Relay Pursuit-Vathana: A Novel Optimization Approach for Feature Selection in Software Defect Prediction

Rajakani M.[1], Beaulah Jeyavathana R.*[2], and Kavitha R. J.[3]

***Abstract*—Software defect prediction plays a crucial role in ensuring the quality and reliability of software systems. Feature selection, the process of identifying the most relevant features from a large set of potential features which is essential for building effective defect prediction models. In this paper, we propose a novel feature selection model based on the RelayPursuit-Vathana (RP-Vathana) optimization algorithm, inspired by relay races and pursuit dynamics in biological systems. The proposed model aims to identify an optimal subset of features for software defect prediction, maximizing the predictive performance of the resulting classification model. The RP-Vathana algorithm was integrated with a Naïve Bayes classifier and benchmarked on three datasets (PC5, JM1, KC2) to validate its effectiveness in feature selection for defect prediction. The results show that RP-Vathana significantly outperforms existing wrapper-based methods, obtaining mean accuracies of 94.28%, 93.69%, and 96.35% on PC5, JM1, and KC2, respectively, compared to the 83–90% range of rival techniques. While the parameter-free design improves usability, the algorithm's performance on highly noisy or very small datasets warrants future investigation into hybrid extensions for enhanced robustness.**

***Index Terms*—Software defect prediction, Feature selection, Bio-inspired algorithm, RP-Vathan**

I. INTRODUCTION

In the software development landscape, ensuring product quality and reliability is critical. Software flaws frequently appear despite careful planning and thorough testing, presenting serious difficulties for stakeholders and developers. Software defect prediction(SDP) model is essential for developing better software by enabling early defect identification, optimizing resource allocation, improving software quality, mitigating risks, enhancing decision-making, and fostering continuous improvement in the software development process.[1-2] Fixing errors caused by logic mistakes, insufficient requirement analysis and inaccurate system design takes time and resources, and it can negatively impact user experience and accurate solution.[3-4] As a result, proactive defect prediction techniques are gaining popularity as a means of identifying possible problems before they materialise in operational environments. Machine learning (ML) techniques have emerged as powerful tools for software defect prediction due to their ability to analyze vast amounts of data, extract patterns, and make predictions with remarkable accuracy [5]. By leveraging historical project data, such as code metrics, version control information, and defect reports, ML models can learn to identify patterns indicative of defect-prone areas within software systems.[6] In this study, the objective is to develop a defect prediction model for software development using supervised learning algorithms.
Various software metrics are utilized as decision variables to predict faults within the software. Given the vast dimensionality of the feature set, the aim is to employ feature selection techniques to identify the optimal subset of features that contribute most significantly to defect prediction accuracy.

The wrapper method stands out as the most commonly used approach for feature selection. It selects optimal features based on the performance of the predictive model, potentially enhancing its effectiveness. Moreover, wrapper methods can adapt to various performance metrics tailored to specific problem domains. These methods evaluate individual features while considering their interactions, thereby capturing complex relationships among them, which cannot be assessed by evaluating features individually. The commonly used wrapper approaches for software defect prediction model are: Particle Swarm Optimization (PSO) [7], Boosted Whale Optimization (BWO) [8], Ant Colony Optimization (ACO) [9], Genetic Algorithm (GA) [10], Firefly Algorithm (FA) [11] and Hybrid Grey Wolf Optimization (HGWO) [12].

In recent years, there has been a growing interest among researchers in developing optimization algorithms that operate without the need for control parameters because of the reduced complexity and improved convergence rates. Rao [32] developed the Rao Optimization algorithm, which updates the population set by leveraging the features of the best and worst individuals, without requiring any control parameters. This approach yields improved results. For instance, a parameter-free optimization algorithm [31]] has been successfully applied to the document classification problem demonstrating its robustness across domains. This highlights the overall effectiveness of such algorithms, while our work specifically targets their application in software defect prediction. Although promising, these approaches rely solely on the best and worst individuals in the population for updates, neglecting the contributions of intermediate solutions, which can restrict exploration of the search space.

[1, *2,3] Assistant Professor
Department of Data Science and Business Systems Department of Computational Intelligence
Department of Electronics and Communication Engineering
[1, *2] SRM Institute of Science and Technology, University College of Engineering Panruti, Panruti, India
[1] (e-mail: rajakanijes@gmail.com)
[*2] (e-mail: rbvathanaj@gmail.com)
[3] (e-mail: rjkresearch@gmail.com)

DOI: 10.36244/ICJ.2026.1.1

To address these shortcomings, the present study introduces a novel parameter-free and deterministic optimization algorithm, named RP-Vathana, for feature selection in SDP. Unlike conventional metaheuristics, RP-Vathana eliminates the reliance on randomness and control parameters by adopting a deterministic update strategy based on competitive interactions among all superior individuals. This ensures stable, consistent performance while preserving the balance between exploration and exploitation. The detailed motivation behind the algorithm is presented in Section II, while its working principle and computational framework are explained step by step in Section III.

The study design involves three major steps: (i) extraction of features from benchmark software defect datasets (PC5, JM1, KC1), (ii) application of the RP-Vathana algorithm for optimal feature selection, and (iii) classification of defect-prone modules using Naïve Bayes, with extensive comparisons against state-of-the-art metaheuristic algorithms across multiple independent runs.

The principal contributions of this study are:

- **A Novel Parameter-Free Algorithm**: We introduce RP-Vathana, a deterministic optimization algorithm for feature selection. This design effectively overcomes the inherent limitations of randomness and manual parameter tuning required in conventional methods.
- **Demonstrated Superiority**: We prove that RP-Vathana yields superior, more stable performance (higher classification accuracy and robustness) compared to current state-of-the-art algorithms on standard Software Defect Prediction (SDP) datasets.
- **New Direction in Optimization:** Our work highlights deterministic optimization as a promising methodological alternative to randomness-driven metaheuristics, offering new insights for SDP and broader machine learning applications.

