Journal of Information Technology Management

Print ISSN: 2008-5893

Online ISSN: 2423-5059

Regression Test Suite Minimization Using Modified Artificial Ecosystem Optimization Algorithm

Abhishek Singh Verma*

*Corresponding Author, Assistant Professor, CSED, School of Engineering & Technology, Sharda University, Greater Noida, India. E-mail: abhiverma2005@gmail.com

Ankur Choudhary

Professor, Department of Computer Science & Engineering, Sharda University, Greater Noida, India. E-mail: ankur.tomer@gmail.com

Shailesh Tiwari

Professor, Department of Computer Science & Engineering, ABES Engineering College, Ghaziabad, India. E-mail: shail.tiwari@yahoo.com

Abstract

Now a day's software is the baseline for the success of any organization. There is a huge demand of quality software in the customer-oriented market. Regression testing makes it possible but it's a costly affair. Regression test suite minimization is way to reduce this cost but it is NP hard problem. This paper proposes an effective approach for regression test suite minimization using Artificial Ecosystem Optimization algorithm. To improve its performance a modified Artificial Ecosystem Optimization algorithm is proposed for Test case minimization. To evaluate the performance of proposed approach experiment is conducted in controlled parameter setting on open-source subject program from SIR repository. The results are collected and analyzed in comparison to existing approaches using statistical test. The test results reflect the superiority of proposed approach.

Keywords: Test Suite Minimization, Regression testing, AEO, MAEO.

DOI: 10.22059/jitm.2021.80023

Introduction

Software development life cycle comprises of various stages including an inevitable one called maintenance. Regression testing is conducted during maintenance and consumes almost 50-60% of total software development cost. Apart from it timely delivery of software products within budget in every maintenance activity is still a big challenge for industry. Hence, there exists a scope of improvement to reduce time and cost by improving the regression testing process (Augustsson, 2012).

Regression test case optimization is an effective way to overcome such issues. There exist three different types of regression test suite optimization such as: Minimization, Selection and Prioritization (Yoo & Harman, 2010). Test suite minimization works by reducing the size of the original test suite that satisfies some test adequacy criteria. Test case selection focuses on selecting a subset of test cases from the original test suite. Test case prioritization on the other hand doesn't reduce the size of the original test suite rather it reorders the existing test cases in some order so that they achieve some test adequacy criteria in decreasing/increasing order. In this paper author focuses on Test Case Minimization technique. To better understand the problem first we present a formal description of problem statement. "Software is a collection of programs $P: P_1, P_2, P_3, \ldots \ldots P_N$ with test suites $TS_1, TS_2, TS_3, \ldots \ldots TS_N$ respectively. For a program P_i the test suite TS_i consists of a set of reusable test cases $TS_i: TC_i^1, TC_i^2, TC_i^3, \ldots . TC_i^M$. F denotes the K faults exist in P_i which can be covered by TS_i. We derive a minimized subset of test suite $TS_i' \subseteq TS_i$ such as $TS_i': TC_i^1, TC_i^2, TC_i^3 \ldots . . TC_i^L$ where L < M. All the faults F must be covered from the test suite TS_i' to achieve the desired fault coverage."

Software products are evolutionary in nature and hence the test suite size keeps growing with every maintenance cycle. So, retest whole test suite after maintenance becomes very costly and time consuming. Regression test suite minimization is an effective solution to reduce time and effort required during regression testing. But the test case minimization becomes complex and transform to a NP hard problem, due to the presence of redundant test cases in existing test suite.

Literature revealed that various heuristic and Meta heuristic approaches have been applied to solve the test case minimization problems. But the scope of optimization still exists. Various nature inspired optimization algorithm such as Bat Search algorithm, Harris Hawk Optimization algorithm, Crow Search algorithm etc. (Kaur et al., 2017; Kaur at al., 2020; Chaudhary et al., 2019) have been utilized for regression test case optimization but there exist a room for improvement.

In this paper a new approach for Test Suite Minimization using Hybrid Artificial Ecosystem Optimization algorithm has been proposed. The main contribution of the paper is as follows:

- Proposed Test Case Minimization approach using Artificial Ecosystem Optimization (AEO) algorithm.
- Updated the performance of proposed approach by changing exploration and exploitation process of AEO.
- Rigorous statistical analysis have been conducted which reflects the superiority of proposed approach for test case Minimization over Bat Search algorithm, , Harris Hawk Optimization algorithm, Crow Search algorithm etc. (Kaur et al., 2017; Kaur et al., 2020; Chaudhary et al., 2019).

The structure of this paper is as follows. Section-2 briefly discusses the related works in the same area. Section-3 describes the proposed approach. Section-4 discloses the experimental setup and formulate the research questions. Section-5 report the results and answer the research questions formulated. Section 6 discusses the threat to validity and finally, section-6 concludes the paper with future extension of the proposed work.

Related Work

Test Suite Minimization became important if the test suites are not maintained properly, this may result in enormous test execution cost and diminish the regression testing benefits. Various researchers have conducted various studies in test suite minimization and reduction.

