International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I5P224
Volume 69 Issue 5, 177-185, May 2021

An Improved Uniform Illustration Based Regression Testing By A Novel Heuristic Based Machine Learning Model

U.Sivaji[1], Dr.P.Srinivasa Rao[2]

[1]*Research Scholar, Department of Computer Science and System Engineering, Andhra University, Vishakhapatnam, Andhra Pradesh, India.*

[2]*Professor, Department of Computer Science and System Engineering, Andhra University, Vishakhapatnam, Andhra Pradesh, India*

sivajiu17@gmail.com

Abstract

The regression testing process is defined as the testing progression, which is utilized for verifying the software or code changes without altering the original characteristics of the code. Nevertheless, the execution process is required high resources and time that reduced the accurate detection rate. In this research, a novel Optimized Levy C4.5 Mechanism (OLCM) approach is introduced for performing regression testing. Here, the selected test cases are ordered based on the weightage of test cases and it effectively detects the faults. Moreover, the fitness function of the proposed OLCM module is performed the regression testing and enhance the performance of the system. Moreover, the developed OLCM module is implemented using Network Simulator 2 that is attaining a high detection rate with lower execution time and resource utilization. Additionally, the obtained results are validated with prevailing regression testing methods for evaluating the efficiency of the proposed OLCM approach.

Keywords: *Regression testing, test suite, C4.5 algorithm, test case prioritization, levy flight optimization*

I. INTRODUCTION

Regression testing method is utilized for identifying the changes in a code or software without modifying the complete working process of the original software [1]. Commonly, the software testing is demarcated as the partial or complete collection of implemented test cases that arerequired to re-execute to ensure the prevailing functionalities work fine [2]. For regression testing, the utilized code is initially debugged for identifying the bugs. Subsequently, the necessary modifications are developed for fixing [3]. The process of regression testing is performed on the appropriate test cases, which involves the affected and modified parts of the program or code [4]. Moreover, software maintenance involves error clear, enhancement, optimization, and removing the prevailing feature, which may affect the system performance so, the regression testing process is required [5]. In this, the test case selection is necessary that is categorized by returnable and obsolescent test cases [6]. Moreover, the collection of test cases is named as a test suite that can be grouped for the purpose of test execution processes [7]. Subsequently, the regression testing includes the testing prevailing software for authenticating that processing characteristics are can't affect by the developed work [8]. Additionally, these testing are necessary for identifying the deficiencies primarily, which is performed based on several stages that are selecting, minimization, arrangement, execution of the test cases, and flaw mitigation [9].

Also, the regression testing progression is focused on the functional tests that involve quality assurance testing that is employed as the verification process [10]. The regression testing is not dependent on the encoding codes like C#, C++, and Java that can be verified in which the changes can affect the original features of the current program or codes [11]. Also, regression testing is referred to as the action of preservation level that is performedon the improved codes or software for determining the reliability of the programs [12]. The priority selection of the test cases is employed for detecting the faults in an earlier stage, which is assists in regression testing [13]. Here, theutilized test cases are required for evaluating the modifications and identify the faults in a particular time [14]. Moreover, TCP is utilized for scheduling the test cases for attaining the orders of test cases [22].

Subsequently, several approaches like Combinations of Code Coverage based Prioritization (CCCP), Honey Bee based Fuzzy rule (HBF), deep learning [15], and Kernel Fuzzy with c-means clustering (KFCM) are utilized in the regression testing process. However, these existing methods are affected by many limitations that are reduced the performance of the testing process. In this, the existing regression testing models are influenced by high execution time, duplicate faults detection, resource utilization, and lower faults detection, which are reduced the efficiency of the models. Therefore, a novel machine learning approach is developed in this paper to overcome the existing issues.

The structure of this article is designed as Section 2 details the recent kinds of literature, and the system frame and issues arestated in Section 3. Moreover, section 4 detailed the introduced approach, then the results of the developed replica are explained in section 5 and the arguments are concluded in section.6.

II. RELATED WORKS

Some of the current literatures based on regression testing are described as follows,

The regression testing process has enhanced the fault detection rate by utilizing the test cases for execution. Moreover, the regression testing is depending on the progress of TCP for that the approach of code coverage is employed to monitor the prioritization. Here, Huang et al [16] developed the coverage model that is the combination of the concepts like combination coverage and code coverage. The introduced approach is named Combinations of Code Coverage based Prioritization (CCCP) that can identify the faults that increased the strength of the testing.

The frequent changes in the software are permitted in the environments of Continuous Integration (CI) that creates software development. In this, the approaches based on test case prioritization (TCP) are utilized for determining the faults. Lima et al [17] demonstrated the approach based on TCP in the CI (TCPCI) environment. Thus, the developed TCPCI model diminished the testing issues, CI discriminations, features like test case instability, and parallel execution.

