Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.061532

ARTICLE

A Feature Selection Method for Software Defect Prediction Based on Improved Beluga Whale Optimization Algorithm

Shaoming Qiu, Jingjie He, Yan Wang* and Bicong E

School of Information Engineering, Dalian University, Dalian, 116622, China

*Corresponding Author: Yan Wang. Email: wy@dlu.edu.cn

Received: 26 November 2024; Accepted: 21 February 2025; Published: 19 May 2025

ABSTRACT: Software defect prediction (SDP) aims to find a reliable method to predict defects in specific software projects and help software engineers allocate limited resources to release high-quality software products. Software defect prediction can be effectively performed using traditional features, but there are some redundant or irrelevant features in them (the presence or absence of this feature has little effect on the prediction results). These problems can be solved using feature selection. However, existing feature selection methods have shortcomings such as insignificant dimensionality reduction effect and low classification accuracy of the selected optimal feature subset. In order to reduce the impact of these shortcomings, this paper proposes a new feature selection method Cubic Traverse Ma Beluga whale optimization algorithm (CTMBWO) based on the improved Beluga whale optimization algorithm (BWO). The goal of this study is to determine how well the CTMBWO can extract the features that are most important for correctly predicting software defects, improve the accuracy of fault prediction, reduce the number of the selected feature and mitigate the risk of overfitting, thereby achieving more efficient resource utilization and better distribution of test workload. The CTMBWO comprises three main stages: preprocessing the dataset, selecting relevant features, and evaluating the classification performance of the model. The novel feature selection method can effectively improve the performance of SDP. This study performs experiments on two software defect datasets (PROMISE, NASA) and shows the method's classification performance using four detailed evaluation metrics, Accuracy, F1-score, MCC, AUC and Recall. The results indicate that the approach presented in this paper achieves outstanding classification performance on both datasets and has significant improvement over the baseline models.

KEYWORDS: Software defect prediction; feature selection; beluga optimization algorithm; triangular wandering strategy; cauchy mutation; reverse learning

1 Introduction

Software quality assurance has become a critical challenge in software engineering as the size and complexity of software systems continue to grow. Software defects can not only compromise reliability but also lead to high maintenance costs or even severe system failures. Therefore, SDP has received widespread attention as an effective means of prevention and detection. By analyzing historical code and project data to build a SDP model, potential defect-prone modules can be identified early in the software development process, thereby reducing development risks and improving development efficiency [1]. Previous experiments [2] have predicted the defect tendency of software modules by analyzing various metrics in software source code. These metrics reflect the complexity of the code and related information in the development process, but too many metrics may lead to poor quality of the prediction model. This is because there will be redundant metrics that are irrelevant to defect prediction. Using feature selection can avoid this

problem. Feature selection is crucial in SDP. It aims to eliminate irrelevant or redundant features, thereby enhancing model performance, improving training efficiency, and avoiding overfitting [3,4]. In SDP, we may use multiple features such as Coupling Between Objects (CBO), Depth of Inheritance Tree (DIT), Lack of Cohesion in Methods (LCOM), Number of Children (NOC), Response For a Class (RFC), and Weighted Methods per Class (WMC). These features reflect different complexities and design characteristics of classes, but some of them may be redundant or irrelevant. For example, CBO and RFC are often highly correlated because class coupling and response set size are closely related. In contrast, DIT may show a weaker correlation with defect occurrence, making it an irrelevant feature. Feature selection helps boost the prediction accuracy of SDP and speeds up training by eliminating unnecessary or irrelevant features. Therefore, feature selection helps the model focus on truly useful features, enhancing its generalization ability.

Goyal et al. [5] proposed to use lion optimization algorithm for feature selection to enhance the performance of SDP. Malhotra et al. [6] introduced a SDP model that uses a two-phase grey wolf optimization for feature selection, and Chantar et al. [7] proposed using an enhanced binary grey wolf optimizer for text classification. The performance of the population-based metaheuristic algorithm is impacted by initialization and parameter control. In particular, if the population position is not initialized with sufficient randomness and the search space may not be fully explored, which means not all features will be traversed and redundant or irrelevant features will not be effectively removed. Goyal [8] proposed a genetic evolution algorithm for feature selection. Genetic algorithms involve several parameters, including population size, crossover rate, and mutation rate, etc. The choice of these parameters significantly affects the final performance of the algorithm. If these parameters are set unreasonably, the algorithm may converge too slow, thus affecting the final performance of defect prediction. To solve these problems, this study introduces a novel feature selection method CTMBWO based on BWO. As far as we know, BWO has not been used for feature selection in the SDP field. This method generates the initial individual population through Cubic chaotic mapping, which is more random and uniform and can effectively avoid falling into the local optimal solution. And an update mechanism based on the combination of Cauchy mutation and reverse learning is used to effectively boost the diversity and the speed of global convergence of the population. When the current population is not optimal, the triangle wandering strategy is used to update the selected population. At the same time, the algorithm uses a small number of parameters.

The contributions of this paper are as follows:

(1) In the initialization stage of CTMBWO, the Cubic chaotic map is introduced to generate a more evenly distributed initial population and strengthen the algorithm's global search ability.
(2) In the exploration phase of CTMBWO, the triangle walk strategy is used to further expand the search range of the algorithm and improve the search accuracy.
(3) During the whale fall phase of CTMBWO, an update mechanism based on the combination of Cauchy mutation and reverse learning is used to effectively enhance the distribution and the speed of global convergence of the population. In addition, a sparrow alert mechanism is also integrated to further ensure that the beluga can avoid getting stuck in the local optimal solution.

2 Related Work

Feature selection is crucial in SDP that aims to select the feature subset that has the greatest impact on the prediction results from a lot of features. The dataset may contain a large number of irrelevant and redundant features in SDP, and the excessive number of features may increase the time cost of training the classifier and may reduce the performance of the classifier. When dealing with high-dimensional features, feature selection methods are currently widely used. Feature selection methods are divided into filtering method, wrapping method, and ensemble learning method.

(1) Filtering method

The filtering method is an unsupervised feature selection method that scores features by evaluating the correlation or divergence of each feature with the target variable individually, without relying on a specific model. This method does not require model training, so it is computationally efficient and suitable for processing large-scale data sets. Based on the scoring results, the filtering method selects features based on a set threshold or the top N highest-scoring features. Some commonly used filtering methods are Chi-Square (CS), Information Gain (IG), Relief method, etc.

(2) Wrappering method

The wrappering method closely combines feature selection with the model training process and systematically assessing the performance of various feature subsets to identify the most effective ones. The core idea of the wrappering method is to use a specific machine learning algorithmto train the model and evaluate the pros and cons of each feature subset based on the model's predictive performance. This method can obtain a better feature set, especially when the feature dimension is high. Commonly used methods include forward selection and exhaustive search.

(3) Ensemble learning method

The ensemble learning method is a combination of filtering and wrapping methods. Features are selected by comprehensively selecting the results of multiple filtering or wrapping methods, or the output of the filtering method is used as the input of the wrapping method to screen features. The ensemble method combines the advantages of filtering and wrapping methods, and often produces better prediction performance than models based on single filtering or wrapping methods.

Population-based metaheuristic algorithms are often used for feature selection, especially in high-dimensional data or complex problems. They can explore the feature space by simulating the natural evolution process and select the most advantageous feature subset.

Das et al. [9] introduced a cutting-edge feature selection method known as the Golden Jackal Optimization (GJO) algorithm, a metaheuristic inspired by the hunting strategies of the golden jackal. The method can be further applied to multi-objective optimization problems by using feature selection. The researchers integrated this algorithm with four classification models to extract significant feature subsets from SDP datasets.

Arasteh et al. [10] proposed an innovative feature selection method by changing the Binary Chaos-based Olympiad Optimization Algorithm. Using this method to get the features that have the greatest impact on the prediction results. Integrating the method with classification models can greatly improve the precision and accuracy of software module classification.

Kukkar et al. [11] proposed a feature extraction technique by improving the ant colony optimization (ACO) to find out more relevant features for SDP. At the same time, the algorithm was integrated with machine learning to boost prediction performance. The results demonstrated a significant improvement in accuracy compared to the baseline methods.

Anbu et al. [12] used the firefly algorithm for feature selection. Fireflies tend to fly towards brighter areas, and their flight direction and distance are influenced by two important factors: brightness and attractiveness. These factors are used to guide the selection of solutions, with fireflies attracted to one another based on brightness differences, ultimately searching for the global optimal solution. Through this process, a representative feature subset is selected.

