RES MILITARIS
REVUE EUROPEENNE D ETUDES EUROPEAN JOURNAL OF MILITARY STUDIES
Social Science Journal

Enhancing Software Quality through Meta-Heuristic Algorithm for Optimized Test Case Execution Prioritization and Effective Fault Detection

Priti Singh, Hari Om Sharan , C.S. Raghuvanshi
Faculty of Engineering and Technology, Rama University, Mandhana, Kanpur, Uttar Pradesh, India
preetirama05@gmail.com

Abstract
Software testing is a critical phase in the software development lifecycle, aimed at ensuring the reliability and functionality of software modules or applications. It involves rigorous examination of software artifacts and behavior to validate its functionalities and identify any potential errors. The ultimate goal of software testing is to enhance the overall quality of the developed software, thereby bolstering customer satisfaction and maximizing profitability in the software market. While various methods and tools have been employed for software testing in the past, many fail to provide comprehensive guidelines for improving software quality. Moreover, existing testing frameworks often lack stage-by-stage output with precise error definitions, hindering effective error mitigation strategies. Additionally, regression testing models reliant on feedback mechanisms tend to introduce computational and time complexities. To address these challenges, the prioritization of test cases (PTC) emerges as a pivotal process for enhancing fault detection rates and refining error descriptions to elevate software quality. In this context, this study leverages an Extended FireFly Algorithm (EFFA) to formulate an advanced prioritization methodology. The EFFA capitalizes on the swift and efficient search capabilities of the FireFly Algorithm (FFA), significantly enhancing fault detection accuracy while mitigating inherent drawbacks. Implemented and tested within the DOTNET software environment, the proposed EFFA prioritization methodology demonstrates superior performance metrics, as evidenced by improved Fault Detection Rates (FDR) and reduced processing times.

Keywords: Test Case Prioritization, Test Case Minimization, Whale Optimization, Rider Optimization, Fault Detection Rate, Precision, Recall, F-Measure.

Introduction

Testing is the major process that plays a vital role in software development and various software application development. Software Testing (ST) is one of the significant tasks in the software development life cycle. The ST process is divided into two parts as manual and automated testing. In manual testing, the human is involved in testing the software, but a software tool is used to test the newly developed software in automated testing. In recent times, the software industry plays a King role in the industry of computing and information technology. Information processing and computing complexity is reduced in terms of accuracy in calculation and processing time.

Table-1. Essential Features of Test Cases

Features	**Description**
Date of Creation	The date at test case is created
Date of Execution	The date at test case is executed
Test-ID	A unique identification number is assigned to each test case.
Test Summary	Summary about the testing.
Pre-Condition	Set of constraints need to be fulfilled during test case execution
Steps	Steps we have to follow before doing testing
Input Data	Various types of data we enter when testing the Login of a system
Severity	Severity is the impact that the bug/error can be made for the system, is assigned as a low, medium, or high.
Priority	The order that the bug/error should be fixed.
Expected Results	The expected result mentions the customer requirement of the system.
Actual Results	In the actual result, we mention the real way of relevant component is working.
Pass/Fail	The result of the test case is pass or fail
Executed By	Test case designer name
Developer Comment	The developer comments on getting pass/fail result

Thus, it requires software testing model to identify and detect faults for enhancing the software quality. Although a software system is tested efficiently, it might require certain functionalities to be implemented. During regression testing, the software system gets altered. Even after alternation, the system's standard is also well maintained [1].

The test case is the primary and essential factor of software engineering which helps to examine the software quality. In any software testing the primary factors (test cases) are considered initially to determine the other additional factors in the testing process in accordance to the application. Before developing a test case, it needs to obtain knowledge about the system's needs. For that, user requirements, use case models, developer notes, and other factors are referred to. The test case designer should discuss the development section with the team leader, project manager, and other team members. The major features used in the TCP process is given in Table-1.

The elements given in Table-1 vary for various test cases. Prioritising the execution order of the test cases is focussed as the major research problem. To do that, designing a good test case is more important. Several TCP methods are available in the software development and testing industries. But recent testing industries worldwide use only seven TCP methods, shown in Figure-1. This paper uses the type-2, type-2, type-3, and type-4 shown in Figure-1 to increase the fault detection rate, decreasing the time, through which the ratio of the fault detection with enhanced software quality.

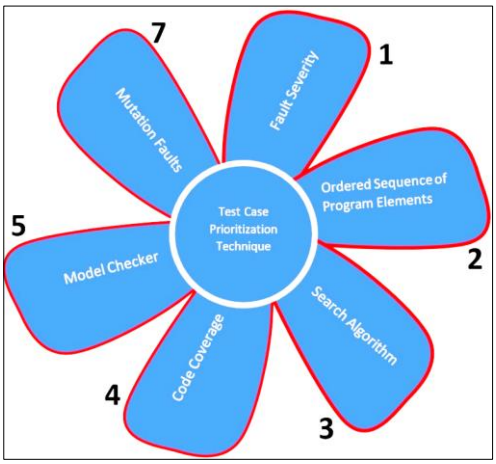

Figure-1. Various Categories of Test Case Prioritization Methods

While using regression testing, a significant amount, such as $\frac{2}{3}$ of the entire budget is utilized for software testing and its maintenance; hence, this testing method is quite expensive [2]. Thus50% of the cost is utilized [3]. In cases like critical software systems, regression testing requires more efficiency in the testing process.