Test suite size plays an important role in maintain testing cost, to reduce the test suite size a heuristic approach has been proposed which identify and eliminate the redundant & obsolete test cases based on different degree of essentialness for data flow testing (Harrold et al., 1993). To find out an optimal subset of test suite an another heuristic approach based on GRE (Greedy Redundant and Essential) has been proposed (Chen & Lau, 1998a). Lee and Chung proposed another approach for selecting optimal subset of test suite by utilizing enhanced zero–one optimal path set selection method (Chung & Lee, 1997; Lee & Chung, 2000). Chen and Lau performed a simulation study on above discussed approaches and concluded that no single approach performs better than other two approaches in all the cases and solve the test suite minimization problem using divide and conquer approach for finding minimal and optimal representative test subsets (Chen & Lau, 1998b, 2003). A bi-criteria decision making model for test suite minimization considering test suite size and error detection rate as two criteria has been proposed (Black et al., 2004). Sprenkle et al. compared the applied approaches (Harrold et al., 1993; Black et al., 2004) for user-session-based testing

of different web applications (Sprenkle et al., 2005). Jeffery et al. modified the Harrold, Gupta and Jean, 1993 approach keeping selective redundant test cases and claimed better performance (Jeffrey & Gupta, 2005; Jeffrey & Gupta, 2007).

Few researchers continued to explore the performance of existing heuristic approaches such as greedy approach, Harrold, GRE and integer linear programming approaches on Java programs (Harrold et al., 1993; Littlewood & Sofer, 1987)(Black et al., 2004; Chen & Lau, 1998a).In their empirical study authors have analyzed the differences and similarity of all the selected approaches (Zhang et al., 2011).To perform early estimation of errors and reduce the testing time and effort, Gupta et al. proposed an approach by utilizing decision table generated from software requirement specification and claimed approx. 33% of saving in cost and time (Gupta et al., 2014). A greedy based approach for coverage based test suite reduction has been proposed and claimed the better code coverage in comparison to Harrold and Black approach (Harris & Raju, 2015). Khan et al. taken multiple test adequacy criteria's statement, branch , path adequacy, requirement and code coverage with minimum size test suite and claimed good performance (Khan et al., 2017). A fault coverage based test suite optimization approach inspired from Harrold approach has been proposed and this approach has further been compared against Greedy, Additional Greedy, HGS, and Enhanced HGS approaches and claimed better accuracy (Agrawal et al., 2019).

Regression test suite minimization, being an NP problem, various researcher explored its solutions using search-based approaches. Yoo and Harman proposed a pareto optimal test case selection approach taking coverage and cost as a multi objective criteria (Yoo & Harman, 2007). Maia et al. proposed multi objective approach for test case selection problem by utilizing multi objective genetic algorithm and claimed better performance than random search approach (Loiola & Maia, 2009). Yoo and Harman further utilized pareto optimal multi-objective optimization to solve test suite minimization (Yoo & Harman, 2010). A hybrid technique by combining Bee Colony optimization and genetic algorithm has been proposed, in this approach crossover parameter have been borrowed from genetic algorithm to generate and update the new solutions (Suri et al., 2011). You et al. proposed a genetic algorithm based time aware test suite reduction approach which removes the redundant test cases and minimize the total test case execution time (You & Lu, 2012). A fuzzy logic based approach has been proposed for multi objective test suite minimization problem (Haider et al., 2012), and also performed an empirical analysis (Haider et al., 2013) on computational and fuzzy based approach for safe reduction of test suites. The analysis claimed that the fuzzy based approach is safer then computational approaches for test suite minimization (Haider et al., 2013). Ahmad utilized cuckoo search optimization technique to solve configuration-aware structural testing the results was compared and claimed better then GA, PSO and other similar approaches (Ahmed, 2015). Turner et al. utilized NSGA II approach for test suite minimization on Mockito framework (Drake et al., 2016). Gupta et al. proposed multi

objective test suite optimization approach for detection and localization of test cases (Gupta et al., 2020).

The approaches discussed above claim that both heuristic and metaheuristic-based approaches have been proposed to address the test suite minimization problem but still a scope of optimization exists. In this paper first authors have applied original AEO approach for minimizing test case in test suite. In order to improve performance a modified approach is also proposed using Artificial Ecosystem Optimization. The proposed approach is further utilized for test suite minimization.

Proposed Approach: Test Case Minimization Using AEO AND MAEO

Nature is ocean of knowledge and inspiration; it's an excellent example is ecological system. Nature intelligently maintains a perfect balance in ecological system. Zhao et al.(Weiguo et al., 2019) took inspiration from nature and proposed an effective nature inspired algorithm called artificial ecosystem-based optimization (AEO) algorithm. AEO mimics the production, consumption, and decomposition behaviour of living creatures (Weiguo et al., 2019). The main aim of every meta-heuristics is to maintain balance between exploration and exploitation during their search process (Yang, n.d.; Kumar et al., 2018). In AEO, production, consumption, and decomposition behaviour is considered as operators (Weiguo et al., 2019). The role of production operator is to maintain the balance among exploitation and exploration behaviour while the consumption operator is utilized for to improve the exploration and at last decomposition operator is utilized to promote the exploitation behaviour. In this paper authors have proposed modified AEO to improvise the exploitation and exploration process by utilizing Equilibrium optimizer (Faramarzi et al., 2020) equations in AEO algorithm for test case minimization problem. Authors have used same problem statement as discussed in introduction section to explain the algorithmic process.

In order to achieve optimal solution AEO and its modified version follows certain rules (Weiguo et al., 2019):

1. The producer, consumer, and decomposer are three types of organisms included in the ecosystem.

2. As an individual, there is only one producer as well as only one decomposer in the population.

3. Rest of the population is treated as consumers, which are categorized as a carnivore, an herbivore, or an omnivore with equal probability.

4. The energy level of individuals is evaluated by cost function. Higher the value of cost function reflects higher energy of the individual for minimization problem.

A. Production:

The producer is considered as the worst individual of the ecological system, which may produce food energy from natural resources such as water, sunlight & carbon dioxide. The decomposer is the one who provides nutrition to the producer. The producer will be updated or generated from the random value between lower and upper limits of the search space (test suites) TS_{rand} and the decomposer TS_N.