Wang et al. [13] proposed a binary adaptation of the Gray Wolf Optimizer algorithm aiming at identifying the most impactful features within the dataset to solve the problems of excessive feature volume

in the training dataset. Through feature selection, the features that have the greatest impact on the SDP are selected. These features include the number of lines of code and complexity, etc. At the same time, experiments show that accuracy, Recall, and F1 have been significantly improved.

Sekaran et al. [14] proposed a new feature selection method that combines mutation-enhanced Salp Swarm Optimizer (MBSSO) with rough set theory. Rough set theory offers a structured approach to examining the relationships and dependencies among features. It refines the search space by assessing fitness scores and incorporates a mutation enhancement strategy to evade getting stuck at local optima. SDP is performed after feature selection using kernel extreme learning machine.

Alsghaier et al. [15] proposed combining genetic algorithm (GA) with support vector machine (SVM) classifier and particle swarm algorithm (PSO) to obtain better SDP performance. The experimental results show that when applied to limited scale datasets, integrating GA with SVM and PSO can obtain good SDP results and overcome the limitations of previous studies.

Inspired by the above techniques, this paper also uses a population-based metaheuristic algorithm CTMBWO for feature selection. The algorithm uses chaotic mapping [16] to change the initialization method of the population, making the distribution of the population more uniform and random. At the same time, mutation operations are performed during each iteration to find the optimal population and optimal feature subset faster, thereby improving the SDP performance.

3 Proposed Methodology

3.1 Beluga Whale Optimization Algorithm

Beluga whale optimization algorithm (BWO) was proposed by Zhong et al. in 2022 [17], which is a heuristic algorithm based on the beluga's lifestyle, as shown in Fig. 1. The algorithm simulates the beluga's swimming behavior (a), praying behavior (b), and death behavior (c) and they are modeled as the exploration, exploitation, and whale-fall phases.

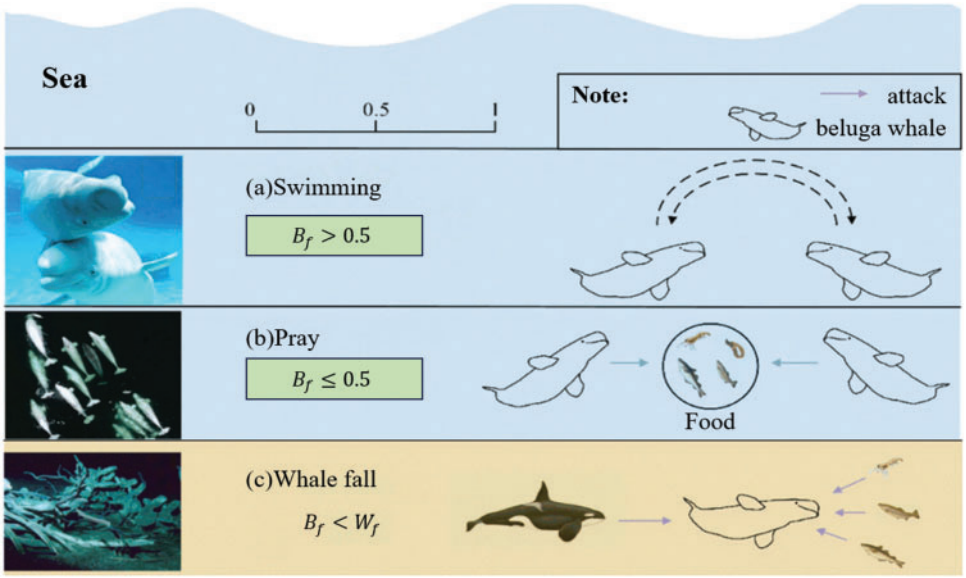

Figure 1: The lifestyle of Beluga, (a) Swimming behavior; (b) Pray behavior; (c) Whale fall behavior

In the feature selection task, BWO searches for the optimal feature subset by representing each feature as a binary vector and simulating the strategy of beluga whales swimming around their prey. The fitness of each beluga whale is calculated and evaluate their performance. The algorithm updates the position of the beluga whale based on the fitness value and gradually selects the most useful features for SDP.

By comparing the balance factor B_f, the BWO decides whether to enter the exploration or exploitation phase. When $B_f > 0.5$ the algorithm is in the exploration phase, and when $B_f \leq 0.5$, it shifts to the exploitation phase. The probability W_f and B_f together determine whether the whale fall phase occurs and it happens when $B_f < W_f$. The mathematical model of B_f and W_f are:

$$B_f = B_0(1 - T/2T_{\max}) \tag{1}$$

$$W_f = 0.1 - 0.05T/T_{\max} \tag{2}$$

where T is the iteration number currently being executed, $T_{\max}$ is the maximum iterative number, B_0 is a random number between $(0, 1)$.

3.1.1 Initialization Phase

In the initialization stage, BWO randomly generates a population of beluga whales and calculates the fitness value corresponding to each population. It determines whether the current population is optimal by comparing the fitness value. Using the matrix X to represent the obtained population.

3.1.2 Exploration Phase

When $B_f > 0.5$, the algorithm enters the exploration phase. In this phase, the beluga whales swim randomly in all directions to locate the global optimal solution and avoid getting trapped in local optima. The updated position of the beluga whale is expressed as:

$$\begin{cases} X_{i,j}^{T+1} = X_{i,p_j}^{T} + (X_{r,p_1}^{T} - X_{i,p_j}^{T})(1 + r_1)\sin(2\pi r_2) & j = even \\ X_{i,j}^{T+1} = X_{i,p_j}^{T} + (X_{r,p_1}^{T} - X_{i,p_j}^{T})(1 + r_1)\cos(2\pi r_2) & j = odd \end{cases} \tag{3}$$

where T is the current iteration, $X_{i,j}^{T+1}$ is the new position of the ith beluga whale on the jth dimension, $p_j(j = 1, 2, \cdots, d)$ is a randomly chosen dimension from the d-dimension, X_{i,p_j}^{T} is the position of the ith beluga whale on p_j dimension, X_{r,p_1}^{T} is the current positions of the rth beluga whale (r is the beluga whale chosen at random), r_1 and r_2 are random number between (0,1), $\sin(2\pi r_2)$ and $\cos(2\pi r_2)$ denote the fin orientation of the mirrored beluga, with odd and even choices.

3.1.3 Exploitation Phase

The exploitation phase happens when $B_f \leq 0.5$. In this stage beluga whale uses the Levy flight strategy [18] to enhance convergence to approach the target solution more accurately by hunting prey.

3.1.4 Whale Fall

The whale fall happens when $B_f < W_f$. In this stage the beluga whale abandons inferior prey or focuses on catching high-quality prey and further converging to the optimal solution. This stage ensures that the algorithm focuses on optimizing the current best solution, thereby improving the accuracy and reliability of the final result.

3.2 Boosted Beluga Whale Optimization Algorithm CTMBWO

As the feature dimension increases, the feature selection method based on BWO will easily fall into local optimal solutions and uneven population distribution that makes it difficult to choose a feature subset that is both highly accurate in classification and has a limited number of features. In BWO, the distribution of its population is shown in Fig. 2a. Most of the populations are distributed in the lower left corner and the distribution is not uniform. Therefore, when selecting features, processing these densely distributed populations in the lower left corner increases the possibility of obtaining a local optimal solution. The population distribution of CTMBWO is shown in Fig. 2b, which shows that it is evenly distributed. This ensures that all populations will be calculated, and the solution obtained is also the global optimal solution rather than the local optimal solution.