Nayak et al [18] developed the Honey Bee optimization model based on the Fuzzy rule (HBF) for improving the detection rate of faults in TCP. This optimization model is employed to reduce the test cases with the use of test selection by the pre-existing test suite, which is performed based on forager bees and scout bees. Hence, this approach has enhanced the rate of fault detection using the function of bees.

The regression testing of large projects is challengeable for identifying the test cases which is executed at the end of every release because that required more time and cost. So, Ali et al [19] developed the TCP method and integration model (TCPI) for enhancing the quality of releases using agile software. Here, the developed approach processed using two kinds of phases; primarily, the TCP process is performed using the clustering model, and secondly, select the test cases which have the ability of failure with higher frequency. Thus, it enhanced the fault detection percentage and reduced the amount of identifying duplicate faults.

The TCP in regression testing is developed by Harikarthik et al [20] that primarily create the test cases, which are clustered using Kernel Fuzzy with c-means clustering (KFCM) approach. The test cases are categorized as relevant and irrelevant and here, the relevant cases are selected for providing TCP. For providing TCP, the KFCM approach utilized the model of an enhanced artificial neural network. Moreover, the process of weight optimization is done with the use of a whale algorithm.

Table 1:Related works based on regression testing

Author	Method	Merits	Demerits
Huang et al [16]	CCCP	It can easily identify the faults.	Processing time is high
Lima et al [17]	TCPCI	Reduced the testing issues.	The detection of duplication faults is large.
Nayak et al [18]	HBF	Enhanced the fault detection rate.	Large amount of data is not used.
Ali et al [19]	TCPI	Reduced the detection of duplicate faults.	Resource utilization is large.
Harikarthik et al [20]	KFCM	It required less memory and time.	Fault detection rate is low.

The key functions of the introduced manner is given below,

- Primarily, the test cases are collected from the test suite for processing the testing model.
- Additionally, develop the Optimized Levy C4.5 Mechanism (OLCM) for performing regression testing process.
- Moreover, the developed OLCM model performs the functions like test case minimization, TCP, scheduling, and regression testing.
- Here, the levy fitness function is introduced in the developed C4.5 Mechanism for attaining high-efficiency outcomes in regression testing.
- The simulation of this model is finished using the NS-2 tool and the proposed model effectively analyses the software performance.
- Moreover, the performance metrics of the proposed OLCM model are calculated and evaluated using prevailing methods in terms of fault detection rate, precision, accuracy, and recall.

III. SYSTEM MODEL ANDPROBLEM STATEMENT

Normally, the regression testing models are utilized for enhancing the fault detection rate with the use of test case execution. This progression is a difficult task in the software improvement and maintenance manner. While modifying the program, the test cases are utilized to execute and the validation of attained results and the old outcome to reduce the unwanted modifications. If the validation of two outcomes is the same then that is not affecting the performance of the software. The system model for regression testing is represented in fig.1.

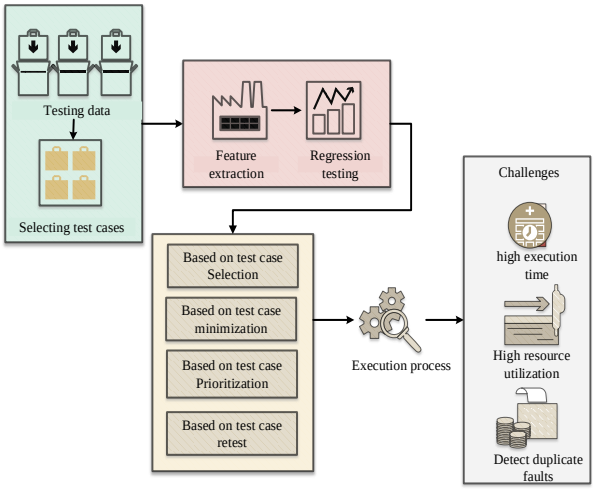

Fig.1System model for regression testing

However, the large test suits are required more time and resources while executing the testing process. Moreover, test suite minimization is one of the difficulties while attaining extreme test coverage. To determine the frequency range of each test are the difficult tasks in regression testing. Thus, the developed machine learning approach enhances the performance with lower execution time, lower resource utilization, and high detection rate.

IV. Proposed OLCM Methodology

The process of regression testing is stated as the software testing progressions, which can be used to validate the system based on the modifications in a program. However, these testing techniques are utilized more time, and resources for executing regression testing used a large number of test cases. In this research, a novel Optimized Levy C4.5 Mechanism (OLCM) is introduced to overcome the above issues. Additionally, the proposed OLCM process is represented in fig.2.