Table-1. Statistical Information of Features of Test Cases

Test Case	Fault revealed by test case									
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
TC1	1	0	0	0	1	0	0	1	1	1
TC2	1	0	0	0	1	1	1	0	0	0
TC3	0	0	0	0	1	1	1	1	1	1
TC4	1	1	1	1	1	0	0	0	1	0
TC5	1	1	1	0	1	0	0	1	0	0

The test cases are prioritized based on their attributes; hence it is essential to optimize them effectively and reduce fault detection time. For example, 5 test cases and their attributes are given in Table-2. The table shows that TC3 and TC4 are superior to others since their faults are detected earlier and speedily. Comparing other testing techniques with the proposed regression technique, the primary difference is that it reuses the software test module of the previous software version. However, in some test cases, the test modules run all the test cases involved in that modules. For this, it requires more effort. Consider an example where the software system contains40,000 lines of codes, in which the time it takes to run all the 40,000 lines of codes might take many weeks or a month to run the whole test module [4, 5]. Hence another technique must be adopted to reuse test modules more competently.

Most of the earlier methods like regression testing consumes more cost and some of them have tried to reduce the cost of it. In the earlier days, several methods arose to reduce the cost of regression testing. A similar set of testing is selected and executed to minimize the cost of testing process. For lowering the validation suite, an EFF method is introduced to group the sets that helps to cover the overall test scenario. Moreover, the earlier techniques have focussed on fault detection during the testing process. Based on the faults detected, time taken for fault detection, and the cost are increased. Thus, the earlier approaches may not be suitable for all the test cases while selecting the subset. Ordering & reusing the test case also

helps to increase the software quality and minimize errors. A technique is introduced to schedule the test cases to attain an output efficiently. The test cases run depending upon the size of the validation suite. Planning the test case is optimal if the validation suite is extended. Firstly, preparing the test case at the initial level helps to overcome the time and resource restrictions.

Choosing the test and reducing the cost plays a vital role in prioritizing techniques. In a smaller validation suite, scheduling takes place after the selection and reduction. Scheduling technique varies from one to another. EFF method is implemented in this paper to schedule the test cases. Mainly, the scheduling is based on information gathering and finding errors. Thus, the performance is calculated and compared to the other techniques. The developer can quickly identify the errors earlier while increasing the fault rate. Thus, EFFmethod is implemented for enhancing its performance & improving the detection of fault rate. Several definitions for the defects, mistake, error, failure, and fault are discussed in IEEE Standard Glossary of Software Engineering Terminology" [24, 25]. An incorrect output obtained is termed as a “mistake”. So any kind of defects may occur in the software module, which needs to be identified and corrected before delivering to the customer.

Contribution of the Paper

In this paper, an Extended FireFly Algorithm is proposed for TCP, and it incorporates local and global searching strategies.

The EFFA detects the faults in the test cases faster using distance customization and data emphasizing.

This EFFA uses various benchmark test case datasets in the experiment for further verification.

The obtained results are compared with the existing methods in terms of Fault Detection Rate and process time. The overall process of the proposed model is depicted in Figure-2.

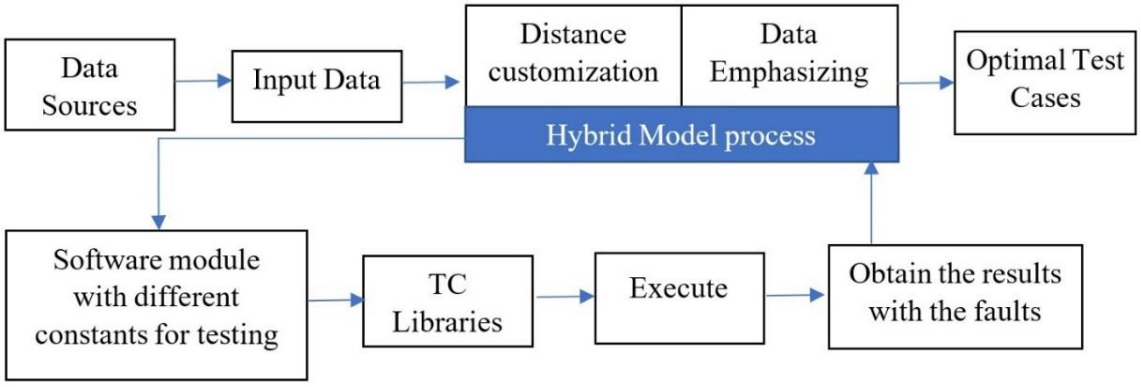

Figure-2. Contribution of Paper

Literature Survey

In this paper, the EFF method is implemented and compared with some existing techniques, such as prioritizing test cases. Some of the widely used regression techniques are SVM [6], K–Nearest Neighbour (KNN) [7]) logistic regression [8], and some specific techniques from neural networks [9]. Ensemble learner [10] is created by joining all the outputs of various SVM. For estimating the performance of the EFF method and proving its efficiency, we have evaluated three complex industrial systems here.

In 2019, Nayak S. et al. [11] analyzed the general model of TCP to improve the FDR and process time. Various factors such as FDR, number of errors detected, the ability of the test case with respect to error detection and their suitability. The resulting test scenario requirement is analyzed using other prioritization techniques. The consequence of the strategy shows a higher standard level of a detected fault. In 2016 Ammar, A. *et al*. [12] analyzed an improved method for organizing the entire test suite without utilizing random positioning. The ability of the method was verified by testing with the controlled model. In 2016, Ansari A. *et al*. [13] examined test priority techniques that require test cases to be put together to ensure that well-designed test cases have significant deficiencies. In their article, the ACO was used as the priority process to minimize the cost and effort. The study results showed that the organization of test cases limits the time, effort, and cost of the test and finds significant flaws in the software. In 2010 Huang, C.Y., *et al*. [14] developed a weighted opportunity flow chart to understand the unweighted graphical user interface of the test case with the position. From the experiment, the efficiency is verified in terms of FDR and time.