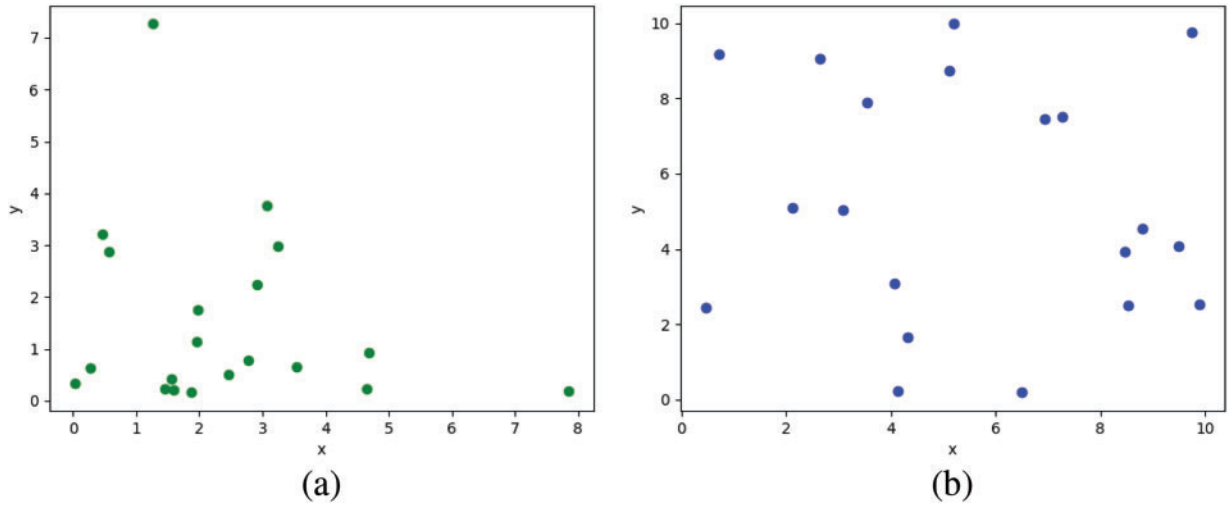

Figure 2: Comparison of population distribution between BWO and CTMBWO, (a) The population distribution of BWO, (b) The population distribution of CTMBWO

3.2.1 Cubic Chaos Mapping

A diverse population will have a significant impact on the predicted results. However, the way BWO generates populations often leads to an uneven distribution, which in turn reduces diversity. To improve the global search capability, it is essential for the beluga whale population to be evenly distributed across the entire search space. Chaotic mapping [16] is known for its randomness, regularity and ergodicity, which can help achieve even distribution and increase population diversity. Therefore, this paper introduces a chaotic iterative mapping based on the Cubic, the formula is as follows:

$$x_{n+1} = \rho x_n (1 - x_n^2) \tag{4}$$

where x_n denotes the position of the nth population, ρ denotes the mixing factor and the Cubic mapping has better chaotic traversal when $x_0 = 0.3$, $\beta = 0.259$. By using Eq. (4), the new population initialization is obtained as:

$$X' = \begin{pmatrix} x'_{1,1} & x'_{1,2} & \cdots & x'_{1,d} \\ x'_{2,1} & x'_{2,2} & \cdots & x'_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x'_{n,1} & x'_{n,2} & \cdots & x'_{n,d} \end{pmatrix} \tag{5}$$

and calculate the fitness value:

$$F_{fv} = \begin{pmatrix} f(x'_{1,1}, x'_{1,2}, \cdots, x'_{1,d}) \\ f(x'_{2,1}, x'_{2,2}, \cdots, x'_{2,d}) \\ \vdots \\ f(x'_{n,1}, x'_{n,2}, \cdots, x'_{n,d}) \end{pmatrix} \tag{6}$$

where n is the size of the beluga whale population, d is the number of dimensions associated with the design variable.

3.2.2 Triangle Wandering Strategy

(1) The distance dis between the beluga whale and its conspecifics is expressed as:

$$dis = X_{best}^{T} - X_{i}^{T} \tag{7}$$

where X_i^T is the current position for the beluga whale, X_{best}^T is the best position for the beluga whale at iteration T.

(2) The range of the beluga whale's walking step length $range$, is expressed as:

$$range = \alpha \cdot dis_1 \tag{8}$$

where α is a normally distributed random number.

(3) Define the direction β of the beluga whale's movement and express it as:

$$\beta = 2\pi\alpha \tag{9}$$

where α is a normally distributed random number.

(4) Calculate the final distance $Dist$ between the wandering position and the same type by using Eqs. (7) and (9):

$$Dist = dis^2 + range^2 - 2 \cdot dis \cdot range \cdot \cos(\beta) \tag{10}$$

(5) The updated position is expressed as:

$$\begin{cases} X_{i,j}^{T+1} = X_{i,p_j}^{T} + (X_{r,p_1}^{T} - X_{i,p_j}^{T})(1 + r_1)\sin(2\pi r_2) + Dist \cdot rand(0,1) & j = even \\ X_{i,j}^{T+1} = X_{i,p_j}^{T} + (X_{r,p_1}^{T} - X_{i,p_j}^{T})(1 + r_1)\cos(2\pi r_2) + Dist \cdot rand(0,1) & j = odd \end{cases} \tag{11}$$

where T is the current iteration, $X_{i,j}^{T+1}$ is the new position for the ith beluga whale on the jth dimension, $p_j(j = 1, 2, \cdots, d)$ is a randomly chosen dimension from d-dimension, X_{i,p_j}^T is the position of the ith beluga whale on p_j dimension, X_{r,p_1}^T is the current positions of the rth beluga whale (r is the beluga whale chosen at random), r_1 and r_2 are random number between (0, 1), $\sin(2\pi r_2)$ and $\cos(2\pi r_2)$ denote the fin orientation of the mirrored beluga, with odd and even choices.

3.2.3 Cauchy Mutation Combined with Reverse Learning

During the whale fall phase, Cauchy mutation and reverse learning strategies are used to update the beluga whale's position to avoid getting stuck into local optima.

In BWO, the target position is updated based on the position change after each iteration and the fitness value is recalculated so the new position replaces the target position. However, the target position remains unaffected which may lead to the algorithm trapped by local optimal solution. Therefore, a strategy of combining Cauchy mutation and reverse learning is proposed to randomly update the target position based on a random probability P to prevent the algorithm from converging to the local optimal solution. The mathematical model of the whale fall stage is:

$$X_i^{T+1} = r_5 X_i^T - r_6 X_r^T + r_7 X_{step} + X_{new}^{T+1} \tag{12}$$

$$P_s = -\exp\left(1 - \frac{T}{T_{\max}}\right)^{10} + \omega \tag{13}$$

$$X_{best}^{T^*} = u_b + r(l_b - X_{best}^T) \tag{14}$$

$$\begin{cases} X_{new}^{T+1} = b_3(X_{new}^T - X_{best}^{T^*}) \\ X_{new}^{T+1} = Cauchy(0,1)X_{best}^T \end{cases} \tag{15}$$

where r_5, r_6, r_7, r are random numbers from 0 to 1, X_{step} is the movement size during the whale fall stage, $X_{step} = (u_b - l_b)\exp(-C_2 T/T_{\max})$ where C_2 is the scaling factor for the movement size, $C_2 = 2W_f \times n$, u_b and l_b are the maximum and minimum limits, ω is the adjustment factor, and the optimal result of the function is obtained when ω is equal to 0.05, $X_{best}^{T^*}$ is the reverse solution of the generation T target solution X_{best}^T, X_{new}^{T+1} is the generation $T+1$ target solution, $Cauchy(0,1)$ is the standard Cauchy distribution function, b_3 is the pseudo-information exchange coefficient, $b_3 = \alpha e^{(-\beta T)/T_{\max}}$, α is the random initial value, β is the decay rate, T is the iteration number currently being executed, $T_{\max}$ is the maximum iterative number.

3.2.4 Sparrow Alert Mechanism

In the whale fall phase, the failed individuals can share useful information with other individuals by simulating the behavior of alert individuals in the sparrow alert mechanism, thereby transmitting environmental risks. This information may include the search space situation around the failed individuals and help other individuals avoid being confined to the same solution or further optimize their own solutions. The formula is:

$$X_i^{T+1} = X_i^T + \alpha(X_{new}^{T+1} - X_i^T) + \beta R \tag{16}$$

where X_i^{T+1} is the position of the current individual i in generation $T+1$, X_{new}^{T+1} is the target solution of generation $T+1$, α is a scaling factor to control the direction of the individual's movement, β is a scaling factor to control the scope of the randomized search and R is a random vector, which is used to guide the individual to perform the randomized search and increase the stochasticity and diversity of the exploration.

3.2.5 Summary of CTMBWO Algorithm

Fig. 3 is the algorithm flow chart of CTMBWO.

