IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-12, Dec 2018 E-ISSN: 2347-2693

# Regression Test Case Minimization with Firefly based Algorithm

**Ajmer Singh[1*], Vandana[2], Rajvir Singh[3]**

[1, 2, 3]CSE Department, DCRUST, Murthal, India

[*]*Corresponding Author: ajmer.saini@gmail.com, Tel.: +91-9813687398*

**Available online at: www.ijcseonline.org**

Accepted: 15/Dec/2018, Published: 31/Dec/2018

***Abstract*—** Software testing process ordinarily expends no less than half of the aggregate cost required in programming advancement. Programming advancement associations spend significant part of their financial plan and time in testing related tasks. Software testing is an indispensable component in the Software Development Life Cycle (SDLC) and can outfit brilliant outcomes; if directed appropriately and successfully in an improved way. Lamentably, Software testing is frequently less formal and thorough than it ought to. Regression testing means to reveal all the undesired reactions of code corrections on rest of the code. Regression testing ensures that settling of programming deficiencies does not present whatever other issues, which were absent prior. Regression testing is iterative process, where size and many-sided quality of experiments continues expanding. Along these lines, Optimization of experiments is profoundly sought to finish the regression testing inside settled time and cost limitations. Streamlining of experiments amid regression testing is an open research problem as there is no single procedure which can supersede every other system on all parameters. Along these lines, researchers ought to evolve new experiment minimization systems for regression testing to improve its feasibility in view of different parameters. This paper reports a work on building up a novel minimization procedure for regression testing utilizing firefly based optimization.

***Keywords*—** Regression Testing,Test case Minimization, Soft computing ,Object Oriented Testing, Software Maintenance,

## I. INTRODUCTION

Test Software testing is one of the essential parts of SDLC. The main objectives of software testing are to firstly check if the system fulfils the specified requirements and then executing a program with the intent of finding errors. Software testing is a practice used to help and differentiate the accuracy, security, performance and dependability for a selected program. Software testing is a vital component and one of the essential part or phase in various SDLC models while most of the budget and time is consumed by this phase only, so according to various findings and research approximately 40% of cost and 70% of time is devoted to this phase. Testing is an imperative research zone in software engineering, prone to end up noticeably much more vital later on. The testing of program is an imperative method for evaluating and deciding the quality of any software product.

There are many strategies for testing software under test (SUT), White box and Black box testing. White box testing aim is to determine a method that goes through internal functionality to figure out the possible bugs and errors and further subdivided in two ways: dynamic and static testing [1]. It is also known as structural testing. It has certain test data inputs which are used to test all the code paths. While Black box testing is a kind of testing in which the main agenda is to transform a set of defined inputs into expected output and this whole process is performed internally and unaware of how this process take place. Various techniques of performing black box testing are cause-effect graphing technique, boundary value analysis, decision table based testing and equivalence class testing [2]

Test suite minimization is an enhancement issue to discover a negligibly estimated subset of the experiments in a suite that activities an indistinguishable arrangement of scope prerequisites from the first suite. The key issue in minimization of test suites is to expel the experiments in a suite that tends to be repetitive in the suite as for the scope of some specific arrangement of program prerequisites. One of the major objectives in test case minimization is to reduce the redundant and obsolete test cases during testing so as to faster the application under test by correcting faults and removing errors in the beginning to lower the chances of failures. In this paper regression based test case minimization method is used. Regression testing means to reveal all the undesired reactions of code corrections on rest of the code. Regression testing ensures that settling of programming deficiencies does not present whatever other issues, which were absent prior. Regression testing is iterative process, where size and many-sided quality of experiments continues expanding.

Recent studies shows that various Evolutionary Algorithms like Artificial Immune System, Simulated annealing, Cuckoo Search Algorithm (CSA) etc. are being functionalized in the field of Software Engineering to obtain optimal solutions.

 

Nowadays, test case minimization and test case generation on multi-objectives are trending in research community rather than working on single objectives.

This paper presents regression test case minimization with Firefly based Optimization Algorithm. The organization of this article is as follows section II consists of related work of different researchers in the same field, section III describes the Firefly algorithm. Section IV highlights the results and their inferences. Finally section 5 concludes the research with some possible future avenues.

## II. RELATED WORK

Test case minimization is a vital part in regression testing. Alongside the quick expanding of the scale and multifaceted nature of test suites, the issues turn out to be increasingly hard to unravel.

Authors in [3] observed that regression testing is a costly, however imperative process. Unfortunately, there might be inadequate assets to take into account the re-execution of all experiments amid regression testing. In this circumstance, test case prioritization techniques plan to enhance the viability of regression testing by executing most valuable tests first. The study of [4] presented experimental work on regression experiment prioritization concentrated on avaricious calculations. Be that as it may, it is realized that these calculations may deliver problematic outcomes since they may develop comes about that signify just nearby minima inside the pursuit space. By differentiation, meta-heuristic and transformative hunt calculations intend to maintain a strategic distance from such issues.

