OAPR-HOML'1: Optimal automated program repair approach based on hybrid improved grasshopper optimization and opposition learning based artificial neural network

T.MAMATHA[1], B. RAMA SUBBA REDDY[2], C SHOBA BINDU[3]
[1]Research Scholar, Dept. of CSE, JNTUA University, Anantapuramu, AP, INDIA
mamathat7@gmail.com
[2]Professor & HOD, Dept. of CSE, SV College of Engineering, Tirupati, INDIA
rsreddyphd@gmail.com
[3]Professor , Dept. of CSE , JNTUA University, Anantapuramu, AP, INDIA
shobabindhu@gmail.com

Abstract
Over the last decade, the scientific community has been actively developing technologies for automated software bug fixes called Automated Program Repair (APR). Several APR techniques have recently been proposed to effectively address multiple classroom programming errors. However, little attention has been paid to the advances in effective APR techniques for software bugs that are widely occurring during the software life cycle maintenance phase. To further enhance the concept of software testing and debugging, we recommend an optimized automated software repair approach based on hybrid technology (OAPR-HOML'1). The first contribution of the proposed OAPR-HOML'1 technique is to introduce an improved grasshopper optimization (IGO) algorithm for fault location identification in the given test projects. Then, we illustrate an opposition learning based artificial neural network (OL-ANN) technique to select AST node-level transformation schemas to create the sketches which provide automated program repair for those faulty projects. Finally, the OAPR-HOML'1 is evaluated using Defects4J benchmark and the performance is compared with the modern technologies number of bugs fixed, accuracy, precession, recall and F-measure.

Keywords:
Fault location identification, AST node-level transforms, software testing, automatic program repair, grasshopper optimization.

1. Introduction

With the wider use of software, there are concerns about the functionality and performance of the software. To ensure more reliability, the program constantly checks for bugs and errors. On the other hand, it is almost impossible to create error-free software without considering scientific and ethical development methods. Software Reliability Development Process Over the last 30 years, several software reliability development models have been proposed to determine the course of development. [1]. SRGM can be very useful in that it helps management make important decisions such as checking resource allocation and changing software publishing time. Software testing is a very popular method used to improve software quality. Software testing combines two modes of reliability and provides simple and practical automated testing and writing test with high cost / performance ratios. Therefore, software testing is widely used in the software industry [2] [3]. Because software testing is based on the software process model, this does not guarantee that all software vulnerabilities will be fixed. Therefore, one issue that requires special attention is the reliability of the software obtained after testing. However, the purpose of software testing in the software reliability development process is not clear. It supports the latest software reliability testing by examining the quantitative link between software testing and software reliability [4]. Software testing methods have traditionally been divided into white and black box tests. The white box test can be used when the tester has access to internal data structures and instructions, while the black box test does not indicate knowledge of internal performance [5] [6].

Software testing is an important tool to ensure software quality. There are functional testing, data flow testing, limit value testing, random testing and many other software tests. During software testing, trial cases are selected and the software is tested [7]. Defects are detected and removed one by one, which improves the reliability of the test program. However,

Manuscript received April 5, 2022
Manuscript revised April 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.4.33

the purpose of routine software testing for the reliability of given software is not clear [8]. One reason for this is that software testing not only confirms the existence of software vulnerabilities, but also the ability to confirm the absence of software vulnerabilities. In principle, the number of software errors remaining in the program is unknown. Another reason is the ambiguous reliability of the software between the process profile, the test profile, and the working profile [9]. Debugging is the process of detecting and removing software code errors (also known as "errors") that cause crashes. Troubleshooting is the use of bugs and errors to prevent software or system crashes. Configuration is difficult when different subsystems or modules are tightly connected, as any modification of one module can cause errors in another. Sometimes a debugging program will take longer than the code [10][11]. Incorrect debugging models improve the performance of the software but reduce the user expectation as the prediction speed can be fixed or gradually reduced [12]. Such an assumption is not always correct; as it will increase over time until an input error enters the test process. Troubleshooting was not initially familiar with the operation and use of the software. After removing the errors, they can read the program and introduce new errors [13][14]. However, after learning the error correction program and removing many simple errors, it rarely makes new errors. Therefore, the performance of the error rate increases over time and does not decrease [16]. If experienced software programs have found bugs, they can be found and removed. Each debugging method is a new process that does not take into account debugging errors. Therefore, it is necessary to eliminate errors to understand the settings of the newly released software. Enriches knowledge about malicious programs and programs and features and uses such knowledge to prevent new bugs [17]. For example, the test phase creates trial cases that allow the system to test whether it meets the initial requirements. If the system is modified later, for example, the initial tests may be repeated to see if there are any initial requirements to improve its performance [18] [19]. Therefore, the knowledge gained at the meeting should be carefully compared to the actual and expected outcomes of some of the sub-issues that contribute to the rest of the program or program development cycle. Unfortunately, the answer is that this very useful information is usually ignored after a bug fix session. Therefore, we can conclude that software development will allow you to spend more than 50% of your time doing things that do not meet the basic life cycle principle [20].

Our contributions: An optimal automated program repair approach is proposed using hybrid techniques (OAPR-HOML'1). The main contributions of proposed OAPR-HOML'1 technique are list as follows:

- An improved grasshopper optimization (IGO) algorithm is used for fault location identification in the given test projects.
- Opposition learning based artificial neural network (OL-ANN) technique is used to select AST node-level transformation schemas to create the sketches which provide automated program repair for those faulty projects.
- Finally, OAPR-HOML'1 is evaluated using Defects4J benchmark and the performance is compared with the existing state-of-art techniques in terms of number of bugs fixed, accuracy, precession, recall and F-measure.

The rest of the paper is organized as follows: Sect. 2 describes the recent works related to APR techniques. Sect. 3 provides the problem methodology and system model of proposed OAPR-HOML'1 technique. Sect. 4 gives the working function of proposed OAPR-HOML'1 technique with the proper mathematical analysis. Then, the simulation results of proposed and existing techniques are discussed in Sect. 5. Finally, the paper concludes in Sect. 6.