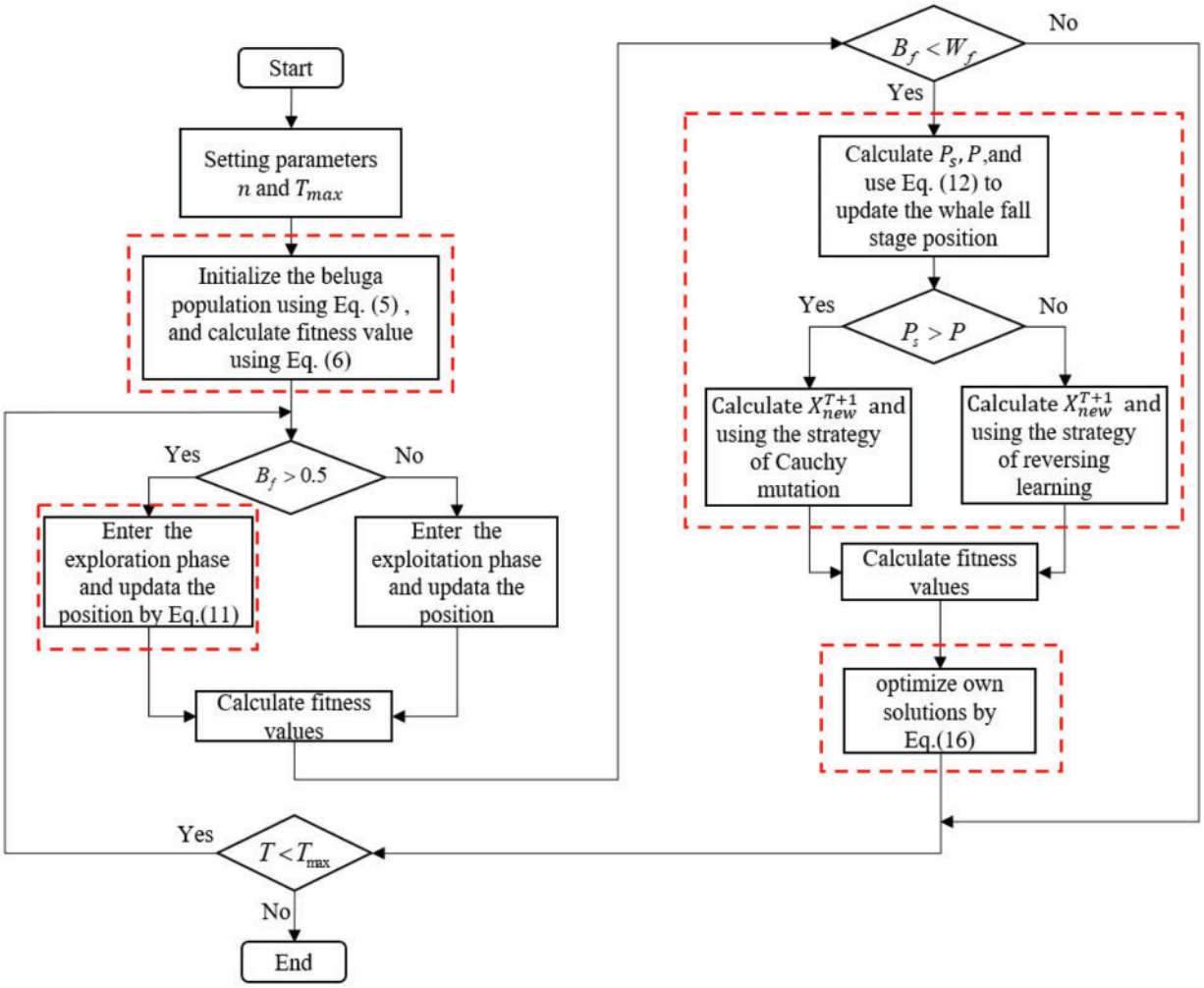

Figure 3: The algorithm flow chart of CTMBWO

The specific algorithm of CTMBWO is as shown in Algorithm 1.

Algorithm 1: CTMBWO

Input: Objective function $f(x)$, Beluga whale population size$N$, Maximum number of iterations T_{max}.
1: Initialize the beluga population and calculate individual fitness value using Eqs. (5) and (6)
2: while($T < T_{max}$)
3: Calculate B_f
4: if($B_f > 0.5$)
5: Enter the exploration phase and update the position using Eq. (11)
6: else
7: Enter the exploitation phase and update the position
8: if($B_f < W_f$)
9: Calculate$P$, P_s, C_2, X_{step} and updata the position using Eq. (12)
10: if($P_s > P$)
11: Calculate X_{new}^{T+1} using the strategy of Cauchy mutation

(Continued)

Algorithm 1 (continued)

12: else
13: Calculate X_{new}^{T+1} using the strategy of reversing learning
14: Using Eq. (16) to help other individuals optimize their own solutions.
15: $T = T + 1$
16: else
17: $T = T + 1$
18: end while
Output: the global optimal solution

4 Experimental Design

4.1 Experimental Condition

The experiments in this paper are conducted using Python 3.9 on 64-bit Windows 10 operating system, with Pycharm and an i7-10700F CPU.

4.2 Datasets

The datasets used in this paper are 19 projects from NASA and PROMISE. The datasets details are shown in Table 1.

Table 1: Detail of the datasets

Dataset	Version	#File	Features	#Defective file	%Defective file
ant	1.4, 1.5, 1.6, 1.7	1567	20	330	21.1
camel	1.2, 1.4, 1.6	2445	20	549	22.5
jedit	3.2, 4.0, 4.1, 4.2	1257	20	292	23.2
log4j	1.0, 1.1, 1.2	449	20	256	57
lucene	2.0, 2.2, 2.4	782	20	438	56
poi	1.5, 2.0, 2.5, 3.0	1378	20	707	51.3
synapse	1.0	157	20	16	10.2
velocity	1.4, 1.5, 1.6	639	20	367	57.4
xalan	2.4, 2.5, 2.6	2416	20	908	37.6
xerces	1.3	453	20	69	15.2
CM1	–	505	38	48	9.5
KC1	–	2107	22	325	15.4
KC3	–	458	444	43	9.3
MC1	–	9466	39	68	0.7
MW1	–	403	38	31	7.7
PC1	–	1107	38	76	6.9
PC2	–	5589	37	23	0.4
PC3	–	1563	38	160	10.2
PC4	–	1458	38	178	12.2
PC5	–	17186	39	516	3

By further observing the specific feature values of each project in Table 1, it can be found that the feature value distribution in the data set has a large range of differences. For example, in project ant-1.5, the value of WMC is 9, the value of DIT is 6, the value of LCOM3 is 0.8125, and the value of AMC is 17.2222. This difference in numerical range will affect the model's processing of features. To address this problem we use Min-Max normalization [19] that scales the data and ensures that all numbers have the same range, thereby eliminating the numerical differences between different features. The formula is:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{17}$$

where $X_{\max}$ and $X_{\min}$ are the maximum and minimum values of the features. According to the Pareto principle, about 80% of software defects occur in about 20% of software modules [20]. And the sample size for particular categories in the dataset that we use is far less than that in other categories. To effectively address this problem, we adopted the Synthetic Minority Over-sampling Technique (SMOTE) [21], which generates synthetic samples by oversampling the minority class to balance the class distribution.

To assess the stability and generalization capability of the proposed method, we employ 5-fold cross-validation. The dataset is split evenly into 5 subsets, with 4 subsets used for training in each iteration and the remaining one used for testing. This process is repeated 5 times, and the final result is the average of all 5 runs. By doing so, each subset is used as a test set at least once, enhancing the model's overall reliability and robustness. At the same time, we employ a strategy combining cross-validation and early stopping to further prevent overfitting.

4.3 Baseline Model

The benchmark models used are four well-known models GAPSO [15], Fed-OLF [22], MFWFS [23], and SLSTM [24].

4.4 Evaluate Metrics

We evaluate the prediction model's performance using widely recognized metrics, accuracy, precision, recall, F1, and the area under the curve (AUC). In addition, we use the Matthews Correlation Coefficient (MCC) that is a more comprehensive performance indicator. The formulas are:

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{18}$$

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

$$Recall = \frac{TP}{TP + FN} \tag{20}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{21}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{22}$$

where *TP* is True Positive that refers to the cases where the sample is positive, and the model also predicts it as positive, *TN* is True Negative that refers to the cases where the sample is negative, and the model predicts it as negative as well, *FN* is False Negative that refers to the cases where the actual class is positive, but the model predicts it as negative, *FP* is False Positive that refers to the cases where the actual class is negative, but the model predicts it as positive. AUC is the area under the receiver operating characteristic curve (ROC). ROC

curve is an effective tool for evaluating the performance of binary classification models. By plotting the false positive rate and recall at different classification thresholds, it helps us gain a comprehensive understanding of the model's performance under various conditions. The AUC measures the discriminatory ability of the classification model, with values closer to 1 indicating better classification performance.

