International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Cuckoo Search Algorithm for Test Case Prioritization in Regression Testing

Priyanka Dhareula, Anita Ganpati

ABSTRACT: ***There are countless optimization problems that have been accelerated by Nature Inspired Metaheuristic Optimization Algorithms (NIMOA) in the earlier decades. NIMOA have gained huge popularity owing to their effective results. In this study NIMOA namely, Cuckoo Search Algorithm (CSA) is used to prioritize (order) the test cases for Regression Testing (RT). Prioritizations aids in the execution of higher priority test cases to give early fault detection. This research adopts the aggressive approach of reproduction made by Cuckoos to prioritize the test cases for RT. Average Percentage of Fault Detected (APFD) metrics is used in this paper for validations of results. APFD metrics is used to compare the performance of CSA with Flower Pollination Algorithm (FPA) and traditional approaches for Test Case Prioritization (TCP). Two java applications are used for the study. CSA is implemented in Java on eclipse platform. It is learnt from the study that APFD results of CSA outperformed the FPA for both the applications namely Puzzle Game and AreaandPerimeter. It is inferred from the results that prioritized set of test cases given by NIMOA outperformed the APFD results of traditional approaches and also CSA performed better than the FPA for TCP.***

Key Terms: Cuckoo Search; Flower Pollination; Regression Testing; Test Case Prioritization; APFD

I. INTRODUCTION

The lifespan of a software development considers Regression Testing (RT) as one of the most crucial phases. With the change in organizational requirements it becomes necessary to incorporate the changes in the software under consideration. Changes may result in unintended behavior of the software, which is further identified in RT. RT is performed to determine whether previous tested code has not altered its functionality due to new changes made in the software. RT is a time extensive and costly process. Since the test suite employed to perform RT is very large in size, it is rationally not possible to perform such exhaustive testing due to budget and time constraints as the software has to be delivered within given time limits. Therefore, it becomes imperative to constitute a mechanism to reduce the size of test suite to perform effective testing that can find maximum faults in minimum possible time.

Manuscript published on 30 September 2019
* Correspondence Author
Priyanka Dhareula*, Computer Science Department, H. P. University, Shimla – 171005, H. P., India
Anita Ganpati, Computer Science Department, H. P. University, Shimla – 171005, H. P., India

Many traditional Regression Test Case Optimization (RTCO) techniques are known to exist in the past literature. Broadly they are categorized as Retest all, Minimization [1], Selection [12, 9] and Prioritization [12]. Alone these techniques have not resulted in effective optimization of the test suite and henceforth, there is a need to augment them with improvised version of optimization techniques known as the Nature Inspired Metaheuristic Optimization Algorithms (NIMOA).

Numerous NIMOA are known to have produced optimized results in the field of RTCO. Many NIMOA have existed for past few decades namely; Genetic Algorithm (GA), Differential Evolution, Ant Colony Optimization (ACO), Bee Colony Algorithms, Particle Swarm Optimization, The Firefly Algorithm, Cuckoo Search Algorithm (CSA) [16], The Bat Algorithm, Harmony Search, The Flower Pollination Algorithms (FPA); are few to name. Therefore, when the traditional RTCO approaches [5, 6] are combined with NIMOA [11] it will result in much more effective results for RT. In this paper Cuckoo Search Algorithm (CSA) [16] is used to prioritize the test cases to perform RTCO. CSA was developed by Xin-She Yang and Suash Deb in 2009 [19]. Cuckoos have mesmerizingly beautiful sound and a very aggressive reproduction approach. This research adopts the aggressive approach of reproduction made by Cuckoos to prioritize the test cases for RT. Therefore, CSA is used in this study to perform TCP without any prior knowledge of faults covered by the test cases. Average Percentage of Fault Detected (APFD) metrics and time of execution is used to determine the effectiveness of CSA for TCP. For the validations of the results APFD values of CSA is compared with the APFD values of FPA [4] and traditional approaches for TCP in RT.