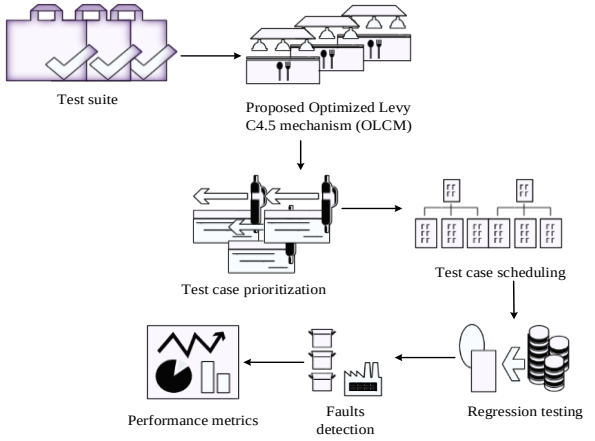

Fig.2. Process of OLCM methodology

Primarily, the developed OLCM approach generated the test cases for performing the further process. Consequently, the processes of TCM and TCP are carried out by the developed OLCM model. Moreover, the TCP process in the testing is to schedule the test cases and detect the faults in an earlier stage. These test cases are utilized for evaluating the modifications in a program. Therefore, the developed approach is performing the regression testing process and detects the faults in the program. Also, it calculates the performance metrics that are validated with existing methods for identifying the efficacy of the proposed approach.

A. Process of OLCM for regression testing

The developed OLCM model is performed on the test case database for processing the regression testing, which has the ability to identify the performance of the software or code. The proposed OLCM model involves the processes like test case selection, TCP, and regression testing that is detailed below.

- **Test case selection**

Initially, the test suite is selected from a net source that involves the group of test cases with various categories of programs like C, C++, Java, and C#, which contains test data, steps, and conditions.The proposed Optimized Levy C4.5 Mechanism uniformly selects theC# test cases for processing. The developed OLCM model is selecting the testsuite that is mentioned as T_S and the selected test casesare mentioned in eqn.(1),

$$T_S = T_i;(i = 1,2,3,.....,n) \quad (1)$$

Where, T_i denotes the n number of test cases for C# program. Afterwards, the test case minimization process is carried out by the proposed model.

- **Test case minimization**

The levy flight optimization (LFO) [21] algorithm is utilized for identifying the best location that behavior is considered in this paper for recognizing the faults. Here, the faults are represented as $F_i;(i = 1,2,.....,k)$ that are detected using the proposed OLCM and the faults detection rate are calculated. The rate of fault recognition is defined as the average quantity of faults per minutes by the test case that is calculated using eqn.(2),

$$F_t(i) = \left(\frac{no.of\_faults}{time}\right)*10 \quad (2)$$

Moreover, the impact of bugs are identified based on the severity (S) of test cases that is calculated using eqn.(3),

$$F_I(i) = \left(\frac{S(i)}{Max(S)}\right)*10 \quad \textbf{(3)}$$

Hence, the severity of the faults is identified and the impacts are calculated for performing the testing process. Here, the highest rates of bugs in the test cases are removed and the remaining test cases are processed.

Algorithm 1: process of Optimized Levy C4.5 Mechanism	
Start	
{	
Initialize the dataset T_S	***//test suitefor C# program***
$int\, T_S = T_i;(i = 1,2,3,.....,n)$	***//n number of test cases***

$T_i \leftarrow train()$ ***//dataset training***

Test case minimization()
// Fault detection()

$for \forall F_i$

calculate $F_t(i)$ ***//fault detection rate***

Find severity $F_I(i)$

end for

//Test case prioritization ()

Calculate weightage of the test cases

$T_w(i) \rightarrow weightage$

If $T_W(i) \rightarrow high$ **then** 1st priority

else

If $T_W(i) \rightarrow low$ **then** it is last priority

end if

// execution process

Regression testing()

Calculate the fitness function

While $(t < \max\_testing)$ *//t-regression testing*

Run the test cases $k(T_i)$ *using decision nodes*

$k_j = k(T_j)$

Update(t)

If $k_i > k(T_j)$***do***

$k_j \leftarrow k(T_j), k_i \leftarrow k(T_i)$

end if

Run the test cases $k(T_j)$

$k(T_i) = k(T_j)$

end while

// calculate parameters

output best solution

}

Stop

- **Process of Test case prioritization (TCP)**

The TCP process can be utilized for ordering the test cases that can provide the priority for test cases, which is diminished the cost, time, and resource utilization. In this research, the utilized test cases are prioritized based on the weightage of the test cases on the basis of decision trees by OLCM. Here, the weightage of the test cases are calculated using eqn.(4),

$$T_W(i) = F_t(i) + F_I(i) \quad (4)$$

Where, $F_I(i)$ denotes the impact of faults and $F_t(i)$ is the fault detection rate. Therefore, the utilized test cases are orderedusing the calculation of test cases weightage. Here, if the weightage is high then it is taken as 1st priority and each test caseisarranged in the descending order, which are performed the regression testing.