5 Result

To evaluate the performance of CTMBWO, we use 5-fold cross-validation on four classifiers, namely SVM [25,26], RF [27], DT [28], and KNN [29], to conduct experiments and obtain the experimental results of Accuracy, AUC, MCC, F1, and Recall. Table 2 is the experimental results of the same version (training and testing are completed in the same project and the same version). Table 3 is the experimental results of the cross-version (Training is done with the lower version, and testing is conducted with the higher version).

To further compare the performance of the four classifiers, we calculated their fitness values on several datasets, with the results shown in Fig. 4. And Table 4 presents the running time (in seconds) of the four classifiers after 100 iterations on each dataset, with the average value bolded.

Table 2 illustrates that the experimental results obtained when using RF as a classifier are better than the other three classifiers. It can be concluded that the overall experimental results of projects Lucene and Log4j are worse than those of other projects; the overall experimental results of MC1 and PC2 are better than those of other projects. By specifically analyzing the detail of datasets in Table 1, projects MC1 and PC2 have 9466 and 5589 instances, respectively, while projects Lucene and Log4j have only 782 and 449 instances. From this, it is evident that the results of SD are closely linked to the number of instances in the project.

Table 2: Accuracy, AUC, MCC, F1, Recall results of CTMBWO on DT, KNN, RF and SVM

	DT					KNN					RF					SVM				
Dataset	**ACC**	**AUC**	**MCC**	**F1**	**Recall**	**ACC**	**AUC**	**MCC**	**F1**	**Recall**	**ACC**	**AUC**	**MCC**	**F1**	**Recall**	**ACC**	**AUC**	**MCC**	**F1**	**Recall**
ant-1.7	0.862	0.862	0.723	0.862	0.86	0.867	0.915	0.734	0.868	0.869	0.9	0.961	0.8	0.901	0.907	0.822	0.863	0.645	0.818	0.794
camel-1.2	0.752	0.762	0.505	0.757	0.777	0.731	0.772	0.466	0.742	0.783	0.773	0.833	0.546	0.767	0.754	0.684	0.696	0.37	0.695	0.727
camel-1.4	0.863	0.863	0.727	0.867	0.882	0.848	0.895	0.703	0.859	0.917	0.929	0.971	0.858	0.93	0.941	0.778	0.826	0.558	0.77	0.737
camel-1.6	0.842	0.844	0.684	0.847	0.846	0.833	0.884	0.672	0.849	0.91	0.895	0.952	0.788	0.899	0.918	0.754	0.773	0.509	0.776	0.827
ivy-1.0	0.751	0.761	0.527	0.756	0.865	0.781	0.727	0.6	0.788	0.917	0.745	0.762	0.523	0.755	0.881	0.789	0.72	0.581	0.777	0.822
ivy-1.1	0.934	0.934	0.869	0.936	0.96	0.927	0.969	0.861	0.931	0.999	0.965	0.99	0.932	0.967	0.988	0.911	0.954	0.823	0.914	0.941
ivy-1.2	0.92	0.921	0.839	0.923	0.916	0.893	0.929	0.799	0.891	0.976	0.927	0.967	0.853	0.92	0.939	0.864	0.894	0.73	0.854	0.89
jedit-3.2	0.874	0.873	0.75	0.876	0.871	0.862	0.902	0.75	0.88	0.996	0.865	0.935	0.736	0.875	0.928	0.848	0.884	0.698	0.833	0.895
jedit-4.0	0.857	0.859	0.717	0.851	0.89	0.878	0.889	0.753	0.866	0.859	0.904	0.949	0.807	0.895	0.887	0.784	0.821	0.576	0.783	0.848
jedit-4.1	0.872	0.872	0.743	0.872	0.879	0.883	0.899	0.767	0.883	0.883	0.87	0.943	0.739	0.869	0.865	0.821	0.873	0.653	0.802	0.728
log4j-1.0	0.889	0.89	0.779	0.896	0.882	0.934	0.938	0.873	0.942	0.999	0.91	0.951	0.822	0.915	0.895	0.918	0.929	0.835	0.922	0.906
log4j-1.1	0.796	0.796	0.594	0.786	0.754	0.806	0.852	0.611	0.808	0.818	0.783	0.817	0.567	0.780	0.769	0.841	0.785	0.699	0.82	0.727
log4j-1.2	0.975	0.975	0.95	0.977	0.98	0.877	0.924	0.767	0.874	0.802	0.981	0.997	0.962	0.982	0.969	0.886	0.918	0.773	0.891	0.869
lucene-2.0	0.71	0.705	0.413	0.665	0.696	0.755	0.738	0.494	0.702	0.7	0.71	0.773	0.434	0.687	0.773	0.736	0.684	0.457	0.684	0.692
lucene-2.2	0.755	0.763	0.516	0.75	0.8	0.774	0.765	0.545	0.743	0.708	0.793	0.85	0.59	0.786	0.824	0.699	0.635	0.399	0.687	0.720
xalan-2.4	0.882	0.881	0.763	0.873	0.888	0.883	0.924	0.774	0.881	0.949	0.905	0.962	0.814	0.901	0.948	0.826	0.874	0.652	0.816	0.845
xalan-2.6	0.731	0.721	0.463	0.734	0.728	0.778	0.804	0.563	0.764	0.708	0.782	0.842	0.568	0.774	0.735	0.753	0.78	0.533	0.713	0.606
CM1	0.933	0.933	0.868	0.934	0.955	0.911	0.947	0.831	0.917	0.985	0.943	0.984	0.886	0.943	0.953	0.846	0.876	0.698	0.854	0.904
KC1	0.881	0.872	0.762	0.877	0.868	0.857	0.913	0.721	0.862	0.922	0.898	0.96	0.796	0.895	0.888	0.748	0.817	0.504	0.761	0.825
KC3	0.929	0.927	0.859	0.932	0.951	0.929	0.969	0.862	0.934	0.984	0.942	0.982	0.883	0.943	0.948	0.889	0.92	0.778	0.892	0.898
MC1	0.996	0.997	0.992	0.996	0.998	0.993	0.997	0.987	0.993	0.999	0.998	0.999	0.996	0.998	0.999	0.957	0.98	0.914	0.958	0.978
MW1	0.945	0.944	0.89	0.942	0.941	0.929	0.978	0.868	0.931	0.999	0.965	0.989	0.93	0.963	0.958	0.896	0.953	0.792	0.891	0.893
PC1	0.953	0.954	0.906	0.953	0.96	0.937	0.974	0.879	0.941	0.992	0.962	0.993	0.924	0.963	0.973	0.891	0.922	0.786	0.897	0.942
PC2	0.994	0.995	0.987	0.994	0.994	0.988	0.995	0.975	0.988	0.998	0.996	0.999	0.992	0.996	0.997	0.925	0.963	0.849	0.924	0.924
PC3	0.917	0.917	0.835	0.919	0.934	0.899	0.959	0.807	0.906	0.974	0.945	0.987	0.89	0.946	0.961	0.846	0.902	0.693	0.851	0.88
PC4	0.942	0.945	0.884	0.942	0.937	0.918	0.955	0.841	0.923	0.98	0.963	0.992	0.926	0.964	0.981	0.905	0.952	0.817	0.918	0.966
Average	0.878	0.881	0.753	0.876	0.886	0.874	0.91	0.76	0.873	0.909	0.895	0.936	0.793	0.893	0.907	0.834	0.853	0.666	0.827	0.838

Table 3: Time comparison of four classifiers

Dataset	DT	KNN	RF	SVM
ant-1.7	15.20	53.14	503.96	290.65
camel-1.4	18.42	65.96	575.85	524.16
camel-1.6	17.11	72.37	636.51	699.95
ivy-1.2	12.65	29.20	388.53	83.12
jedit-3.2	5.83	19.56	311.17	35.21
jedit-4.0	7.73	25.49	345.98	55.03
jedit-4.2	9.23	30.96	391.86	76.00
log4j-1.0	4.56	13.65	299.46	15.53
log4j-1.2	5.28	20.43	322.41	35.66
lucene-2.2	5.33	16.54	320.66	32.99
lucene-2.4	6.05	21.78	343.54	49.43
poi-2.0	8.28	26.51	378.99	74.19
synapse-1.0	5.98	16.17	301.97	21.78
velocity-1.6	5.53	17.64	319.97	34.05
xalan-2.4	16.34	77.20	553.82	377.59
xalan-2.6	9.87	42.38	429.37	287.85
xerces-1.3	8.89	40.10	402.53	322.51
CM1	23.01	161.27	530.61	200.33
KC1	34.90	165.18	992.06	2759.81
KC3	16.95	156.01	473.69	158.80
MC1	421.40	1010.33	4032.26	30725.03
MW1	16.67	148.18	440.24	135.56
PC1	40.35	213.42	823.34	1004.19
PC2	329.60	636.35	3664.39	17699.06
PC3	74.42	271.01	1178.41	1997.21
PC4	52.75	251.03	1098.35	1177.78
PC5	1278.34	1837.32	11940.09	118255.98
Average	**90.77**	**201.45**	**1185.19**	**6560.35**

Table 4 shows that when using CTMBWO for cross-version defect prediction, the average accuracy is above 60%, the F1 value varies greatly for different projects, and the average MCC value is low, with the highest being 0.2963. It can be concluded that the prediction performance of using CTMBWO for cross-version prediction is weak. The main reason is that we only consider traditional features and do not further explore the semantic features and other dependencies between codes.