The mathematical representation of production operator in AEO is as follows:

$$TS_1(itr+1) = (1-a)TS_N(itr) + a * TS_{rand} \quad (1)$$

$$a = \left(1 - \frac{itr}{TI}\right) r1 \quad (2)$$

$$TS_{rand} = r(UB - LB) + LB \quad (3)$$

where $TS1$ represents production operator, TS_{rand} represents random test suite, TS_N represents best test suite so far or decomposer, a is a linear weight coefficient between $[0-1]$ to maintain balance between random test suite TS_{rand} and decomposer TS_N for updating $TS1$ (Production operator), r and r1 are random numbers ranges between $[0-1]$, itr denotes the current iteration and TI denotes total iterations.

In AEO, the linear weight coefficient decreases linearly from 1 to 0 using equation 2, while the proposed MAEO uses exponential function to decrease the value of a from 1 to 0 over the iterations. As shown in figure1 the exploitation and exploration will ranges 70% and 30 %.

$$a = \left(1 - \frac{itr^2}{TI^2}\right) r1 \quad (4)$$

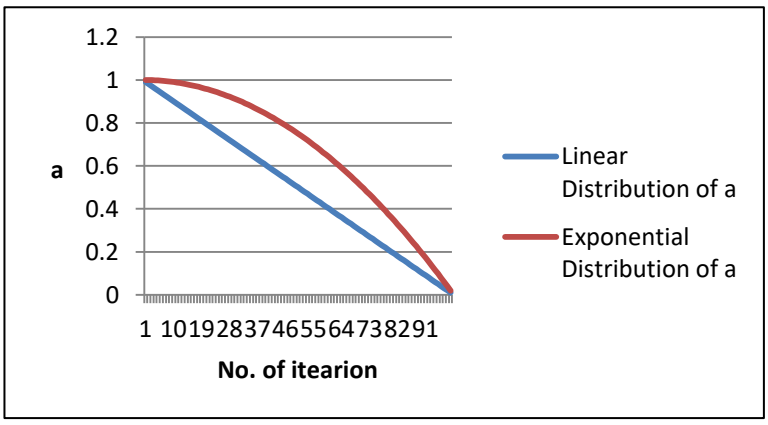

Figure 1. Linear and exponential distribution of "a"

B. Consumption

The consumption operator is performed by all the consumers. Each consumer may eat either a randomly selected lower energy consumer or a producer or both.