- **Regression testing**

Regression testing is identifying the modification in code that cannot change the original function of the software. In this process, the test suite is selected for processing that involves the group of test cases, which are utilized for test execution. After the TCP process, the test execution is carried out on the test cases that are mentioned by $k(T_i)$. Here, the test execution is done with the use of OLCM decision trees. The decision trees of the OLCM has split the test cases and executed at lower time duration. Consequently, the identified faults are minimized and the attained test cases are mentioned as $k(T_j)$. The proposed model has effectively identified the bugs in the test cases while performing regression testing. Finally, the regression testing performed on the code for identifying any changes is occurred or not. The fitness function run the test case if the condition $k(T_i) = k(T_j)$ is satisfied then the function of the code is not changed.

Table.2 Detected faults and processing time of every test cases

Test cases	Number of fault detected	Execution time
T1	2	8
T2	1	12
T3	2	9
T4	3	13
T5	1	7

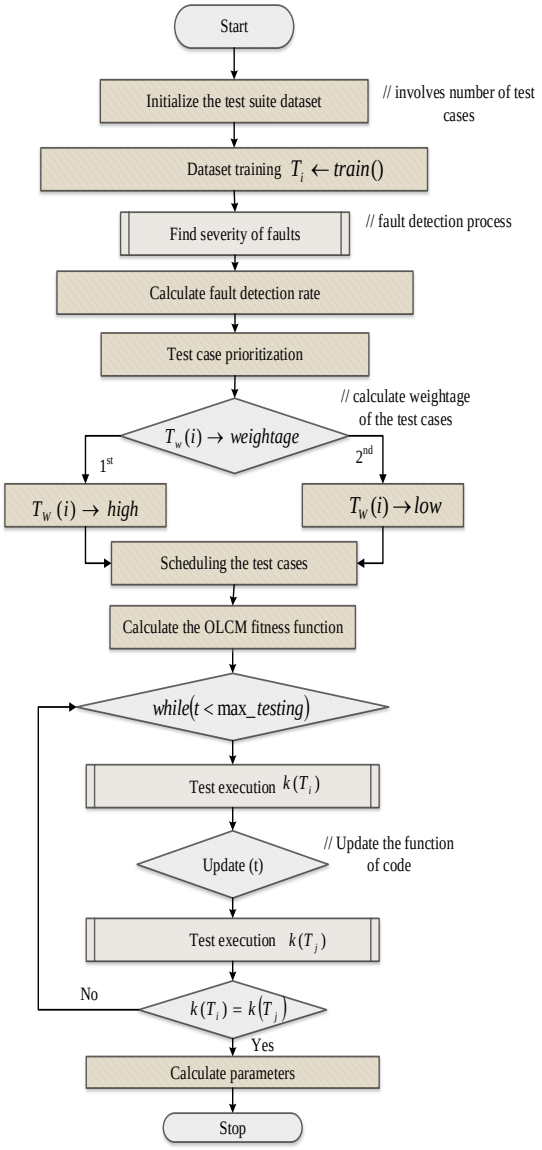

Fig.3 block diagram of the proposed OLCM approach

The procedure of the proposed OLCM approach is given in algorithm 1 and the block diagram representation is shown in fig.3.

V. RESULTS AND DISCUSSION

In this research, the introduced OLCM model has been implemented using network simulator 2 for performing the investigation of software performance of regression testing. The developed approach utilized the test suite for a testing process that involves number of test cases. Here, this model performs the TCP, faults detection, and regression testing in the test cases. Moreover, the performance parameters like accuracy, fault detection rate, execution time, recall, and precision are calculated.

A. Case study

The proposed model is utilized for performing the regression testing process that is solved in this section. In this approach, the test suite is considered for testing that involves the amount of test cases. Let the 5 numbers of test cases T1 to T5 and faults F1 to F5 for processing. Also, considered detected faults with execution time for the test cases are mentioned in table.2.

Primarily, the test cases are diminished using OLCM function and the fault detection rate is calculated using eqn.(1). Also, some examples for FD rate calculation of the test cases are given in table.3.

Table.3 FD rate for sample test cases

Test cases	FD rate
T1	$F_t(T1) = (2/8)*10 = 2.5$
T2	$F_t(T2) = (1/12)*10 = 0.83$
T3	$F_t(T3) = (2/9)*10 = 2.2$
T4	$F_t(T4) = (3/13)*10 = 2.3$
T5	$F_t(T5) = (1/7)*10 = 1.4$

Consequently, the fault impacts in the test cases are calculated based on the severity of fault using eqn.(2). Here, the maximum severity of fault is considered as 15 and attained outcomes are mentioned in table.4.

