An enhanced Whale Optimisation Algorithm for test case suite selection: Ambha WOA

Bhawna Jyoti[1], Aman Kumar Sharma[2]

[1]Research Scholar, Computer Science Department, Himachal Pradesh University, Shimla, India
[2]Professor-Computer Science, Himachal Pradesh University, Shimla, India.

Abstract:
Background: In the software industry, it is a challenging task to meet the everchanging needs of the user and set harmony of software modifications with customers' expectations. Retesting of modified software using regression testing techniques provides confidence that newly added modules have not affected on reliability and normal functioning of the software. **Aim:** In regression testing, test case suite selection techniques aim to find a subset of selected test cases that maximize fault revealing capability and reduce test case suite size as well as execution time. We aim to develop an enhanced metaheuristic algorithm based on the whale optimization algorithm for test case selection problem. **Method:** Whales are used as search agents to determine selected test case suite with the objective of finding the global optimal solution for TCS problems. **Results:** Ambha_WOA algorithm is implemented on 12 test case suites taken from GitHub and SIR repositories. Performance metrics – APFD metric, classification accuracy, Fault detection ratio metric, precision and recall are considered to compare proposed algorithm with classical whale optimization algorithm. To validate results, NP-statistical tests (Wilcoxon sign rank and F test) are conducted.**Conclusion**: Metaheuristic algorithms are efficient to deal with TCS problems in regression testing by giving an optimal solution.

Keywords: Whale Optimization Algorithm, Test case selection problem, APFD metric, code coverage, Regression testing, software maintenance phase testing.

NOMENCLATURE

Whale Optimization Algorithm (WOA), Classification Metric (C), Average Percentage of Fault Detection (APFD), Test case selection (TCS), Test case suite Reduction (TSR).

1. Introduction

In this software-dependent real-world scenario, it is very important to maintain the quality and reliability of software. When there is a change in customers' requirements, software is updated in the maintenance phase to meet those particular objectives without affecting actual performance of the modified software [24]. Regression testing provides confidence that modifications done in one part of the software have not produced incorrect performance measurements in unchanged parts of the software [25]. Regression test case suites are needed as new modules are integrated into software during the maintenance phase. These test case suites increase in dimensions (with the addition of new test cases) and take more time as well as resources which is becoming a challenge in the software industry these days [32]. To overcome issues concerned with resource and time constraints regression testing techniques are used. Techniques include test case suite reduction (TSR), test case suite selection (TCS) and test case prioritization (TCP). TSR involves removal of redundant (R) or obsolete test cases, TCS deals with choosing a subset of test cases that fulfills all the test case requirements as original test case suite does and TCP concerns with ideal ordering of test cases with the objective of early detection of faults in a software system. Test Case Selection techniques are based on the selection procedure of test cases that are being executed after system changes that exist between new and previous versions of the system [26][32].

Given: The program (P), the modified version of P (P1), and a test case suite (T).
Problem: Find a subset of T (T1), with which to test the modified version of P (P1). [32]

Optimal solutions are needed to handle real-world problems like warehousing, scheduling, planning etc., within reasonable time constraints. Heuristic algorithms are unable to find the global optimum solution and get trapped in the local optimum search space. So, metaheuristic algorithms are attracting the attention of many researchers and are widely accepted to handle complex NP-hard problems. The whale optimization algorithm is widely used in many engineering domains and gave promising results.[30][31][33]

This paper is organized as: Section 2 describes a related study about test case selection in regression testing. Section 3 presents the basic ideology and proposed algorithm based on the Whale optimization algorithm. Section 4 highlights experimental results, Section 5 gives comparative analysis and discussion, Section 6 concludes the study as well as gives necessary future directions.

2. Related study

Panichella et al. [1] proposed genetic algorithm for test case selection by introducing diversification using genetic algorithms. Orthogonal individuals are initialized as a population and diversities- distance crowding of test cases, ranking selection technique, as well as migration among the population, are used for test case selection. Kaur et al. [3] bestowed bat and cuckoo (nature-inspired) algorithms for selecting test cases in regression testing. Performance evaluation is determined by the number of faults detected in a source code program. Hand seeded faults are introduced in a dataset taken from SIR repository leads to a research gap. Bayeing et al. [2] proposed branch coverage criteria along with diversity introduced by using genetic algorithm. Siemen program and space program are used for their empirical study.

Souza et al. [4] proposed a multi-objective particle swarm optimization technique based on functional requirement coverage for the TCS problem. Crowding distance of test cases as well as Roulette wheel selection are used. Only two test case suites (each having only 80 test cases) are used for implementing their experimental work that leave a space to improve. Harikarthik et al. [5] proposed hybrid neural network and whale optimization algorithm for test case prioritization. Initially test case are generated for java cloud project. Test cases are clustered using fuzzy means clustering technique. Only relevant test cases cluster is used for TCP. For weight optimization process, whale optimization is used and for prioritization purpose artificial neural network is used. Zhang et al. [6] proposed hybrid regression test case selection using multiple granularities of class and combine the strength of classical regression testing. Method level changes are considered for test case selection and two parameters- selected test ratio, end to end duration time are used for performance evaluation.

Rothermel et al. [8] presented a survey describing basic guidelines for evaluation of test case selection technique. Code coverage and fault detection are two basic parameters upon which reliability of test case suite depends. Singh et al. [12] put forward TCS technique based on def-use associative relationships between variables. Selected test cases are prioritized by using def-use criteria. Empirical study was done on only six C programs that leads to research gap.

Zhang et al. [10] put forward clustering based TCS algorithm by identifying execution profiles of source code program but less focus is paid on total running time of test case suite. Wong et al. [11] used code complexity as well as fault handling capabilities of selected test case suite. Xu et al. [19] presented a “comparison by using graph” algorithm for TCS problem but only seven AspectJ java programs are considered for their empirical study. Various test case suite selection techniques are discussed in Table 1:

Table 1: Research Gaps in existing work

Technique Proposed	Criteria used for selection	Metrics used for evaluation	Proposed outcome solutions
Metaheuristic algorithm- bat algorithm and cuckoo search [3]	Source code execution	Number of faults detected and execution time.	Authors used the echolocation movements of bats and exploit search space by using the foraging behavior of bats. Good results are observed in reducing execution time but the initial population was restricted to select only 5 test cases at each run out of 567 test cases of lexical Analyzer program. (Initial population constraints are put to obtain an optimal test case suite)
Software development life cycle [13]	Requirement coverage and source code coverage	Number of faults detected and execution time	The authors prepositioned a framework that relies on requirements as well as source code coverage. A small project (Academic Time Synchronization Project) is used for their proposed algorithm.
Metaheuristic algorithms [14]	Source code coverage	APFD metric and execution time metrices are used.	Artificial bee colony (ABC), Cuscuta search (CS) and genetic algorithm (GA) are used for test case selection problem. Only eight test cases are used for their evaluation work.
Ant colony optimization [15]	Code coverage	APFD metric and execution time metrices are considered.	Foraging behavior of ant system is used for searching best search solutions found and test case suite is selected based on shortest path followed by ants. Small test case suite (eight test cases) is used for their empirical study.
Test factors and Ant colony optimization [16]	Code coverage	Execution time and faults detected by test cases are chosen as criteria for test case selection.	Authors considered code coverage, Importance, volatility, complexity, Fault rate and execution time as test vectors. To obtain test requirement coverage matrix test vectors are converted into binary numbers. No source code program is taken to show their empirical evaluation.
Genetic algorithm [17]	Code coverage	Execution time of test cases	Primary fitness(F1) function defined in Genetic algorithm is calculated based on test adequacy and secondary fitness function(F2) function is calculated based on running time of test cases. No focus is paid on fault revealing capability of test cases.
Clustering approach [18]	Code coverage, code complexity and historical information of test cases.	Execution time and fault detection ratio metrices are used by authors,	Fault detection ratio of test cases is identified and cluster formation is done. Small test case suite of 15 test cases "Microsoft Dynamics Ax" project is considered for their empirical work.