2. Related works

Li et al., [21] the reliability model of the test coverage software indicates not only the incomplete error change but also the environmental uncertainty based on the persistent toxic process. Normally, the program is tested in a specific control environment, but developers can use it in working conditions unknown to different users. Several models of NHPP software development reliability have been developed to measure software reliability, but the general assumption of these models is that the work environment is similar to a growing environment. In fact, the unpredictability of software operating conditions significantly affects software reliability and the unpredictable environment. So when a software system operates in a field environment, its

reliability in terms of theoretical reliability and other areas usually differs from similar software. This study proposes a new model based on the speed of detection of a test coverage error and investigates the ambiguity of the coverage ID in terms of operating conditions. They compare the performance of a particular model with many existing NHPP SRGMs. Li et al., [22] proposed a state-based error localization approach. The executed trace to be analyzed for the observed failure is then compressed using a set of trace points at each stage of the dynamic pro graph. Zhang et al., [23] studied the most difficult problem of testing a domain distribution loop in an SDN environment. They have introduced two new test protocols that can be used for domain loop tests. Both protocols are protected, meaning that each protocol protects personal information about its location and configuration. The first protocol, based on random sampling with very little error power, was very effective in rapidly reducing sample errors Although the first is less effective, the second protocol provides 100% accuracy of results based on a secure package cross-test. They provided rigorous evidence to ensure security and accuracy, and our protocols perform well when tested with real-time network data. Gazzola et al., [24] proposed a method that regulates knowledge in this area through a review of a group of 108 documents, software automation techniques, descriptions of methods and approaches, their relatively representative examples, and a review of open challenges and empirical evidence published to date. Experiments and research the combination of methods, techniques and heuristics and automated repair techniques creates a growing and multifaceted research framework. Kong et al., [25] studied detail about 180 seeds for 17 small and large projects and how to correct real mistakes. They examined the repair results of five representative automated programs, including GenProg, RSRepair, Brute-force-based technique, AE and Kali based technology, on repair results. We will further examine the results of various material programs and trial rooms regarding the performance and effectiveness of program repair techniques.

Qiu et al., [26] presented a computers work long hours at certain loads, under controlled stress conditions, to accelerate database malfunctions. Second, it examines and formulates models of mathematical relationships of data species. Such relationship models are used for TTF / MTTF extrapolation under different operating conditions and are needed to reduce system reliability assessment time. Jiang et al., [27] localize and address these gaps and compare these strategies with existing strategies. The results indicate future automated program repair instructions. Over the last decade, more and more attention has been paid to auto-repair techniques designed to automatically create the correct patches in real-world defects. Various technologies and tools have been proposed and developed. Gupta et al., [28] presented the report proposes an approach to multi-case trial testing and various types of distorted coverage and compares it with current approaches. SPEA-2, NSGA-2 and VEGA algorithms were used for test analysis, and tests were performed on malicious 4j database applications. The results of the study indicate that the proposed approach has the potential to reveal more errors compared to existing approaches. All errors found by incorrect estimation of localization can be improved or compared with existing approaches. Software development is an ongoing process. Testing and debugging at all stages of software development are the most important steps.

The main purpose of testing is to detect maximum errors quickly. After fixing the error, it should be removed using the appropriate debugging method. Two steps are completed one after the other, which requires different information. Kim et al., [29] proposed the effectiveness of context-based change program (CCLA) technology in selecting changes, correcting location selection, and integrating changes that are key features that drive bit hunt gap. The CCA collects short sub-modifications and their AST contexts and uses them to identify words only if the CCA is used. They evaluated CCA performance with a unique collection of 54k (221k) updates, from links written by approximately 5K men. Results show that CCA corrects 90.1% of the changes required for the test packet connection, and less than 5% of changes occur intentionally. They found that collecting additional changes would only be useful if effective search engine navigation supported the environment. In addition, the CCA Repair Model 44J found 70% more defects at the patch repair site than the SCFL tool alone. Caballero et al., [30] proposed a method for debugging and testing in integrated frameworks, where each step creates useful information for the other and re-uses the results of each step. Some

frameworks are instantly common in very different programming languages. Erlong (Functional), Java (Required, Object Oriented), SQL (Data Request). Test results obtained using the Erlong program confirms the operation of the framework. It provides a common integrated framework for troubleshooting and testing. This simplifies each step and increases efficiency at various stages of the software development cycle and reduces overall effort. Ye et al., [31] proposed about the features of Quixbugs and test the program for 10 repair tools; Our main results are: 16/40 Non-standard applications Quixbugs may contain at least one test package; Quixbugs produced a total of 338 trusted patches using the tools reviewed, of which 53.3% matched the patches according to our guidelines. Liu et al., [32] proposed a method how the most recent measurements of relative APR systems are. The impact of repairs is unknown, and the estimated dimensions are often unclear because the design results (in the approach and evaluation system) are not always known. To significantly reduce the reliability of design results in a program repair approach, we complete a comprehensive review of patch construction systems and offer eight criteria for evaluating the performance of APR equipment. Finally, they present test data of 11 major program repair systems to highlight some of the warnings and specific dimensions in the literature. They believed that the widespread adoption of these measures in the community would lead to the development of practical and reliable tools for program repair. The summary of research gap is given in Table 1.

Table 1 Research Gap summary

Ref	Methodology	Testing and debugging	Parameters
21	NHPP software reliability growth models and Improved NCD method	Fault detection rate	Number of bugs fixed
22	state-based fault-localization approach	minimum debugging frontier set (MDFS)	Number of bugs fixed
23	Software-Defined Networking	inter-domain loop tests	Precession
24	Generate-and-validate techniques	repairing and healing	Number of bugs fixed
25	GenProg, RSRepair, Brute-force	seeded and real faults	Precession
26	stress testing method	TTF or Mean TTF	Number of bugs fixed
27	Defects4J method	Patch correctness	Precession
28	Multi-objective selection	fault localization score	Precession
29	Context-based change application (CCA)	Navigating patch search space	Number of bugs fixed
30	Declarative debugging method	single unified framework	Number of bugs fixed
31	automated patch correctness assessment techniques	Quixbugs	Precession
32	Patch generation system	Eight evaluation matrices	Number of bugs fixed

3. Proposed methodology and System architecture

3.1 Research Gap

Hua [33] have introducing SketchFix, which strongly combines the steps of generation and verification and uses considerable time management behavior for large-scale repair candidates. SketchFix uses the Edsketch Sketch Machine to fill in query search gaps. Disadvantages Experimental evaluation using the 4J mark shows that SketchFix significantly reduces the amount of reassembly and re-implementation compared to other approaches and performs better exposure management at AST node level grains. Several APR techniques have recently been proposed to improve the quality of software testing. [21]-[33]. A variety of hardware and program error classes can be effectively corrected. However, little attention has been paid to the advances in effective APR techniques for software bugs that are widely occurring during the software life cycle maintenance phase. The problem with software is that it automatically fixes bug fixes to significantly reduce debugging costs and improve software. To address this issue, the trial packet-based repair tool reviews the test package provided by Oracle and modifies the non-standard input program to confirm the entire trial package. However, according to recent empirical studies, sketching can be useful [33] is not fully satisfactory, particularly for Java. In addition, the techniques and tools of APR that make knowledge collection challenging differ in many ways. Therefore, in this paper, we focus on creating the optimal APR hardware to improve the performance of a software test run. The main objectives of the proposed OAPR-HOML technology are:

The hybrid optimization and machine learning techniques utilized to improve the quality of software testing and debugging.