Table.4 calculation of bugs severity

Test cases	Severity of bugs
T1	$F_I(T1) = (6/15)*10 = 4$
T2	$F_I(T2) = (9/15)*10 = 6$
T3	$F_I(T3) = (7/15)*10 = 4.6$
T4	$F_I(T4) = (11/15)*10 = 7.3$
T5	$F_I(T3) = (9/15)*10 = 6$

Subsequently, the TCP process is performed based on the weightage of the test cases. Moreover, the weightage of the test cases are calculated using eqn.(4) and attained values are mentioned in table.5,

Table.5 calculation of test case weightage

Test cases	Weightage of test cases
T1	$T_W(T1) = 2.5 + 4 = 6.5$
T2	$T_W(T2) = 0.83 + 6 = 6.83$
T3	$T_W(T3) = 2.2 + 4.6 = 6.8$
T4	$T_W(T4) = 2.3 + 7.3 = 9.6$
T5	$T_W(T5) = 1.4 + 6 = 7.4$

Based on the highest value of weightage for the test cases are taken as the 1st priority and considered as the descending order for prioritization. Hence, the scheduled

test cases are T4, T5, T2, T3, and T1. Finally, the OLCM has effectively detect the bugs of the test cases when performs the regression testing.

B. Performance metrics

The proposed approach performs the regression testing process that calculated parameters like accuracy, fault detection rate; execution time, recall, and precision are validated by existing methods like HBF [18], CTFF [19], and KFCM [20].

a) Accuracy calculation

The effectiveness of the developed OLCM model is identified using the calculation of accuracy. Accuracy of the proposed model is calculated using eqn.(5) that can determined the proposed model efficiency.

$$Accuracy = \frac{exact \quad \det ection}{total \quad \det ection} \tag{5}$$

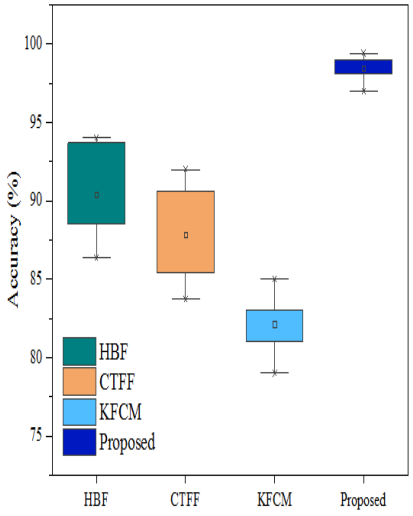

Fig.4 Comparison of accuracy

The accuracy measurement of the developed OLCM method is compared with recent existing approaches like HBF, CTFF, and KFCM, which are detailed in table.6.

Table.6 Evaluation of Accuracy

Test cases	Accuracy (%)			
	HBF	CTFF	KFCM	OLCM [proposed]
10	94	92	85	99.4
20	93.7	90.6	83	99
30	89.56	87.5	82.7	98.78
40	88.5	85.42	81.05	98.09
50	86.35	83.74	79	97

Here, the prevailing methods are attained almost 94% accuracy only. The proposed method has achieved 99.4% high accuracy than other methods that is represented in fig.4.

b) Execution time

The time is required for completing entire process by the proposed approach is mentioned as execution time. Here, the time period is measured when the test cases are performed the regression testing process that is calculated using eqn.(6),

$$Et = \frac{Starting\_time - ending\_time}{total\_time} \tag{6}$$

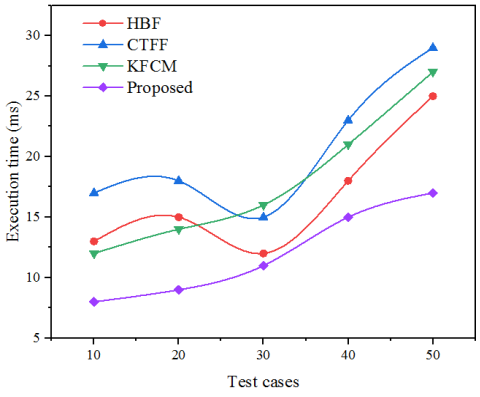

Fig.5. Comparison of execution time

The existing methods are utilized more time duration for performing regression testing process that is reduced the efficiency of the model. The comparison of the execution time is detailed in table.7 and represented in fig.5. The proposed model utilized less time duration to perform the regression testing process than other approaches.