Jeffrey, D. *et al*. [15] in 2008 analyzed a strategy to organize test cases that detect the faults present in the program using a value profile-based approach. It alters the values that causes failure in execution to produce the desired output. The point-by-point trial study and results give fascinating bits of knowledge into the viability of utilizing necessary cuts for test case prioritization to accomplish a high pace of fault detection. In 2020, Taneja, D. *et al*. [16] analyzed test case reductions in object-oriented (OO) testing. Freely accessible information identified with Open-Source Software have been utilized for framework assessment. A linear regression (LR) model containing faulty information and different object-oriented estimations for insurance have been created. The cost estimation is depending on the weight of the test cases and the methodology empowered in the software programming.

In 2020, Mohapatra, S.K. *et al.*, [17] developed a local search for test case minimization (TCM). In their work, they attempted to utilize an intelligent search algorithm to tackle the TCM issue. They found that the technique could give delegate packages quickly. From the tests, it was discovered that the delegate set blunder detection limit was 100%. The RF factor for all subjects was 0.0, and the RS factor was higher on STAGE contrasted with different strategies. Trial perceptions and results investigation recommends that the methodology has a beneficial application. In 2018 Hashim, N.L *et al.* [18] developed a TCP method using FFA algorithm and obtained reduced complexity with the cost.

Limitation and Motivation

From the detailed survey carried out, one can understand that the earlier research works have not calculated the reliability of the similarity of the test cases. Some of the earlier methods estimated distance for only one string, which is not solving the redundancy problem since space weight is equal for all the TCs. The ranking process done by the earlier methods is not practical, for example, Swarm Intelligence and other algorithms in [25-27]. One of the significant problems of most of the earlier methods is time complexity. Thus, this paper aims to solve the above-said problems by integrating the hybrid model with the nature-inspired algorithm. That helps solve problems by providing ample search space and more iterations, which can also be customized. It uses local as well as global searching methods.

Software Testing

One of the significant tasks in the software development life cycle is software testing, where it executes a software module to examine the output following the input data. The software testing is introduced in a precise manner using some notations as:

Let $\boldsymbol{SM}$ be the software module, $\boldsymbol{I}$ be the input feed into $\boldsymbol{SM}$, and the correct hypothetical version of the $\boldsymbol{SM}$ is $\boldsymbol{F}$. A test case $\boldsymbol{t}$ is considered an element of $\boldsymbol{I}$, where each $\boldsymbol{I}$ is the member of the test suite $\boldsymbol{T}$, and for all test case $\boldsymbol{t}$, a failure condition is verified by $\boldsymbol{P}(\boldsymbol{t}) \neq \boldsymbol{F}(\boldsymbol{t})$. In case if there are no faults generated by $\boldsymbol{T}$, then it doesn't mean that the quality of the SM is 100% correct because generally, $\boldsymbol{I}$ is infinite. However, if the design of $\boldsymbol{T}$ is accurate, it is assured that the quality of $\boldsymbol{SM}$ is high. During the testing process, the $\boldsymbol{SM}$ is modified and copy it as $\boldsymbol{SM'}$, and the create $\boldsymbol{T}$ for testing $\boldsymbol{SM}$**.** There are two different categories of information used in the EFF prioritization method to arrange the execution in such as manner to maximize the objective function, such as reducing the errors and increasing the quality in the $\boldsymbol{SM}$**.** The first category is coverage information (CI) based on prioritizing the test cases. The second is fault-exposing-potential (FEP) information for estimating the ability of the test cases regarding fault detection in the EFF Prioritization Method. The CI-based PM involves the control-flow-graph method, Node, edge, and test history coverage. At the same time, the FEP based PM uses different probability values calculated to estimate the ability of EFF PM. They are Execution, Infection, and Propagation probabilities. But adding more coverages, the cost of the test cases is increased. But, one of the main aims of this paper is to decrease the size of the $\boldsymbol{T}$ rather than scheduling the execution of the $\boldsymbol{T}$.

Problem Statement of Test Case Prioritization

The Test Case Prioritization (TCP) is used to schedule all the test cases for increasing the value of the Objective Function (OF). The prioritization mainly focuses on expanding the similarity ratio to make the OF better by executing the TC in a particular order.

Given

The test suit T is considered a test suite, the permutations of T is calculated as PT, and a function f is used for relating PT to real numbers.

Problem

Obtain $T' \in PT, suchthat, (\forall T'')(T'' \in PT)(T'' \neq T')[f(T') \geq f(T'')]$

Thus, the hybrid model generates the set of all TCs for obtaining the best test sequences to assign the priority for executing them.