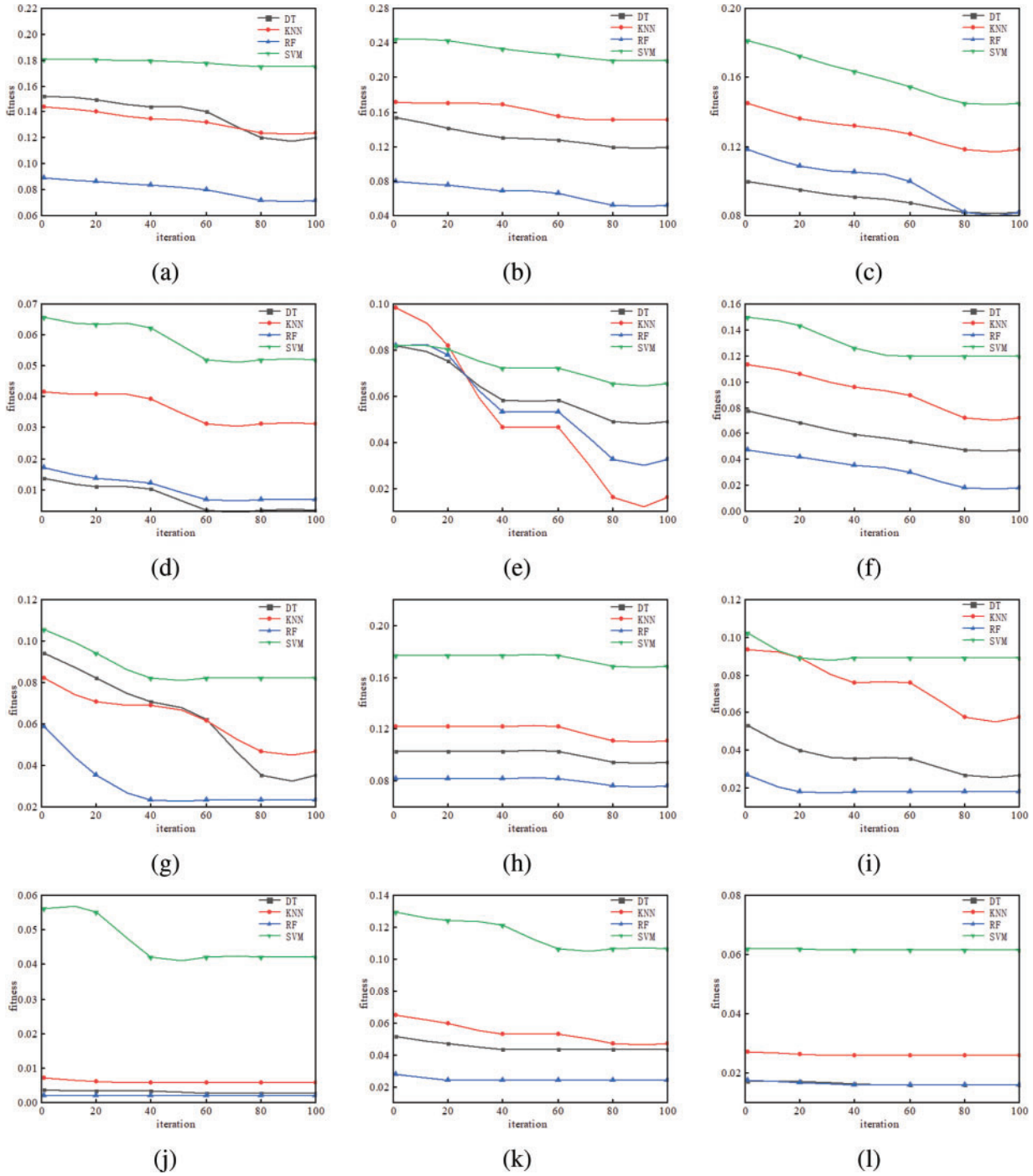

Figure 4: The fitness values of four classifiers on some datasets, (a) The fitness value of ant-1.7; (b) The fitness value of camel-1.4; (c) The fitness value of jedit-3.2; (d) The fitness value of jedit-4.3; (e) The fitness value of log4j-1.0; (f) The fitness value of poi-2.0; (g) The fitness value of synapse-1.0; (h) The fitness value of xalan-2.4; (i) The fitness value of MW1; (j) The fitness value of MC1; (k) The fitness value of PC1; (l) The fitness value of PC5

Table 4: Accuracy, AUC, MCC, F1 results of CTMBWO on DT, KNN, RF and SVM

		DT				KNN				RF				SVM			
Source	Target	ACC	AUC	MCC	F1	ACC	AUC	MCC	F1	ACC	AUC	MCC	F1	ACC	AUC	MCC	F1
ant-1.4	ant-1.6	0.659	0.543	0.11	0.339	0.622	0.621	0.177	0.432	0.684	0.603	0.133	0.335	0.641	0.579	0.151	0.399
ant-1.6	ant-1.7	0.671	0.664	0.318	0.479	0.688	0.695	0.379	0.535	0.598	0.608	0.242	0.573	0.719	0.726	0.38	0.535
camel-1.2	camel-1.4	0.686	0.668	0.276	0.415	0.668	0.693	0.237	0.388	0.642	0.569	0.239	0.461	0.633	0.676	0.172	0.345
camel-1.4	camel-1.6	0.591	0.618	0.256	0.387	0.686	0.654	0.205	0.389	0.658	0.628	0.224	0.431	0.682	0.65	0.19	0.378
jedit-3.2	jedit-4.0	0.569	0.547	0.452	0.599	0.63	0.647	0.426	0.582	0.629	0.614	0.365	0.643	0.628	0.815	0.373	0.538
jedit-4.0	jedit-4.1	0.693	0.691	0.441	0.576	0.713	0.73	0.436	0.595	0.77	0.8	0.459	0.595	0.75	0.793	0.411	0.575
lucene-2.0	lucene-2.2	0.602	0.616	0.242	0.598	0.622	0.637	0.264	0.637	0.627	0.657	0.285	0.632	0.622	0.634	0.256	0.644
lucene-2.2	lucene-2.4	0.558	0.55	0.098	0.616	0.644	0.668	0.307	0.667	0.612	0.643	0.201	0.7	0.602	0.681	0.224	0.624
poi-1.5	poi-2.5	0.584	0.605	0.366	0.731	0.708	0.72	0.389	0.759	0.687	0.7	0.482	0.795	0.785	0.853	0.551	0.825
poi-2.5	poi-3.0	0.584	0.591	0.177	0.628	0.666	0.707	0.328	0.715	0.637	0.653	0.275	0.685	0.679	0.762	0.393	0.7
velocity-1.4	velocity-1.5	0.611	0.498	0.106	0.743	0.624	0.498	0.056	0.746	0.645	0.41	0.056	0.775	0.624	0.535	0.052	0.746
velocity-1.5	velocity-1.6	0.589	0.639	0.272	0.569	0.624	0.685	0.317	0.59	0.615	0.653	0.347	0.66	0.654	0.689	0.327	0.589
xalan-2.4	xalna-2.5	0.562	0.548	0.138	0.306	0.579	0.604	0.156	0.461	0.573	0.623	0.171	0.321	0.608	0.633	0.223	0.486
xalna-2.5	xalan-2.6	0.583	0.579	0.266	0.571	0.601	0.633	0.362	0.645	0.592	0.581	0.304	0.619	0.624	0.683	0.446	0.597
Average		0.615	0.606	0.259	0.542	0.648	0.659	0.291	0.589	0.645	0.628	0.272	0.584	0.663	0.695	0.297	0.571

5.1 Comparison with Baseline Models

It can be concluded from Fig. 4 that the fitness values of the four classifiers are ranked as RF, DT, KNN, and SVM on most datasets. From Table 3, we observe that DT and KNN have the shortest running times among the classifiers. To further test the effectiveness of CTMBWO in feature selection, we compared CTMBWO with 4 prominent methods, namely GAPSO, Fed-OLF, MFWFS, and SLSTM. Table 5 shows a comparison between our method and the baseline methods(with the maximum value is bolded). Our results are the average of the outcomes obtained using KNN and RF as classifiers. And the comparison between CTMBWO and the baseline models' average values is shown in Fig. 5. The '–' in Table 5 indicates missing values, meaning there is no relevant data available in the original paper. We also attempted to reach out to the authors, but they did not share their code with us.