II. RELATED WORK

Yang [19] discussed about various applications of CSA in the dominion of engineering and science. Their study stated that tuning of parameters for metaheuristic algorithms demands more research. It is identified in their work that optimal results can be achieved by using CSA in various studies. Srivastava et al. [14] gave an algorithm to produce reduced test suite using CSA. Activity diagram is used to understand the connection between modules, thereafter the input is given to a graph whose output serves as the input to proposed strategy. Warning System for temperature is used as the case study. Proposed technique is compared with ACO and GA. As per their study CSA produced optimal results in lesser number of iterations. Ahmed et al. [2] in their study used CSA to perform combinatorial test case optimization. Real world case study is used to analyze the effectiveness of the optimized set of test cases.

Their strategy proved effective in covering faults in the given program. Bacanin [3] implemented CSA in Java and named it CSApp. NetBeans IDE is used for the implementation of CSA. Four benchmark functions namely; Ackley, Sphere, Griewank, Rastrigin were used. CSApp is used to optimize these four functions. Their application resulted in superior results. Nagar et al.

[12] used CSA for selection and prioritization of test cases on the basis of maximum fault coverage in lesser time. Java is used to implement their work. MATLAB is used to represent the results graphically. Ahmed [1] performed minimization of test cases for Configuration- Aware structural testing using CSA. Initially combinatorial optimization is performed to produce reduced test suite. Mutation testing is further used to filter the test cases. They have compared their work with well-known strategies. In their study CSA delivered optimal results most of the time. Srivastava et al. [14] used CSA and Tabu Search Algorithm (CSTS) for test case data generation. CSA is used to produce new test cases. Tabu Search Algorithm maintains bad and good test cases and also keeps account of redundant test cases. Triangle classification problem is used as the case study. Proposed approach is compared with GA and Tabu Search Algorithm, GA and ACO. Their approach performed much better than other techniques.

Kumar et al. [9] optimized the test cases by using PSO. Further they prioritized the optimized test suite by using Improved Cuckoo Search Algorithm (ICSA). APFD and Problem Tracking Reports (PTR) metrics are used to determine the effectiveness of proposed method. Their empirical results showed improvement over other techniques. Dhareula and Ganpati [4] performed TCP using FPA for RT. They used the pollination mechanism of flower for prioritizing the test cases. Three java applications were used. The algorithm was implemented in java using eclipse. APFD metrics and time of execution is used to evaluate the results. APFD results of FPA for three applications are compared with random and random reverse prioritization of test cases. The results show that FPA outperformed for TCP for all the applications.

III. REGRESSION TEST CASE OPTIMIZATION TECHNIQUES

RT is one of the most important activity of maintenance phase. It is not possible to perform exhaustive RT due to very large size of the test suite. As the size of test suite increases with the evolution in software, RT demands effective ways to reduce the number of test case due to budget and time constraints. There are few main techniques known in literature for RTCO namely; Test Case Prioritization (TCP), Selection, Minimization and Retest all technique. Let 'P' be a program under consideration and 'P'' be modified version of 'P'. Let test suite 'T' be for 'P'. For RT 'T' will be reused for 'P'' to determine the correctness of 'P'' [13]. RT needs to be performed in minimum budget and time constraints. Therefore, it is not practically possible to execute all the test cases in T for 'P''. Further optimization techniques are used to reduce 'T' to 'T'', so that 'T'' can efficiently cover all the possible errors in given time and cost constraints. For this study TCP is used. TCP technique sets an order (priority) for the test cases in a given test suite. The order is set on the basis of some criteria for the test cases. Prioritizing the test cases will reduce the time and cost of performing RT as only higher priority test cases will be executed that will fulfill the given criterion. Only those test cases will be executed which gives early rate of fault detection.