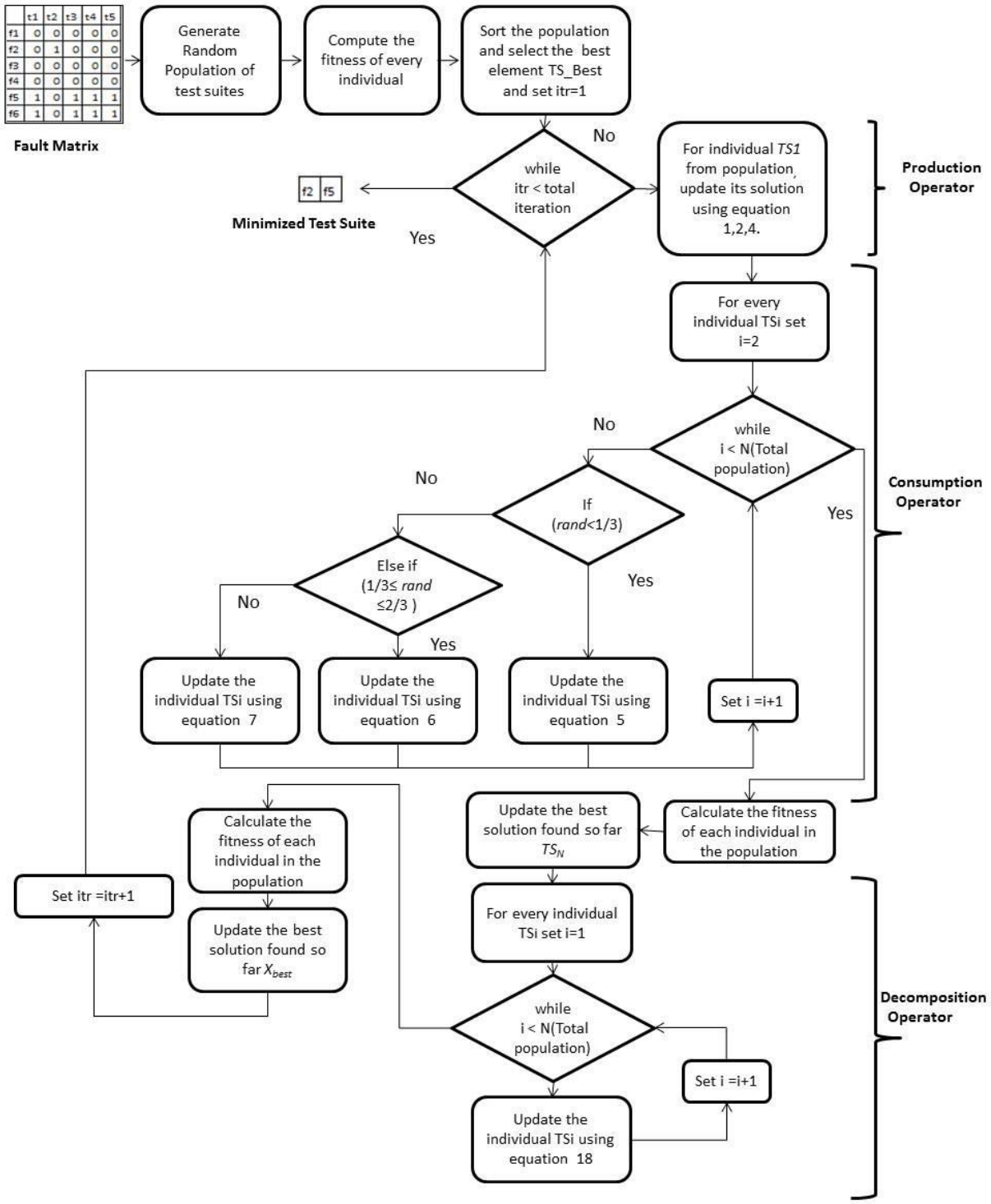

Figure 2. Proposed Approach of Regression Test Case Minimization using AEO

In AEO, Levy flight is considered as a mathematical operator to perform random walk-in order to explore the search space. In AEO, Levy flight is proposed as a consumption factor which is a simple free random walk. Each Consumer may use this consumption factor for hunting their food and there are various consumption approaches adopted by different varieties of consumers.

The mathematical representation of consumption operator in AEO is as follows:

a) Herbivore: Herbivore consumer eats only the producer.

$$TS_i(itr+1) = TS_i(itr) + C.\left(TS_i(itr) - TS_1(itr)\right), where\ itr\ \epsilon\ 2{,}3\ \ldots\ldots N \quad (5)$$

b) Carnivore: Carnivore consumer eats only randomly chosen consumer having a high energy level.

$$TS_i(itr+1) = TS_i(itr) + C.\left(TS_i(itr) - TS_j(itr)\right), where\ itr\ \epsilon\ 3\ \ldots\ldots N \quad (6)$$

where $j = randi(|2\ i-1|)$

c) Omnivore: Omnivore consumers can eat both either a producer or a consumer having a higher energy level.

$$TS_i(itr+1) = TS_i(itr) + C.\left(r2.\left(TS_i(itr) - TS_1(itr)\right)\right) + (1-r2)\left(TS_i(itr) - TS_j(itr)\right) \quad (7)$$

$where\ itr\ \epsilon\ 3\ \ldots\ldots N$ and $j = randi(|2\ i-1|)$

C. Decomposition

The decomposition is most important operator and very essential process to maintain the functioning of an ecosystem as well as generates required nutrients for producer to help them in their growth. In the process of decomposition, when each individual of the population dies, the decomposer will break down chemically or decay its remains.

Here D (decomposition factor), e and h (weight coefficient) are three main parameters to spread the decomposer to each individual.

The mathematical representation of consumption operator in AEO is as follows:

$$TS_i(itr+1) = TS_i(itr) + D.\left(e.\left(TS_n(itr) - h.TS_i(itr)\right)\right)\ \ where\ itr\ \epsilon\ 1{,}2\ \ldots\ldots N \quad (8)$$

$$D = 3u, \qquad u \sim N(0,1) \tag{9}$$

$$j = r3.randi(|1\ 2|) - 1 \tag{10}$$

$$h = 2.r3 - 1 \tag{11}$$

In AEO, decomposition balance exploitation behaviour by using equations 8,9,10,11. In MAEO, authors have utilized equations from Equilibrium optimizer (Faramarzi et al., 2020).

The mathematical representation of decomposition operator in MAEO is as follows:

Generate random vectors of $\vec{\lambda}, \vec{r}$ random number vector of size [1- Dim] ranges between $[0-1]$. $a_1 = 1$, GP=0.5 to better maintain exploitation and exploration. The

$$\overrightarrow{TS_{Rand\_P}} = randi(TS) \tag{12}$$

$$Construct\ \vec{F} = a_1\ sign\ (\vec{r} - 0.5)[\ e^{-t\vec{\lambda}} - 1] \tag{13}$$

$$Construct\ \overrightarrow{GCP} = \ 0.5r_1 \qquad r_2 \geq GP \tag{14}$$

$$Construct\ \vec{G}_0 = \ \overrightarrow{GCP}\ (\overrightarrow{TS_{Rand\_P}} - \ \vec{\lambda}\ \overrightarrow{TS_i}\ ) \tag{15}$$

$$Construct\ \vec{G} = \ \vec{G}_0.\vec{F} \tag{16}$$

$$TS_i(itr+1) = TS_{N_i}(itr) + \left(TS_i - TS_{Rand\_P}\right).F_i + \left(\frac{G_i}{\lambda_i V}\right)(1 - F_i) \tag{17}$$

In Eq. 17 first part of the equation reflects the decomposer represents the best solution so far in this iteration the second term represents the movement in the direction of random consumer which behave like direct search and third term is used for global exploration.

Experimental Setup

Experimental setup contributes significantly to evaluate the quality of any research, keeping this in consideration. We have conducted the research experiments to evaluate the performance of proposed algorithm. The algorithms and their control parameters considered for this study is as follows:

Table 1. Control parameter setting for Experimental Setup

Algorithm / Control Parameters				
MAEO	Population size (N)	No of Iterations	a1	
	30	500	1	
AEO	Population size (N)	No of Iterations		
	30	500		
BAT	No. of Bats	Frequency	Loudness	Pulse Rate
	30	75	0.75	0.25
CROW	Population size (N)	Awareness Probability	Flight Length	No of Iterations
	30	0.1	2	500
HHO	Population size (N)	No of Iterations		
	30	500		

The proposed algorithm as well as existing approach is evaluated on benchmarked dataset consisting of 12 versions of 5 open-source programs. These versions are retrieved from benchmarked Software Artifact Repository in regression testing (Do et al., 2005) under controlled experimental setting. Table 2 present summarized details of fault data.