II. RELATED WORKS

Software defect prediction (SDP) has been studied extensively, with applications in within-project [13–15], cross-project [16–18], and heterogeneous settings [19–20]. Most approaches employ supervised learning, where prediction accuracy depends heavily on the choice of features [21].

A. Feature Selection Approaches: Filter, Wrapper and Hybrid

Feature selection methods are generally classified as filter, wrapper, and hybrid. Filter methods apply statistical criteria, wrappers evaluate subsets using classifiers, and hybrids combine both. Alsaeedi et al. [26] reported that machine learning–based models generally outperform traditional filter-only approaches, while Aleem et al. [27] benchmarked 11 algorithms across 15 NASA datasets and found NB and SVM to be consistently effective. This indicates that careful feature subset selection remains a key factor for reliable SDP.

B Parameter-Dependent vs Parameter-Free Metaheuristics Approaches

Metaheuristic search has also been widely adopted for feature selection. Parameter-dependent methods (e.g., GA, PSO) require careful tuning and are often vulnerable to getting trapped in local optima, which limits their global search ability. Despite these drawbacks, they have shown effectiveness in domains such as image classification [28], tuberculosis detection [29], and SDP [30]. In contrast, parameter-free approaches reduce reliance on parameter tuning and improve robustness, though their application in SDP remains limited. This suggests an opportunity to explore parameter-free metaheuristics for more stable and scalable feature selection in SDP.

C Benchmarking Approaches and Limitations

Benchmarking studies play a central role in comparing SDP methods. Aleem et al. [27] demonstrated that NB and SVM often outperform other classifiers across datasets, while further evaluations confirm that SVM, MLP, and Bagging consistently achieve high accuracy (~89%), low error (MAE ≈ 0.10–0.14), and strong F-measure (~0.94). Conversely, KNN frequently underperforms, with accuracy below 75% and higher error rates. These findings reinforce the effectiveness of ensemble and margin-based methods, but also highlight the variability of results across datasets.

Beyond conventional classifiers, neuro-fuzzy techniques [22-24] and software agent–based approaches [25] have been explored to enhance test-case generation and improve defect detection. Hybridizing these approaches with machine learning methods offers potential for more accurate and adaptable prediction models.

However, benchmarking efforts face several limitations, including dataset imbalance, lack of diversity across repositories, and inconsistencies in evaluation protocols. Therefore, there is a need for more comprehensive benchmarking strategies that can ensure fair comparisons and generalizable conclusions.

D. Proposed Method: RP-Vathana Feature Selection

To address the limitations identified in prior studies, we propose RP-Vathana, a parameter-free optimization algorithm for feature selection in software defect prediction. Unlike parameter-dependent metaheuristics, RP-Vathana requires no control parameters, which enhances adaptability across diverse optimization tasks and eliminates the risk of poor performance due to improper tuning.

The algorithm is inspired by relay-race dynamics, where team members adjust their pace in response to one another. In this framework, candidate solutions cooperate and refine their search strategies relative to their peers, enabling the algorithm to escape local optima and move toward globally competitive solutions.

RP-Vathana has been applied to decision-metric selection in software defect prediction and has demonstrated effectiveness across three benchmark datasets, confirming its robustness and potential as a scalable feature selection approach.

E. Background and Motivation

In a relay race, teamwork and cooperation among team members are vital for success. Typically, teams follow a structured approach to maximize efficiency and speed. In an individual race, each runner focuses on matching the fitness

level of the fastest participant without assistance. However, in a relay race, teams consist of four participants who must collaborate and adjust their fitness levels based on each other's abilities. This means that a runner can receive support from their teammates even if they underperform at certain stages of the race.

To win a race, it's essential to adjust your pace according to the faster runners. You must increase your speed gradually, starting by overtaking those ahead of you. In many population-based optimization algorithms, individuals' positions are typically updated relative to the best individual in the population. However, in this research, a novel optimization algorithm called Relay Pursuit - Vathana (RP-Vathana) is introduced, which draws inspiration from relay races. In RP-Vathana, a relay race structure is utilised to enhance the position updates of weaker individuals within the population.

The remainder of the manuscript is organized as follows: Section III introduces the theoretical concept of the RP-Vathana algorithm, while Section IV details its implementation. Section V covers the experimental setup, parameter settings for benchmark algorithms, evaluation metrics, and an in-depth analysis of the results. Section VI addresses potential threats associated with our proposed algorithm, and Section VII and VIII provides the Limitations/Future scopes and conclusion respectively.

III. Working principle of Relay Pursuit-Vathana algorithm

This section describes the working principle of the algorithm followed by the mathematical model of the algorithm.

The algorithm begins by generating a random population of N individuals, each with a unique fitness evaluated using a Bayesian Information Criteria (BIC). These individuals are then sorted based on their fitness values in descending order. Next, the population is divided into 'm' groups, with each group containing an equal number of individuals. The highest-fitness individual overall is selected as the best individual, while the top N/m individuals in the sorted population become the leaders of each group.

Let us consider the initial population set $POP= (I_1, I_2, I_3,..I_N)$, where I_j represents j-th individuals. $Sorted\_POP = [I_{R1},I_{R2},..,I_{RN}]$ where I_{Rj} represents individuals with rank j. $POPgroup = [POP_1,POP_2,POP_3,..,POP_{N/m}]$ where POP_i represents the i-th population subset. Let N denote the population size and m the number of subsets, where the first m ranked individuals act as leaders. The population is partitioned into disjoint subsets based on modular arithmetic.