IV. CUCKOO SEARCH ALGORITHM

CSA is one of the latest NIMOA developed by Xin- She Yang and Suash Deb in 2009 [10, 19]. Cuckoo are birds with scintillating sounds they make and they are also known for their aggressive strategy for reproduction [17]. Cuckoo species such as *Guira* and *Ani* Cuckoos place their eggs in common nests, they may even get rid of eggs of other birds to increase their own eggs hatching probability. Maximum Cuckoo species involve in obligate brood parasitism by using other host birds' nest for laying their eggs. Brood parasitism can be classified into three types viz., cooperative breading, intraspecific brood parasitism and nest takeover. CSA relies on brood parasitism. Parasitic Cuckoos normally choose a host nest where they lay their own eggs. Host eggs hatch later as compared to Cuckoo eggs. As the cuckoo egg hatches, it blindly pushes the host eggs out of the nest, which increases the probability of food share for Cuckoo chicks provided by its hosts bird [17, 18]. This natural process of survival of Cuckoo birds can be best explained as the survival of the fittest and optimum reproduction of birds in number as well as fitness. Henceforth, the behaviour of Cuckoos inspires this research to use CSA for TCP in RT. Figure 1 represents the CSA used for TCP [19].

Objective function f(x), $x= (x_1, \ldots, x^d)^T$

Generate initial population of n host nests x_i

While (t< MaxGeneration) or (stop criterion)

Get a cuckoo randomly

Generate a solution by Levy flight [e.g., Eq. (1)]

Evaluate its solution quality or objective value f_i

Choose a nest among n (say, j) randomly

If ($f_i < f_j$)

Replace j by the new solution i

End

A fraction (p_a) of worse nests are abandoned

New nests/ solutions are built

Keep best solution (or nest with quality solution) [used Eq. (10)]

Rank the solutions and find the current best

Update t ← t+1

End while

Postprocess result and visualization

Figure 1: Cuckoo Search Algorithm Used for TCP

Objective function (f_n) for this study is the code coverage detail of each test case. Where each test case represents the egg or the cuckoo. Maximum or the initial population of host nest is kept to be 3. Host nest here represents the test suites. Initially all the three nests namely, N1, N2, N3 will contain one-one egg each picked up randomly. Stopping criteria for the algorithm is when all the test cases from the original test suite (TS_o) have been traversed. Levy flight is used to perform global random

walk to pick the new solution x_i^{t+1}, (egg/ cuckoo) using eq. (1) [19].

$x_i^{t+1} = x_i^t + \alpha + \text{Levy}(\lambda)$ …Eq. (1)

Levy ~ $v = t^{-\lambda}$, $(1<\lambda<=3)$, $\alpha=1$ …Eq. (2)

$x_i^{t+1} = x_i^t + \alpha L(s, \lambda)$ …Eq. (3)

where $L(s, \lambda) = \frac{\lambda\Gamma(\lambda)\sin\left(\frac{\pi\lambda}{2}\right)}{\pi} \frac{1}{s^{1+\lambda}}$, $(s>>s_0>0)$ …Eq. (4)

Here λ and α are constants. α is the step size scaling factor which is greater than zero. (λ) in Eq. (2) signifies the standard gamma function, here this distribution is effective for big steps s>0. Theoretically, it is essential that $s_0>>0$ (in Eq. 4), but practically it can be as small as 0.1. However, it is necessary to generate pseudo- random step sizes that appropriately obey the levy distribution. Mantegna method is used to draw the step size 's' by using two Gaussian distribution 'U' and 'V' (in Eq. 5) by the following transformation [19].

$S = \frac{U}{|V|^{1/\lambda}}$, U~N $(0, \sigma^2)$, V~N (0,1) …Eq. (5)

Here U~ $(0, \sigma^2)$ signifies sample that are drawn from gaussian normal distribution having zero mean and a variance of σ^2. The variance can be calculated by Eq.6.