- The machine learning technique used to enhance the reliability of software testing.
- To study and analyze the different optimization and machine learning techniques for ARP
- Introduce an optimization technique for compute optimal location of faults which reduce the fault searching cost, testing cost and time.
- Introduce a machine learning technique to provide the automatic fault repair in a program which enhances the accuracy and quality of testing.
-

3.2 System architecture of proposed OAPR-HOML'1 technique

The system architecture of proposed OAPR-HOML'1 technique is shows in Fig. 1 which gives the detailed structure of working function. First, the proposed OAPR-HOML'1 technique computes the exact fault location to identify the suspicious part using IGO algorithm. Then, we applied OL-ANN technique to compute the AST node-level transformation schemas for sketches creation. These maps are drawn directly and executed against the test kit. We discuss the schemes used after the AST node change position to address suspicious locations in the test program.

- Expression Transformer: If the false statement contains AST node variables, fixed values, or field defects, the node object will move to the specified hole.
- Operator Transformer: If there is a binary expression with an arithmetic operator in the wrong line, that binary expression will move into the hole.
- Overloading Transformer: If the incorrect account has a reset mode, specify the parameter types and different types of parameters and call to create exposure holes.
- Condition Transformer: The subsection refers to the left and right external expressions associated with the operator.
- If-condition transformer: Use the if-condition before the false statement with the conditionality hole.
- Return-statement transformer: Include a response statement before an incorrect statement. If the current system revenue type is empty, file a blank income statement, otherwise an impact hole based on the system revenue type will be provided.
-

4. Optimal automated program repair approach (OAPR-HOML'1)

4.1 Suspicious location detection using IGO algorithm

Selection of characteristics is a prerequisite for many machine learning tasks, such as classification and clustering. It helps maintain important and relevant characteristics and avoids unnecessary and inappropriate features. For classification, feature selection is a subset of features that reflect the most important and unique characteristics of events in each class. Here we applied grasshopper optimization algorithm for selecting the optimal features. The locust optimization algorithm implements the optimal behavior of locusts in nature. Like the other set of algorithms, each locust represents a candidate solution that is generated approximately at the beginning and then according to the evaluation process; become the best locust leader. The leader will attract him to the other locusts. Gradually all the locusts move towards the cats. Below is the mathematical structure of IGA.

$$Y_j = R_j + F_j + B_j \tag{1}$$

The social interaction is denoted by R_j, the force of gravity is denoted by F_j, and B_j indicates air consumption is defined.

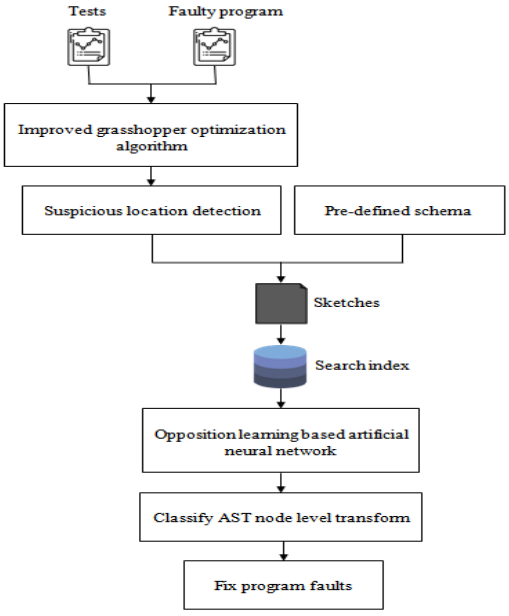

Fig. 1 System architecture of proposed OAPR-HOML'1 technique

In the locust movement process, social interaction plays an important role in R_j, which can be achieved.

$$R_j = \sum_{i=1, i\neq j}^{M} s(c_{ji})\hat{c}_{ji} \tag{2}$$

The c_{ji} indicate the Euclidean distance between two locusts available by $c_{ji} = | y_i - y_j |$. r is the activity used to assess the intensity of social interaction can be assessed as follows

$$r(s) = ge^{\frac{-\tau}{j}} - e^{-s} \tag{3}$$

G denotes gravity, while W denotes attractive length. Below is the formula for calculating the gravitational factor

$$F_j = -f\hat{e}_f \tag{4}$$

g is a gravitational constant that refers to a uniform vector directed toward the center of the earth.
Below is the formula for calculating the wind direction coefficient.

$$B_j = v\hat{e}_l \tag{5}$$

V represents the value of continuous sliding and e_l is unit vector in the direction of the wind. The locust motion formula can be expressed as follows.

$$Y_j = \sum_{\substack{i=1 \\ i\neq j}}^{M} r(| y_i - y_j |)\frac{y_i - y_j}{c_{ji}} - f\hat{e}_f + v\hat{e}_l \tag{6}$$

Let Y_j^c denote the grasshopper position j in the cth dimension. The improved equation is as follows.

$$Y_j^c = d_1\left(\sum_{\substack{i=1 \\ 1\neq j}}^{M} d_2 \frac{va_c - wa_c}{2} r(| y_i - y_j |)\frac{y_i - y_j}{c_{ji}}\right) + \hat{t}_c \tag{7}$$

Slope coefficient parameters d1 and d2 are used to simulate the process of locust locusts gradually approaching the feed position and eventually feeding on the mass. As the number of repetitions increases, d1 is used to reduce the search target, while d2 is used to reduce the severity and interference effect. The parameter update formula is as follows

$$d_j = dMAX - w\frac{dMAX - dMIN}{W} \tag{8}$$

Algorithm 1 Suspicious location detection using IGO algorithm

Input	: Position of grasshopper
Output	: Cluster centroid
1	Initialize the parameters $dMAX$, $dMIN$
2	Initialize the initial population
3	Calculate the fitness of each individual of the population
4	Assign t to the individual with the highest fitness value in the initial population
5	Update d_j using $d_j = dMAX - w\frac{dMAX - dMIN}{W}$
6	For each individual do
7	Normalize the distances between individuals
8	Update the position of individual using $Y_j^c = d_1\left(\sum_{\substack{i=1 \\ 1\neq j}}^{M} d_2 \frac{va_c - wa_c}{2} r(| y_i - y_j |)\frac{y_i - y_j}{c_{ji}}\right)$
9	If the individual exceeds the boundaries, bring them back
10	End for
11	Re-evaluate the fitness of each individual in the population
12	If there is a better solution, replace t with it
13	End

The clustering method is defined: C offers a large file.

$$C = c1, c2...c_{ji}...cm \tag{9}$$

dz is the z center of the cluster is called the vector.

$$dz = (dz1, dz2...dzi,..., dzT), dzi \tag{10}$$

Cluster I and C are the length of the cluster.

$$d_{zi} = \frac{\sum_{j=1}^{m}(b_{zj})c_j}{\sum_{i=1}^{s_j} bz_i} \tag{11}$$

The cosine is the standard value I use to calculate the similarity points between two vectors (e.g. document and cluster centroid) because C1 is the document number 1 and C2 is the cluster centroid.

$$\cos(c1,c2) = \frac{\sum_{i=1}^{T} l(T_i,c_1)\times l(T_i,c_2)}{\sqrt{\sum_{i=1}^{T} l(T_i,c_1)^2}\sqrt{\sum_{i=1}^{T} l(T_i,c_2)^2}} \tag{12}$$

The working function of the suspicious location detection using grasshopper optimization algorithm is described in algorithm1.

