9269**	6.9	**2.7**	6.8	**2.7**
W—T—L	3.80	3.20	1.80	**1.20**	3.70	1.53	3.30	**1.47**

E. Comparison with well-known algorithms

In this part, we will confirm the effectiveness of the proposed ESBHHO and EVBHHO versions by comparing them with well-established algorithms including BGSA, BGOA,bWOA, BB, GA, and BALO. For this purpose, all algorithms were implemented and tested on the same proposed model using the same common configurations.

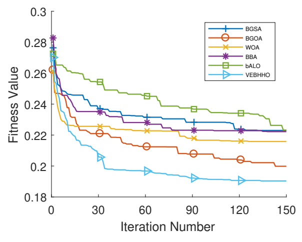
Based on the reported AUC results in Table V, it can be recognized that EVBHHO outperforms other algorithms in about 60% of datasets. Based on the F-Test ranking, the EVBHHO ranked first, followed by BGOA, ESBHHO, BALO, BWOA, BBA, BGSA, and GA, respectively.

The excellent performance of the proposed multi-swarm HHO over other algorithms can be justified by its ability to explore more several regions at once. Moreover, the HHO algorithm employes six updating mechanisms to balance between the exploration and exploitation behaviors, which improves the ability to perform well with various problems.

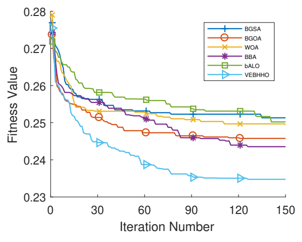
TABLE V: Comparison with other well-known algorithms in term of AUC results

Dataset	ESBHHO	EVBHHO	BGSA	BGOA	WOA	BBA	GA2	bALO
ant-1.7	0.7683	**0.7727**	0.7624	0.7710	0.7671	0.7644	0.7612	0.7665
camel-1.0	0.8013	**0.8107**	0.7792	0.8028	0.7836	0.7794	0.7538	0.7821
camel-1.2	0.6464	**0.6467**	0.6368	0.6423	0.6336	0.6362	0.6280	0.6461
camel-1.4	0.7051	0.7029	0.7009	**0.7098**	0.6928	0.7030	0.6863	0.7067
camel-1.6	0.6731	0.6762	0.6648	0.6788	0.6566	0.6597	0.6586	**0.6813**
jedit-3.2	0.8271	0.8270	0.8202	0.8312	0.8146	0.8173	0.8105	**0.8337**
jedit-4.0	0.7602	**0.7661**	0.7510	0.7583	0.7504	0.7589	0.7404	0.7553
jedit-4.1	0.8071	0.8133	0.7996	**0.8139**	0.8008	0.7984	0.7893	0.8080
jedit-4.2	0.8256	**0.8290**	0.8139	0.8279	0.8177	0.8159	0.8090	0.8176
jedit-4.3	0.8053	**0.8081**	0.7690	0.7911	0.7303	0.7590	0.7458	0.7819
log4j-1.0	0.8214	**0.8297**	0.8174	0.8233	0.8194	0.8172	0.7963	0.8110
log4j-1.1	0.8337	0.8395	0.8330	**0.8413**	0.8346	0.8331	0.8018	0.8344
log4j-1.2	**0.7767**	0.7761	0.7590	0.7724	0.7469	0.7568	0.7359	0.7691
xalan-2.4	0.7467	**0.7521**	0.7364	0.7476	0.7432	0.7391	0.7303	0.7425
xalan-2.7	0.9006	**0.9269**	0.8651	0.9021	0.9171	0.8746	0.8889	0.8893
Rank (F-test)	3.00	**1.80**	6.07	2.27	5.40	5.80	7.73	3.93

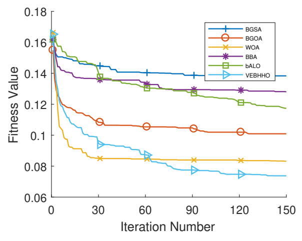
The convergence behaviors for selected datasets are demonstrated in Fig. 3. It can be observed that EVBHHO achieves faster acceleration rate of convergence to the best solution.