$\sigma^2 = \left[\frac{\Gamma(1+\lambda)}{\lambda\Gamma(1+\lambda)/2} \frac{\sin(\frac{\pi\lambda}{2})}{2^{(\lambda-1)/2}}\right]^{1/\lambda}$ …Eq. (6)

the formula looks complicated, but it is just a constant for a given λ. For example, when λ=1, the gamma function becomes

$(1+\lambda) = 1$, $((1+\lambda)/2) = 1$ …Eq. (7), and

$\sigma^{2=} \left[\frac{1}{1*1} \frac{\sin\left(\pi*\frac{1}{2}\right)}{2^0}\right]^{1/1} = 1$ …Eq. (8),

therefore, $x_i^{t+1} = x_i + 1(1)$ …Eq. (9)

$x_i^{t+1} = x_i + 1$ in Eq. (9) is the new solution acquired by global search. For the quality or fitness consider the code coverage of the test case, x_i^{t+1}. The test cases are placed in randomly selected nests or test suite. Randomly a nest among three nests will be chosen. When a test case arrives, it is placed randomly in any one of the nests. If the already present test case has coverage more than or same as the arriving test case, then the new test case is discarded. Once all the test case in TS_o have been traversed, all the nests are analysed for their fault coverage. Nest with the maximum value for APFD as shown in Eq. (10) are kept and rest of the nest are discarded. Keeping the best solution or nest with quality solutions. The final nest will be termed as TS_1, containing prioritized set of test cases. Let TS be the test suite having n test cases and let F be the set of m faults identified by TS. Let Fi be first test case in original test suite TS_o which identifies fault i. Therefore, APFD for TS_i is formulated as shown in Eq. (10).

$$APFD = 1 - \frac{F_1+F_2+F_3+\ldots.F_m}{nm} + \frac{1}{2n} \quad \text{…Eq. (10)}$$

The test cases now in TS_1 are executed on priority basis. Starting from priority one, only those test case from TS_1 are executed which give complete fault coverage. Those higher priority test cases which give complete fault coverage are put in final prioritized test suite termed as TS_p. TS_p gives maximum fault coverage in reduced time of execution by removing those test cases that give redundant fault coverage. Effectiveness of the final prioritized test suite TS_p is determined with the help of APFD metrics in Eq. (1).

V. IMPLEMENTATION

Implementation of CSA for TCP in RT is done in Java using Eclipse IDE. Two applications written in Java, as were used in [Dhareula and Ganpati 2019] are used for empirical evaluation in this study also. Namely, Puzzle Game and AreaandPerimeter. For both the applications test scripts are designed using TestNG tool [7]. To get the code coverage of all the test cases EclEmma tool is used [8]. For the validation of results five versions of Puzzle Game and eight versions of AreaandPerimeter applications were used [4]. Each version was seeded with a unique fault.

VI. RESULTS

Puzzle Game application is used for the explanation in this study. The application consists of 33 test case and the original test suite is named as TS_o= {T1, T2, …, T33}, which consists of test cases in no particular order. Five unique faults have been induced in each version of the application viz., {F1, …, F5). TS_o is subjected to CSA and the output is a prioritized set of three test suites namely; TS_1/ N1= {T1, T2, T6}, TS_2/ N2= {T2, T3, T19}, TS_3/N3= {T3, T4, T5}. All the three test suites are separately exercised upon the application under consideration to determine the rate of fault coverage. The results of fault coverage are shown in Table 1.

Table 1: Fault Coverage Results of all the Nests in Puzzle Game

Faults ↓ / Test Cases →	N1 (TS_1)			N2 (TS_2)			N3 (TS_3)		
	T1	T2	T6	T2	T3	T19	T3	T4	T5
F1 (Version1)	(1)					(1)		(1)	
F2 (Version2)		(1)		(1)					(1)
F3 (Version3)			(1)		(1)		(1)		
F4 (Version4)		(1)	1	(1)	1		(1)		
F5 (Versio5)	(1)	1	1	(1)	1	1	(1)	1	1

Circled cells in Table 1 are the test cases that identifies a given fault first in sequence for a given version. APFD result of all the nests is shown in the Table 2 below.