Whale optimization algorithm is a metaheuristic algorithm that is widely used in solving NP-hard problem as well as in various engineering disciplines. In the computer science domain, whale optimization is used in wireless sensor networks, cryptography, data mining applications, cloud-computing, neural network-based applications, Robotics and cryptography [40]. In applied mathematics, different optimization problems are solved by the whale optimization algorithm. In aeronautical engineering domain, WOA is used in Air Foil design, navigation and RADAR. So, whale optimization algorithm is efficient and it can

be used to solve test case selection problem in regression testing domain. [37][38]

3. Proposed Work

We propose an enhanced metaheuristic algorithm for TCS problem in regression testing.

3.1 **Basic ideology**: It is a swarm-intelligence-based optimization algorithm proposed by Mirjalili and Lewis in 2015. Swarm-based Metaheuristic algorithms use interactions among social creatures to obtain collective intelligence to solve complex situations in real world scenario [37][40]. Whales are intelligent creatures that have spindle cells in brains responsible for judgments about situations. This algorithm is based on Megaptera novaeanglide whale (also known as humpback whale). It involves two-phase mechanisms- Encircling the prey by exploring search space and exploitation of search space by attacking prey. These whales use bubble-net foraging behavior (spiral movement in an upward direction) for feeding of small fishes' present near the surface. Location of prey is recognized in search space to find candidate solution and all other search agents will update their position vectors following spiral movement of candidate solution found by whale. [37][38][39]

Table 2: Description of Whale Optimization Algorithm.

Encircling prey Phase	
$\vec{D} = \lvert\vec{C}.\vec{Y}1(t) - \vec{Y}(t)\rvert$	Here $\vec{Y}1$ is position vector of best solution that has already been obtained, $\vec{Y}$ is starting position vector, t is iteration number, C is the coefficient vector and D is collective behavior that represents candidate solution.
$\vec{Y}\,(t+1) = \vec{Y}1(t+1) - \overrightarrow{A}.\vec{D}$	This equation defines position vector for whale to search optimal solution. Here t is iteration number and A is coefficient vector.
$\overrightarrow{A} = 2.\vec{a}.\vec{r} - \vec{a}$	Here A is coefficient vector and r belong to [0,1] and vector a decrease from 2 to 0.
$\overrightarrow{C} = 2.\vec{r}$	Here C is coefficient vector and r belong to [0,1].
Intensification and exploration Phase	
$\overrightarrow{A} = 2.\vec{a}.\vec{r} - \vec{a}$	Shrinking encircling mechanism is followed by decreasing value of a over the course of iterations from 2 to zero.
$\overrightarrow{Y}(t+1) = \begin{cases} Y1(t) - \overrightarrow{A}.\vec{D}, & if\ p \le 0.5 \\ D.e^{bl}.\cos(2\pi l) + Y1(t) & if\ p \ge 0.5 \end{cases}$	Spiral updating position is obtained by calculating by distance of i^{th}whale and prey. Here p indicates random number in [0,1]

$\vec{D} = \lvert\vec{C}.\vec{Y}1random - \vec{Y}\rvert$	$\vec{Y}1\ random$ is position vector selected randomly for renewing positions of whales.
$\vec{Y}\ (t+1) = \vec{Y}1(random) - \overrightarrow{A}.\vec{D}$	This equation gives next random position vector in search space by spiral updating mechanism.

In classical WOA, randomization selection procedure for candidate solution is followed and positions of whales are updated accordingly. In order to enhance exploitation of search space, we use roulette wheel selection procedure instead of random selection [40[41].

3.2 **Proposed algorithm**:

An enhanced whale optimization algorithm is proposed for getting best optimal solution. In our proposed algorithm, crossover (C) and mutation (M) operators are introduced to increase exploitation of global search space. Main equations are described in Table3.

Table 3: Equations used in Proposed algorithm Ambha_WOA

$\vec{X}(t+1) = \vec{X} - \overrightarrow{A}.\vec{D}$	$\vec{X}$ is position of whale search agent, and D is collective behavior that represents candidate solution, t is iteration number and A is coefficient vector.
$r = 0.9 + \frac{0.9 * (i-1)}{maximum\ number\ of\ iterations}$	The parameter r (resultant solution) which decremented linearly from 0.9 to zero involves in shrinking encircling mechanism and depends upon iteration number.
$X_i^{t+1} = Crossover(X^{mutation}, X_i^t)$	Crossover operation is implemented and $X^{mutation}$is candidate solution obtained from mutation operation having X_i^t whale current position at t iteration.
$X^d = \begin{cases} X1d & if\ p \geq 0.5 \\ X2d & if\ p < 0.5 \end{cases}$	Crossover operation is implemented (X1d, X2d) and intermediate solution X^d is obtained with the same probability p and having dimension d. in this case, X1d is first reference parent and X2d is the second reference parent.

Pseudocode of Ambha_WOA is described as under:

Generate initial population X_i (i=1,2,3 .... n)
Calculate objective function value (APFD)metric of each candidate solution for test case selection
while(t<max_iteration)
for each solution s,
calculate mutation_rate (r),
update values of a, A, c and p
if (p<0.5 && |A| <1)
apply mutation operation on X^* (best solution)
given mutation_rate (r) to get $X^{mutation}$
perform crossover operation
elseif2(|A|>1)
select a random whale search agent X,
Apply mutation operator on X,
get $X^{mutation}$
perform crossover operation and assign new whale position of X_t to output of crossover.
end if2,
elseif1(p>=0.5)
end if1,
end for,
calculate fitness objective function of each solution,
if there is better candidate solution update X^*,
t=t+1,
end while, return X^{best} solution.
Output: Selected test case suite.

Fig. 1: pseudocode of proposed algorithm 'Ambha_WOA'

4. Experimental Setup

For implementing our experimental work, we use java Eclipse IDE (version 2021-12-4.22.0) on windows 11 having i5 processor. For test case selection problem, we use three criteria-coverage of statement_code, modification_code and fault-finding capacity of test cases present in a test case suite. We have used EclEmma tool for calculating code coverage. Junit test case suites are being run on junit tool and feature vector representation table is formulated by instance*feature (instance-test case, feature-statement coverage, modification coverage, fault coverage). Feature vector is represented by numerical data ranging from [0,1]. MATLAB 2016b is used for test case selection by Ambha_WOA. Number of candidate solutions are taken ten (10), number of maximum iterations are taken hundred (100), lower bound for search agents is taken as 10 and upper bound for search agents is taken as 10.