Table 2. Dataset retrieved and their characteristics

Objects	flex v1 object	flex v2 object	flex v3 object	flex v4 object	flex v5 object	Grep object	Gzip object	nano-xml v1 object	nano-xml v2 object	nano-xml v3 object	nano-xml v5 object	xml-security object
Total number of faults	19	20	17	16	9	18	16	5	7	7	9	5
Total number of test cases	567	567	567	567	567	199	214	15	214	216	216	15

The research questions and research hypothesis section discuss the research questions and hypothesis formed to evaluate the performance of proposed approach over the existing literature.

Research Questions

We have formulated following three research questions to compare the performance of proposed and existing approaches available in literature.

- Research Question 1: How is the performance of the MAEO algorithm as compared to the rest of the algorithm adopted for regression test case minimization problem?
- Research Question 2: Is there any effect of the subject program on the performance of the algorithms?
- Research Question 3: Does minimized test suite size effect fault revealing ability of algorithms?
- Research Question 4: How the computational cost of the MAEO algorithm is as compared to the rest of the algorithm adopted for regression test case minimization problem?

Research Hypothesis

The following hypothesis is designed to answer our research questions RQ1, RQ2 and RQ3 respectively:

Ho: Overall Performance of MAEO = Overall Performance of AEO = Overall Performance of BA= Overall Performance of HHO= Overall Performance of CSA

Ha: Overall Performance of MAEO ≠ Overall Performance of AEO ≠ Overall Performance of BA≠ Overall Performance of HHO ≠ Overall Performance of CSA

Ho: Algorithmic performance is independent of Subject Program

Ha: Algorithmic performance is dependent of Subject Program

Ho: RTM_5 = RTM_10 = RTM_15

Ha: RTM_5 ≠ RTM_10 ≠ RTM_15

Ho: Computational Cost of MAEO = Computational Cost of AEO = Computational Cost of BA= Computational Cost of HHO= Computational Cost of CSA

Ha: Computational Cost of MAEO ≠ Computational Cost of AEO ≠ Computational Cost of BA≠ Computational Cost of HHO ≠ Computational Cost of CSA

Results & Discussion

The experiment is conducted on subject programs under control parameters setting as shown in Table 2 and 3. In this experiment 15 runs of each algorithm is evaluated and means of these runs is utilized for performance evaluation. The values highlighted in bold are best from one or the other viewpoint. Table 3 present the means of dependent variable fault covered.

Table 3. Mean value of Faults Covered by proposed and adopted algorithms on different test suite sizes and subject programs

Subject Program	Test suite size	% Test Cases Covered				
		MAEO	AEO	BA	HHO	CSA
flex v1	5	100	100	96	100	100
	10	100	100	100	100	100
	15	100	100	100	100	100
flex v2	5	98	95	75	89	87
	10	100	95	81	92	90
	15	100	98	82	93	92
flex v3	5	98	93	59	92	90
	10	100	100	74	99	98
	15	100	100	79	100	99
flex v4	5	100	100	100	100	100
	10	100	100	100	100	100
	15	100	100	100	100	100
flex v5	5	100	100	100	100	100
	10	100	100	100	100	100
	15	100	100	100	100	100
Grep	5	100	100	58	98	100
	10	100	100	71	100	100
	15	100	100	80	100	100
Gzip	5	100	100	52	88	98
	10	100	100	78	98	100
	15	100	100	78	99	100
nano v1	5	100	100	100	100	100
	10	100	100	100	100	100
	15	100	100	100	100	100
nano v2	5	100	100	100	100	100
	10	100	100	100	100	100
	15	100	100	100	100	100
nano v3	5	100	100	100	100	100
	10	100	100	100	100	100
	15	100	100	100	100	100
nano v5	5	100	99	75	88	99
	10	100	100	82	97	100
	15	100	100	89	99	100
xmlSec	5	100	100	99	100	100
	10	100	100	100	100	100
	15	100	100	100	100	100

- ***Answer to Research Question 1:***

The bold highlighted mean values reflect the fault coverage by different approaches on different subject programs and different minimized test suites. The highlighted means reflect that the proposed MAEO performs superior to other compared approaches. However, to confirm the claim of superiority a two- way ANOVA test is performed. The test results will show the effect of independent variables separately and in combination on dependent variable fault coverage. The tests are conducted in SPSS 20 tool and results are shown in Table 4 below.

Table 4. Two-way ANOVA of independent variables (Algorithm, Test suite size, subject program) and their combined effects on fault coverage at 95% Confidence Interval (α = 0.05)

Tests of Between-Subjects Effects					
Dependent Variable: Fault_Coverage					
Source	**Type III Sum of Squares**	**Df**	**Mean Square**	**F**	**Sig.**
Corrected Model	36997.683[a]	179	206.691	3247.265	0
Intercept	155283.917	1	155283.917	2439622.637	0
Algo	237.613	4	59.403	933.266	0
Subject	36259.461	11	3296.315	51787.486	0
TS_Size	23.490`	2	11.745	184.52	0
Algo * Subject	392.076	44	8.911	139.996	0
Algo * TS_Size	17.325	8	2.166	34.024	0
Subject * TS_Size	39.079	22	1.776	27.907	0
Algo * Subject * TS_Size	28.639	88	0.325	5.113	0
Error	160.4	2520	0.064		
Total	192442	2700			
Corrected Total	37158.083	2699			

All the significance values listed in column 6 of table 4 are lesser than α, so we can claim that all the approaches are statistically significant from each other. Further we have conducted the post –hoc test Tukey's HSD at 95% confidence interval to perform a pair wise comparative analysis of approaches for fault coverage and the results are shown in table 5 below:

Table 5. Pair wise Comparison of different algorithms with respect to Fault Coverage at α = 0.05