Table 2: APFD Values for all the Nests in Puzzle Game

	Nest 1 (TS_1)	Nest2 (TS_2)	Nest 3 (TS_3)
APFD Results	$APFD = 1 - \frac{1+2+6+2+1}{5*33} + \frac{1}{2*33}$ APFD= 1-0.07272+0.01515 APFD= 1- 0.08787 **APFD= 0.91213**	$APFD = 1 - \frac{19+2+3+2+2}{5*33} + \frac{1}{2*33}$ APFD= 1- 0.16969+ 0.01515 APFD= 1- 0.18484 **APFD= 0.81516**	$APFD = 1 - \frac{4+5+3+3+3}{5*33} + \frac{1}{2*33}$ APFD= 1-0.10909+ 0.15151 APFD= 1- 0.2606 **APFD= 0.7394**
Time of Execution	0.352 seconds	0.347 seconds	0.272 seconds

As it is evident from Table 2 that all the nests are efficient in determining all the faults in given versions of Puzzle Game. It is also evident from Table 2 that Nest N1 (TS_1) is having maximum value for APFD metrics hence making it possibly the best nest with most efficient eggs/ test cases. Therefore, it can be stated that N1 is selected as the final prioritized test suite TS_p for Puzzle Game. TS_p for Puzzle Game gives total fault coverage as was deliberated by TS_o. Therefore, final order of test cases in TS_p= {T1, T2, T6}. The screen shot of priority of test cases and time of execution of TS_p for Puzzle game and random order of execution of TS_o as implemented in eclipse is shown in Figure 2 and Figure 3 respectively.

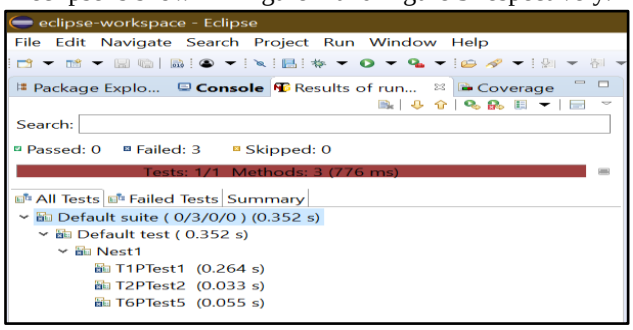

Figure 2: TS_p ordering and Time of Execution for Puzzle Game

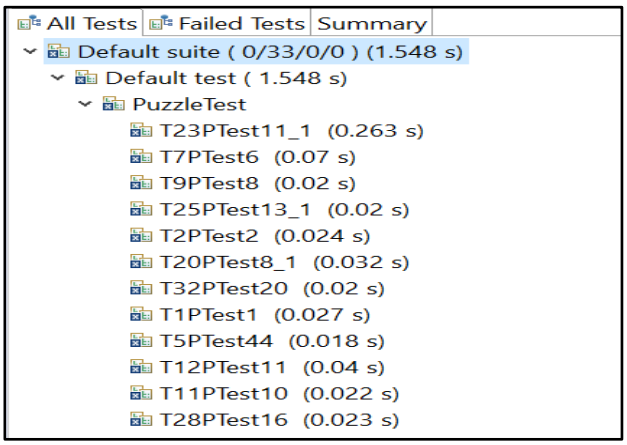

Figure 3: TS_O Random ordering and Time of Execution for Puzzle Game

Further a comparative study using APFD results of CSA with FPA and random order of execution of TS_o for the two application is performed. The results of comparative analysis are summarised in Table 3. For AreaandPerimeter nests (N2 and N3) failed to identify all the faults in given application. Nest N1 identifies all the faults in given application and is identified as TS_p as shown in Table 3