Table: 4 Description of datasets used for empirical study

Sr no.	Subject program	branches	methods	test classes	No of versions	test cases	No of faults
1	oryx_banking	665	313	74	2	156	10
2	omega_lib	715	318	55	2	165	11
3	max_hosp	746	256	56	2	188	14
4	ap_axi	806	248	96	2	159	16
5	studentApi	806	266	81	2	170	15
6	u_bank	830	298	89	2	178	16

7	delta_lib	794	232	90	2	220	17
8	apex_hosp	841	242	91	2	262	15
9	st_travis	241	282	15	2	350	17
10	student-service	225	311	28	2	322	14
11	ax_bank	229	345	35	2	359	12
12	user_interaction	339	368	42	2	358	11

Test cases are classified as true positive(tp), true negative(tn), false positive (fp) and false negative(fn). The following table describes metrices taken for evaluation.

Table: 5 Description of metrics used in evaluation of algorithm

Sr. No	Metric	Defined as	Description
1	Fault Detection Ratio (FDR)	$\frac{Number\ of\ faults\ identified}{Total\ number\ of\ faults} * 100$	It is defined as ratio of number of faults detected to total number of faults present in system program.
2	Average Percentage of Faults detection (APFD)	$1 - \frac{1}{m * n}\sum_{i=n}^{m} \mathrm{TF} + \frac{1}{2n}$	This metric is used to identify average of fault detection capability of n test cases having m faults in system program.
3	Classification Accuracy	$\frac{TP + TN}{TP + FP + TN + FN}$	It is defined as ratio of correctly classified test cases distribution to total test cases present in a test case suite.
4	Execution Time	T=total running time of test case suite.	It measures cost effectiveness of test case suite in terms of running time of test case suite.
5	Precision	$\frac{TP}{TP + FP}$	It is defined as ratio of number of correctly identified test cases to all true positive as well as false positive outcomes of test cases.
6	Recall	$\frac{TP}{TP + FN}$	It is defined as ratio of test cases (correctly classified to class P) to all test cases (which should be classified to that particular class P).
7	F-measure	$\frac{2 * recall * precision}{recall + precision}$	It represents harmonic mean of recall and precision.

5. Results

Results and comparative study of proposed algorithm with classical WOA are described as under:

1) **Fault Detection Ratio**: This metric helps to identify fault detection capability of test case suite. We observed that proposed algorithm has shown better results as compared to classical whale optimization algorithm.

Table 6: Comparative values of FDR of Ambha_WOA and WOA

Sr No.	Subject program	FDR(WOA)	FDR(Ambha_WOA)	Increase (%)
1	oryx_banking	77.9	91.6	17.5
2	omega_lib	71.6	90	25.6
3	max_hosp	61.1	87.5	43.2
4	ap_axi	68.9	95.7	38.8
5	studentApi	71.5	91.6	28.1
6	u_bank	82.7	92.3	11.6
7	delta_lib	79.8	90	12.7
8	apex_hosp	76.8	100	30.2
9	st_travis	73.8	92.5	25.3
10	student-service	79.1	90.7	14.6
11	ax_bank	74.6	93.3	25.0
12	user_interaction	71.1	94.4	32.7

It is observed in above Table that there is good increase of FDR metric in ap_axi (38%), max_hosp (43.2 %), user_interaction (32.7%) by using Ambha_WOA.

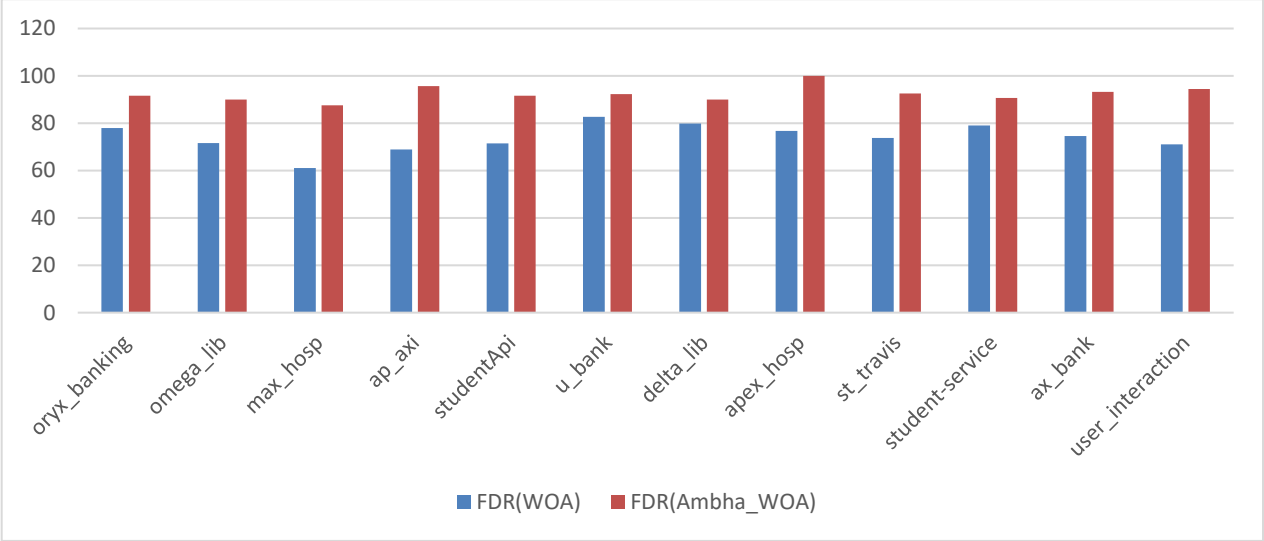


Fig 2: Graphical comparison of FDR metric of WOA vs Ambha_WOA.

2) **Average Percentage of Faults Detection**: In the following Table 7, it is seen that proposed algorithm has significantly improved APFD metric in case of omega_lib, max_hosp, u_bank and delta_lib.

Table 7: Comparative values of APFD metric of WOA and Ambha_WOA

Sr no.	Subject program	APFD (random order)	APFD(WOA)	APFD(Ambha_WOA)	Increase (%) WOA /Ambha_WOA
1	oryx_banking	0.456	0.751	0.912	21.4
2	omega_lib	0.457	0.521	0.899	72.5
3	max_hosp	0.458	0.567	0.887	56.4
4	ap_axi	0.432	0.657	0.891	35.6
5	studentApi	0.468	0.675	0.822	21.7
6	u_bank	0.498	0.536	0.836	55.9
7	delta_lib	0.412	0.599	0.898	49.7

8	apex_hosp	0.437	0.611	0.811	32.7
9	st_travis	0.456	0.654	0.854	30.5
10	student-service	0.465	0.675	0.875	29.6
11	ax_bank	0.455	0.679	0.872	28.4
12	user_interaction	0.466	0.673	0.873	29.7

Furthermore, it is observed that good increase in percentage (72.5%) in case of omega_lib dataset. Graphical comparison of APFD metric is given in following Figure 3:

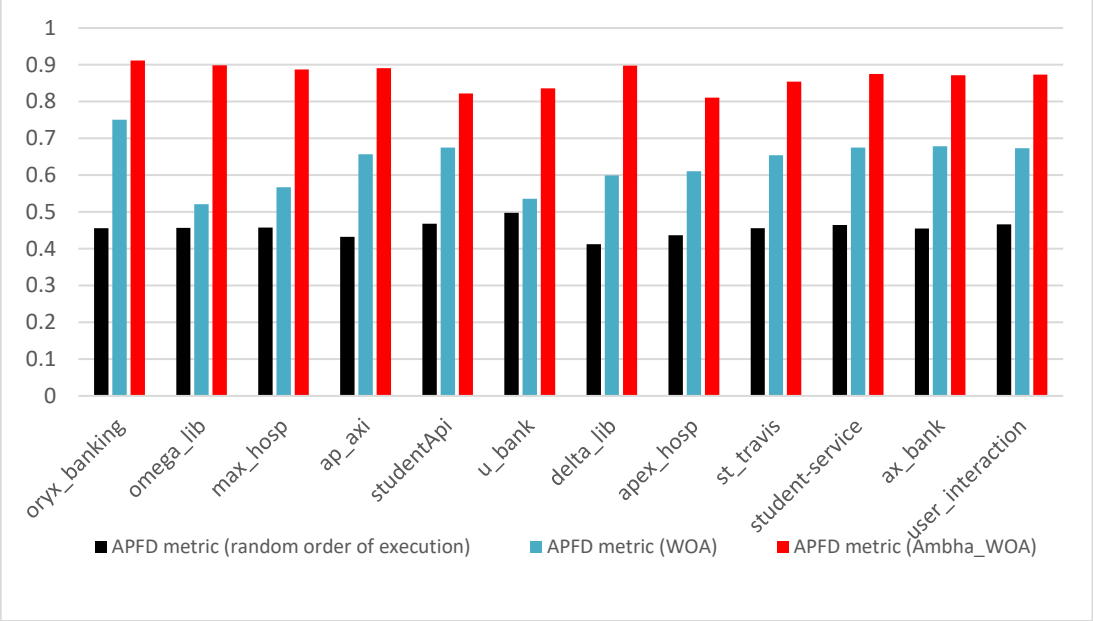


Fig 3: Comparison of APFD metric

3) **Classification Accuracy**: This metric is useful to know accuracy of classified test cases. The following Table 8, presents comparison of WOA with Ambha_WOA.The dataset oryx_banking (99.1), max_hosp (98.8), student-service (96.8) have good accuracy values by using Ambha_WOA and good increase in percentage i.e., student-service (26.8%) as compared with classical whale optimization algorithms in same datasets.

Table 8: Comparative analysis of classification accuracy of Ambha_WOA with WOA

Sr no.	Subject program	WOA	Ambha_WOA	Increase (%)
1	oryx_banking	89.6	99.1	10.6
2	omega_lib	81.3	95.3	17.2
3	max_hosp	85.5	98.8	15.5
4	ap_axi	82.5	98.4	19.2
5	studentApi	80	95.6	19.5
6	u_bank	82	96.5	17.6
7	delta_lib	89	96.5	8.4
8	apex_hosp	86.5	96.4	11.4
9	st_travis	84.3	93.5	10.9
10	student-service	76.3	96.8	26.8
11	ax_bank	79.5	96.5	21.3
12	user_interaction	78.6	96.5	22.7

It can be shown graphically in Figure 4,values of Ambha_WOA have shown good results as compared to WOA.

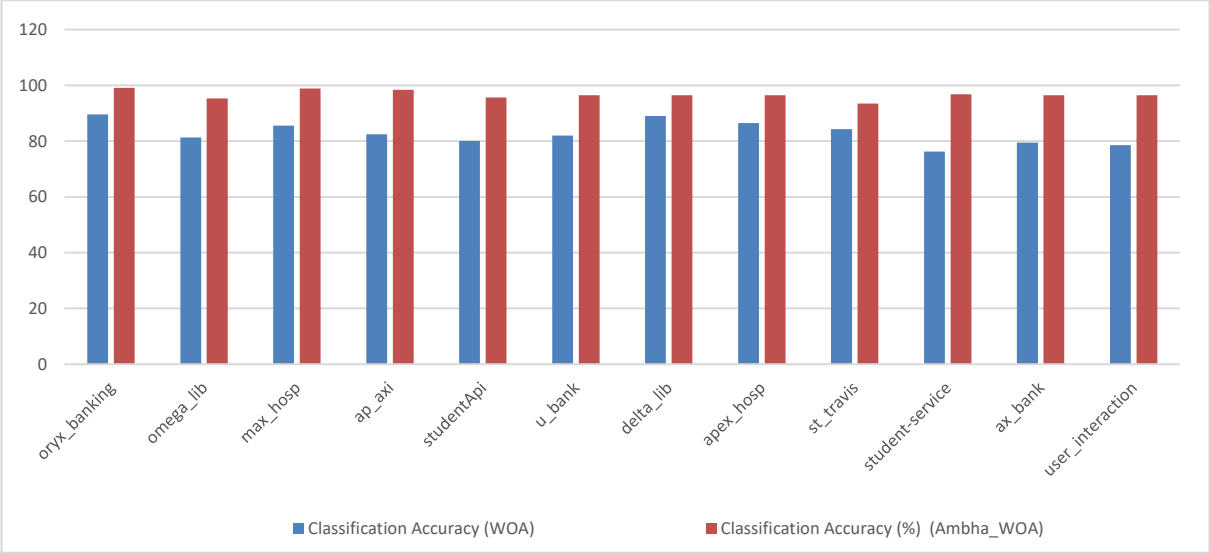


Fig 4: Comparative Analysis of Classification Accuracy metric

4) Execution Time

In the following Table 9, execution time of running a test case suite is reduced in omega_lib, max_hosp, ap_axi, u_bank and user_interaction.

Table 9: Comparative values of execution time of test case suite of WOA and Ambha_WOA

Sr no.	Subject program	WOA	Ambha_WOA	decrease (%)
1	oryx_banking	31.1	23.9	23.1
2	omega_lib	21.5	11.9	44.6
3	max_hosp	28.9	11.6	59.8
4	ap_axi	23.8	9.8	58.8
5	studentApi	22.4	9.7	56.6
6	u_bank	26.7	10.9	59.1
7	delta_lib	18.8	12.9	31.3
8	apex_hosp	19.8	11.7	40.9
9	st_travis	21.3	12.9	39.4
10	student-service	22.3	12.9	42.1
11	ax_bank	24.5	14.7	40
12	user_interaction	24.5	10.2	58.3

Graphically, it can be presented as shown in the following figure 5:

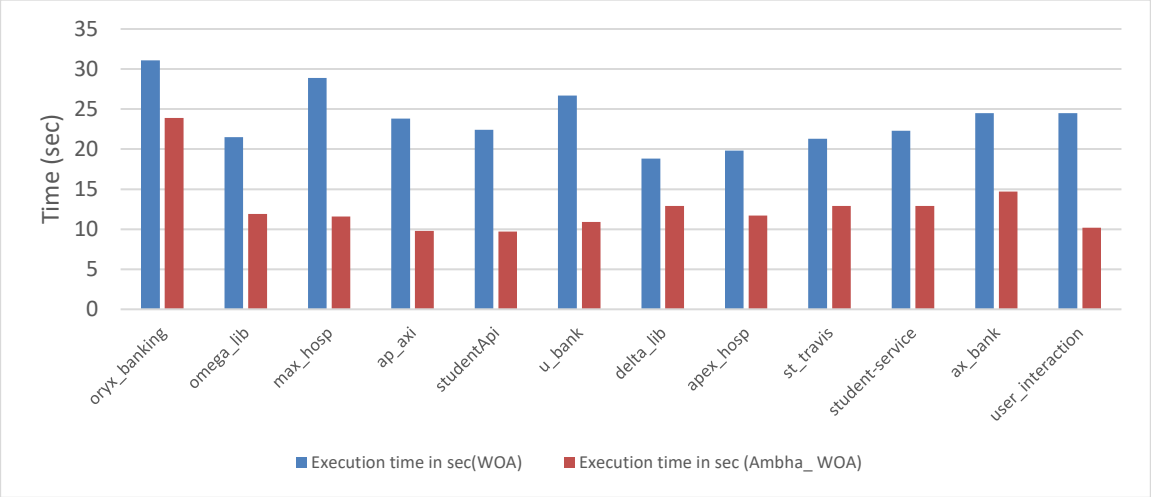


Fig. 5: Comparative Analysis of Classification Accuracy metric

5) Precision

The precision metric has increased results in omega_lib, apex_hosp and student_service.