Table.7 Evaluation of Execution time

Test cases	Execution time (ms)			
	HBF	CTFF	KFCM	OLCM [proposed]
10	13	17	12	8
20	15	18	14	9
30	12	15	16	11
40	18	23	21	15
50	25	29	27	17

c) Fault detection rate (FD)

FD rate is calculated for identifying the maximum quantity of faults will be discovered while doing software testing by the OLCMapproach, which is calculated using eqn.(7).

$$FD = 1 - \left(\frac{T_1F_1 + T_2F_2 + \ldots\ldots\ldots + T_nF_m}{mn} \right) + \frac{1}{2m} \quad (7)$$

Where, T_iF_i denotes the test cases used for performance, m represents the entireamount of bugs that is recognized in the code and n denotes the total quantity of test cases.

Table.8 Evaluation of FD rate

Test cases	Fault detection rate (%)			
	HBF	CTFF	KFCM	OLCM [proposed]
10	82.5	85	62	94
20	78.3	83	61	93
30	75.4	80	56	91.5
40	79.6	76	57	89
50	73.8	67	53	88.6

The fault detection rate of the developed OLCM approach has validated with existing methods like HBF, CTFF, and KFCM, which is detailed in table.8. Here, the proposed OLCM approach has effectively detected the bugs in the program than other techniques that are illustrated in fig.6.

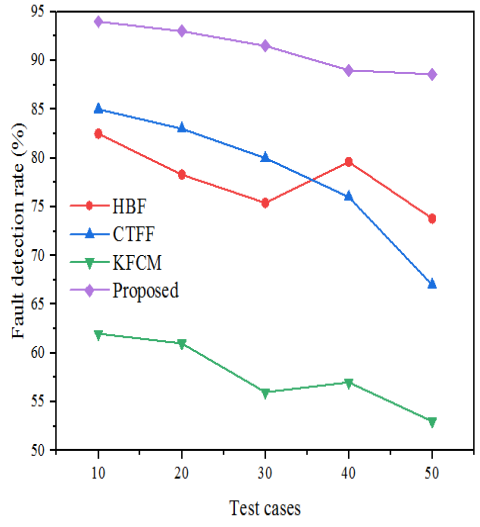

Fig.6. Comparison of FD rate

The proposed model identified the maximum quantity of bugs while executing regression testing progression. The rate of FD of the OLCM approach is 94% that is an efficient outcome while comparing with prevailing techniques.

d) Calculation of Precision

Precision is measurement of finding the exact detection of faults with the use of OLCM approach, which is calculated by eqn.(8),

$$P = \frac{exact \quad \det ection}{exact \quad \det ection + false \quad \det ection} \quad (8)$$

Table.9 Evaluation of precision

Test cases	Precision (%)			
	HBF	CTFF	KFCM	OLCM [proposed]
10	87	90	75	98
20	85.56	89.57	73.6	97.06
30	81.08	87.8	71.45	95.78
40	79.67	83.7	68.56	93.5
50	76.8	81.9	65.8	91

The precision value of the developed OLCM model is validated with the existing approaches like HBF, CTFF, and KFCM, which are detailed in table.9. The evaluation of the proposed OLCM method has achieved 98% high precision value while comparing with other approaches.

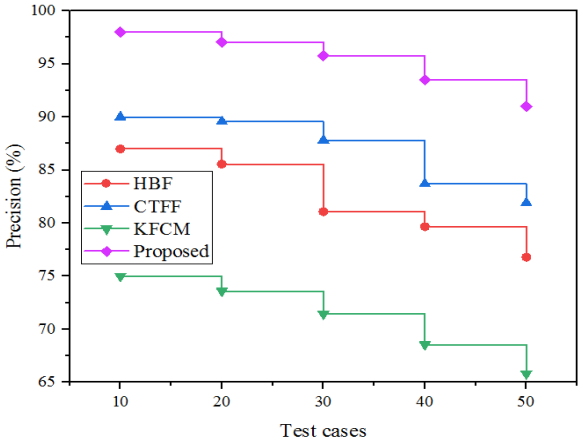

Fig.7 Comparison of Precision

The existing KFCM method was achieved 75% precision value, the HBF model was achieved 87%, and the CTFF method was attained almost 90% precision value. Here, the OLCM manner has attained 98% that is represented in fig.7.

e) Calculation of Recall

Recall value is measured for finding the proficiency of the FD value with the use of OLCM approach for the quantity of test cases, which is computed using eqn.(9)

$$R = \frac{exact\_positive \quad detection}{detected \quad outcomes} \tag{9}$$

Table.10 Evaluation of recall

Test cases	Recall (%)			
	HBF	CTFF	KFCM	OLCM [proposed]
10	91	97	78	98.5
20	89	96	74.8	97
30	87.96	94.57	72.75	96.06
40	85.07	92	69.47	94.78
50	82.67	89	64.56	92.5

The recall value of the developed OLCM model is validated with the existing approaches like HBF, CTFF, and KFCM, which are detailed in table.10. The evaluation of the recall is done using several quantities of test cases and the proposed method has reached a 98.5% recall value that is higher than other approaches.