4.2 Select AST node-level transformation schemas using OL-ANN

A typical neural network of sorting problems consists of 4 different types of layers: input layer, hidden layer, soft-max layer, and output layer. Problems with misdiagnosis and classification can now become a problem for neural network-based classification. First, the two labels are common and lead to error detection, which is a binary classification problem. In this case, a neural network can be built using two different data sets: one for normal data and the other for error data type. The artificial neural network (OL-ANN) classification, which is based on the study of opposition, is used to identify program errors and use them to correct program errors.

Opposition-based Learning (OBL) is a machine intelligence algorithm that reflects its counter-evaluation while taking a closer look at the solution of the current evaluator and the current candidate. It is a new concept for computational intelligence used in many optimization methods to improve solution outcomes. Weight and other parameter values in traditional neural network systems are approximate. Random weight tries to reach the global optimal or approach the optimal weight to achieve the optimal solution with minimal errors. Starting with a random load took more time to reach optimal solutions. As a result, if a random start is too close to the optimal load, this can accelerate the accumulation. To accelerate the integration speed of the neural network algorithm, we used the inverse weight values to obtain the global optimal weight values. Opposition-based learning can be defined as follows:

Assume $z \in [b,a]$ as a real number. The opposition number z is stated as

$$\hat{z} = b + a - z \tag{13}$$

Similarly, the inverse weight M in dimensional space, let

$$Z = (z_1, z_2,..., z_M) \tag{14}$$

Where $z_1, z_2,..., z_M \in S$ and $z_j \in [b_j, a_j] \forall_j \{1,2,...,M\}$. The opposition points are defined with the help of its coordinates.

$$\hat{z} = (\hat{z}_1, \hat{z}_2,..., \hat{z}_M) \tag{15}$$

$$\hat{z} = b_j + a_j - z_j \tag{16}$$

At the same time, opposition-based learning can be defined using inversion values and inverse weight determination, the values $\hat{z} = (\hat{z}_1, \hat{z}_2,..., \hat{z}_M)$ is the function of opposition of $eZ = (z_1, z_2,...z_M)$. Then, both the values are compared as $f(z) \geq f\left(\hat{z}\right)$. Therefore, $\hat{z}$ has good fitness when compared with z. Here, calculate the slope of all loss activities in relation to the weight found in the network. At the correct size of the input nodes,

the hidden layers and the output layers m, l and n are the total size of their input events which is y_{qj} which indicates the q's illustration of the input value of j^{th}. The weight of l^{th} node is the j^{th} node of the hidden input layer which is denoted as u_{lj}. The actual weight of the node i and the output of the output layer l from the indirect output of l are represented as y_{il}.

$$w_{ql} = g\left(Net_{ql}\right) = g\left(\sum_{j=0}^{m} u_{lj} y_{qj}\right) \tag{17}$$

For the i^{th} node, the layer nodes output is

$$x_{qi} = g\left(Net_{qi}\right) = g\left(\sum_{j=0}^{m} z_{il} w_{ql}\right) \tag{18}$$

Where, the standard sigmoid function is selected as the stimulating activity in the following equation,

$$g(y) = \frac{1}{1+e^{-y}} \tag{19}$$

In the below equation the global error function is defined.

$$D = \sum_{q=1}^{Q} D_Q = \frac{1}{2} \sum_{q=1}^{q} \sum_{i=1}^{n} \left(T_{qi} - x_{qi}\right)^2 \tag{20}$$

For sample q the error is indicated as D_Q. The ideal result is represented as T_{qi}. Below equation are the weight correction formulas.

$$\Delta\omega_{il} = \eta \sum_{q=1}^{q} \left(\sum_{i=1}^{n} \delta_{qj} \omega_{il} \right) w_{ql}\left(1 - w_{ql}\right) y_{qj} \tag{21}$$

In general, the learning rate range is 0.1- 0.3 which is represented as η. In the output layers, weight adjustment of neurons is examined as follow,

$$\Delta u_{lj} = \eta \sum_{q=1}^{q} \left(\sum_{i=1}^{n} \delta_{qj} \omega_{il} \right) w_{ql}\left(1 - w_{ql}\right) y_{qj} \tag{22}$$

The algorithm 2 represents the function of opposition learning artificial neural network.

Algorithm 2 Classify transform schema using OL-ANN

Input : Neurons
Output : Weight

1 Initialize the values for the input
2 Compute the opposition number by
$\hat{z} = b + a - z$
3 Evaluate the opposition point using
$\hat{z} = b_j + a_j - z_j$
4 Compare the values as
$f(z) \geq f\left(\hat{z}\right)$
5 Estimate the node of the output layer
6 Select the standard sigmoid function by
$g(y) = \frac{1}{1+e^{-y}}$
7 Obtain the weight corrections
8 End

5. Results and Discussion

To evaluate the performance of proposed OAPR-HOML'1 technique in this section with the open source frequently used datasets for Java-targeted APR research. First, we discuss the description of dataset, and present the implementation details. Then, we discuss the comparative analysis of proposed OAPR-HOML'1 technique with the existing state-of-art techniques.

5.1 Dataset description

The proposed OAPR-HOML'1 technique is evaluate through the Defects4J (v1.0), It does not have its flaws 357 by 5 open source projects are JFreeChart (chart C), Closure compiler (Closure Cl), Apache commons-Lang (Lang L), Apache commons-Math (Math M) and Joda-Time (Time T). In general, the whole Defects4J is too

large for simulation analysis, so we arbitrarily choice faults after every plan to validate the performance of proposed OAPR-HOML'1 technique. The detailed description of each project is discussed as follows:

Table 2 Description of Defects4J (v1.0) benchmark dataset

Project	Number of bugs	Lines of code	Number of test cases
C	26	96,000	2,205
CL	133	90,000	7,927
L	65	22,000	2,245
M	106	85,000	3,602
T	27	28,000	4,130
Total	357	321,000	20,109

From table 2, we know that a amount of bugs, lines of codes and amount of test cases of project C is 26, 96000 and 2205 respectively. The number of bugs, lines of codes and the number of test cases of project CL is 133, 90000 and 7927 respectively. The number of bugs, lines of codes and the number of test cases of project L is 65, 22000 and 2245 respectively. The number of bugs, lines of codes and the number of test cases of project M is 106, 85000 and 3602 respectively. The number of bugs, lines of codes and the number of test cases of project T is 27, 28000 and 4130 respectively. Then, the performance of proposed OAPR-HOML'1 technique is compared with the existing state-of-art APR techniques are SketchFix, SimFix, Astor, Nopol, ACS, HDRepair, CapGen, ELIXIE, kPAR and jKali.