Hybrid Model Construction

The proposed EFF algorithm is a hybrid algorithm that incorporates two significant functions: customizing the distance and data emphasizing. The correlation coefficient matrix is used to measure distance customization, whereas TFIDF (Text Frequency In Data Frequency) measures data highlighting. The hybrid process is carried out into two levels. In the **first hybrid process**, the distance customization process is carried out. The term string spacing is used to measure the counts of the test cases, whereas distance customization is used for measuring the string spacing that occurred in various test cases expressed mathematically

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) if min(i,j) = 0 \\ min\begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 others \\ lev_{a,b}(i-1,j-1)+1(a_i \neq b_j) \end{cases} \end{cases} \quad \textbf{(1)}$$

In equation-(1), $\boldsymbol{lev_{a,b}(i,j)}$ denotes the distance customization among the TCs a and $\mathbf{b}(\boldsymbol{i,j})$ denotes the indexes, $\boldsymbol{a_i}$ is the $\boldsymbol{i^{th}}$character in $\boldsymbol{a}$, and $\boldsymbol{b_j}$ denotes the $\boldsymbol{j^{th}}$character in $\boldsymbol{b}$, where the distance is calculated by

$$\boldsymbol{d_i = lev_{a,b}(i,j)} \quad \textbf{(2)}$$

From equation 2, the overlapped distance for the same test cases is obtained by,

$$\boldsymbol{P_i = max(i,j) - lev_{a,b}(i,j)} \quad \textbf{(3)}$$

The distance customization (**dc**) is calculated as

$$\boldsymbol{Q_i = lev_{a,b}(i,j)} \quad \textbf{(4)}$$

From equation–(3), the similarity coefficient is calculated for the test cases $\boldsymbol{a}$ and $\boldsymbol{b}$ is,

$$\boldsymbol{S_{i,j} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i}} \quad \textbf{(5)}$$

From the $S_{i,j}$, the correlation coefficient matrix M_i is obtained using the following equation:

$$M_i = \begin{bmatrix} 0 & S_{1,2} & \dots & S_{1,n} \\ S_{2,1} & 0 & \dots & : \\ : & \dots & \dots & S_{n-1,n} \\ S_{n,1} & \dots & \dots & 0 \end{bmatrix} \quad \textbf{(6)}$$

The second level of the hybrid process estimates the significance of the data emphasizing with the help of finding the number of test data $n_{k,j}$ that occurred in each test case, and it is calculated as,

$$\boldsymbol{tfidf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} * log \frac{|D|}{|\{d: d \ni t_{i,j}\}|}} \quad \textbf{(7)}$$

It significantly denotes the DE of unique test cases, where,

$n_{i,j} \rightarrow$ The number of times the term occurred in test case **d**.

$|\{d: d \ni t_{i,j}\}| \rightarrow$ # of test data $t_{i,j}$present in d.

$|D| \rightarrow$ # test cases

$t_{i,j} \rightarrow$ denotes the i^{th} test data present in j^{th} test case.

The above-said levels of the hybrid process are integrated with the Firefly algorithm to design the Extended Firefly algorithm, explained below.

Extend FFA

In the EFFA, all the parameters are initialized at the beginning.

Step-1: The FF agents are initialized by the brightness value is:

$$Brightness_{i,j} = w_i / \left(\frac{n_{i,j}}{\Sigma_k n_{k,j}} * log \frac{|D|}{|\{d: d \ni t_{i,j}\}|} \right) \quad \textbf{(8)}$$

w_i denotes the weight factor value assigned to the i^{th} test case.

Step-2: Evaluation Criteria

For evaluating the proposed EFFA, the APFD value is calculated. The authors in [19] utilized APFD for detected fault rate in TCP experiments. The technique used for optimal sequencing in this experiment. This paper assigns 0 to 100 as the range for attaining the

APFD. From the value, the more significant value shows the better failure rate detection. It can be expressed using the value APFD formula is as:

$$APFD = 1 - \frac{TF_1+TF_2+TF_3+\cdots+TF_N}{n\times m} + \frac{1}{2n} \qquad \textbf{(9)}$$

$n \rightarrow$ total number of test case sets

$m \rightarrow$number of defects in the software module

$TF_i \rightarrow$number of defects (I) identified at i^{th} location

$$Faultdetectionrate \leftarrow max\ (APFD) \qquad \textbf{(10)}$$

The enormous value of the term APFD denotes the fast fault detection rate.

EFFA Prioritization Model

The previous section explained the initialization of various preset parameters under various constraints needed for the EFF algorithm. This section provides the basic principles of the EFF algorithm and utilizes the same in the TCP environment. Also, it gives information about the functionalities of the hybrid model based on EFFA. Before implementing the EFFA, the FF algorithms need to be understood clearly, as shown in the following section.

Firefly Algorithm

The fireflies get attracted to each other by their brightness value. Thus, the objective function is created based on the brightness emitted.

$$OFV = max\left(f\left(x_{i,j}\right)\right) \qquad \textbf{(11)}$$

Based on this, a new optimization algorithm is created. In Figure-1, the full functionalities of the FF algorithm are illustrated.

Considerations

1. All fireflies are attracted to each other. And fireflies can attract other flies dynamically.
2. Considers the OFV of the problem.
3. The fireflies get attracted only because of their brightness. The other species get attracted to fireflies when their brightness is high.
4. If no brightness is emitted, the flies move randomly in search of bright light.
5. To continue the above process until all the flies, get visited.

Figure-1 illustrates the entire functionalities of FFA used in the software testing environment. Initially, the objective function is defined for finding the solution to the problem. Then, the flies' distance matrix and brightness attraction level are calculated. By encoding them the degree of FF, attraction is calculated. The movements of all flies depend on the brightness value of the flies. The movements of the fireflies are stopped when all the flies get visited. The entire flight paths are accounted for finding the optimal way. Thus, the optimal TCs are represented by the shortest path of the flies.