For each leader $l \in \{1,2,3,..,m\}$, the corresponding subset is defined as

$$s_l = \{i \in \{1,2,\dots,N\}: i \equiv l(mod\ m)\} = \left\{l + km : k = 0,1,2,\dots,\left\lfloor\frac{N-l}{m}\right\rfloor\right\}$$

In other words, if the individuals are denoted as, x_1,x_2,x_N sorted in the ascending order of the rank (where x_i corresponds to the i-th ranked individual), then the l-th subset can be expressed as

$$POP_l = \{x_i : i \in s_l\} = \left\{x_{l+km} : k = 0,1,\dots,\left\lfloor\frac{N-l}{m}\right\rfloor\right\}$$

For instance, if the population size is 20 and m is 4, each group will have 5 individuals. The individuals with rank *1* to *4* are selected as leaders. The subsets are organised such that individuals with rank *1,5,9,13,17* form the first subset, individuals with rank *2,6,10,14,18* form the second subset, and so on. The leader of each subset corresponds to the individual with the lowest index within that subset.

After organizing the individuals into groups, the velocity of each individual, except the leader candidate is update based on the velocity of best performing candidates within the same group. Let v_{ij} denote the velocity of the j-th variable in the i-th individual, and v_{kj} represent the velocity of the j-th variable of the k-th best candidate in the population g.

The velocity is updates using Eq.1 and Eq.2:

$$v_{ij}^g = v_{ij}^g + \gamma.\left(v_{kj}^g - v_{ij}^g\right) \quad (1)$$

$$x_{ij}^g = x_{ij}^g + v_{ij}^g \quad (2)$$

This process continues for a maximum number of iterations or until the fitness function converges. Subsequently, the candidate with the maximum fitness value is selected as the best candidate.

Algorithm 1 RP-Vathana optimization algorithm

```
procedure RP-VATHANA(featureVector)
    Design objective function
    Generate initial population set POP_1
    Calculate the fitness of each candidate using BIC
    x_Best: candidate x with best fitness in POP_1
    while T<T_Max do
        Sort all the candidates in POP_{T-1} in descending order
        for each Group g from 1 to m do
            for each candidate l from g to N do
                Append candidate X_l to group 'g'
                l:=l+m
            end for
        end for
        for each Group g from 1 to m do
            for each candidate i in the group g do
                X_new : Move the candidate X towards all the better
                candidates in the group 'g' using Equations 3 and 4
                X_New1 : Move X_New towards X_Best using equations 1 and 2
                Keep best among X,X_New,X_New1 in the POP_T
            end for
        end for
        Update X_Best in POP_T
    end while
    return X_Best
```

A. Mathematical model of RP-Vathana optimization:

This section describes about the mathematical model of proposed RP-Vathana optimization algorithm

1. Population initialization:

Assume that the size of the population is N, and each candidate has d number of variable. The i-th candidate is represented as :

$x_i=\{x_{i1},x_{i2},..,x_{id}\}$ where x_{ij} represents the j-th variable of the candidate

2. Fitness Evaluation:

The Bayesian Information Criteria is employed as the fitness function to evaluate the fitness of each candidate. This is a minimization problem so that the candidate with minimum value is selected as best candidate, represented as x_{best}.

3. Grouping and Movement:

Assume that the entire population set is divided in to m number of sub groups. Each group has *N/m* candidates. The initial population set is sorted in descending order and divided into m groups and candidates are selected based on their rank.

In each group the velocity of the candidates is updated based on the better performing candidate in the same group. For each individual x_i in each group g, the position of variable *j* is updated as follows.

$$v_{ij}^{g} = v_{ij}^{g} + \gamma . \sum_{k=1}^{K} \left(v_{kj}^{g} - v_{ij}^{g} \right) \quad (3)$$

$$x_{ij}^{g} = x_{ij}^{g} + v_{ij}^{g} \quad (4)$$

Where,

v_{ij}^{g} is the velocity of the j-th variable of the i-th candidate in the group 'g'

γ is a random value ranging from 0 to 1. The introduction of the random variable *r* in the modification step adds stochasticity to the algorithm. This stochastic element facilitates exploration of the search space, mimicking the inherent randomness observed in biological processes.

v_{kj}^{g} is velocity the j-th variable of the k-th better performing candidate in the group 'g'

x_{ij}^{g} is the j-th variable of the i-th candidate in the group 'g'

K is the number of better performing candidates in the group 'g'

In the Eq.3, if the difference between velocity of the j-th variable of candidate *i* and better candidate *k* is positive, encourages the individual *i* to move towards including the j-th variable. If it is negative, encourages the individual *i* to move towards excluding the j-th variable.

Repeat the steps 2 and 3 for maximum number of iterations. (T_{Max}) and the candidate with minimum BIC is selected as optimal candidate. Algorithm 1, shows the working principle of the RP-Vathana optimization algorithm.

Fig.1 shows the flow chart of the RP-Vathana algorithm. The candidates are ranked by the fitness values and the best candidate is selected as X_{best}. The population set is divided into m subsets. Candidates are placed into the subsets based on their ranks. First '*N/m* ranked candidates are set as leader of the respective subset and the leader candidates are moved towards the X_{best} candidate. All the remaining candidates *X* in the group g are moved towards the better performing candidate in g. If the new candidate X_{new} is better than *X* then X_{new} is moved towards X_{best} and kept as X_{new1}. The better candidate among *X*, X_{new} and X_{new1} is retained in the population set.

IV. Implementation of RP-Vathana optimization algorithm

This section describes the implementation details of the proposed algorithm

A. Data Preprocessing using RobustScaler

To mitigate the effect of outliers on feature scaling, we applied the RobustScaler method. Unlike standard scaling, which relies on the mean and standard deviation, RobustScaler uses the median and interquartile range (IQR) to transform the features. The scaling formula is given by:

$$x' = \frac{x - median(x)}{IQR(x)} \quad (5)$$

where *x* is the original feature value, median(x) is the median of the feature and $IQR(x) = Q_3 - Q_1$ represent the interquartile range. This approach ensures that the transformation is robust to outliers, improving the stability and performance of subsequent algorithms.