5.2 Analysis of APR techniques repair efficiency

The simulation assessment is performed to investigate the forms of human resistance with proposed OAPR-HOML'1 technique. Known to our knowledge, currently 128 bugs that are ready to be corrected, at least 89 of which is a tool APR. Defects and errors using version 4, prove that they are not corrected, 267 in all literature tool. This is a major challenge to the research MAR. In the following Tables 3-7, 'Y' defines the bug is properly secured, 'X' defines a bug is reasonable but incorrect and '?' indicates bug is not generating fix. Table 3 describes the performance comparison of repair efficiency of proposed and existing APR techniques for chart C project. In this simulation, we select the 20 bugs from chart C project in random manner. It is clearly depicts the repair efficiency of proposed OAPR-HOML'1 technique is very high compare to the existing APR techniques. The proposed OAPR-HOML'1 technique fixes the 15 number of bugs among 20 bugs. However, the existing APR technique fixes the 8, 2, 5, 0, 5, 3, 0, 0, 0, 1 and 0 numbers of bugs for SketchFix, SimFix, Astor, Nopol, ACS, HDRepair, CapGen, ELIXIE, kPAR and jKali respectively. Table 4 describes the performance comparison of repair efficiency of proposed and existing APR techniques for Closure CL project.

In this simulation, we select the 20 bugs from Closure CL project in random manner. It is clearly depicts the repair efficiency of proposed OAPR-HOML'1 technique is very high compare to the existing APR techniques. The proposed OAPR-HOML'1 technique fixes the 15 number of bugs among 20 bugs. However, the existing APR technique fixes the 9, 6, 8, 2, 3, 2, 2, 3, 3 and 2 numbers of bugs for SketchFix, SimFix, Astor, Nopol, ACS, HDRepair, CapGen, ELIXIE, kPAR and jKali respectively. Table 5 describes the performance comparison of repair efficiency of proposed and existing APR techniques for Lang L project. In this simulation, we select the 20 bugs from Lang L project in random manner. It is clearly depicts the repair efficiency of proposed OAPR-HOML'1 technique is very high compare to the existing APR techniques. The proposed OAPR-HOML'1 technique fixes the 15 number of bugs among 20 bugs. However, the existing APR technique fixes the 14, 8, 9, 3, 3, 0, 1, 3, 3, and 2 numbers of bugs for SketchFix, SimFix, Astor, Nopol, ACS, HDRepair, CapGen, ELIXIE, kPAR and jKali respectively.

Table 3 Repair efficiency comparison for proposed and existing APR techniques for chart C project

Projects	OAPR-HOML'1	SketchFix	SimFix	Astor	Nopol	ACS	HDRepair	CapGen	ELIXIE	kPAR	jKali
1	Y	Y	Y	Y							
3	Y	Y	Y	Y			?		X	?	
4	Y	Y	X	Y			?		X		
5	Y	Y	X	Y		?	?		X		
7	Y	Y	X	Y		?					
8	Y	X		X		?				?	
9	Y	X		X	?	Y					
11	Y	X		X	?	Y					
12	X	X	?	?	?	Y					
13	X	X	?	?				?		X	
14	X	X	?	?				?			
15	X	X						?			
17	Y	X									
18	Y	Y			X					Y	
19	Y	Y			X	X				X	
20	Y	Y	X		X	X			X		
22	Y	?	X			X			X		
23	X	?	X						X		
24	Y	?	X	X							
26	Y	X	?	X							

Table 4 Repair efficiency comparison for proposed and existing APR techniques for chart CL project

Projects	OAPR-HOML'1	SketchFix	SimFix	Astor	Nopol	ACS	HDRepair	CapGen	ELIXIE	kPAR	jKali
2	Y	Y	X	Y							
5	Y	Y	X	X			X		X	?	X
10	X	X	Y	Y		Y	?	X	Y	Y	
18	Y	Y	Y	X	Y	Y	X	?	X		Y
21	Y	Y	Y	Y		?					
22	X	Y	X	X		?				X	
31	Y	X		Y	X	X	?	Y			
33	Y	Y	?	Y	X	Y			X	?	
40	Y	X			X	X	?				
51	X	X	X	Y				?		X	X
57	Y	X	?	X				X	X		
62	X	X						Y		Y	Y
70	Y	Y	X						?		
73	X	X	Y	?	?					Y	X
79	Y	Y	?		?	?	Y			X	
106	Y	X	Y	?	?	?			Y		
109	Y	?	Y	?		?			X		
115	Y	Y		Y			Y	?	Y	X	?
125	Y		X	X	Y						
126	Y	X	?	Y			X		?	?	?

Table 5 Repair efficiency comparison for proposed and existing APR techniques for chart L project

Projects	OAPR-HOML'1	SketchFix	SimFix	Astor	Nopol	ACS	HDRepair	CapGen	ELIXIE	kPAR	**jKali**
6	Y	Y	Y	Y							
7	X	Y	Y	Y			X		X	Y	
10	Y	?	Y	Y			X		?		
16	Y	Y	Y	Y	?	?	X		?	?	
21	X	Y	Y	Y							**Y**
24	Y	?	?	Y	?					Y	
26	X	Y	?	Y		X					**Y**
27	Y	Y	?	Y	X	X					
33	Y	Y	X	X	X	X					
35	Y	Y	X	X				X		Y	
38	X	?	X	X				X	X		**X**
39	Y	Y				?	?	Y			
41	Y	Y	Y			?					
45	Y	Y			Y					X	**X**
46	Y	Y	Y	Y	Y	Y				X	
50	Y	Y			Y	Y	?	?	Y		
51	Y	X	X	?		Y			Y		
53	Y	X							Y	?	**?**
55	Y	X	X	?	X			?			
57	X	Y	Y			?			X		**?**

Table 6 describes the performance comparison of repair efficiency of proposed and existing APR techniques for Math M project. In this simulation, we select the 20 bugs from Math M project in random manner. It is clearly depicts the repair efficiency of proposed OAPR-HOML'1 technique is very high compare to the existing APR techniques. The proposed OAPR-HOML'1 technique fixes the 10 number of bugs among 10 bugs. However, the existing APR technique fixes the 6, 3, 3, 3, 0, 2, 0, 0, 0 and 1 numbers of bugs for SketchFix, SimFix, Astor, Nopol, ACS, HDRepair, CapGen, ELIXIE, kPAR and jKali respectively. Table 7 describes the performance comparison of repair efficiency of proposed and existing APR techniques for Time T project. In this simulation, we select the 20 bugs from Time T project in random manner. It is clearly depicts the repair efficiency of proposed OAPR-HOML'1 technique is very high compare to the existing APR techniques. The proposed OAPR-HOML'1 technique fixes the 10 number of bugs among 10 bugs. However, the existing APR technique fixes the 5, 3, 3, 2, 0, 0, 0, 0, 1 and 1 numbers of bugs for SketchFix, SimFix, Astor, Nopol, ACS, HDRepair, CapGen, ELIXIE, kPAR and jKali respectively.