Dependent Variable: Fault_Cov						
Tukey HSD						
(I) Algo		Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
MAEO	AEO	.05*	0.015	0.01	0.01	0.09
	BA	.82*	0.015	0	0.77	0.86
	HHO	.17*	0.015	0	0.13	0.22
	CSA	.14*	0.015	0	0.1	0.18
AEO	MAEO	-.05*	0.015	0.01	-0.09	-0.01
	BA	.77*	0.015	0	0.72	0.81
	HHO	.12*	0.015	0	0.08	0.17
	CSA	.09*	0.015	0	0.05	0.13
BA	MAEO	-.82*	0.015	0	-0.86	-0.77
	AEO	-.77*	0.015	0	-0.81	-0.72
	HHO	-.64*	0.015	0	-0.68	-0.6
	CSA	-.67*	0.015	0	-0.72	-0.63
HHO	MAEO	-.17*	0.015	0	-0.22	-0.13
	AEO	-.12*	0.015	0	-0.17	-0.08
	BA	.64*	0.015	0	0.6	0.68
	CSA	-0.03	0.015	0.242	-0.07	0.01
CSA	MAEO	-.14*	0.015	0	-0.18	-0.1
	AEO	-.09*	0.015	0	-0.13	-0.05
	BA	.67*	0.015	0	0.63	0.72
	HHO	0.03	0.015	0.242	-0.01	0.07

The results observed from table 5 above shows that the pair wise algorithms is significantly different among others in term of their fault coverage as all the P-values are lesser than α. The result evident from homogenous subset of fault coverage of different algorithms in table 6 answers the Research question1 that proposed MAEO works better than other adopted Algorithms. The MAEO is superior then BA, HHO, CSA and AEO algorithm.

Table 6 Homogenous Subsets of Mean Fault Coverage of Algorithms

Fault_Coverage					
Tukey HSD					
Algo	N	Subset			
		1	2	3	4
BA	540	7			
HHO	540		7.65		
CSA	540		7.68		
AEO	540			7.77	
MAEO	540				7.82
Sig.		1	0.242	1	1

- ***Answer to Research Question 2***

Table 3 and Figure 3 also help us to answer our Research Question 2. The figure 3 shows that except flex v4, nano v3 and xmlSec, there is a significant difference in the performance of algorithm. Table 3 also strengthen the claim of research question 2. So, we can conclude that a test suite size of 10 test cases is fair enough to perform regression testing without significantly affecting the fault detection effectiveness of RTM approaches.

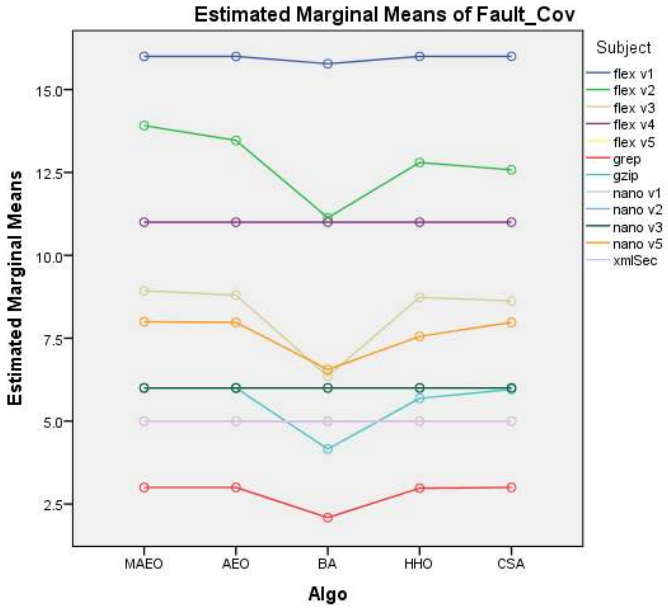

Figure 3. Algorithm Performance vs Subject program

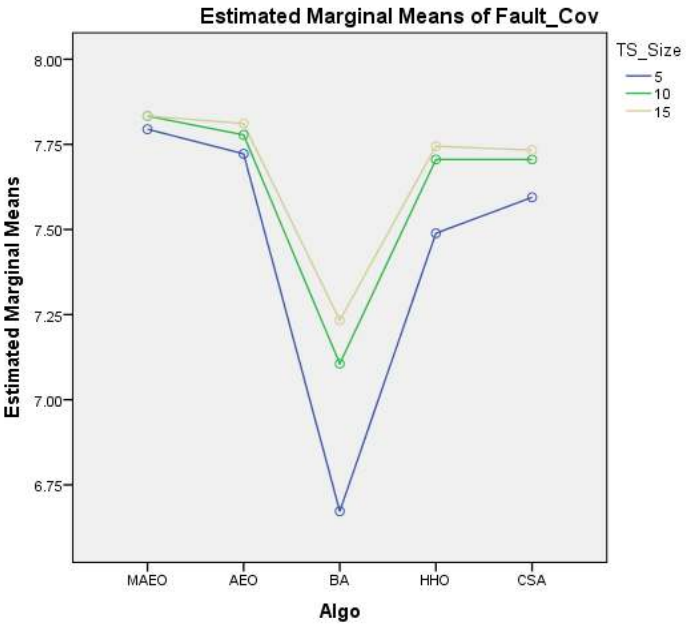

Figure 4. Algorithm Performance vs Test Suite size

- ***Answer to Research Question 3***

Table 3, 7 and Figure 4 answer the 3rd research question. It can be clearly seen from figure 4 that on 5 test suite size the algorithm performance is less as compare to 10 and 15 test suite size. It can be clearly seen from table 3and 7 that the algorithm performance varies with minimized size of test suite.