B. Population representation scheme

Each candidate in the population set is represented as binary feature set of size n_f where n_f is the number of design metrics in the software metrics data set. This feature set is represented as binary values (0,1) where the value of 1 represents the inclusion of the design variable and value of 0 represents its absence. Suppose if the size of feature set is 10 and the i-th candidate is represented as x_i= {1,0,1,1,0,0,1,1,0,1}. It indicates that the features *f1,f3,f4,f7,f8* and *f10* are present in the candidate solution.

C. Initialize the population table:

To generate the initial population set with a size specified by N, which is calculated as

$N = n_f \,.\, 0.10$, each candidate solution undergoes the following process:

Firstly, the continuous search space is transformed into a discrete search space using the sigmoid function. Random values are generated for each feature within the defined range of that feature. These randomly generated values represent the initial state of each feature in the solution. Subsequently, the sigmoid function is applied using Eq.6 to these random values, converting them into discrete values within the range of 0 to 1.

$$Sigmod(x) = \frac{1}{e^{-x} + 1} \quad (6)$$

Each candidate solution, denoted as xi, is represented as a vector $\{v_1, v_2, .. v_d\}$, where each v_j represents a random value generated for the j-th variable in the i-th solution, *r(i,j)*. Initially, the value of v_j is assigned randomly within the defined range of the variable *r(i,j)*. Subsequently, the sigmoid function is applied to map the value v_j to a value between 0 and 1. Finally, the Eq.7 is utilized to convert the resulting continuous value into a binary value.

$$x_{ij} = floor(Sigmod(x) + rand()) \; mod \; 2 \quad (7)$$

D. Fitness evaluation

The fitness of candidates is assessed using the BIC (Eq.8) as an objective function. BIC uses error rate and number of features used as the key for evaluating the performance of the candidate. Since it is a penalty-based evaluation method, the candidate which shows less error rate with minimum number of features gets a low penalty.

$$BIC = n \cdot \log MSE + k \cdot \log n \quad (8)$$

were n denotes the number of instances in the dataset, MSE represents the mean square error rate, k is the number of features.

MSE is calculated using the Eq.9

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad (9)$$

were N is the number of instances in the data set , y_i is the actual value of the instance i, $\hat{y}_i$ is the predicted value of the instance i.

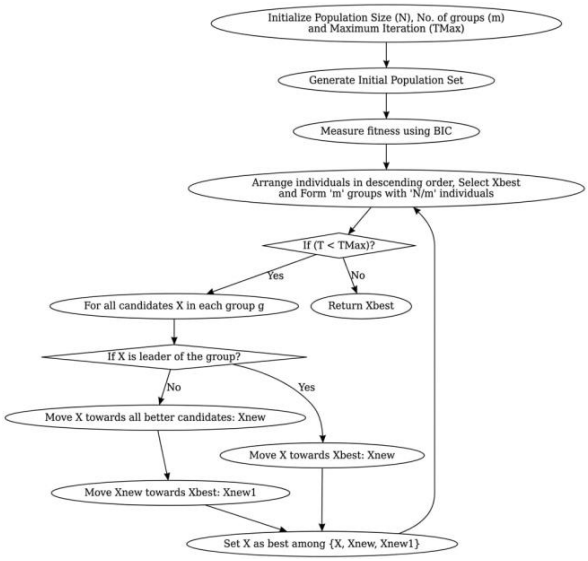

Fig.1 RP-Vathana optimization algorithm flow chart

V. EXPERIMENT AND VALIDATION

A. Experimental set up

Navie Bayes (NB) classifier are used for classification purpose. The model is trained for 80% of the samples and remaining samples are used for validation purpose. 5-fold cross validation process is used for classification. The entire samples are divided in to five equal folds. On each iteration, one fold is used for training the model and the remaining folds are used for testing. This process is repeated for all five possible combinations of training and testing folds. Highest accuracy achieved across the five iteration of 5-fold cross validation is selected as overall accuracy. To compare the effectiveness of the proposed approach, we evaluate it alongside several state-of-the-art algorithms, including Particle Swarm Optimization (PSO) [7], Boosted Whale Optimization (BWO) [8], Ant Colony Optimization (ACO) [9], Genetic Algorithm (GA) [10], Firefly Algorithm (FA) [11], and Hybrid Grey Wolf Optimization [12]. These algorithms are parameter-dependent, meaning their performance heavily relies on parameter values. Table.1 shows the parameters used for the benchmark algorithms to be compared.

Each algorithm is independently executed 30 times, and the average results are documented for analysis and comparison.

TABLE I
ALGORITHM SPECIFIC PARAMETER VALUES

Algorithm	Parameters	Value(s)
GA	C – Crossover method	One-point
	Pc – Crossover probability	0.5
	M – Mutation method	Swap
	PM – Mutation probability	0.3
PSO	W – Inertia weight	0.1
	C1 – Local learning coefficient	0.4
	C2 – Global learning coefficient	0.9
FA	α – Randomization parameter	0.1
	B0 – Base attraction	1
	Γ – Absorption coefficient	1
BWO	α – Linearity decreasing parameter	α ∈ [0,2]
	b – Constant	1
	l – Randomization parameter	[-1,1]
HGWO	α – Controlling parameter	α ∈ [0,2]

B. Performance metrics used:

The following well known metrics are used for evaluating the performance of the proposed RP-Vathana algorithm: accuracy, recall, precision and F1 score.

To calculate the above metrics True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) values are calculated from the predicted values of the testing instances.