Extended Firefly Algorithm

The FF algorithm is extended into EFFA which provides importance to the searching ability. The optimal flies are obtained from local to global candidate set. The distance customization is calculated for choosing the optimal move to get the next target. The entire task of the EFFA is given below:

The OF is to be optimized and formulated using the brightness value. The fitness function value of the EFFA is created by:

$$max\left(f(x_{i,j})\right) = max_k\left\{\frac{W_{i-1}}{Random(\cdot)} * Brightness_{i,j} | i-1 \in arrnaged, i \in unarranged\right\}$$

(12)

Were,

W_{i-1} →denotes theTC weight

$i-1$ →indicates the test case number

$Arranged$ → says the index of the TC; based on that, the priority is determined.

$UnArranged$ → says the index of the TC where their priority is not specified.

$Random(\cdot) =$

$f(x_{i,j})$ → is the fitness function value for $x_{i,j}$

k → number of test cases with highest FF value, till not visited and included in $Set_{candidate}$.

The EFF algorithm is written as a sequence of steps, as:

Algorithm_EEFA( )

{

1. Initialize the OFV, BV, FA.
2. Set all the parameters $\boldsymbol{\beta, \alpha, \gamma, \varepsilon^t}$, and maximum generation.
3. The $\boldsymbol{X_i}$ and $\boldsymbol{X_j}$ has the information about the firefly agent $x_{i,j}$.
4. Let T_{best} be the optimal test sequences, executed in the order.
5. Update BV, OF, and FA by creating the $f(x_{i,j})$ for the $agent-j$ of the $testcase-i$.
6. Apply $Brightness_{i,j} = w_i / \left(\frac{n_{i,j}}{\Sigma_k n_{k,j}} * log\frac{|D|}{|\{d:d \ni t_{i,j}\}|}\right)$ for initializing $x_{i,j}$ value as the entry of the FA.
7. A node is called X_i when reaches the FA. The formula used to update the distance of X_i is

$$X_i^{t+1} = X_i^t + \beta e^{-\gamma S_{i,j}^2}\left(X_i^t - X_j^t\right) + \alpha\varepsilon^t \qquad \textbf{(13)}$$

In the above equation, β and α represent the constants. The absorbing the light rate is denoted by β, and is commonly considered as **1**. The value of $\alpha \in [0,1]$, ε is the random factor used for constant distribution. The attraction coefficient is denoted as γ, similarity coefficient $S_{i,j}$ finds the similarity between X_i and X_i, stored in M_s.

8. Next, Node is selected using a selection strategy, follows the steps as:
   The distance is calculated for nodes in $Set_{candidate}$ and X_i^t using the equation-(13)
   Choose the Node having the smallest distance X_{i+1}^t for $(i+1)^{th}$ sequence.
   Based on the movement of FA, the path is recorded, and the corresponding Node's distance $X_{i+1}^t$is updated in the hybrid model.
   Repeat all processes given in step-5 until the $UnArranged$ becomes empty.
9. Obtain the set of all optimal sequences, with:
   Restore all initial constraints with the position of the TC is to starting position.
   Repeat the above processes given in steps 1 to 6.
   Return the result as the optimal test sequences from the entire flight path.

Datasets

To implement, experiment, and evaluate the performance of the proposed EFFA, some of the benchmark datasets are used. Some of the Software Artifacts Infrastructure (SAI)

repositories are Flex [20], GZIP [20, 21], and GREP [20, 22], with their program execution commands, which are referred from [23]. The details of the dataset are given in Table-2.

Table-2. Dataset and Details

Dataset	Error Version	No. of. Test Cases
Flex [20]	21	567
GZIP [20, 21]	55	217
GREP [20, 22]	17	809

Experimental Results and Discussion

In addition to the experiment carried out using the benchmark datasets, the proposed EFF algorithm is evaluated by applying the same in real-time applications such as Hospital Management and Library Management systems. The total size of the HM and LM application is 74702 and 715682, respectively. The following are the different evaluation metrics used to estimate the performance of the proposed method for prioritizing and minimizing test cases. For evaluating the performance of the proposed EFF algorithm, some of the performance measures are calculated. Also, the performance of the proposed method is estimated according to the FDR, precision, recall, and F-measure.

Precision

Failure in the TCs is measured through Precision value which gives the ratio of failure out of total set $TCF \subseteq TC$, which has been selected in a test set $TS \subseteq TC$

$$precision = \frac{T_P}{F_P + T_P}$$

Where $T_P$represents the true positive and F_P represents the false positive.

Recall

Recall is the overall wrong classification of TCs detection from the test set. The more failure test cases are accurately chosen based on the highest score is:

$$Recall = \frac{T_P}{F_N + T_P}$$

Where T_P represents the true positive and F_N represents the false negative.

F1-Score

One of the other performance measures is F-Score, which is also calculated to evaluate the performance using the following formula as:

$$\boldsymbol{F - measure = 2 \times \left(\frac{Precision \times Recall}{Precision \times Recall}\right)}$$

Fault Detection Rate (FDR)

One of the important factors used in this paper is FDR, used to verify the overall performance of the test cases. The average value obtained for the FDR is from 0 to 100. The highest value indicates the betterness and lowest value indicates the poorness of the method. It can be understood by the mathematical expressions, such as:

Let

T -> test suit = {tc1, tc2, tc3,…tcn}and

F = m faults find by T
Where T' = order(T)
TF(i) denotes the faults obtained from i^{th} test-case and the fault is fault-i. The average FDR obtained during the execution of TC is estimated as:

$$Avg(FDR) = 1 - \frac{TF_i + TF_i + \cdots + TF_i}{nm} + \frac{1}{2n}$$

The performance of the proposed EFF algorithm is compared with the other methods regarding APFD and execution time. The comparison has been carried out using various benchmark datasets.