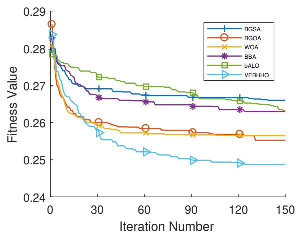
(a) camel-1.0



(b) jedit-4.0



(c) xalan-2.7



(d) xalan-2.4

Fig. 3: Convergence curves behavior on selected datasets

V. Conclusion and Future Directions

In this paper, a novel feature selection method that is based on an enhanced binary HHO algorithm was proposed to enhance the perfromance of a learning algorithm for SFP problem. Moroever, the ADYSON technique was used to rebalance the datasets. Three classifiers were used with the objective function to evaluate the selected features by the optimization algorithm. Fifteen datasets in the field of SFP were used to assess the performance of the proposed approach. The obtained results indicated that the enhanced HHO approach augmented with ADASYN method recorded the best results among all other algorithms when used with LDA classfier.

The future directions are related to the following: first, exploring the efficiency of other binarization techniques as well as other versions of TFs. Second, different classifiers can be tested with the proposed model, such as Artificial Neural Networks.

References

[1] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009, vol. 74.

[2] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to algorithms*. The MIT Press, 2009, vol. 3rd edition.

[3] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*. Springer Science & Business Media, 2006, vol. 57.

[4] T. Thaher, M. Mafarja, B. Abdalhaq, and H. Chantar, "Wrapper-based feature selection for imbalanced data using binary queuing search algorithm," in *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, Oct 2019, pp. 1–6.

[5] M. Mafarja and S. Mirjalili, "Whale optimization approaches for wrapper feature selection," *Applied Soft Computing*, vol. 62, pp. 441–453, 2017.

[6] M. Mafarja, I. Aljarah, A. A. Heidari, H. Faris, P. Fournier-Viger, X. Li, and S. Mirjalili, "Binary dragonfly optimization for feature selection using time-varying transfer functions," *Knowledge-Based Systems*, vol. 161, pp. 185–204, 2018.

[7] V. Kumar and S. Minz, "Feature selection: A literature review," *Smart Computing Review*, vol. 4, pp. 211–229, 01 2014.

[8] H. He and E. Garcia, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, pp. 1263 – 1284, 10 2009.

[9] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1872–1879, 2015.

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

[11] 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*, pp. 1–1, 2020.

[12] E. Erturk and E. A. Sezer, "Iterative software fault prediction with a hybrid approach," *Applied Soft Computing*, vol. 49, pp. 1020–1033, 2016.

[13] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Computing*, vol. 22, no. 4, pp. 9847–9863, Jul 2019.

[14] W. Rhmann, B. Pandey, G. Ansari, and D. Pandey, "Software fault prediction based on change metrics using hybrid algorithms: An empirical study," *Journal of King Saud University - Computer and Information Sciences*, 2019.

[15] R. Jayanthi and L. Florence, "Software defect prediction techniques using metrics based on neural network classifier," *Cluster Computing*, vol. 22, no. 1, pp. 77–88, Jan 2019.

[16] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849 – 872, 2019.

[17] B. Crawford, R. Soto, G. Astorga, J. García, C. Castro, and F. Paredes, "Putting continuous metaheuristics to work in binary search spaces," *Complexity*, vol. 2017, 2017.

[18] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5. IEEE, 1997, pp. 4104–4108.

[19] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Bgsa: binary gravitational search algorithm," *Natural Computing*, vol. 9, no. 3, pp. 727–745, 2010.

[20] T. Thaher, A. A. Heidari, M. Mafarja, J. Dong, and S. Mirjalili, *Binary Harris Hawks Optimizer for High-Dimensional, Low Sample Size Feature Selection*, 01 2020, pp. 251–272.