Table 10: Comparative values of precision metric of WOA and Ambha_WOA

Sr no.	Subject program	WOA	Ambha_WOA	Increase (%)
1	oryx_banking	0.671	0.762	13.5
2	omega_lib	0.621	0.821	32.2
3	max_hosp	0.665	0.761	14.4
4	ap_axi	0.778	0.871	11.9
5	studentApi	0.665	0.761	14.4
6	u_bank	0.761	0.891	17.0
7	delta_lib	0.671	0.782	16.5
8	apex_hosp	0.712	0.962	35.1
9	st_travis	0.698	0.821	17.6
10	student-service	0.712	0.961	34.9
11	ax_bank	0.776	0.971	25.1
12	user_interaction	0.778	0.891	14.5

Graphically, it can be represented as shown in Figure 6:

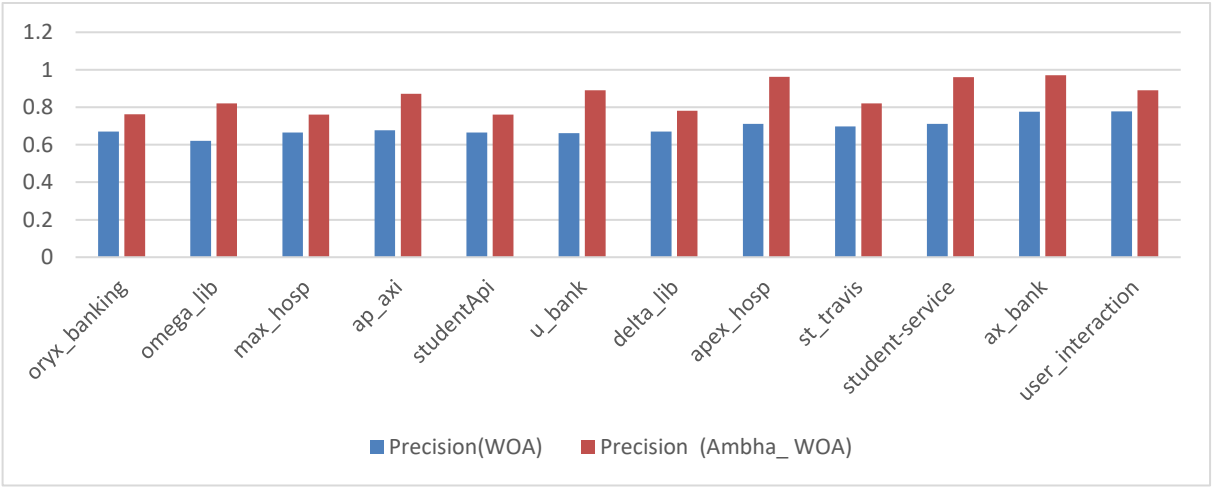


Fig. 6: Comparative Analysis of Precision metric

6) Recall

Good values are observed in oryx_banking(0.981), delta_lib(0.98),st_travis(0.923), student-service(0.991) and user_interaction(0.989).

Table 11: Comparative values of Recall metric of WOA and Ambha_WOA

Sr no.	Subject program	WOA	Ambha_WOA	Increase (%)
1	oryx_banking	0.761	0.981	28.9
2	omega_lib	0.791	0.91	15.1
3	max_hosp	0.682	0.881	29.1
4	ap_axi	0.762	0.911	19.5
5	studentApi	0.721	0.9	24.8
6	u_bank	0.661	0.892	10.8
7	delta_lib	0.671	0.989	47.3
8	apex_hosp	0.791	0.911	15.1
9	st_travis	0.611	0.923	51.0
10	student-service	0.787	0.991	25.9
11	ax_bank	0.767	0.977	27.3
12	user_interaction	0.765	0.989	29.2

Further, it can be presented as shown in the following figure 7:

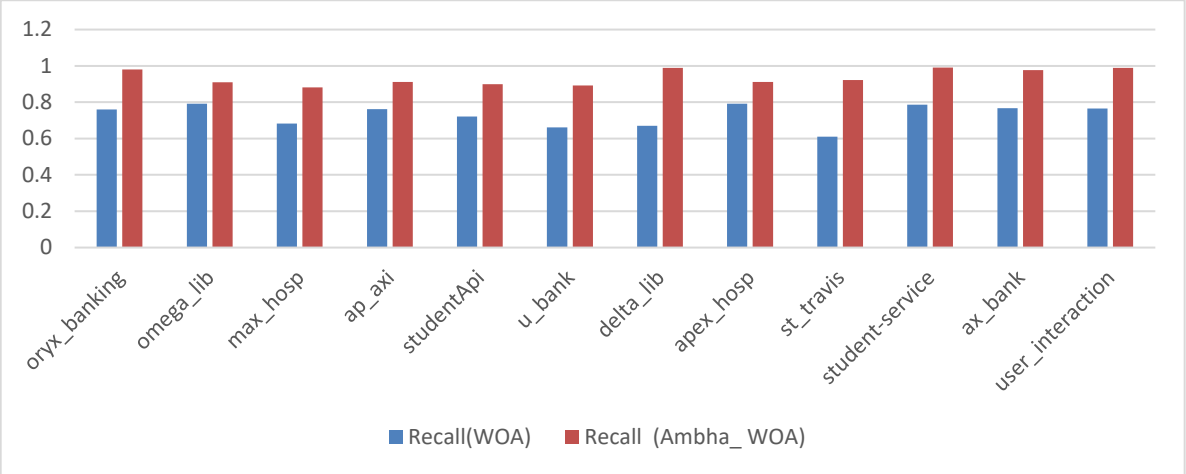


Fig. 7: Comparative Analysis of Recall metric

7) F-measure

Ambha_WOA have been proved better in terms of F-measure in datasets apex_hosp (0.921), st_travis (0.961), student-service (0.921) and user_interaction (0.911).