For evaluating the performance, the existing methods FA, GREEDY, HRA, PSO, and the proposed EFFA are examined individually over various datasets, and the APFD is calculated. Table-4 shows the performance comparisons on the FIEX dataset. Table-5 shows the performance comparisons on the GREP dataset. Table-6 shows the performance comparisons on the GZIP dataset. From the overall comparison, it has been found that the proposed EFFA obtained a very good APFD than others.

Table-3. Performance Comparison of APFD For FIEX Dataset

APFD	FA	GREEDY	HRA	PSO	EFFA
	0.954	0.942	0.956	0.949	0.976

Table-4. Performance Comparison of APFD For GREP Dataset

APFD	FA	GREEDY	HRA	PSO	EFFA
	0.952	0.933	0.953	0.940	0.968

Table-5. Performance Comparison of APFD For GZIP Dataset

APFD	FA	GREEDY	HRA	PSO	EFFA
	0.951	0.934	0.949	0.939	0.971

Once again, the performance is compared by repeating the execution for various datasets for various test cases from multiple test case sets. Table-6 shows the performance comparison among the earlier and proposed EFFA on FIEX dataset with 20 TCs. For all the TCs, the obtained APFD using EFFA is high than others.

Table-6. Performance Comparison of APFD For FIEX Dataset (20 Test Cases)

	GREEDY	PSO	FA	H - RA	EFFA
1	94.5	95.5	96.1	95.9	97.5
2	92.5	95.8	96.0	96.3	97.1
3	91.9	95.2	95.5	95.4	96.8
4	93.2	95.0	95.5	95.4	96.3
5	93.3	95.6	95.5	95.4	96.6
6	94.3	94.8	95.5	95.4	96.0
7	94.5	95.9	95.6	95.7	96.2
8	93.1	94.5	95.7	95.8	97.3
9	93.3	93.9	95.6	95.6	96.7
10	94.5	93.9	95.5	95.5	97.9
11	93.1	95.4	95.6	95.6	97.2
12	93.8	95.4	95.5	95.5	96.5
13	93.9	94.4	95.6	95.4	96.2
14	94.0	94.3	95.5	95.6	97.8
15	94.2	94.7	94.7	94.6	95.3
16	94.3	94.6	95.9	96.0	98.6
17	93.3	95.5	95.6	95.7	97.7
18	94.0	94.6	95.7	95.8	96.9
19	94.0	94.6	95.9	96.4	97.5
20	93.9	95.9	96.0	96.1	98.8

The comparison of performance factors with the earlier methods is given in Table-7 with respect to GREP dataset with 20 TCs. For all the TCs, the obtained APFD using EFFA is high than others. Table-9 shows the comparison of performance factors between earlier and proposed EFFA on GZIP dataset with 20 TCs. For all the TCs, the obtained APFD using EFFA is higher than others.

Table-7. Performance Evaluation (FDR For GREP (20 Test Cases))

	GREEDY	PSO	FA	H - RA	EFFA
1	94.4	94.6	95.0	94.9	96.2
2	92.5	94.7	95.8	95.0	96.9
3	92.4	93.8	95.3	95.6	96.4
4	93.2	94.5	95.0	95.1	97.1
5	92.4	93.9	95.5	95.4	96.8
6	93.4	95.0	95.4	95.4	96.3
7	92.4	94.9	94.6	94.9	95.8
8	93.4	92.9	95.4	95.6	97.6
9	93.5	93.5	95.4	95.2	97.9
10	93.4	94.2	94.5	95.2	96.1
11	93.3	94.3	95.7	94.9	97.4
12	93.4	94.0	95.0	95.7	96.4
13	94.4	93.1	95.2	95.1	96.5
14	93.3	93.3	94.5	95.2	97.3
15	92.1	93.9	94.4	94.8	95.6
16	93.2	94.2	95.4	94.9	96.4
17	92.1	95.0	95.1	95.1	96.7
18	93.2	94.9	95.2	95.2	97.2
19	92.1	94.0	94.7	95.0	96.2
20	93.2	93.0	95.4	95.8	96.9

Table-8. Performance Evaluation (FDR For GZIP (20 Test Cases))

	GREEDY	PSO	FA	H - RA	EFFA
1	92.2	93.7	95.0	95.1	96.7
2	92.4	94.0	94.5	94.5	95.3
3	93.1	93.8	94.6	94.5	95.6
4	92.9	93.4	95.4	95.4	96.1
5	92.2	94.4	95.0	95.2	96.9
6	93.4	93.0	94.5	94.3	95.8
7	93.3	94.8	95.5	95.3	96.4
8	92.2	94.7	95.5	95.4	96.8
9	93.0	93.6	94.4	94.3	95.2
10	93.4	93.5	94.4	95.2	96.6
11	93.1	93.3	95.4	95.6	96.3
12	94.2	94.8	95.4	95.4	96.1
13	92.9	92.9	94.6	94.3	95.5
14	93.4	93.5	95.5	95.7	97.2
15	92.2	93.8	95.5	95.7	96.9
16	93.3	94.5	94.8	94.3	95.9
17	92.2	93.8	94.7	95.1	96.1
18	93.5	93.4	94.4	94.8	95.7
19	94.3	94.4	95.2	95.5	96.4
20	93.4	92.9	95.1	95.0	96.0

Findings

1. EFFA obtained the highest mean APFD and less time complexity for the custom dataset.
2. EFFA obtained higher APFD for multiple datasets comparing with GREEDY, PSO, FA, HFA, and EFFA.
3. EFFA obtained high APFD for 20 TCs from various datasets.