The authors in [5] studied the effectiveness of time ware prioritization technique for regression testing. The results indicate that time aware approach can be helpful in minimizing the overall cost of test case execution. The study in [6] presented a review of test case minimization techniques. The authors discussed various techniques like Heuristic H, GRE, and Divide and Conquer approach Genetic algorithm, selective redundancy, TestFilter, Integer Linear Programming based, DILP, Cluster analysis and set theory based.

Authors of [7] presented a technique in which the tests cases were minimized using configuration-aware structural testing. Cuckoo search algorithm (CSA) is used to optimize and search for an optimal solution by covering the *d*-tuples list. For the purpose of evaluation, a user-configurable system is used as a case study. In addressing the usefulness of the study more effectively, different faults were seeded in the software-under-test through mutation testing techniques. The approach of [8] applied ACO technique which generated optimal paths in the control flow graph. The optimal paths will cover the whole software with minimum redundancy. In the proposed algorithm, the suit of paths is prioritized in a new way so that we can decide which paths are to be tested first. The parameters used in the experiment are transition count, current node, feasible set, probability, updated pheromone, updated heuristic, updated weight, path, and priority.

The researcher [9] worked on regression testing where distinguished 27 papers announcing 36 exact reviews, 21 examinations and 15 contextual investigations. In all out 28 systems for regression test determination are assessed. They exhibit a subjective investigation of the discoveries, a review of systems for regression test choice and related experimental confirmation. No strategy was discovered unmistakably unrivaled since the outcomes rely on upon many shifting components and further they recognized a requirement for observational reviews where ideas are assessed as opposed to little varieties in specialized usage.

Study of [10] focused on Ant Colony Optimization (ACO) technique for solving issue of test case prioritization. Finite state machine is used for prioritizing the test cases from a given test suite to generate optimal solutions. According to their study the Ant colony Optimization algorithm is more effective and optimistic to solve the problems related to test case prioritization. The study of [11] analyzed the literature to review the work on multi-objective test case minimization techniques using evolutionary methods. The authors conclude that very less work has been done using multi-objective optimization technique for the test case minimization. Authors also suggest the possible avenues in this filed based on evolutionary methods like Particle Swarm optimization (PSO), Baits optimization and firefly based optimization.

### *A. Regression Testing*

Regression Testing also known as approval testing is a costly movement as it devours a lot of time and cost. Regression testing is a part of maintenance phase where it is verified that after certain alterations or modifications in the program or code new fault are introduced or not. The main aim of this kind of testing is to rerun the tests performed earlier and rechecking the existence of faults after the alteration in the program code [12]. Basically there are three activities carried out during the process of regression testing that are: Test selection, Test minimization and Test prioritization. In software testing regression testing is required to detect faults, fixing faults and adding new feature to the software without exploiting the basic functionalities of the software.

 

Regression testing is carried out using automation tools not manually as time required for the process is high so it becomes very time consuming to do all the testing part manually as well as maintaining the quality of the product. It is a kind of black box testing where the main agenda is to transform a set of defined inputs into expected output and this whole process is performed internally and unaware of how this process take place.

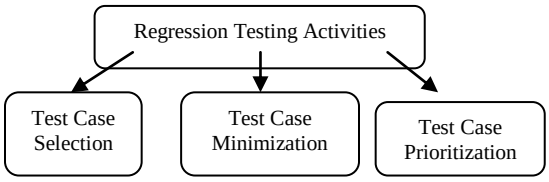


Fig 1. Activities of Regression Testing [13]

Regression testing guarantees changed projects against unintended revisions. Since a few surely understood programming disappointments can be faulted for not testing changes and revisions in a product framework completely and legitimately, numerous procedures have been created to bolster proficient and viable regression testing [14]

**a. Re-Test All**

Amid advancement of some product items, their test suites turn out to be to a great degree vast. So it would be exorbitant to run the entire test suite both as far as time and human exertion, which in the long run winds up being extremely costly regarding cash also. Along these lines, this is illogical to retest the whole test suite. [15].

**b. Selective Re-testing**

Re-test All Method is unfeasible, on the grounds that it requires the re-running the whole test suite. Then again, if just a couple test cases are run, it may be deficient for distinguishing a considerable lot of the issues in altered programming, so the entire regression testing will flop as being inconsistent

*B. Test Suite Minimization (TCM)*

In these procedures expel the repetitive experiments for all time to diminish the extent of test suite. Be that as it may, the blame location ability of a test suite may diminish because of decrease in the quantity of experiments. Test suite minimization is an enhancement issue to discover a negligibly estimated subset of the experiments in a suite that activities an indistinguishable arrangement of scope prerequisites from the first suite. The key issue in minimization of test suites is to expel the experiments in a suite that tends to be repetitive in the suite as for the scope of some specific arrangement of program prerequisites. The conduct of tests adequacy regarding designated assets (work, cash, time, calculation) is asymptotic. This implies, among every conceivable test, just a piece of them is financially helpful [16]. Test suite minimization is an enhancement issue to discover a negligibly estimated subset of the experiments in a suite that activities an indistinguishable arrangement of scope prerequisites from the first suite