Table 5: The results of F1 and AUC

	F1					AUC				
Dataset	GAPSO	Fed-OLF	MFWFS	SLSTM	Ours	GAPSO	Fed-OLF	MFWFS	SLSTM	Ours
ant	0.879	0.611	0.809	**0.913**	0.891	0.842	0.747	0.837	**0.966**	0.926
camel	0.866	0.411	0.761	0.891	**0.895**	0.846	0.635	0.850	**0.952**	0.932
jedit	0.869	0.607	0.722	**0.981**	0.954	0.820	0.840	0.982	**0.995**	0.981
lucene	0.394	–	0.831	0.764	**0.807**	0.500	–	**0.900**	0.838	0.838
poi	0.702	0.589	0.613	0.796	**0.952**	0.696	0.753	0.912	0.843	**0.966**
xalan	0.681	0.588	**0.901**	0.813	0.899	0.646	0.747	**0.948**	0.874	0.911
xerces	0.920	0.656	0.901	0.952	**0.970**	0.872	0.787	0.891	0.982	**0.992**
CM1	0.950	0.427	–	**0.981**	0.939	0.907	0.723	–	**0.996**	0.979
KC1	0.875	0.404	0.753	0.885	**0.890**	0.850	0.637	0.868	0.949	**0.955**
MC1	0.988	0.256	0.712	0.999	**0.999**	0.995	0.786	0.933	0.999	**0.999**
PC1	0.968	0.326	–	**0.993**	0.964	0.940	0.720	–	**0.993**	0.986
PC3	0.949	0.428	0.803	**0.976**	0.933	0.904	0.676	0.924	0.976	**0.979**
PC4	0.935	0.557	0.854	**0.988**	0.961	0.878	0.720	0.938	0.987	**0.992**
Average	0.845	0.488	0.787	0.917	**0.928**	0.823	0.731	0.908	0.949	**0.957**

Based on the results in Table 5, it can be observed that the F1 and AUC values of CTMBWO show significant improvements compared to the four baseline methods. The F1 score increased by 0.3%–74%, and AUC improved by 4.9%–33.8%. Additionally, the average F1 and AUC values of CTMBWO are higher than

those of the baseline methods. These improvements indicate that CTMBWO is more effective at capturing the underlying patterns in the data, leading to higher accuracy and better performance in classification tasks. The increase in the F1 score suggests that the model performs better in handling class imbalance, achieving a better balance between recall and precision. The rise in AUC indicates that the model is more capable of distinguishing between different classes, especially when faced with complex and incomplete input data, making more accurate predictions. These results highlight the advantages of CTMBWO in defect prediction tasks and validate its effective improvement over baseline methods.

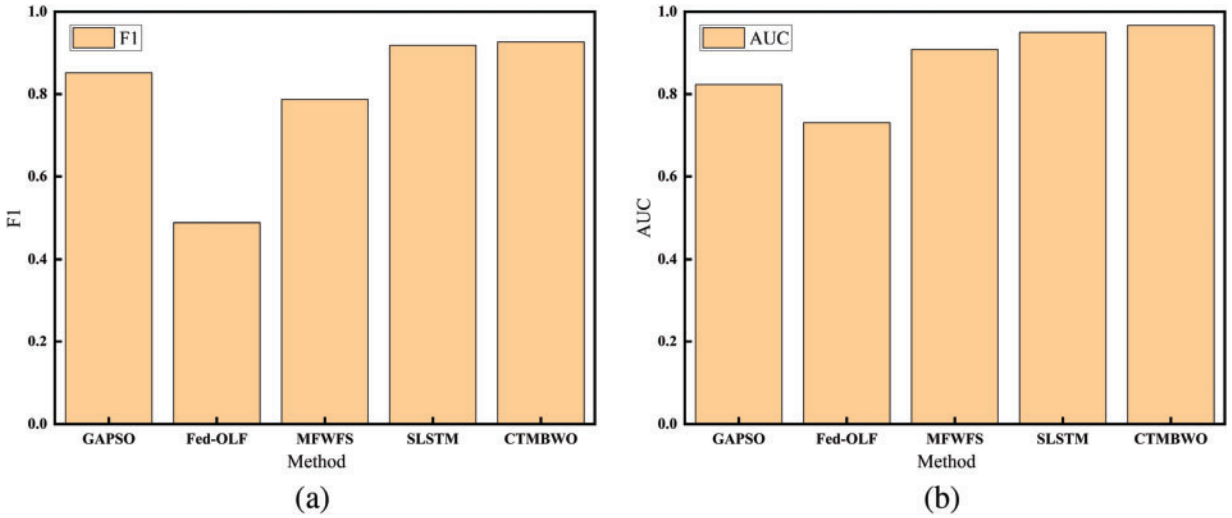

Figure 5: The comparison of the average values of F1 and AUC

5.2 Ablation Experiment

To further verifying the feasibility of our method, we conducted ablation experiments to compare the effects of using CTMBWO and other methods. Table 6 presents the results of ablation experiments. The methods tested include a baseline (Method1) without using feature selection, and two methods that incorporate Cubic chaotic mapping and an update mechanism combining Cauchy mutation and reverse learning. Method2 initializes the population using Cubic chaotic mapping, resulting in a 5.71% improvement in accuracy compared to Method1. Method3 builds on Method2 by adding a position update mechanism and achieves a further 3.46% improvement in accuracy. In Table 6, Cubic is the initialization of the population using Cubic chaotic mapping, C_M&O_L are the Cauchy variation and reverse learning, and T_W is the triangular wandering strategy.

In addition, we discussed the influence of parameters on the experimental results. The hyperparameter T_{max} has minimal impact on the experimental results because we use the strategy of early stopping and cross validation to avoid overfitting. Therefore, we focus primarily on analyzing how the population size N affects the results. We set the population size to 10, 20, and 30 and the experimental results are shown in Table 7. From Table 7, we can see that as the population size increases, both the Accuracy and AUC values of the model generally improve. This suggests that a larger population size enhances the model's performance. However, the population size cannot be increased indefinitely, as each individual in the population needs to be processed as the population grows, which significantly increases both computational time and space complexity, and may slow down the speed of convergence. In other words, the algorithm will take longer to reach the optimal solution. For this reason, we chose a population size of 30, which strikes a balance between resource usage and model performance.

Table 6: The results of ablation experiment

	Cubic	C_M&O_L	T_W	Accuracy	AUC	F1	Recall
Method1				0.772	0.845	0.772	0.789
Method2	✓			0.829	0.873	0.825	0.843
Method3	✓	✓		0.863	0.894	0.862	0.877
CTMBWO	✓	✓	✓	**0.878**	**0.907**	**0.884**	**0.9**

Table 7: The results of Accuracy and AUC for different population size

	Accuracy			AUC		
Dataset	***N* = 10**	***N* = 20**	***N* = 30**	***N* = 10**	***N* = 20**	***N* = 30**
ant	0.88	0.882	0.889	0.926	0.927	0.929
camel	0.778	0.788	0.79	0.833	0.826	0.829
jedit	0.858	0.875	0.885	0.907	0.907	0.924
log4j	0.89	0.904	0.903	0.926	0.93	0.921
lucene	0.747	0.765	0.781	0.761	0.793	0.809
poi	0.837	0.846	0.857	0.865	0.88	0.884
synapse	0.926	0.919	0.919	0.959	0.939	0.945
velocity	0.873	0.892	0.904	0.913	0.923	0.929
xalan	0.795	0.798	0.806	0.848	0.841	0.846
xerces	0.882	0.885	0.893	0.937	0.938	0.939
Average	0.847	0.855	0.863	0.888	0.89	0.896

5.3 The results of Paired t-Tests

Table 8 summarizes the results obtained from the paired t-test at a significance level of 5%. The results show that CTMBWO significantly outperforms the three baseline models: GAPSO, Fed-OLF and MFWFS. Although the significance of CTMBWO compared to SLSTM is not as pronounced, we can conclude from the comparison of the average values in Table 5 and Fig. 5 that CTMBWO's classification performance is clearly superior to that of SLSTM.