TP indicates the number of correctly classified positive instances, TN indicates number of correctly classified negative instances, FP indicates number of negative instances classified positively and FN indicates number of positive instances classified negatively.

Accuracy is the percentage of samples that are correctly classified. (Eq.10)

$$ACC = \frac{(TP+TN)}{N} \qquad (10)$$

Precision is the ratio between the correctly classified positive values to the number of instances that are classified as positive. (Eq.11)

$$Precision = \frac{TP}{(TP+FP)} \qquad (11)$$

Recall is the percentage of positive values that are correctly classified.(Eq.12)

$$Recall = \frac{TP}{TP+FN} \qquad (12)$$

F1 score combines the precision and recall values (Eq.13)

$$F1 - Score = \frac{TP}{\left(TP + \frac{1}{2} * (TP + TN)\right)} \qquad (13)$$

C.Result and discussion

The effectiveness of the proposed software defect prediction method is assessed using three well-known benchmark datasets [PC5, JM1, KC1], which are accessible through the NASA-MDP public dataset repositories. Table 2 provides summaries of these benchmark datasets for reference.

TABLE II
NASA DATASET DESCRIPTION

Dataset	No. of features	Total no. of instances	No of defective instances	No of non-defective instances
PC5	39	17186	516	16670
JM1	21	10885	8779	2106
KC1	21	2109	326	1783

The comparison results are presented in Table 3, depicting the performance of our proposed method alongside other algorithms using NB classifier on the PC5 dataset. The proposed RP-Vathana achieves 94.28% accuracy with precision: 0.973, recall: 0.9347 and F1-Score:0.9504, which are comparatively better than other algorithms.

TABLE III
ACCURACY COMPARISON ON PC5 DATA SET USING NB CLASSIFIER

	ACC %	ERR	Precision	Recall
GA	85.79%	14.21%	0.9697	0.8579
PSO	88.71%	11.29%	0.9712	0.8771
FA	89.14%	10.86%	0.9715	0.8814
BWO	90.44%	9.56%	0.9736	0.9044
HGWO	92.94%	8.06%	0.974	0.9094
RP-Vathana	94.28%	5.72%	0.967	0.9109

Table 4 presents the comparative results for defect prediction rates such as accuracy, error rate, precision, recall, and F1-score on the JM1 dataset using NB classifier. Notably, the proposed feature selection method, RP-Vathana, has a macro-averaged F-score of 0.9142 for the NB classifier. Our RP-Vathana-based software defect prediction method achieves an impressive 93.69% classification accuracy. These results highlight the superiority of our proposed feature selection method on the publicly available JM1 dataset.

TABLE IV
ACCURACY COMPARISON ON JM1 DATA SET USING NB CLASSIFIER

	ACC %	ERR	Precision	Recall
GA	85.78%	14.22%	0.89	0.8578
PSO	86.70%	13.30%	0.8958	0.867
FA	87.13%	12.87%	0.8986	0.8713
BWO	89.94%	10.06%	0.9139	0.8944
HGWO	91.93%	8.07%	0.9243	0.9093
RP-Vathana	93.69%	6.31%	0.9177	0.9107

Classification accuracy, error rate, precision, recall, and F1-score are the evaluation metrics shown in Table 5 that are obtained from the proposed RP-Vathana method and other feature selection algorithms that use NB classifier on KC1 data set. Compared to other approaches, our proposed algorithm notably achieves high classification accuracy. In particular, RP-Vathana software defect prediction model outperforms other feature selection methods with a better success rate of 96.35%. Moreover, the proposed model performs better in terms of recall, F1-score, and macro-averaged precision.

TABLE V
ACCURACY COMPARISON ON KC1 DATA SET USING NB CLASSIFIER

	ACC %	ERR	Precision	Recall
GA	86.72%	13.62	0.9062	0.8672
PSO	88.63%	11.37%	0.9058	0.8663
FA	89.05	10.95%	0.9191	0.8905
BWO	91.97%	8.03%	0.9267	0.9037
HGWO	94.84%	5.16%	0.9488	0.9384
RP-Vathana	96.35%	3.65%	0.9795	0.9635

TABLE VI
PERFORMANCE METRICS COMPARISON OF PRO-POSED ALGORITHM AND BASELINE ALGORITHMS ACROSS 30 RUNS

		ACCURACY (%)				
		Maximum	***Minimum***	***Mean***	***Median***	***Std.Dev***
PC5 Data Set	***RP-Vathana***	94.28	89.32	91.43	91.34	1.27
	HGWO	92.94	85.73	89.63	89.36	1.9
	BWO	90.44	83.75	87.31	86.72	2.08
	FA	89.14	82.98	86.37	86.95	1.98
	PSO	88.71	83.89	86.62	86.7	1.28

	GA	85.79	81.58	83.81	83.95	1.59
JM1 Data Set	***RP-Vathana***	93.69	88.7	91.02	91.29	1.38
	HGWO	91.93	87.08	89.28	88.96	1.45
	BWO	89.94	85.22	87.51	87.59	1.52
	FA	87.13	82.14	84.75	84.99	1.61
	PSO	86.7	82.07	84.19	84.33	1.49
	GA	85.78	80.04	82.73	82.68	1.68
KC2 Data Set	***RP-Vathana***	96.35	92.57	94.25	94.17	1.16
	HGWO	94.84	90.69	92.85	92.83	1.18
	BWO	91.97	88.24	90.06	90.2	1.21
	FA	89.05	84.77	86.81	86.86	1.2
	PSO	88.63	84.46	86.09	85.95	1.21
	GA	85.93	81.94	83.88	83.72	1.3

Table 6 summarizes the maximum, minimum, mean, median, and standard deviation of the accuracy scores obtained from 30 runs of our proposed algorithm compared to other state-of-the-art algorithms. The proposed algorithm shows competitive performance with a higher mean accuracy and lower standard deviation, indicating both effectiveness and stability. The detailed statistical measures provide a clear and comprehensive understanding of how the proposed method performs relative to others.