5.3 Comparative analysis

In this section, we compare the performance of proposed OAPR-HOML'1 technique with the existing state-of-art APR techniques are SketchFix, SimFix, Nopol, ACS, HDRepair, CapGen, ELIXIE, kPAR and jKali in terms of number of bugs fix, accuracy, precession, recall and F-measure. The detailed description of those performance metrics are as follow: Accuracy (Acc) defines the ratio of the number of bugs correctly fixed against the number of total bugs in the project. Precision (Pre) defines number of errors troubleshooting correct account number was first created errors reliable link was first created. Recall (Re) defines that many errors at the beginning reliable links to patches but also raise an error. Finally, the F-measure (F-m) defines the ratio of the precession + recall with precession × recall. Table 8 describes the performance comparison of proposed and existing ARP techniques on the Defects4J benchmark in terms of number of bugs fix, accuracy, precession, recall and F-measure.

Table 6 Repair efficiency comparison for proposed and existing APR techniques for chart M project

Projects	OAPR-HOML'1	SketchFix	SimFix	Astor	Nopol	ACS	HDRepair	CapGen	ELIXIE	kPAR	jKali
5	Y	Y	Y	Y							
33	Y	X	Y	X	Y		Y		X	?	
50	X	X	X	X	Y		?		X		X
59	Y	Y	X	Y		?	Y		X		X
70	Y	Y			Y	X				X	
82	X	Y	Y		?	X			X		
85	Y	Y	?	Y		X			X		
93	Y	Y	X						X		?
97	Y	?	X	?	?						
105	X	?	?	X							Y

Table 7 Repair efficiency comparison for proposed and existing APR techniques for chart T project

Projects	OAPR-HOML'1	SketchFix	SimFix	Astor	Nopol	ACS	HDRepair	CapGen	ELIXIE	kPAR	**jKali**
1	X	Y		X							
3	Y	Y							X	?	
4	Y	Y		X		?			X		**?**
8	X	?	Y		Y					Y	
9	Y	Y	Y	Y		Y				X	**Y**
15	Y	?	X	Y	Y	?	?		X		
18	Y	?	X	Y		X	X		X		
20	Y	X	?						X		**X**
24	Y	X	Y	?	X		?				
29	X	Y	X								

Table 8 Comparative analysis of proposed and existing APR techniques

Metrics	OAPR-	SketchFix	SimFi	ACS	Nopo	HDRepa	CapGen	ELIXI	kPAR	jKali
C	10/12	6/8	4/8	2/2	1/6	0/2	4/4	4/7	3/10	0/6
CL	15/20	3/5	6/8	0/0	0/0	0/7	0/0	0/0	5/9	0/0
L	9/12	3/1	9/13	3/4	3/7	2/6	5/5	8/12	1/8	0/0
M	19/20	7/8	14/26	12/16	1/21	4/7	12/16	12/39	7/18	1/14
T	4/5	0/1	1/1	1/1	0/1	0/1	0/0	2/3	1/2	0/2
Total	**57/69**	**19/26**	**34/56**	**18/23**	**5/35**	**6/23**	**21/25**	**26/41**	**17/47**	**3/17**
Acc (%)	**91.25**	63.93	59.36	69.92	9.39	23.21	80.72	60.89	29.2	3.21
Pre (%)	**85.69**	73.1	60.7	78.3	14.3	26.1	84	63.4	36.2	4.5
Re (%)	**82.37**	62.5	63	61.2	12.3	25.3	73	61.2	28.95	3.921
F-m (%)	**80.78**	63.9	64.12	59.3	17.38	23.92	74.1	64.3	27.39	5.32

Fig. 2 shows the number of bugs fix comparison of proposed OAPR-HOML'1 technique with the existing modern APR methods such as SketchFix, SimFix, ACS, Nopol, HDRepair, CapGen, ELIXIE, kPAR and jKali. It is clearly depicts the number of bugs fix of proposed OAPR-HOML'1 technique is very high in terms of 66/62, 40/18, 68.4/66, 91.2/49.2, 89.4/66.6, 63.1/63.7, 54.3/40.5, 70.1/31.8 and 94.7/75.3 higher than the existing APR techniques such as SketchFix, SimFix, ACS, Nopol, HDRepair, CapGen, ELIXIE, kPAR and jKali respectively. Fig. 3 shows the accuracy comparison of proposed OAPR-HOML'1 technique with the existing modern APR methods such as SketchFix, SimFix, ACS, Nopol, HDRepair, CapGen, ELIXIE, kPAR and jKali. It is clearly depicts the accuracy of proposed OAPR-HOML'1 technique is very high in terms of 29.93%, 34.94%, 23.38%, 89.7%, 74.56%, 11.53%, 33.27%, 68% and 96.48% higher than the existing APR techniques such as SketchFix, SimFix, ACS, Nopol, HDRepair, CapGen, ELIXIE, kPAR and jKali respectively.