Table 12: Comparative values of F-measure metric of WOA and Ambha_WOA

Sr no.	Subject program	WOA	Ambha_WOA	Increase (%)
1	oryx_banking	0.621	0.821	32.2
2	omega_lib	0.665	0.761	14.2
3	max_hosp	0.778	0.871	11.9
4	ap_axi	0.665	0.761	14.4
5	studentApi	0.761	0.891	17.0
6	u_bank	0.771	0.882	14.3

7	delta_lib	0.712	0.862	21.0
8	apex_hosp	0.798	0.921	15.4
9	st_travis	0.712	0.961	34.9
10	student-service	0.676	0.921	36.2
11	ax_bank	0.678	0.891	31.4
12	user_interaction	0.791	0.911	15.1

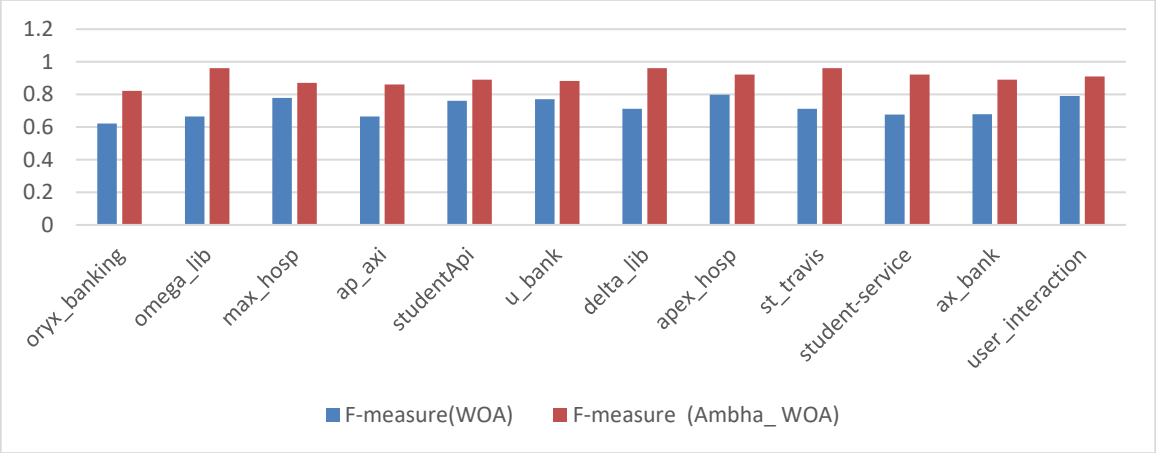


Fig. 8: Comparative Analysis of F-measure metric

6. Non-parametric statistical tests

To prove correctness of our 'Ambha_WOA' algorithm, we conducted statistical test on software

IBM SPSS 22 (an open-source statistical software). Tests of normality areaccompanied to see variation of data from normal.

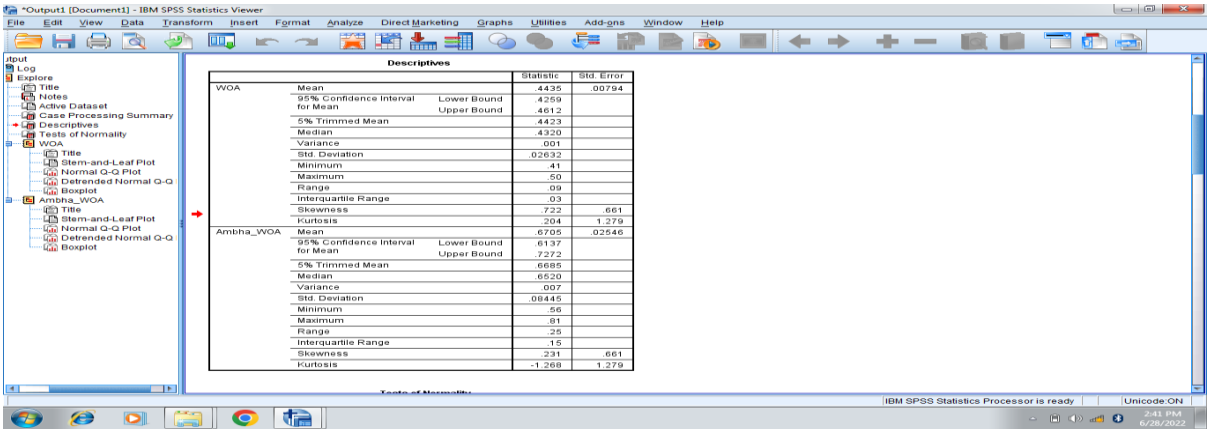


Fig. 9: Comparative Analysis of statistical parameters taken for WOA and Ambha_WOA

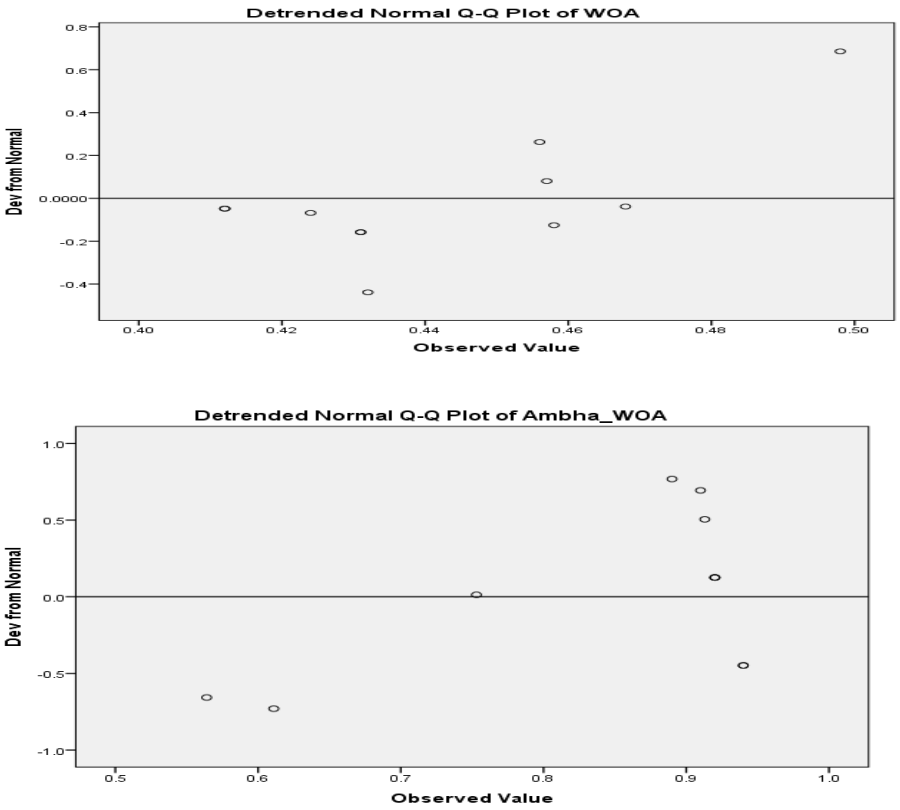
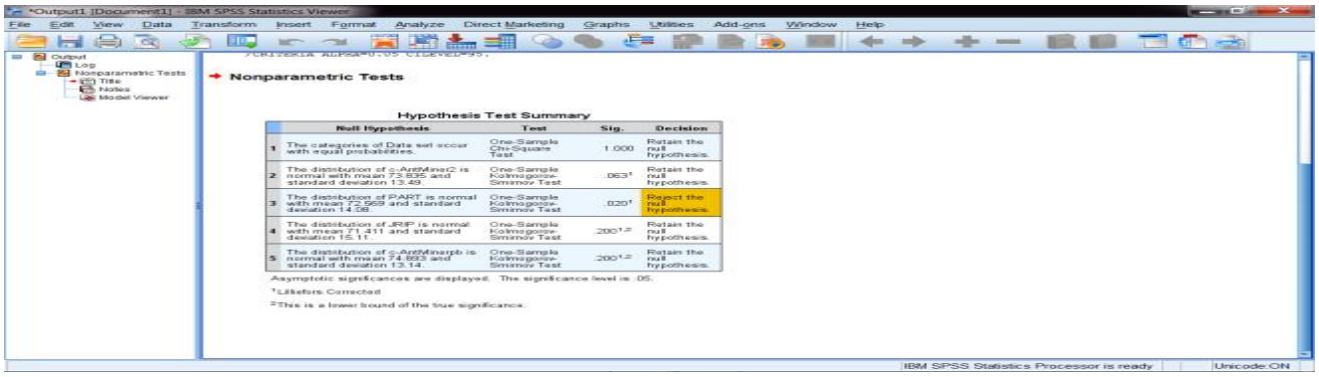


Fig. 10: Normality test conducted on dataset (APFD metric considered) for WOA and Ambha_WOA.

a) Wilcoxon Signed (WS) test

Sum of positive ranks and sum of negative ranks are used to exercise this test.For doing comparison of WOA and Ambha_WOA (proposed), We took 'α=0.05', n_i is difference of scores between two datasets.