Conclusion

This study is dedicated to the development and deployment of a test case prioritization framework leveraging nature-inspired algorithms. Specifically, an Extended Firefly Algorithm (EFFA) is engineered as a hybrid model, combining customized distance metrics and data emphasis with the Firefly Algorithm (FFA) to optimize the efficiency of the prioritization process. Implemented within the DOTNET software environment, the EFFA methodology undergoes rigorous validation to assess its efficacy. Performance evaluation metrics including Average Percentage of Faults Detected (APFD), time complexity across multiple datasets, and total test cases (Tcs) are meticulously analyzed. Comparative assessments against established methodologies provide valuable insights into the superiority of the proposed EFFA. The experimental findings underscore the effectiveness of the EFFA approach, as evidenced by superior APFD scores and reduced time complexities when compared to existing methods.

Future Work

The precision, recall, and F-score values are compared in the experiment, and the performance is evaluated in future work.

References

[1]. G. Rothermel and M.J. Harrold. Analyzing regression test selection techniques. IEEE Transactions on Software Engineering, Vol. 22, No. 8, pp. 529-551.

[2] R. Pressman. Sojtw. Eng.: A Practitioner's Approach. McGraw-Hill, New York, NY, 1987.

[3] H.K.N. Leung and L. White. Insights into regression testing. In Proc. of the Conf. on Software. Maintenance., pages 60-69, October 1989.

[4] K. Onoma, W-T. Tsai, M. Poonawala, and H. Suganuma. Regression testing in an industrial environment. Comm. of the ACM, 41(5):81-86, May 1988.

[5] G. Rothermel, M.J. Harrold, J. Ostrin, and C. Hong. An empirical study of the effects of the minimization on the fault detection capabilities of test suites. In Proc. of Conf. on Sojtw. Maint., pages 34-43, November 1998.

[6] T. Joachims, "Optimizing search engines using clickthrough data". In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02). ACM, pp. 133-142. doi: 10.1145/775047.775067

[7] I. H. Witten, E. Frank, and M. A. Hall. 2017. Data Mining: Practical Machine Learning Tools and Techniques (4th ed.). Morgan Kaufmann Publishers Inc. ISBN: 978-0-12-804291-5

[8] D. W. Hosmer Jr., S. Lemeshow, and R. X. Sturdivant. Applied logistic regression. Vol. 398. John Wiley & Sons, 2013. doi:10.1002/9781118548387

[9] I. Goodfellow, Y.Bengio, A. Courville, "Deep learning". Vol. 1. Cambridge: MIT press, 2016. ISBN: 0262035618
[10] H. K. N. Leung and L. White, "Insights into regression testing (software testing)," Proceedings. Conference on Software Maintenance, 1989, pp. 60-69. doi: 10.1109/ICSM.1989.65194
[11]. Nayak, S., Kumar, C., & Tripathi, S. (2017). Enhancing Efficiency of the Test Case Prioritization Technique by Improving the Rate of Fault Detection. Arabian Journal for Science and Engineering, 42(8), 3307–3323.
[12]. Ammar, A., Baharom, S., Ghani, A.A.A. and Din, J., 2016, December. Enhanced weighted method for test case prioritization in regression testing using unique priority value. In 2016 International Conference on Information Science and Security (ICISS) (pp. 1-6). IEEE.
[13]. Ansari, A., Khan, A., Khan, A., & Mukadam, K. (2016). Optimized Regression Test Using Test Case Prioritization. Procedia Computer Science, 79, 152–160.
[14]. Huang, C.-Y., Chang, J.-R., & Chang, Y.-H. (2010). Design and analysis of GUI test-case prioritization using weight-based methods. Journal of Systems and Software, 83(4), 646–659.
[15]. Jeffrey, D., & Gupta, N. (2008). Experiments with test case prioritization using relevant slices. Journal of Systems and Software, 81(2), 196–221.
[16]. Taneja, D., Singh, R., Singh, A. and Malik, H., 2020. A Novel technique for test case minimization in object oriented testing. Procedia Computer Science, 167, pp.2221-2228.
[17] Mohapatra, S.K., Mishra, A.K. and Prasad, S., 2020. Intelligent Local Search for Test Case Minimization. Journal of The Institution of Engineers (India): Series B, pp.1-11.
[18]. Hashim, N.L. and Dawood, Y.S., 2018. Test case minimization applying firefly algorithm. International Journal on Advanced Science, Engineering and Information Technology, 8(4-2), pp.1777-1783.
[19]. Hutchins, M. and Foster, H. and Goradia, T. and Ostrand, T., "Experiments on the effectiveness of dataflow- and control flow-based test adequacy criteria", Proceedings of the 16th International Conference on Software Engineering, May, 1994, pages 191-200.
[20]. https://sir.csc.ncsu.edu/portal/usage.php
[21] https://docs.oracle.com/cd/E36784_01/html/E36870/gzip-1.html
[22] https://docs.oracle.com/cd/E88353_01/html/E37839/gzgrep-1.html
[23] https://alvinalexander.com/blog/post/linux-unix/how-grep-search-compressed-gzip-gz-text-file/
[24]. P. Bourque and R.E. Fairley (eds), Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society, 2014; Available athttp://www.swebok.org
[25]. Piscataway, NJ, USA: The Institute of Electrical and Electronic Engineers, Inc. (IEEE). Available at: http://www.computer.org/portal/web/swebok.
[26]. P. R. Srivatsava, B. Mallikarjun, and X. S. Yang, "Optimal test sequence generation using firefly algorithm," Swarm Evol. Comput., vol. 8,pp. 44– 53, Feb. 2013.
[26]. Amir Hossein Gandomi, Xin She Yang, Amir Hossein Alavi, Mixed variable structural optimization using Firefly algorithm, Computers and Structures 89 (23–24) (2011) 2325–2336.