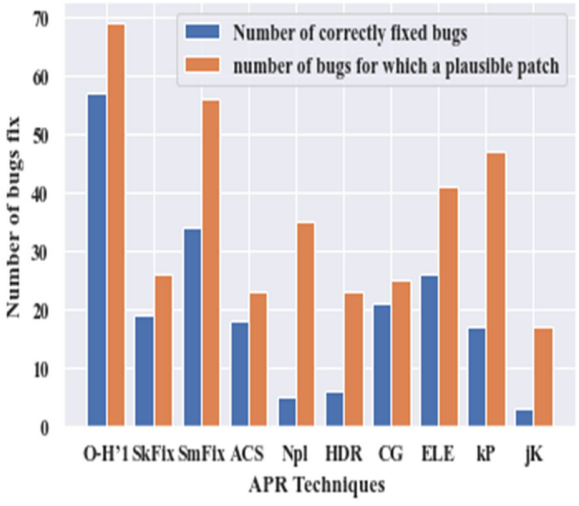

Fig. 2 Performance comparison of number of bugs fix

Fig. 4 shows the precession comparison of proposed OAPR-HOML'1 technique with the existing modern APR methods such as SketchFix, SimFix, ACS, Nopol, HDRepair, CapGen, ELIXIE, kPAR and jKali. It is clearly depicts the precession of proposed OAPR-HOML'1 technique is 14.69%, 29.16%, 8.62%, 83.31%, 69.54%, 1.97%, 26.01%, 57.75% and 94.75% very high in terms of higher than the existing APR techniques such as SketchFix, SimFix, ACS, Nopol, HDRepair, CapGen, ELIXIE, kPAR and jKali respectively. Fig. 5 shows the recall comparison of proposed OAPR-HOML'1 technique with the existing modern APR methods such as SketchFix, SimFix, ACS, Nopol, HDRepair, CapGen, ELIXIE, kPAR and jKali. It is clearly depicts the recall of proposed OAPR-HOML'1 technique is 24.1%, 23.5%, 25.7%, 69.2%, 11.37%, 69.28%, 25.70%,64.85% and 95.23%very high in terms of higher than the existing APR techniques such as SketchFix, SimFix, ACS, Nopol, HDRepair, CapGen, ELIXIE, kPAR and jKali respectively. Fig. 6 shows the F-measure comparison of proposed OAPR-HOML'1 technique with the present modern APR methods such as SketchFix, SimFix, ACS, Nopol, HDRepair, CapGen, ELIXIE, kPAR and jKali. It is clearly depicts the F-measure of proposed OAPR-HOML'1 technique is 20.8%, 20.6%, 26.5%, 78.48%, 70.3%, 8.2%, 20.4%, 66.09% and 93.4%very high in terms of higher than the existing APR techniques such as SketchFix, SimFix, ACS, Nopol, HDRepair, CapGen, ELIXIE, kPAR and jKali respectively.

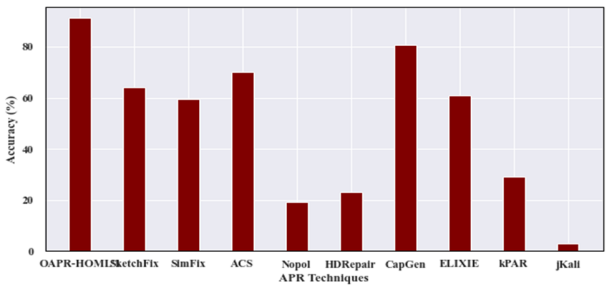

Fig. 3 Performance comparison of accuracy

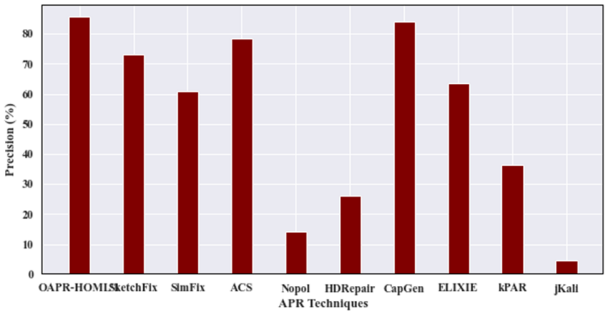

Fig. 4 Performance comparison of precession

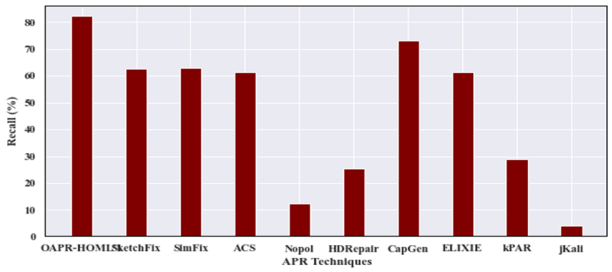

Fig. 5 Performance comparison of recall

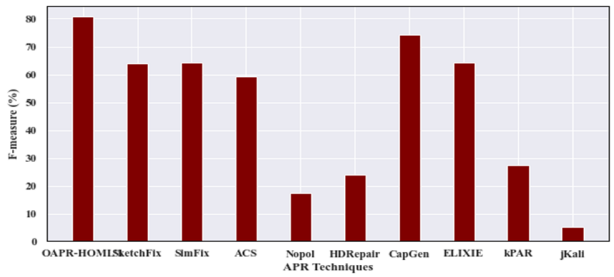

Fig. 6 Performance comparison of f-measure

6. Conclusion

An improved grasshopper optimization (IGO) algorithm is used for fault location identification in the given test projects and opposition learning based artificial neural network (OL-ANN) technique is used to select AST node-level transformation schemas to create the sketches which provide automated program repair for those faulty projects. From simulation results, we observe the average accuracy of proposed OAPR-HOML'1 is n% higher than the existing state-of-art APR techniques; the average precession of proposed OAPR-HOML'1 is n% higher than the existing state-of-art APR techniques; the average recall of proposed OAPR-HOML'1 is n% higher than the existing state-of-art APR techniques; and the average F-measure of proposed OAPR-HOML'1 is n% higher than the existing state-of-art APR techniques. Finally, the OAPR-HOML'1 is evaluated using Defects4J benchmark and the with respect to the number, which is like a performance bugs have been state-of-the-art technology run, accuracy, precession, recall and F-measure.

Acknowledgments:

Authors thank, K. Abijith Rao (CEO, SNIST) and Management of Sreenidhi Institute of Science and Technology, JNT University Anantapuramu for providing a assistance to establish working environment in the lab to carry out my present research. Authors also acknowledge, Prof. C V Tomy (Director), T. Shiva Reddy(Prinicipal), Dr. Aruna varanasi (head of the department) for continuous moral support, help and encouragement.

References

[1] Peng, R., Li, Y.F., Zhang, W.J. and Hu, Q.P., 2014. Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction. *Reliability Engineering & System Safety*, *126*, pp.37-43.

[2] Chen, T.Y., Tse, T.H. and Zhou, Z.Q., 2010. Semi-proving: An integrated method for program proving, testing, and debugging. *IEEE Transactions on Software Engineering*, *37*(1), pp.109-125.

[3] Gokhale, S.S., Lyu, M.R. and Trivedi, K.S., 2006. Incorporating fault debugging activities into software reliability models: A simulation approach. *IEEE Transactions on reliability*, *55*(2), pp.281-292.

[4] Cao, P., Dong, Z., Liu, K. and Cai, K.Y., 2013. Quantitative effects of software testing on reliability improvement in the presence of imperfect debugging. *Information Sciences*, *218*, pp.119-132.

[5] Kondo, K. and Yoshida, M., 2005. Use of hybrid models for testing and debugging control software for electromechanical systems. *IEEE/ASME Transactions on Mechatronics*, *10*(3), pp.275-284.

[6] Hierons, R.M., 2014. Generating complete controllable test suites for distributed testing. *IEEE transactions on software engineering*, *41*(3), pp.279-293.