Table 3: Comparative Analysis for APFD Results of CSA, FPA and Random Ordering of Test Case

Sr. No.	Algorithm		Puzzle Game Application			AreaandPerimeter Application		
			Ordering	APFD Results	Time (in sec)	Ordering	APFD Results	Time (in sec)
1.	Nature Inspired Metaheuristic Techniques	CSA TS_P Ordering	Nest1= {T1, T2, T6}	0.91213	0.352 s	Nest1= {T1, T8, T10, T19, T27, T37, T39, T42, T50, T58, T61, T65, T70, T71, T89, T99, T101}	0.58851	0.08s
2.		CSA Reverse TS_P Ordering	Nest1 (Reverse)= {T6, T2, T1}	0.85758	0.281 s	Nest1 (Reverse)= {T101, T99, T89, TT71, T70, T65, T61, T58, T50, T42, T39, T37, T27, T19, T10, T8, T1}	0.58851	0.08s
3.		FPA TS_P Ordering	{T3, T5, T8}	0.85152	0.286 s	{T1, T27, T40, T59, T81, T99, T108}	0.44698	0.088s
4.		FPA Reverse TS_P Ordering	{T8, T5, T3}	0.8099	0.255	{T108, T99, T81, T59, T40, T27, T1}	0.43696	0.078s
5.	Traditional Approach	Random Ordering of TS_o	{T28, T26, T27, T15, T3, T2, T20, T13, T12, T30, T11, T7, T31, T8, T5, T29, T19, T14, T22, T17, T33, T1, T23, T4, T32, T16, T18, T10, T24, T21, T9, T25, T6}	0.38485	0.94	{T24, T79, T9, T85, T6, T90, T13, T101, T83, T77, T105, …, T8, T21, T37, T58, T14, T44, T23, T89}	0.42368	0.792s

6.		Reverse Random Ordering of TS_o	{T6, T25, T9, T21, T24, T10, T18, T16, T32, T4, T23, T1, T33, T17, T22, T14, T19, T29, T5, T8, T31, T7, T11, T30, T12, T13, T20, T2, T3, T15, T27, T26, T28}	0.27576	0.94s	{T89, T23, T44, T14, T58, T37, T21, T8, …, T105, T77, T83, T101, T13, T90, T6, T85, T9, T79, T24}	0.43363	0.792s

APFD results for CSA order (TS_p), CSA reverse order (TS_p), FPA order (TS_p), FPA reverse order (TS_p), Random Order of (TSo) and Reverse Random Order of (TS_o) along with the time of execution of all the ordering is displayed in Table 3 and graphically represented in Figure 4. The APFD results in Table 3 signifies that NIMOA are efficient in optimizing the test suite in RT. Also, CSA outperformed the FPA for both the applications namely Puzzle Game and AreaandPerimeter. It can be stated that prioritized set of test cases given by NIMOA gave better APFD results in lesser time as compared to traditional approaches.