$Rank_{post} = \sum rank(n_i>0) + \frac{1}{2}\sum rank(n_i=0); Rank_{negt} = \sum rank(n_i<0) + \frac{1}{2}\sum rank(n_i=0);$
Mean $Rank_{post}$ = 4.71 and $Rank_{negt}$= 7.61;level of significance=0.05.
Two-tailed test values = 0.241 and 0.042.We used standard 'stat table' tocompetevalues and it is seen that Ambha_WOA has shown promising results than WOA for TCS problem.



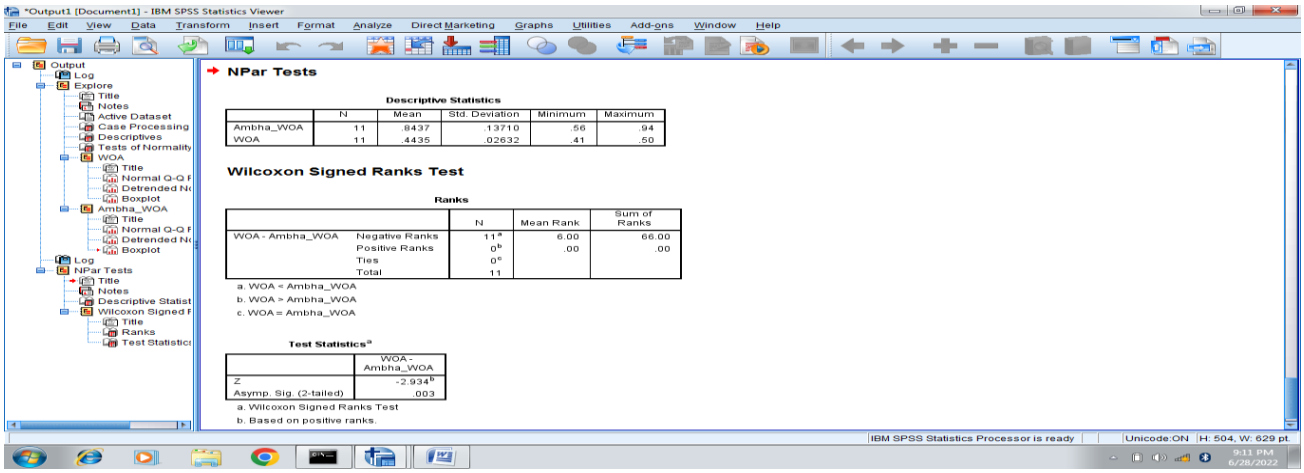


Fig. 11: Screenshots of Wilcoxon Signed Ranks Test conducted for comparing WOA and Ambha_WOA.

b) Friedman (F) test

Friedman test is conducted by using average and break tie between two ranks.

$$(CHI)^2 = \frac{12n}{x(x+1)}\sum(rank)^2 - \frac{x(x+1)}{4}$$

Here (x+1) symbolizesdegree of freedom and '*CHI*' estimates average (ranks) of values.

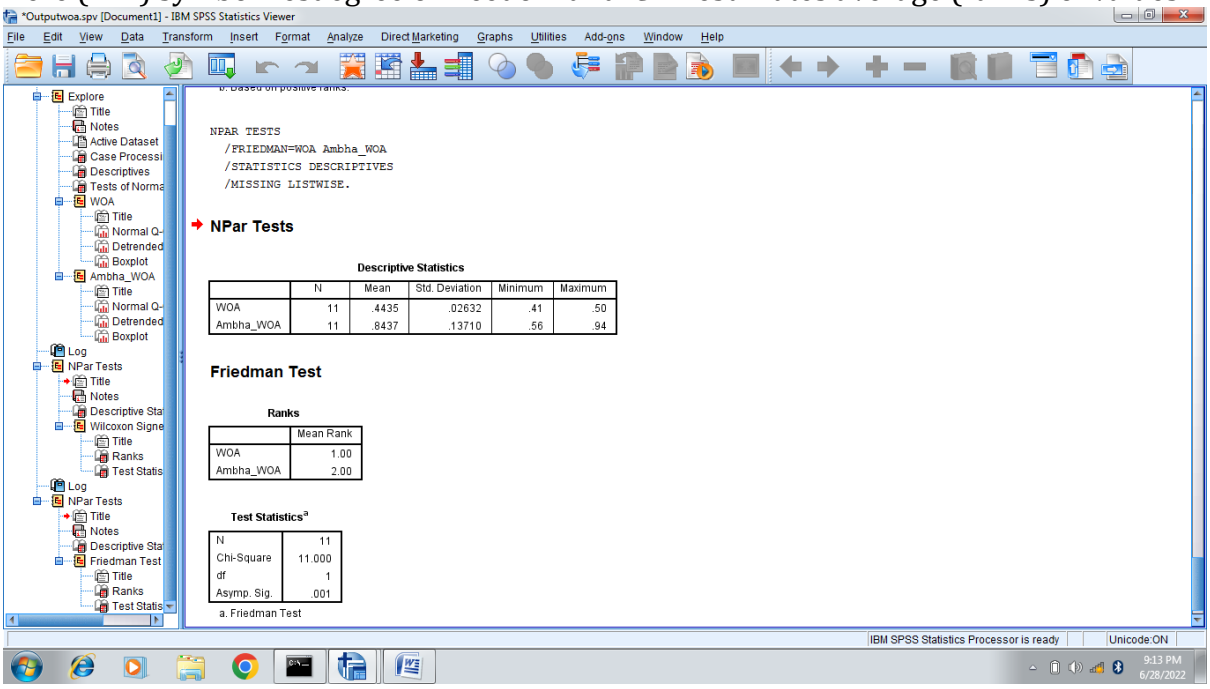


Fig. 12: Screenshots of FriedmanTest conducted for comparing WOA and Ambha_WOA.

It is viewedthat at $'\alpha = 0.05, (CHI)^2 = 12.63'$present in rejection region (according to standard stat table) which inferthat our proposed algorithmdo better thanWOA algorithm.

7. Conclusion

To produce and maintain good quality software's, regression testing is considered as an essential as well as important activity in software industry. Regression test case suites increase in dimensions to meet the everchanging needs of customers playing major role in software modifications. To address issues of size and time constraints, regression testing techniques are proven to be beneficial in maintenance phase of software. Metaheuristic algorithms are proven to be very efficient in dealing with NP-hard real life optimization problems. The whale optimization algorithm is (based on encircling mechanism of whales to hunt the prey) used to solve test case selection problem and it is observed that proposed Ambha_WOA has promising results in APFD metric, Classification Accuracy, F-measure, Recall, Fault detection Ratio and execution time.