[7] Lin, C.T. and Li, Y.F., 2014. Rate-based queueing simulation model of open source software debugging activities. *IEEE Transactions on Software Engineering*, *40*(11), pp.1075-1099.

[8] Huang, C.Y., Kuo, S.Y. and Lyu, M.R., 2007. An assessment of testing-effort dependent software reliability growth models. *IEEE transactions on Reliability*, *56*(2), pp.198-211.

[9] Cotroneo, D., Pietrantuono, R. and Russo, S., 2013. Combining operational and debug testing for improving reliability. *IEEE Transactions on Reliability*, *62*(2), pp.408-423.

[10] Huang, C.Y. and Lin, C.T., 2006. Software reliability analysis by considering fault dependency *reliability*, *55*(3), pp.436-450.

[11] Kapur, P.K., Pham, H., Anand, S. and Yadav, K., 2011. A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation. *IEEE Transactions on Reliability*, *60*(1), pp.331-340.

[12] Popentiu-Vladicescu, F. and Albeanu, G., 2016. Nature-inspired approaches in software faults identification and debugging. *Procedia Computer Science*, *92*, pp.6-12.

[13] Wong, W.E., Sugeta, T., Qi, Y. and Maldonado, J.C., 2005. Smart debugging software architectural design in SDL. *Journal of Systems and Software*, *76*(1), pp.15-28.

[14] Cai, K.Y., Cao, P., Dong, Z. and Liu, K., 2010. Mathematical modeling of software reliability testing with imperfect debugging. *Computers & Mathematics with Applications*, *59*(10), pp.3245-3285.

[15] Tokuno, K. and Yamada, S., 2003. Markovian software reliability measurement with a geometrically decreasing perfect debugging rate. *Mathematical and computer modelling*, *38*(11-13), pp.1443-1451.

[16] Minnerup, P., Lenz, D., Kessler, T. and Knoll, A., 2016. Debugging Autonomous Driving Systems Using Serialized Software Components. *IFAC-PapersOnLine*, *49*(15), pp.44-49.

[17] Chen, J. and Venkataramani, G., 2016. enDebug: A hardware–software framework for automated energy debugging. *Journal of Parallel and Distributed Computing*, *96*, pp.121-133.
[18] Abreu, R., Zoeteweij, P. and Van Gemund, A.J., 2011. Simultaneous debugging of software faults. *Journal of Systems and Software*, *84*(4), pp.573-586.
[19] Serrano, E., Quirin, A., Botia, J. and Cordón, O., 2010. Debugging complex software systems by means of pathfinder networks. *Information Sciences*, *180*(5), pp.561-583.
[20] Wang, J., Wu, Z., Shu, Y. and Zhang, Z., 2015. An I mperfect software debugging model considering log-logistic distribution fault content function. *Journal of Systems and Software*, *100*, pp.167-181
[21] Li, Q. and Pham, H., 2017. NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage. *Applied Mathematical Modelling*, *51*, pp.68-85.
[22] Li, F., Li, Z., Huo, W. and Feng, X., 2016. Locating software faults based on minimum debugging frontier set. *IEEE Transactions on Software Engineering*, *43*(8), pp.760-776.
[23] Zhang, Y., Zhu, B., Fang, Y., Guo, S., Zhang, A. and Zhong, S., 2017. Secure inter-domain forwarding loop test in software defined networks. *IEEE Transactions on Dependable and Secure Computing*, *17*(1), pp.162-178.
[24] Gazzola, L., Micucci, D. and Mariani, L., 2017. Automatic software repair: A survey. *IEEE Transactions on Software Engineering*, *45*(1), pp.34-67.
[25] Kong, X., Zhang, L., Wong, W.E. and Li, B., 2018. The impacts of techniques, programs and tests on automated program repair: An empirical study. *Journal of Systems and Software*, *137*, pp.480-496.
[26] Qiu, K., Zheng, Z., Trivedi, K.S. and Yin, B., 2019. Stress testing with influencing factors to accelerate data race software failures. *IEEE Transactions on Reliability*, *69*(1), pp.3-21.
[27] Jiang, J., Xiong, Y. and Xia, X., 2019. A manual inspection of Defects4J bugs and its implications for automatic program repair. *Science China Information Sciences*, *62*(10), pp.1-16.
[28] Gupta, N., Sharma, A. and Pachariya, M.K., 2020. Testing and debugging: an empirical evaluation of integrated approaches. *Sādhanā*, *45*, pp.1-15.
[29] Kim, J., Kim, J., Lee, E. and Kim, S., 2020. The effectiveness of context-based change application on automatic program repair. *Empirical Software Engineering*, *25*(1), pp.719-754.
[30] Caballero, R., Martin-Martin, E., Riesco, A. and Tamarit, S., 2021. A unified framework for declarative debugging and testing. *Information and Software Technology*, *129*, p.106427.
[31] Ye, H., Martinez, M., Durieux, T. and Monperrus, M., 2021. A comprehensive study of automatic program repair on the QuixBugs benchmark. *Journal of Systems and Software*, *171*, p.110825.
[32] Liu, K., Li, L., Koyuncu, A., Kim, D., Liu, Z., Klein, J. and Bissyandé, T.F., 2021. A critical review on the evaluation of automated program repair systems. *Journal of Systems and Software*, *171*, p.110817.
[33] Hua, J., Zhang, M., Wang, K. and Khurshid, S., 2018, October. Sketchfix: A tool for automated program repair approach using lazy candidate generation. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 888-891).

T Mamatha received her B.Tech Degree in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad,TS, in 2003. She received her M.Tech in Software Engineering at JNTUA University in 2007. At present she is working as an Assistant professor in Computer Science and Engineering , Sreenidhi Institute of Science and Technology,Hyderabad,TS INDIA. Her research interest includes Software Engineering, Software Testing, Machine Learning and Networking

B Rama Subba Reddy received his B.Tech Degree in Computer Science and Engineering From Jawaharlal Nehru Technological University, Anantapur, India, in 1997. M.Tech(CSE)From Mysore university, Mysore in 2002. PhD (Data Mining) From SV University,Tirupati 2014. At present he is working as a Professor &Vice Principal in Computer Science and Engineering Department ,S V Engineering College, Tirupati, AP, INDIA. His research interest includes Network Security , Wireless Communication system and Software Engineering

C Shoba Bindu received her B.Tech Degree in Electronics & Communication Engineering from Jawaharlal Nehru Technological University, Anantapur, India, in 1997. M.Tech. in Computer Science and Engineering from JNTUA University, Anantapuram in 2002. She received her Ph.D in Computer Science and Engineering at JNTUA University in 2010.. At present she is working as a professor in Computer Science and Engineering , JNTUA, Anantapuram, AP, INDIA. Her research interest includes Network Security, Wireless Communication system and Software Engineering