Table 8: The results of paired t-tests

	CTMBWO *vs.* GAPSO	**CTMBWO vs. Fed-OLF**	**CTMBWO vs. MFWFS**	**CTMBWO vs. SLSTM**
F1 p-value	0.0473	0.0001	0.0287	0.0949
AUC p-value	0.0351	0.0003	0.0339	0.1031

6 Conclusion and Future Work

This study proposes a new feature selection method CTMBWO to effectively select the optimal feature subset. CTMBWO aims to exclude redundant and irrelevant features and thus select the most important features. We evaluated the performance of CTMBWO on 19 project using four classifiers and compared the performance with existing models. The results demonstrate that applying CTMBWO in feature selection

significantly enhances the performance of SDP. Specifically, the average F1 value increases by 74%, the average AUC value increases by 33.8%. These improvements show that the algorithm effectively removes redundant and irrelevant features and selects the most valuable features for defect prediction, thereby enhancing the accuracy of the model. Furthermore, the CTMBWO improves population diversity and global search capabilities, effectively avoids local optimal solutions, and thus improves the prediction effect.

The application scenarios of CTMBWO in feature selection problems in machine learning include medical diagnosis, image processing, financial analysis, and text classification. Feature selection helps identify key indicators that are most relevant to the task, improving disease prediction accuracy, enhancing classification precision, optimizing risk assessment and investment decisions, and boosting text classification effectiveness.

However, CTMBWO has some limitations. It mainly performs feature selection based on traditional features and cannot consider the semantic information between codes, which leads to poor performance when predicting cross-version defect prediction. Therefore, future research will focus on exploring methods that combine semantic information and traditional features for SDP.

Acknowledgement: We are grateful to our families and friends for their unwavering understanding and encouragement.

Funding Statement: Not applicable.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Shaoming Qiu, Jingjie He, Yan Wang and Bicong E; data collection and analysis: Shaoming Qiu, Jingjie He and Bicong E; draft manuscript preparation: Shaoming Qiu and Jingjie He. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, Yan Wang, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Pachouly J, Ahirrao S, Kotecha K, Selvachandran G, Abraham A. A systematic literature review on software defect prediction using artificial intelligence: datasets, data validation methods, approaches, and tools. Eng Appl Artif Intell. 2022;111:104773. doi:10.1016/j.engappai.2022.104773.
2. Goyal S, Bhatia PK. Comparison of machine learning techniques for software quality prediction. Int J Knowl Syst Sci. 2020;11(2):20–40. doi:10.4018/IJKSS.
3. Pal S, Sillitti A. Cross-project defect prediction: a literature review. IEEE Access. 2022;10:118697–717. doi:10.1109/ACCESS.2022.3221184.
4. Zhao Y, Damevski K, Chen H. A systematic survey of just-in-time software defect prediction. ACM Comput Surv. 2023;55(10):1–35. doi:10.1145/3567550.
5. Goyal S, Bhatia PK. Software fault prediction using lion optimization algorithm. Int Inf Technol. 2021;13:2185–90. doi:10.1007/s41870-021-00804-w.
6. Malhotra R, Khan K. A novel software defect prediction model using two-phase grey wolf optimisation for feature selection. Cluster Comput. 2024;1–23. doi:10.1007/s10586-024-04599-w.
7. Chantar H, Mafarja M, Alsawalqah H, Heidari AA, Aljarah I, Faris H. Feature selection using binary grey wolf optimizer with elite-based crossover for Arabic text classification. Neural Comput Appl. 2020;32:12201–20. doi:10.1007/s00521-019-04368-6.

8. Goyal S. Genetic evolution-based feature selection for software defect prediction using SVMs. J Circuits Syst Comput. 2022;31(11):2250161. doi:10.1142/S0218126622501614.
9. Das H, Prajapati S, Gourisaria MK, Pattanayak RM, Alameen A, Kolhar M. Feature selection using golden jackal optimization for software fault prediction. Mathematics. 2023;11(11):2438. doi:10.3390/math11112438.
10. Arasteh B, Arasteh K, Ghaffari A, Ghanbarzadeh R. A new binary chaos-based metaheuristic algorithm for software defect prediction. Cluster Comput. 2024;1–31. doi:10.1007/s10586-024-04486-4.
11. Kukkar A, Kumar Y, Sharma A, Sandhu JK. Bug severity classification in software using ant colony optimization based feature weighting technique. Expert Syst Appl. 2023;230:120573. doi:10.1016/j.eswa.2023.120573.
12. Anbu M, Anandha Mala G. Feature selection using firefly algorithm in software defect prediction. Clust Comput. 2019;22:10925–34. doi:10.1007/s10586-017-1235-3.
13. Wang H, Arasteh B, Arasteh K, Gharehchopogh FS, Rouhi A. A software defect prediction method using binary gray wolf optimizer and machine learning algorithms. Comput Electr Eng. 2024;118:109336. doi:10.1016/j.compeleceng.2024.109336.
14. Sekaran K, Lawrence SPA. Mutation boosted salp swarm optimizer meets rough set theory: a novel approach to software defect detection. Trans Emerg Telecomm Technol. 2024;35(3):e4953. doi:10.1002/ett.4953.
15. Alsghaier H, Akour M. Software fault prediction using particle swarm algorithm with genetic algorithm and support vector machine classifier. Softw Pract Exp. 2020;50(4):407–27. doi:10.1002/spe.2784.
16. Feng J, Zhang J, Zhu X, Lian W. A novel chaos optimization algorithm. Multimed Tools Appl. 2017;76:17405–36. doi:10.1007/s11042-016-3907-z.
17. Zhong C, Li G, Meng Z. Beluga whale optimization: a novel nature-inspired metaheuristic algorithm. Knowl Based Syst. 2022;251(1):109215. doi:10.1016/j.knosys.2022.109215.
18. Mantegna RN. Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. Phys Rev E. 1994;49(5):4677. doi:10.1103/PhysRevE.49.4677.
19. Ryu D, Choi O, Baik J. Improving prediction robustness of VAB-SVM for cross-project defect prediction. In: 2014 IEEE 17th International Conference on Computational Science and Engineering; 2014; Sanya, China: IEEE. p. 994–9. doi:10.1109/CSE.2014.198.
20. Fenton N, Bieman J. Software metrics: a rigorous and practical approach. CRC Press; 2014.
21. Fernández A, Garcia S, Herrera F, Chawla NV. SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. J Artif Intell Res. 2018;61:863–905. doi:10.1613/jair.1.11192.
22. Hu X, Zheng M, Zhu R, Zhang X, Jin Z. Fed-OLF: federated oversampling learning framework for imbalanced software defect prediction under privacy protection. IEEE Trans Reliab. 2025. doi:10.1109/TR.2024.3524064.
23. Malhotra R, Chawla S, Sharma A. Software defect prediction based on multi-filter wrapper feature selection and deep neural network with attention mechanism. Neural Comput Appl. 2025;2025:1–28. doi:10.1007/s00521-024-10902-y.
24. Vasishth O, Bansal A. Enhanced software defect prediction using krill herd algorithm with stacked LSTM with attention mechanism. Int J Syst Assur Eng Manag. 2024;2024:1–21. doi:10.1007/s13198-024-02630-2.
25. Weston J, Mukherjee S, Chapelle O, Pontil M, Poggio T, Vapnik V. Feature selection for SVMs. Adv Neural Inf Process Syst. 2000; 647–53. doi:10.5555/3008751.3008845.
26. Nguyen MH, De la Torre F. Optimal feature selection for support vector machines. Pattern Recognit. 2010;43(3):584–91. doi:10.1016/j.patcog.2009.09.003.
27. Breiman L. Random forests. Mach Learn. 2001;45:5–32. doi:10.1023/A:1010933404324.
28. Krawczyk B, Woźniak M, Schaefer G. Cost-sensitive decision tree ensembles for effective imbalanced classification. Appl Soft Comput. 2014;14:554–62. doi:10.1016/j.asoc.2013.08.014.
29. Meiliana, Karim S, Warnars HLHS, Gaol FL, Abdurachman E, Soewito B. Software metrics for fault prediction using machine learning approaches: a literature review with PROMISE repository dataset. In: 2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom); 2017; Exeter, UK: IEEE. p. 19–23. doi:10.1109/CYBERNETICSCOM.2017.8311708.