TABLE VII
PERFORMANCE EVALUATION OF THE PROPOSED MODEL USING MCC AND F1-SCORE ACROSS ALL THREE DATASETS

Dataset	MCC	F1-Score
PC5	0.928	0.938
KC1	0.824	0.970
JM1	0.894	0.914

As summarized in Table 7, the proposed model consistently achieves high MCC and F1-score values across all three datasets. The MCC values indicate a strong correlation between the predicted and actual defect labels, while the F1-scores reflect a balanced trade-off between precision and recall. Together, these metrics confirm the model's robustness and effectiveness in handling the imbalanced class distributions present in the datasets.

A. Comparative Analysis of Selected Features with Existing Studies

The study utilized three benchmark datasets (PC5, JM1, and KC2) that feature a diverse set of software metrics. These metrics capture the essential characteristics of source code modules, including size (e.g., **LOC_TOTAL**), complexity (e.g.,**CYCLOMATIC_COMPLEXITY, DECISION_COUNT**), and maintainability as measured by Halstead metrics (e.g., **HALSTEAD_EFFORT**). Additional features, such as **PARAMETER_COUNT** and **GLOBAL_DATA_COMPLEXITY**, provide structural and modularity insights. These comprehensive metrics serve as the input features for the defect prediction model. To evaluate the effectiveness of RP-Vathana in identifying informative features, we compared the feature subsets selected from the PC5, JM1, and KC2 datasets against those reported in prior studies. Table VIII summarizes the top features obtained by RP-Vathana alongside rankings from existing feature selection approaches.

TABLE VIII
COMPARISON OF TOP SOFTWARE METRICS SELECTED BY RP-VATHANA AND PRIOR STUDIES ACROSS DATASETS

Dataset	Top Features (RP-Vathana)	Top Features (Prior Studies)
PC5	CYCLOMATIC_COMPLEXITY, HALSTEAD_EFFORT, PARAMETER_COUNT, ESSENTIAL_COMPLEXITY, LOC_EXECUTABLE	PARAMETER_COUNT, NUM_OPERANDS, NUM_OPERATORS, NUM_UNIQUE_OPERANDS, NUM_UNIQUE_OPERATORS, HALSTEAD_CONTENT, HALSTEAD_DIFFICULTY [33]
JM1	LOC_EXECUTABLE, DECISION_COUNT, HALSTEAD_DIFFICULTY, NODE_COUNT, MAINTENANCE_SEVERITY	LOC_TOTAL, MAINTENANCE_SEVERITY, ESSENTIAL_DENSITY, CYCLOMATIC_DENSITY, HALSTEAD_LEVEL [34]
KC2	BRANCH_COUNT, HALSTEAD_VOLUME, NUM_UNIQUE_OPERANDS, ESSENTIAL_COMPLEXITY, DESIGN_COMPLEXITY	NUM_OPERANDS, HALSTEAD_DIFFICULTY, HALSTEAD_EFFORT, LOC_EXECUTABLE, DECISION_COUNT [35]

Our findings show that RP-Vathana consistently emphasizes complexity and effort-related metrics, such as **CYCLOMATIC_COMPLEXITY, ESSENTIAL_COMPLEXITY, HALSTEAD_EFFORT**,

and **PARAMETER_COUNT**. This agrees with earlier studies that also highlighted the importance of Halstead and complexity measures for defect prediction

VI. VALIDITY THREATS

This section addresses the validity challenges posed to the proposed feature selection scheme. Although RP-Vathana feature subset selection method outperforms other algorithms in terms of performance, it requires more computational time to execute.

A. Complexity Analysis

The computational complexity of the proposed algorithm is derived by analyzing its major components as presented in Algorithm~1.

Initialization (Lines 2–5): The initial population of N candidates is generated and their fitness values are computed. This requires $O(N)$ operations.

Sorting (Line 7): In each iteration, the population is sorted according to fitness values. This requires $O\ (\ N \log N)$ time.

Group Assignment (Lines 8–13): Each candidate is assigned to one of the mmm groups. Since all NNN candidates are assigned exactly once, this step takes $O(N)$ time.

Candidate Updates (Lines 14–21): Within each group, every candidate is updated with respect to all better candidates in that group. For a group size of approximately N/m, each update requires $\frac{N}{m}$ operations. Since there are N candidates in total, the overall cost of updates per iteration is:

$$O\left(N \cdot \frac{N}{m}\right) = O\left(\frac{N^2}{m}\right)$$

Additional operations, such as updating with respect to the global best and performing comparisons, incur constant-time overheads and do not affect asymptotic complexity.

Iteration over TMax Cycles (Lines 6–23): The above steps are repeated for a maximum of TMax iterations.

By combining the above components, the total time complexity of RP-Vathana is given as:

$$O\left(T_{Max} \cdot \left\{N + N \log N + N + \frac{N^2}{m}\right\}\right)$$

For moderate values of m, the quadratic term $\frac{N^2}{m}$ dominates, and the worst-case asymptotic complexity simplifies to:

$$O(N^2 \cdot T_{Max})$$

This result highlights that the higher computational overhead of RP-Vathana arises primarily from its update mechanism, which considers interactions among all candidates rather than updating with respect to only the global best solution. Although this results in a higher runtime compared to conventional algorithms with time complexity $O(N \cdot T_{Max})$ that update only based on the global best, it enhances exploration capability and reduces the risk of premature convergence.