[27]. M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, H. N. A. Hamed and M. D. Mohamed Suffian, "Test Case Prioritization Using Firefly Algorithm for Software Testing," in IEEE Access, vol. 7, pp. 132360-132373, 2019.

[28] SK Hasane Ahammad, , D. K. J.Saini, Phishing URL detection using machine learning methods, Advances in Engineering Software,Volume 173, 2022,103288, ISSN 0965-9978, https://doi.org/10.1016/j.advengsoft.2022.103288

[29] Yadav, P, S. Kumar, and D. K. J.Saini. "A Novel Method of Butterfly Optimization Algorithm for Load Balancing in Cloud Computing". International Journal on Recent and Innovation Trends in Computing and Communication, vol. 10, no. 8, Aug. 2022, pp. 110-5, doi:10.17762/ijritcc.v10i8.5683.

[30] Jang Bahadur, D. K. , and L.Lakshmanan. "Enhancement of Quality of Service Based on Cross-Layer Approaches in Wireless Sensor Networks". Journal of Theoretical and Applied Information Technology ISSN: 1992-8645, 15th October 2022. Vol.100. No 19 ,

[31] Jang Bahadur, D. K. , and L.Lakshmanan. "Virtual Infrastructure Based Routing Algorithm for IoT Enabled Wireless Sensor Networks With Mobile Gateway". International Journal on Recent and Innovation Trends in Computing and Communication, vol. 10, no. 8, Aug. 2022, pp. 96-103, doi:10.17762/ijritcc.v10i8.5681.

[32] Shailesh Kamble, Dilip Kumar J. Saini, Vinay Kumar, Arun Kumar Gautam, Shikha Verma, Ashish Tiwari & Dinesh Goyal (2022) Detection and tracking of moving cloud services from video using saliency map model, Journal of Discrete Mathematical Sciences and Cryptography, 25:4, 1083-1092, DOI: 10.1080/09720529.2022.2072436

[33] Jang Bahadur, D. K. , and L.Lakshmanan "Improve Quality of Service in Optimization of Job Scheduling using a Hybrid Approach" Xi'an Shiyou Daxue Xuebao (Ziran Kexue Ban)/ Journal of Xi'an Shiyou University, Natural Sciences Edition ISSN:1673-064X Vol: 65 Issue 01 | 2022 DOI 10.17605/OSF.IO/DW57E

[34] Jang Bahadur, D. K. , and L.Lakshmanan "Wireless Sensor Network Optimization for Multi-Sensor Analytics in Smart Healthcare System" Specialusis Ugdymas, ISSN NO: 1392-5369 Vol. 1 No. 43 (2022)

[35] Saini D., An IoT and fog computing enabled intelligent health care monitoring system to the cloud storage, Turkish Journal of Computer and Mathematics Education (TURCOMAT) Vol12 No10(2021), 2085-2091(ISSN: 1309-4653) Scopus Indexed https://turcomat.org/index.php/turkbilmat/article/view/4722
DOI: https://doi.org/10.17762/turcomat.v12i10.4722

[36] Saini D, Novel Approach of Data Transformation for Privacy Assurance using Optimization of Genetic Algorithm, Design Engineering (Toronto) ISSN No:0011-9342 Scopus Indexed

[37] D. K. Jang Bahadur Saini, P. Patil, K. Dev Gupta, S. Kumar, P. Singh and M. Diwakar, "Optimized Web Searching Using Inverted Indexing Technique," 2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT), 2022, pp. 351-356, doi: 10.1109/CSNT54456.2022.9787680.

[38] D. Siddharth, D. K. J Saini, P. Singh, An Efficient Approach for Edge Detection Technique using Kalman Filter with Artificial NeuralNetwork, International Journal of Engineering, Transactions C: Aspects Vol. 34, No. 12, (2021) 2604-2610

[39] "A comparative study on energy-efficient clustering based on metaheuristic algorithms for WSN", International Journal of Advanced Technology and Engineering Exploration, vol. 9, no. 86, 2022. Available: 10.19101/ijatee.2021.874823.
[40] DilipKumar Jang Bahadur, L. Lakshmanan, A Novel Method for Optimizing Energy Consumption in Wireless Sensor Network Using Genetic Algorithm, Microprocessors and Microsystems, 2022,104749,ISSN0141-9331 https://doi.org/10.1016/j.micpro.2022.104749.
[41] Sharma, P. ., R. K. Yadav, and D. J. B.Saini. "A Survey on the State of Art Approaches for Disease Detection in Plants". International Journal on Recent and Innovation Trends in Computing and Communication, vol. 10, no. 11, Nov. 2022, pp. 14-21, doi:10.17762/ijritcc.v10i11.5774
[42] H. Chanyal, R. K. . Yadav, and D. K. J. Saini, "Classification of Medicinal Plants Leaves Using Deep Learning Technique: A Review", International Journal of Intelligent Systems and Applications in Engineering, vol. 10, no. 4, pp. 78–87, Dec. 2022.