Table 7. Homogenous Subsets of Mean Fault Coverage of Test Suite Size

Fault_Coverage				
Tukey HSD				
TS_Size	N	Subset		
		1	2	3
5	900	7.45		
10	900		7.63	
15	900			7.67
Sig.		1.000	1.000	1.000

- ***Answer to Research Question 4***

Table 8 and 9 answers the 4th research question. Table 7 shows that the execution time of MAEO is lesser than AEO and HHO but it is taking much time in comparison to BA and CSA approach. Table 8 also confirm the claim and reject the null hypothesis.

So, it is clear from the table 6, 8 and 9 MAEO is providing better fault coverage as compare to other algorithms and computationally costlier as compare to CSA and BA but better than AEO and HHO.

Table 8. Execution time of adopting algorithm on Regression Test Case Minimization

Report					
Exe_Time					
Algo	Mean	N	Std. Deviation	Variance	Skewness
MAEO	3.78514065	540	0.362517888	0.131	2.826
AEO	4.17900657	540	0.549269359	0.302	1.799
BA	1.3497452	540	0.236788935	0.056	4.605
HHO	4.08997542	540	1.129432657	1.276	2.447
CSA	0.70746535	540	0.185845226	0.035	3.558
Total	2.82226664	2700	1.601025632	2.563	0.135

Table 9. Tukey HSD test on adopting algorithm for Regression Test Case Minimization

Exe_Time					
Tukey HSD					
Algo	N	Subset for alpha = 0.05			
		1	2	3	4
CSA	540	0.707465349			
BA	540		1.349745202		
MAEO	540			3.78514065	
HHO	540				4.08997542
AEO	540				4.179006573
Sig.		1	1	1	0.105

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 540.000.

Threats to Validity

Threats to external validity limit the capability to generalize the results beyond the experiment settings which can be reduced by taking the more realistic experimental setup. To overcome such issues, we have adopted benchmarked subject programs from SIR repository to perform the experiment.

Threat to internal validity mainly affects the independent variables such as algorithms, subject programs and minimized size of test suite in our case. To overcome this threat we adopted algorithms available in literature, similarly subject programs and minimized test suite size. Threat to construct validity reflects the degree up to that the experiment setting imitates the theory. In our case stochastic nature of Meta heuristic algorithms has been addressed by running our experiment 15 times with 500 iterations each to collect the results. Means of these 15 runs has been considered.

Conclusion

Regression test suite minimization is very challenging problem under software maintenance. In each maintenance cycle running all the test cases is costly affair. To these issues this paper proposes an effective Modified Artificial Ecosystem optimization-based approach for test suite minimization. The proposed MAEO with updated formulation maintains optimal balance in exploration and exploitation for test case minimization problem. All the three research questions are answered through empirical study:

Answer to Research Question 1: The proposed Modified AEO based RTM approach is significantly superior to other adopted approaches.

Answer to Research Question 2: Yes, there is significant impact of subject program on performance of algorithms.

Answer to Research Question 3: Yes, there is impact of minimized test suite size on fault revealing ability of algorithms. The Test suite size of 10 test cases is sufficient in mostly cases.

Answer to Research Question 4: Yes, the computational cost of different metaheuristics is not same. Computational cost of MAEO is higher as compare to BA and CSA but lesser than AEO and HHO.

In future we will focus to generalize this approach to various domains of applications and generalize the approach to select the minimum required test case in suite automatically.

References

Agrawal, A. P., Choudhary, A., Kaur, A., & Pandey, H. M. (2019). Fault coverage-based test suite optimization method for regression testing: learning from mistakes-based approach. *Neural Computing and Applications*, *February*. https://doi.org/10.1007/s00521-019-04098-9

Ahmad Khan, F., Bora, D. J., & Gupta, A. K. (2017). An Efficient Heuristic Based Test Suite Minimization Approach. *Indian Journal of Science and Technology*, *10*(29), 1–8. https://doi.org/10.17485/ijst/2017/v10i29/106374

Ahmed, B. S. (2015). Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. *Engineering Science and Technology, an International Journal*, *6*(2), 189–213. https://doi.org/10.1007/s10479-005-3971-7

Augustsson, A. (2012). A Framework for Evaluating Regression Test Selection Techniques in Industry. *Proceedings of 16th International Conference on Software Engineering*, *April*, 201–210.

Black, J., Melachrinoudis, E., & Kaeli, D. (2004). Bi-criteria models for all-uses test suite reduction. *Proceedings. 26th International Conference on Software Engineering*, 106–115. https://doi.org/10.1109/ICSE.2004.1317433

Chaudhary, A., Agarwal, A. P., Rana, A., & Kumar, V. (2019). Crow Search Optimization Based Approach for Parameter Estimation of SRGMs. *Proceedings - 2019 Amity International Conference on Artificial Intelligence, AICAI 2019*, 583–587. https://doi.org/10.1109/AICAI.2019.8701318

Chen, T. Y., & Lau, M. F. (1998a). A new heuristic for test suite reduction. *Information and Software Technology*, *40*(5–6), 347–354. https://doi.org/10.1016/S0950-5849(98)00050-0

Chen, T. Y., & Lau, M. F. (1998b). A simulation study on some heuristics for test suite reduction. *Information and Software Technology*, *40*(13), 777–787. https://doi.org/10.1016/S0950-5849(98)00094-9

Chen, T. Y., & Lau, M. F. (2003). On the divide-and-conquer approach towards test suite reduction. *Information Sciences*, *152*(SUPPL), 89–119. https://doi.org/10.1016/S0020-0255(03)00060-4

Chung, C. G., & Lee, J. G. (1997). An enhanced zero-one optimal path set selection method. *Journal of Systems and Software*, *39*(2), 145–164. https://doi.org/10.1016/S0164-1212(96)00169-0