VII. LIMITATIONS AND FUTURE SCOPE

The parameter-free design of RP-Vathana ensures robustness but limits user control over exploration-exploitation balance. In highly multimodal landscapes, averaging influences from multiple candidates (Centroid Effect) can slow convergence and cause suboptimal solutions. Additionally, the need to aggregate forces from all candidates increases computational overhead, particularly in high-dimensional problems. Future work could address these issues via a dynamically weighted candidate pool to enhance exploration and convergence. Future work can focus on integrating RP-Vathana with other supervised learning models (e.g., SVM, Random Forest, MLP) to validate robustness and to optimize its time complexity for faster execution on large datasets. Hybrid feature selection strategies and selective parameter tuning could further enhance performance in challenging scenarios. To mitigate limitations such as the Centroid Effect and slow convergence in multimodal landscapes, a dynamically weighted candidate pool can be introduced. Additionally, benchmarking against emerging optimization and deep learning approaches can provide a more comprehensive evaluation of generalization and predictive capabilities.

VIII. CONCLUSION

In this study, we presented a novel software defect prediction model called RP-Vathana which is a parameter-free optimization algorithm. We showcased the superior performance of our proposed model with three benchmark datasets (PC-5, JM-1, and KC-1) through extensive experimentation and validation. Our results show that the software defect prediction model based on RP-Vathana consistently outperformed other meta-heuristic algorithms that require control parameters. One of the key strengths of our approach lies in its parameter-free nature, eliminating the need for parameter tuning. Furthermore, the robust performance of our model proves its potential for real-world applications in software defect prediction. In future work, our parameter-free RP-Vathana algorithm combined with auxiliary filter-based methods could potentially improve the accuracy of predictive models. Moreover, investigating parallel computing paradigms may optimise our algorithm's runtime performance considering the computational overhead it carries.

REFERENCES

[1] S. Hamayun, & L.F. Calvo-Flores, "Interpretable Software Defect Prediction from Project Effort and Static Code Metrics." *Applied Sciences*, vol. 14, no. 4, p. 52, 2024.

[2] G. Giray, K.E. Bennin, Ö. Köksal, Ö. Babur, & B. Tekinerdogan, "On the use of deep learning in software defect prediction". *The Journal of Systems & Software*, vol. 195, p. 111 537, 2023. DOI: 10.1016/j.jss.2022.111537.

[3] Z. Li, J. Niu, & X.Y. Jing, "Software defect prediction: future directions and challenges." *Automated Software Engineering*, vol. 31, no. 1, p. 19, 2024.

[4] Y. Jiang, B. Shen, & X. Gu, "Just-In-Time Software Defect Prediction via Bi-modal Change Representation Learning". *The Journal of Systems & Software*, p. 112 253, 2024. DOI: 10.1016/j.jss.2024.112253

[5] Karpagalingam Thirumoorthy, J. A., "Jerold John Britto feature selection model for software defect prediction using binary Rao optimization algorithm," *Applied Soft Computing*, vol. 131, p. 109 737, 2022, ISSN 1568-4946, DOI: 10.1016/j.asoc.2022.109737.

[6] Rohit Vashisht, Abhinav Juneja, Gagan Thakral & Sonam Gupta. "An empirical study of just-in-time-defect prediction using various machine learning techniques", *International Journal of Computers and Applications*, 2024. DOI: 10.1080/1206212X.2024.2328489

[7] R. Malhotra, N. Nishant, S. Gurha, V. Rathi, "Application of particle swarm optimization for software defect prediction using object oriented metrics," in: *2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pp. 88–93, 2021, http://dx.doi.org/10.1109/Confluence51648.2021.9377116.

[8] S.A.H. El-Kenawy, M.M. Eid, M.H.S. El-Shafai, A.M.H.E. Gamil, and D.S. El-Dahshan, "Hybrid binary whale optimization algorithm based on taper shaped transfer function for software defect prediction," *Comput. Mater. Continua*, vol. 78, no. 3, pp. 3177–3199, 2024.

[9] A. Hashemi, M.B. Dowlatshahi, "Exploring Ant Colony Optimization for Feature Selection: A Comprehensive Review." In: *Dey, N. (eds) Applications of Ant Colony Optimization and its Variants. Springer Tracts in Nature-Inspired Computing*. Springer, Singapore, 2024. DOI: 10.1007/978-981-99-7227-2_3

[10] M. Azzeh, R. Al-Sayyed, & F. Al-Tahir, "Software Defect Prediction Using Non-Dominated Sorting Genetic Algorithm and k-Nearest Neighbour Classifier." *e-Informatica Software Engineering Journal*, vol. 18, no. 1, p. 240 103, 2024.

[11] N. Monga and P. Sehgal, "A Framework of Software Defect Prediction using Machine Learning with Updated Firefly Algorithm and Neural Networks," *Journal of Information Systems Engineering and Managemen*t, vol. 10, no. 3, 2025.

[12] S. Sangeetha, & D. S.Rajakumari, "A Hybrid Genetic Based Grey Wolf Optimized Sophisticated Support Vector Machine (SSVM) Model for Software Defect Prediction". *Journal of Advanced Technology and Innovative Research*, vol. 101, no. 19, pp. 164–177, 2023. (Published late 2023, relevant to current work).

[13] H. Chen, X. Li, W. Shi, B. Zhang, & K. Wang, "Cross-Project Defect Prediction Using Transfer Learning with Long Short-Term Memory Networks." *Complexity*, 2024.

[14] A. Al-Sabaawi, B. Al-Khateeb, M.S. Al-Saeed, & N.N. Al-Taisan, "Software Defect Prediction Using an Intelligent Ensemble-Based Model". *IEEE Access*, vol. 12, pp. 3681–3694, 2024.

[15] B. Ghotra, S. McIntosh, A.E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models". In: *ICSE'15. IEEE*, pp. 789–800, 2015.

[16] T. Zimmermann, N. Nagappan, H. Gall, et al. "Cross-project defect prediction: a large-scale experiment on data vs. domain vs. process". In: *FSE/ESEC'09. ACM*, pp. 91–100, 2009.