References

1. Panichella, A., Oliveto, R., Di Penta, M., & De Lucia, A. (2014). Improving multi-objective test case selection by injecting diversity in genetic algorithms. *IEEE Transactions on Software Engineering*, *41*(4), 358-383.
2. Ma, B., Wan, L., Yao, N., Fan, S., & Zhang, Y. (2021). Evolutionary selection for regression test cases based on diversity. *Frontiers of Computer Science*, *15*(2), 1-3.
3. Kaur, A., & Agrawal, A. P. (2017, January). A comparative study of bat and cuckoo search algorithm for regression test case selection. In *2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence* (pp. 164-170). IEEE.
4. Souza, L. S., de Miranda, P. B., Prudencio, R. B., & Barros, F. D. A. (2011, November). A multi-objective particle swarm optimization for test case selection based on functional requirements coverage and execution effort. In *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence* (pp. 245-252). IEEE.
5. Harikarthik, S. K., Palanisamy, V., & Ramanathan, P. (2019). Optimal test suite selection in regression testing with testcase prioritization using modified Ann and Whale optimization algorithm. *Cluster Computing*, *22*(5), 11425-11434.
6. Zhang, L. (2018, May). Hybrid regression test selection. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)* (pp. 199-209). IEEE.
7. Harrold, M. J., Jones, J. A., Li, T., Liang, D., Orso, A., Pennings, M., & Gujarathi, A. (2001). Regression test selection for Java software. *ACM Sigplan Notices*, *36*(11), 312-326.
8. Rothermel, G., & Harrold, M. J. (1996). Analyzing regression test selection techniques. *IEEE Transactions on software engineering*, *22*(8), 529-551.
9. Suri, B., Mangal, I., & Srivastava, V. (2011). Regression test suite reduction using an hybrid technique based on BCO and genetic algorithm. *Special Issue of International Journal of Computer Science & Informatics (IJCSI), ISSN (PRINT)*, 2231-5292.
10. Zhang, C., Chen, Z., Zhao, Z., Yan, S., Zhang, J., & Xu, B. (2010, July). An improved regression test selection technique by clustering execution profiles. In *2010 10th International Conference on Quality Software* (pp. 171-179). IEEE.
11. Wong, W. E., Horgan, J. R., London, S., & Agrawal, H. (1997, November). A study of effective regression testing in practice. In *PROCEEDINGS The Eighth International Symposium on Software Reliability Engineering* (pp. 264-274). IEEE.
12. Singh, Y., Kaur, A., & Suri, B. (2009). Empirical validation of variable based test case prioritization/selection technique. In*International Journal of Digital Content Technology and its Applications*, (pp. 116-123).
13. Siddik, M. S., & Sakib, K. (2014, December). RDCC: An effective test case prioritization framework using software requirements, design and source code collaboration. In *2014 17th International Conference on Computer and Information Technology (ICCIT)* (pp. 75-80). IEEE.
14. Mittal, S., & Sangwan, O. P. (2018). Prioritizing test cases for regression techniques using metaheuristic techniques. *Journal of Information and Optimization Sciences*, *39*(1), 39-51.
15. Singh, Y., Kaur, A., & Suri, B. (2010). Test case prioritization using ant colony optimization. *ACM SIGSOFT Software Engineering Notes*, *35*(4), 1-7.
16. Ahmad, S. F., Singh, D. K., & Suman, P. (2018). Prioritization for regression testing using ant colony optimization based on test factors. In *Intelligent communication, control and devices* (pp. 1353-1360). Springer, Singapore.
17. Mohapatra, S. K., & Prasad, S. (2013, December). Evolutionary search algorithms for test case prioritization. In *2013 International Conference on Machine Intelligence and Research Advancement* (pp. 115-119). IEEE.
18. Carlson, R., Do, H., & Denton, A. (2011, September). A clustering approach to improving test case prioritization: An industrial case study. In *2011 27th IEEE International Conference on Software Maintenance (ICSM)* (pp. 382-391). IEEE.
19. Xu, G., & Rountev, A. (2007, May). Regression test selection for AspectJ software. In *29th International Conference on Software Engineering (ICSE'07)* (pp. 65-74). IEEE.

20. http://travis.ci.com
21. http://spring.io.projects
22. http://github.com
23. http://sir.csc.ncsu.edu
24. Yoo, S., & Harman, M. (2012). Regression testing minimization, selection and prioritization: a survey. *Software testing, verification and reliability*, *22*(2), 67-120.
25. Yang, X. S., & Press, L. (2010). Nature-inspired metaheuristic algorithms second edition.
26. Rothermel, G., & Harrold, M. J. (1996). Analyzing regression test selection techniques. *IEEE Transactions on software engineering*, *22*(8), 529-551.
27. Said, G. A. E. N. A., Mahmoud, A. M., & El-Horbaty, E. S. M. (2014). A comparative study of meta-heuristic algorithms for solving quadratic assignment problem. *arXiv preprint arXiv:1407.4863*.
28. Myers, G. J. (1979). The Art of Software Testing John Wiley and Sons Ltd.
29. Rajabi Moshtaghi, H., Toloie Eshlaghy, A., & Motadel, M. R. (2021). A comprehensive review on meta-heuristic algorithms and their classification with novel approach. *Journal of Applied Research on Industrial Engineering*, *8*(1), 63-89.
30. Stegherr, H., Heider, M., & Hähner, J. (2020). Classifying Metaheuristics: Towards a unified multi-level classification system. *Natural Computing*, 1-17.
31. Rajabi Moshtaghi, H., Toloie Eshlaghy, A., & Motadel, M. R. (2021). A comprehensive review on meta-heuristic algorithms and their classification with novel approach. *Journal of Applied Research on Industrial Engineering*, *8*(1), 63-89.
32. Sommerville, I. (2011). *Software Engineering, 9/E*. Pearson Education India.
33. Fister Jr, I., Yang, X. S., Fister, I., Brest, J., & Fister, D. (2013). A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186*.
34. Hao, J. K., & Solnon, C. (2020). Meta-heuristics and artificial intelligence. In *A Guided Tour of Artificial Intelligence Research* (pp. 27-52). Springer, Cham.
35. Kazmi, R., Jawawi, D. N., Mohamad, R., Ghani, I., & Younas, M. (2017). A Test Case Selection Framework and Technique: Weighted Average Scoring Method. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, *9*(3-4), 15-22.
36. Mahapatra, R. P.,Ranjith A. andKulothungan. (2018). A Framework for improving Test Case Selection And Randomized Prioritization. International Journal of Pure and Applied Mathematics, 18(22), (pp. 1927-36).
37. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software*, *95*, 51-67.
38. Pham, Q. V., Mirjalili, S., Kumar, N., Alazab, M., & Hwang, W. J. (2020). Whale optimization algorithm with applications to resource allocation in wireless networks. *IEEE Transactions on Vehicular Technology*, *69*(4), 4285-4297.
39. Gharehchopogh, F. S., & Gholizadeh, H. (2019). A comprehensive survey: Whale Optimization Algorithm and its applications. *Swarm and Evolutionary Computation*, *48*, 1-24.
40. Rana, N., Latiff, M. S. A., Abdulhamid, S. I. M., & Chiroma, H. (2020). Whale optimization algorithm: a systematic review of contemporary applications, modifications and developments. *Neural Computing and Applications*, *32*(20), 16245-16277.
41. Salgotra, R., Singh, U., & Saha, S. (2019). On some improved versions of whale optimization algorithm. *Arabian Journal for Science and Engineering*, *44*(11), 9653-9691.