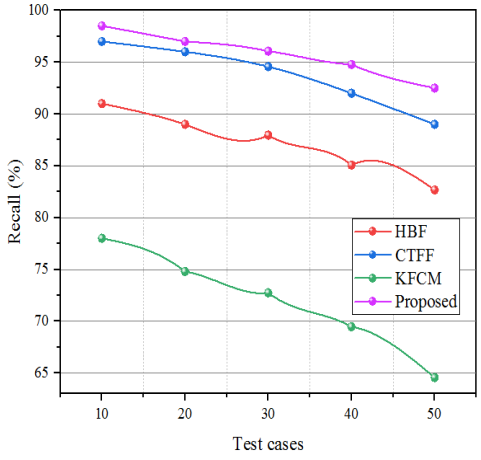

Fig.8 Comparison of Recall

The existing KFCM method was achieved a 78% lower recall value, the HBF model was achieved 91%, and CTFF methods was attained almost 97% recall value. Here, the OLCM manner has attained 98.5% recall and the comparison is illustrated in fig.8.

f) Calculation of memory utilization

Resource utilization has been calculated to identify the effectiveness of the introduced method. In this work, OLCM used test cases with limited memory storage, which is referred to as the number of bytes described in Table 11.

Table.11 Evaluation of Memory utilization

Test cases	Memory utilization (bytes)			
	HBF	CTFF	KFCM	OLCM [proposed]
10	5678	10239	11654	1000
20	6700	11300	12354	2000
30	8700	15445	13457	3000
40	9400	17846	15245	4000
50	10546	19456	16846	5000

The amount of memory required for test events by OLCM is estimated with the techniques in practice. If the memory necessity is low, the method is more efficient, which is achieved by OLCM as illustrated in fig.9.

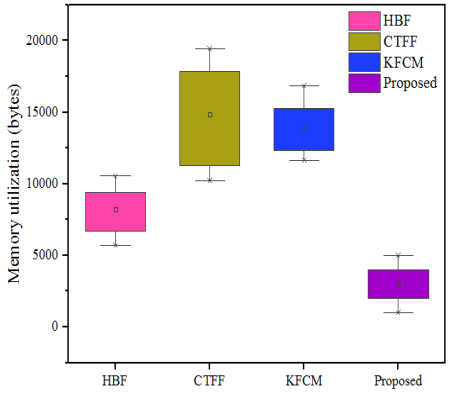

Fig.9 Comparison of memory utilization

The OLCM manner is validated with HBF, CTFF, and KFCM approaches that are utilized a high amount of memory. From fig.9, the proposed OLCM has been utilized a lower quantity of memory that is increased the efficiency and optimized the execution time.

g) F-measure

It is a trade-off between precision and recall that considers both precision value P and recall value R. Therefore, the value of F1-score can be evaluated by eqn. (10).

$$F\_measure = 2\left(\frac{P \times R}{P + R}\right) \tag{10}$$

Table.12 Evaluation of F-measure

Test cases	F-measure (%)			
	HBF	CTFF	KFCM	OLCM [proposed]
10	88.95	94.28	76.4	98.24
20	86.76	93.35	75.67	97
30	85.07	93	73.8	95.65
40	83.6	92.67	69.2	94.13
50	80.76	85.8	65.3	91.74

The F-measure value of the developed OLCM model is validated with the existing approaches like HBF, CTFF, and KFCM, which are described in table.12. The estimation of the proposed method F-measure has attained 98.24% that is efficient than the existing methods, which are described in fig.10.

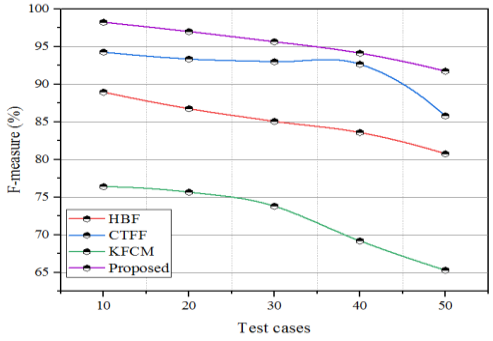

Fig.10 Comparison of F-measure

The proposed OLCM model has attained high F-measure value than other existing approaches, which improved the efficiency for detecting faults while performing regression testing process.