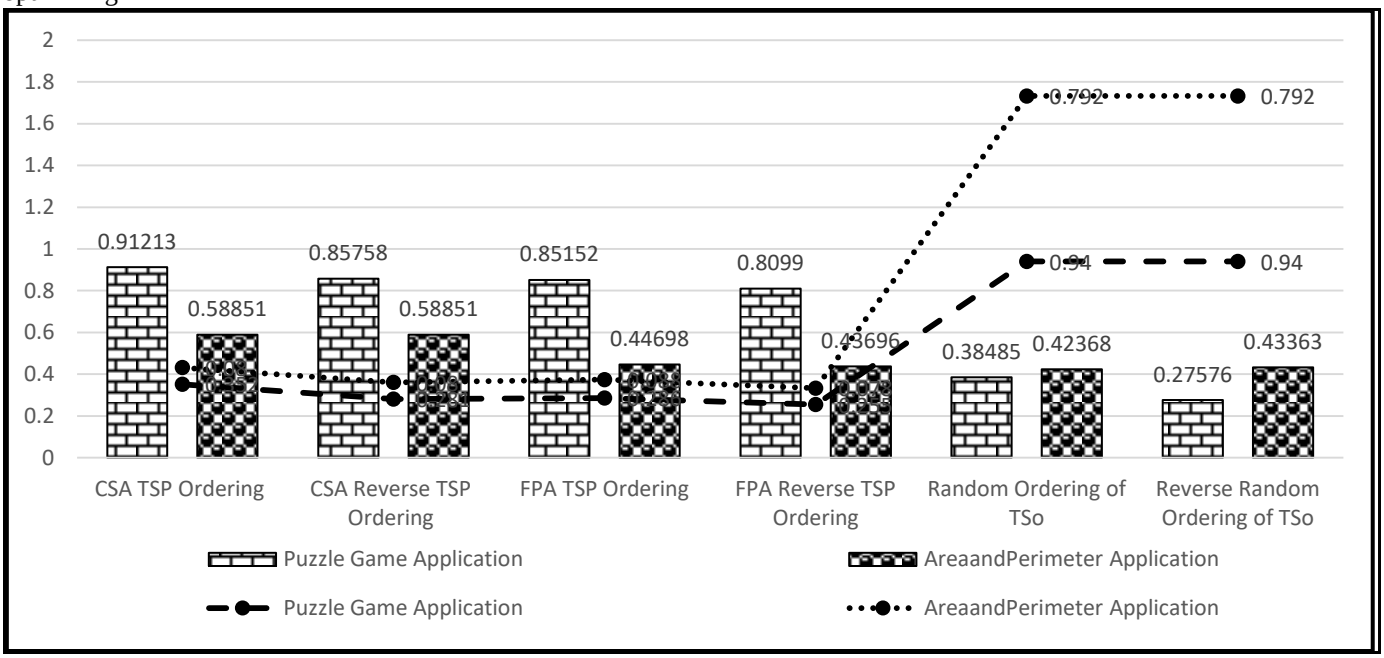

Figure 4: APFD Values with Time of Execution for Different Orderings

VII. CONCLUSION AND FUTURE WORK

In this study CSA is implemented in java to perform TCP for RT. CSA partitions the test cases in different nests by taking into consideration the code coverage of each test case. Three nests have been used to implement CSA. All the nests will contain different set of test cases with different orderings. Test cases in each nest is selected on the basis of their code coverage. Each nest is exercised on the application under consideration to analyse the rate of fault coverage by calculating APFD metrics. Nest with the maximum value for APFD metrics is selected as final nest/ test suite. The procedure is applied for two java applications. For the validation of the results APFD value for CSA order (TS_p), CSA reverse order (TS_p), FPA order (TS_p), FPA reverse order (TS_p), Random Order of (TSo) and Reverse Random Order of (TSo) are compared. It is learnt from the study that APFD for CSA outperformed the FPA for both the applications namely Puzzle Game and AreaandPerimeter. It can be stated that prioritized set of test cases given by NIMOA gave better APFD results as compared to traditional approach and also CSA outperformed FPA. For the future work more NIMOA will be compared for TCP with CSA and FPA. Also, a technique will be proposed for TCP in RT using NIMOA. The proposed technique will be compared with other NIMOA and the traditional ways of TCP for the validation of results.