Do, H., Elbaum, S., & Rothermel, G. (2005). Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering*, *10*(4), 405–435. https://doi.org/10.1007/s10664-005-3861-2

Drake, J. H., Turner, A. J., White, D. R., & Drake, J. H. (2016). *Multi-objective Regression Test Suite Minimisation for Mockito*. 1–6. https://doi.org/10.1007/978-3-319-47106-8

Faramarzi, A., Heidarinejad, M., Stephens, B., & Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. *Knowledge-Based Systems*, *191*. https://doi.org/10.1016/j.knosys.2019.105190

Gupta, A., Mishra, N., & Kushwaha, D. S. (2014). Rule based test case reduction technique using decision table. *Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014*, 1398–1405. https://doi.org/10.1109/IAdCC.2014.6779531

Gupta, N., Sharma, A., & Pachariya, M. K. (2020). Multi-objective test suite optimization for detection and localization of software faults. *Journal of King Saud University - Computer and Information Sciences*, *xxxx*. https://doi.org/10.1016/j.jksuci.2020.01.009

Haider, A A, Rafiq, S., & Nadeem, A. (2012). Test suite optimization using fuzzy logic. *Proceedings - 2012 International Conference on Emerging Technologies, ICET 2012*, *September*, 340–345. https://doi.org/10.1109/ICET.2012.6375440

Haider, Aftab Ali, Nadeem, A., & Rafiq, S. (2013). Computational intelligence and safe reduction of test suite. *ICET 2013 - 2013 IEEE 9th International Conference on Emerging Technologies*, 1–6. https://doi.org/10.1109/ICET.2013.6743502

Harris, P., & Raju, N. (2015). A greedy approach for coverage-based test suite reduction. *International Arab Journal of Information Technology*, *12*(1), 17–23.

Harrold, M., Gupta, R., & Jean, M. (1993). *A Methodology for Controlling the Size of a Test Suite*. *3*, 270–285. https://doi.org/10.1145/152388.152391

Jeffrey, D., & Gupta, N. (2007). by Selectively Retaining Test Cases during Test Suite Reduction. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, *33*(2), 108–123.

Jeffrey, D., & Gupta, N. (2005). Test Suite Reduction with Selective Redundancy. *IEEE International Conference on Software Maintenance*, 549–558. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.1071&rep=rep1&type=pdf

Kaur, Amandeep. (2020). *An Approach To Extract Optimal Test Cases Using AI*. 649–654. https://doi.org/10.1109/confluence47617.2020.9058244

Kaur, Arvinder. (2017). A Comparative Study of Bat and Cuckoo Search Algorithm for Regression Test Case Selection. *2017 7th International Conference on Cloud Computing, Data Science& Engineering – Confluenc*.

Kumar, N., Mishra, B., & Bali, V. (2018). A Novel Approach for Blast-Induced Fly Rock Prediction Based on Particle Swarm Optimization and Artificial Neural Network. In *Lecture Notes in Networks and Systems* (Vol. 34). Springer Singapore. https://doi.org/10.1007/978-981-10-8198-9_3

Lee, J. G., & Chung, C. G. (2000). An optimal representative set selection method. *Information and Software Technology*, *42*(1), 17–25. https://doi.org/10.1016/S0950-5849(99)00052-X

Littlewood, B., & Sofer, A. (1987). A Bayesian modification to the Jelinski-Moranda software reliability growth model. *Software Engineering Journal*, *2*(2), 30–41. https://doi.org/10.1049/sej:19870005

Loiola, C., & Maia, B. (2009). a Multi-Objective Approach for the Regression Test Case Selection Problem. *XLI Brazilian Symposium of Operational Research, XLI SBPO 2009.*, 1824–1835.

Sprenkle, S., Sampath, S., Gibson, E., Pollock, L., & Souter, A. (2005). An empirical comparison of test suite reduction techniques for user-session-based testing of web applications. *IEEE International Conference on Software Maintenance, ICSM*, *2005*, 587–600. https://doi.org/10.1109/ICSM.2005.18

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*, *2*, 2231–5292. https://www.researchgate.net/publication/228460782

Weiguo, Wang, L., & Zhang, Z. (2019). Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. In *Neural Computing and Applications* (Vol. 0123456789). Springer London. https://doi.org/10.1007/s00521-019-04452-x

Yang, X. (n.d.). *Nature-Inspired Optimization Algorithms*.

Yoo, S., & Harman, M. (2007). Pareto efficient multi-objective test case selection. *Proceedings of the 2007 International Symposium on Software Testing and Analysis - ISSTA '07*, 140. https://doi.org/10.1145/1273463.1273483

Yoo, S., & Harman, M. (2010). Using Hybrid Algorithm For Pareto Effcient Multi-Objective Test Suite Minimisation. *Journal of Systems and Software*, *83*(4), 689–701. https://doi.org/http://dx.doi.org/10.1016/j.jss.2009.11.706

You, L., & Lu, Y. (2012). A genetic algorithm for the time-aware regression testing reduction problem. *Proceedings - International Conference on Natural Computation*, *Icnc*, 596–599. https://doi.org/10.1109/ICNC.2012.6234754

Zhang, L., Marinov, D., Zhang, L., & Khurshid, S. (2011). An empirical study of JUnit test-suite reduction. *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, *4*, 170–179. https://doi.org/10.1109/ISSRE.2011.26

Bibliographic information of this paper for citing:

Verma, Abhishek Singh; Choudhary, Ankur & Tiwari, Shailesh (2021). Regression Test Suite Minimization Using Modified Artificial Ecosystem Optimization Algorithm. *Journal of Information Technology Management*, 13(1), 22-41.