VI. CONCLUSION

This work developed an innovative machine learning algorithm named as optimized levy C4.5 mechanism (OLCM) for performing regression testing process. Initially, the test cases are selected for C# program that is processed using OLCM. In this, the selected test cases have accomplished the procedures like test case minimization, TCP, test case scheduling, fault detection, and regression testing. Initially, the test cases are selected from the test suite that is minimized using the fitness function of the OLCM. Moreover, test cases are arranged based on the calculation of test cases. Thus, the proposed method has effectively performed the regression testing than other methods. Also, the OLCM method provided the highest fault detection rate among the prevailing approaches. Additionally, it achieved 99.4% high accuracy for performing the software testing process using OLCM. Moreover, this model is applicable for other programs like C, C++, Java, and so on.

References

[1] Bin Ali, Nauman, et al. On the search for industry-relevant regression testing research. Empirical Software Engineering 24.4 (2019) 2020-2055.

[2] Lübke, Daniel. Selecting and Prioritizing Regression Test Suites by Production Usage Risk in Time-Constrained Environments. International Conference on Software Quality. Springer, Cham, 2020.

[3] Lu, Yun, and Xiuhong Chen. Joint feature weighting and adaptive graph-based matrix regression for image supervised feature Selection. Signal Processing: Image Communication (2020) 116044.

[4] Mahdieh, Mostafa, et al. Incorporating fault-proneness estimations into coverage-based test case prioritization methods. Information and Software Technology 121 (2020) 106269.

[5] Nayak, Soumen, et al. An Improved Approach to Enhance the Test Case Prioritization Efficiency. Proceedings of ICETIT 2019. Springer, Cham, 2020. 1119-1128.

[6] Rajput, Arpit, and Sheetal Joshi. Techniques of Test Case Prioritization. Decision Analytics Applications in Industry. Springer, Singapore, 2020. 443-453.

[7] Khatri, Harsh, ShubhamTorvi, and JayakrishnaKandasamy. Prioritization of sustainability indicators using regression analysis: A case study. Materials Today: Proceedings 22 (2020): 2397-2403.

[8] Al-Sabbagh, KhaledWalid, et al. Improving Data Quality for Regression Test Selection by Reducing Annotation Noise. 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, 2020.

[9] Rhmann, Wasiur, et al. Software fault prediction based on change metrics using hybrid algorithms: An empirical study. Journal of King Saud University-Computer and Information Sciences 32.4 (2020) 419-424.

[10] Grano, Giovanni, et al. Scented since the beginning: On the diffuseness of test smells in automatically generated test code. Journal of Systems and Software 156 (2019) 312-327.

[11] Durelli, Vinicius HS, et al. Machine learning applied to software testing: A systematic mapping study. IEEE Transactions on Reliability 68.3 (2019) 1189-1212.

[12] Anwar, Zeeshan, et al. A hybrid-adaptive neuro-fuzzy inference system for multi-objective regression test suites optimization. Neural Computing and Applications 31.11 (2019) 7287-7301.

[13] Majd, Amirabbas, et al. SLDeep: Statement-level software defect prediction using deep-learning model on static code features. Expert Systems with Applications 147(2020): 113156.

[14] Sharma, Pooja, and AmritLalSangal. Soft Computing Approaches to Investigate Software Fault Proneness in Agile Software Development Environment. Applications of Machine Learning. Springer, Singapore, 2020. 217-233.

[15] Xiao, Lei, et al. LSTM-based deep learning for spatial–temporal software testing, DISTRIBUTED AND PARALLEL DATABASES (2020).

[16] Huang, Rubing, et al. Regression test case prioritization by code combinations coverage, Journal of Systems and Software 169 (2020): 110712.

[17] Lima, Jackson A. Prado, and Silvia R. Vergilio. Test Case Prioritization in Continuous Integration environments: A systematic mapping study, Information and Software Technology 121 (2020) 106268.

[18] Nayak, Soumen, et al. Regression test optimization and prioritization using Honey Bee optimization algorithm with fuzzy rule base, Soft Computing (2020) 1-18.

[19] Ali, Sadia, et al. Enhanced regression testing technique for agile software development and continuous integration strategies., Software Quality Journal (2019) 1-27.

[20] Harikarthik, S. K., V. Palanisamy, and P. Ramanathan. Optimal test suite selection in regression testing with test case prioritization using modified Ann and Whale optimization algorithm. Cluster Computing 22.5 (2019) 11425-11434.

[21] Mokeddem, Diab. Parameter Extraction of Solar Photovoltaic Models Using Enhanced Levy Flight Based Grasshopper Optimization Algorithm, Journal of Electrical Engineering &Technology 16.1 (2021) 171-179.

[22] Sivaji, U., and P. SrinivasaRao. Test case minimization for regression testing by analyzing software performance using the novel method, Materials Today: Proceedings (2021).