[17] Z. Li, J. Niu, XY. Jing, et al. "Cross-project defect prediction via landmark selection-based kernelized discriminant subspace alignment." *IEEE Trans. Reliab*. vol. 70, no. 3, pp. 996–1013, 2021.

[18] Z. Li, H. Zhang, X.Y. Jing, et al. "Dssdpp: data selection and sampling-based domain programming predictor for cross-project defect prediction." *IEEE Trans. Softw. Eng*. vol. 49, no. 4, pp. 1941–1963, 2023

[19] X. Jing, F. Wu, X. Dong, et al. "Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning." In: *FSE'15. ACM*, pp. 496–507, 2015.

[20] Z. Li, X.Y. Jing, X. Zhu, "Heterogeneous fault prediction with cost sensitive domain adaptation". *Softw. Test. Verif. Reliab*. vol. 28, no. 2, 1–22, 2018.

[21] D. Al-Fraihat, Y. Sharrab, A.R. Al-Ghuwairi, H. Alshishani and A. Algarni, "Hyperparameter Optimization for Software Bug Prediction Using Ensemble Learning," in *IEEE Access*, DOI: 10.1109/ACCESS.2024.3380024

[22] Ömer Faruk Arar, Kürşat Ayan, A feature dependent Naive Bayes approach and its application to the software defect prediction problem, *Applied Soft Computing*, vol. 59, 2017, pp. 197-209, ISSN 1568-4946, DOI: 10.1016/j.asoc.2017.05.043.

[23] S. Pandey, R. Mishra, A. Tripathi, "Software bug prediction prototype using Bayesian network classifier: A comprehensive model", *Procedia Comput. Sci*. vol. 132, 2018, pp. 1412–1421, http://dx.doi.org/10.1016/j.procs.2018.05.071.

[24] X. Rong, F. Li, Z. Cui, "A model for software defect prediction using support vector machine based on CBA", *Int. J. Intell. Syst. Technol. Appl*. vol. 15, 2016, p. 19, http://dx.doi.org/10.1504/IJISTA.2016.076102.

[25] S. Rathore, S. Kumar, "A decision tree logic-based recommendation system to select software fault prediction techniques", *Computing*, vol. 99, 2017, pp. 255–285, http://dx.doi.org/10.1007/s00607-016-0489-6.

[26] A. Alsaeedi, M. Khan, "Software defect prediction using supervised machine learning and ensemble techniques: A comparative study", *J. Softw. Eng. Appl*. vol. 12, 2019, pp. 85–100, http://dx.doi.org/10.4236/jsea.2019.125007

[27] S. Aleem, L. Capretz, and F. Ahmed, "Benchmarking Machine Learning Technologies for Software Defect Detection." *International Journal of Software Engineering & Applications*, vol. 6, pp. 11–23. DOI: 10.5121/ijsea.2015.6302

[28] M.R., RJ, K. "Invasive weed optimization with deep transfer learning for multispectral image classification model." *Multimed Tools Appl*., 2023. DOI: 10.1007/s11042-023-17429-9

[29] R.B. Jeyavathana and R. Balasubramanian, "Automatic detection of tuberculosis based on adaboost classifier and genetic algorithm", International Journal of Biomedical Engineering and Technology, vol. 36, no. 3, pp. 203–219, 2021.

[30] M. Anbu, G.S. Anandha Mala, "Feature selection using firefly algorithm in software defect prediction". *Cluster Comput* vol. 22, Suppl 5, pp. 10 925–10 934, 2019. DOI: 10.1007/s10586-017-1235-3

[31] K. Thirumoorthy, K. Muneeswaran, "Optimal feature subset selection using hybrid binary Jaya optimization algorithm for text classification", *Sadhan*., vol. 45, no. 1, 2020, pp. 1–13, http://dx.doi.org/10.1007/s12046-020-01443-w

[32] R. Rao, "Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems." *International Journal of Industrial Engineering Computations*, vol. 11, no. 1, pp. 107–130, 2020.

[33] Y. Wang, A. Patel, & J. Thomas, "Selection of test case features using fuzzy entropy measure and Random Forest for software defect prediction." International Journal of Information and Systems, vol. 24, no. 3, pp. 85–96, 2023. DOI: 10.18280/isi.240306

[34] R. Sharma, M. Gupta, & P. Singh, "Analysis of important features in software defect prediction using SMOTE, RFE and Random Forest." *Journal of Software Engineering Research and Development*, vol. 12, no. 2, pp. 145–160, 2024. DOI: 10.1007/s40595-024-00123-x

[35] S. Lee, & H. Kim, "Feature importance analysis for NASA defect datasets using Random Forest and information gain." *Journal of Computer Science and Technology*, vol. 37, no. 5, pp. 1024–1038, 2022. DOI: 10.1007/s11390-022-2567-5

M. Rajakani is an Assistant Professor at SRM Institute of Science and Technology, Kattankulathur, Chennai. He received his Ph.D. in optimization techniques from Anna University. His research interests include Satellite Image Processing, optimization algorithms, machine learning, and intelligent communication systems. He has published several research articles in high-impact international journals and international Conferences.

R. Beaulah Jeyavathana is an Associate Professor at SRM Institute of Science and Technology, Kattankulathur, Chennai. She received her Ph.D. in Medical Image Analysis from MS University. Her research focuses on artificial intelligence-based medical image analysis, machine learning models for healthcare systems, and optimization techniques. She has authored several research articles published in reputed international journals and conference proceedings.

R. J. Kavitha is an Assistant Professor at University College of Engineering, Panruti. She received her Ph.D. in Wireless Communications from Anna University. Her research interests include signal processing, wireless communications, and machine learning. She is currently supervising five research scholars and has successfully guided four research scholars to completion under Anna University.