REFERENCES

1. AHMED, B.S. 2016. Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. *Engineering Science and Technology, an International Journal*, 19(2), 737-53.
2. AHMED, B.S., ABDULSAMAD, T.S. AND POTRUS, M.Y. 2015. Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the cuckoo search algorithm. *Information and Software Technology*, 66,13-29.
3. BACANIN, N. 2012. Implementation and performance of an object-oriented software system for cuckoo search algorithm. *International Journal of Mathematics and Computers in Simulation*, 6(1), 185-93.
4. DHAREULA, P. AND GANPATI, A. (In press). Flower pollination algorithm for test case prioritization in regression testing. *In 4th International Conference on Information and Communication Technology for Sustainable Development* (ICT4SD 2019) on July 5th -6th, 2019, Goa, India. Springer.
5. DHAREULA, P., AND GANPATI, A. 2015. Prevalent criteria's in regression test case selection techniques: An exploratory study. *In International Conference on Green Computing and Internet of* Things, IEEE, 871-876.
6. DHAREULA, P., AND GANPATI, A. 2016. Identification of attributes for test case reusability in regression test selection techniques. *In 3rd International Conference on Computing for Sustainable Global Development, IEEE,* 1144-1147.

7. TestNG, 2004. https://testng.org/doc/download.html
8. Java Code Coverage for Eclipse, 2006. https://www.eclemma.org/download.html
9. KUMAR, K.S. AND MUTHUKUMARAVEL, A. 2017. Optimal Test Suite Selection using Improved Cuckoo Search Algorithm Based on Extensive Testing Constraints. *International Journal of Applied Engineering Research*, 12(9), 1920-1928.
10. MOHAMAD, A., ZAIN, A.M., BAZIN, N.E. AND UDIN, A. 2013. Cuckoo search algorithm for optimization problems-a literature review. *In Applied Mechanics and Materials*, 421, 502-506. Trans Tech Publications.
11. MOHAMAD, A.B., ZAIN, A.M. AND BAZIN, A.N.E. 2014. Cuckoo search algorithm for optimization problems—a literature review and its applications. *Applied Artificial Intelligence*. 28(5), 419-48.
12. NAGAR, R., KUMAR, A., SINGH, G.P. AND KUMAR, S. 2015. Test case selection and prioritization using cuckoos search algorithm. *International Conference on Futuristic Trends on Computational Analysis and Knowledge Management*, IEEE, 283-288.
13. ROTHERMEL, G., UNTCH, R.H., CHU, C., HARROLD, M.J. 1999. Test Case Prioritization: An Empirical Study. International Conference on Software Maintenance-1999 (ICSM'99).'Software Maintenance for Business Change, IEEE, 79-188.
14. SRIVASTAVA, P.R., KHANDELWAL, R., KHANDELWAL, S., KUMAR, S. AND RANGANATHA, S.S. 2012. Automated test data generation using cuckoo search and tabu search (CSTS) algorithm. *In Journal of Intelligent System*, 21(2), 195- 224.
15. SRIVASTAVA, P.R., SRAVYA, C., ASHIMA, KAMISETTI, S. AND LAKSHMI, M. 2012. Test sequence optimisation: an intelligent approach via cuckoo search. *International Journal of Bio-Inspired Computation*, 4(3), 139-48.
16. YANG, X.S. AND DEB, S. 2009. Cuckoo search via Lévy flights. *World Congress on Nature & Biologically Inspired Computing*, IEEE, 210-214.
17. YANG, X.S. AND DEB, S. 2010. Engineering optimisation by cuckoo search. *arXiv preprint arXiv*.
18. YANG, X.S. AND DEB, S. 2014. Cuckoo search: recent advances and applications. *Neural Computing and Applications*, 24(1), 169-74.
19. YANG, X.S. 2014. Nature-Inspired Optimization Algorithms. *Elsevier*.

AUTHOR PROFILE

Anita Ganpati is a Professor of Computer Science at Himachal Pradesh University, Shimla, India. Her research interest includes Open Source Software, Big Data, Cloud Computing. She has over 54 publications in International Journals and Over 6o papers in Conference Proceedings.

Priyanka Dhareula received her MCA form Guru Jambeshwar University of Science and Technology, Hissar, India. She is currently pursuing her Ph.D. in Computer Science under the supervision of Dr. Anita Ganpati at Himachal Pradesh University, Shimla, India. Her research interest includes Software Testing and Optimization.