*C. Test Case Selection (TCS)*

In test case selection method a portion of the experiments is selected and concentrated only on that piece of work or test cases on which different operations to be performed. [17]

*D. Test Case Prioritization (TCP)*

These procedures distinguish the productive requesting of the experiments to amplify certain properties, for example, rate of blame discovery or scope rate [18]. A critical review by [19], confirm that these prioritization procedures can be valuable to the regression testing

*E. Optimization*

It is the way toward accomplishing certain target utilizing least assets and endeavours in view of an approach superior to anything other contemporary methodologies. "Improvement calculations are pursuit strategies where the objective is to locate an ideal answer for an issue, keeping in mind the end goal to fulfil at least one target capacities, perhaps subject to an arrangement of constraints

## III. FIREFLY ALGORITHM FOR TEST CASE MINIMIZATION

This section highlights the various components of the proposed methodology.

*A. Firefly Behavior*

The blazing light of fireflies is an astounding sight in the mid year sky in the tropical ecosystem and calm areas also known by other name i.e. lightening bug. Most fireflies deliver short, cadenced flashes and there is around 2000 species of fireflies. The example of flashes is frequently interesting for specific animal varieties. [20]. In the given equation (1) the light force I diminish as the separation r increments. Besides, the air retains light, which winds up noticeably weaker and weaker as the separation increments. These two joined elements make most fireflies visible as far as possible separation, generally a few hundred meters during the evening, which is sufficient for fireflies to convey

The basic principles in depicting the standard firefly algorithm are:

1. A firefly gets attracted to other flies irrespective of their sex as all the fireflies are unisex
2. According to the nature of fireflies the attractiveness of

 

the firefly is directly proportional to firefly's brightness. In this phenomenon the less bright firefly tends to move towards the brighter firefly. Both attractiveness and brightness increase as the distance decreases. A particular firefly will move randomly if there is no brighter one.

The landscape of the objective function determines the brightness of a firefly

*B. Measuring Light intensity and attractiveness*

In the firefly calculation, the two vital issues are: definition of the allure and the variety of light power. The light intensity I(r) is obtained as follows,

$$I(r) = \frac{I_s}{r^2} \quad (1)$$

where $I_s$ = Intensity at the source.

Variation in the light intensity I along with distance $r$ for a given medium while keeping the light absorption coefficient fixed is depicted by the following equation,

$$I = I_0 e^{-\gamma r} \quad (2)$$

where $I_0$ = The original light intensity at zero distance $r = 0$. Gaussian form to remove singularity:

$$I(r) = I_0 e^{-\gamma r^2} \quad (3)$$

Equation for depicting the attractiveness of a firefly:

$$\beta = \beta_0 e^{-\gamma r^2} \quad (4)$$

where $\beta_0$ is the attractiveness at r = 0.

The attractiveness function $\beta(r)$ can be any monotonically decreasing functions such as:

$$\beta(r) = \beta_0 e^{-\gamma r^m}, \quad (m \geq 1) \quad (5)$$

The Cartesian distance between two fireflies, respectively,

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \quad (6)$$

The movement of a firefly i attracted towards another firefly j is shown by

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2} (x_j^t - x_i^t) + \alpha \epsilon_i^t \quad (7)$$

*C. Controlling Randomization*

By varying the randomization parameter a we can further improve the convergence of the algorithm. As shown in following equation

$$\alpha = \alpha_\infty + (\alpha_0 - \alpha_\infty) e^{-t} \quad (8)$$

where $t \in [0, t_{max}]$ = The pseudo time for recreations

$t_{max}$ = the maximum number of eras

$\alpha_0$ = the underlying randomization parameter

$\alpha_\infty$ = the last esteem. We can likewise utilize a comparable capacity to the geometrical strengthening plan. That is,

$$\alpha = \alpha_0 \theta^t \quad (9)$$

where $\theta \in (0, 1]$ is the randomness reduction constant.

In most applications, we can use $\theta = 0.95 \sim 0.99$ and $\alpha_0 = 1$. Moreover, in the present variant of the Firefly Algorithm calculation, we don't unequivocally utilize the current worldwide best g*, despite the fact that we just utilize it to decipher the last best arrangements. Our reproductions demonstrated that the productivity may enhance on the off chance that we include an additional term $\lambda \epsilon i(g^* - xi)$ to the refreshing recipe. It merits bringing up that is an arbitrary walk, one-sided toward the brighter fireflies. In the event that $\beta_0 = 0$, it turns into a straightforward arbitrary walk. Moreover, the randomization term can undoubtedly be reached out to different dispersions.

Description of proposed algorithm

In view of the above depiction, firefly algorithm can take care of development reconfiguration issue. The calculation can be separated into two phases, the Particle swarm optimizations arrange and the Genetic algorithm organizes. [21]

Step 1: In this step M hubs are initialized haphazardly along with the maximum cycle time $N_{cmax}$. The hybrid likelihood and transformation likelihood are 0.9 and 0.05 individually.

Step 2: In this step the target work estimations is calculated for all hubs, the position of the hub is stored with the base target work, an incentive as the worldwide best hub.

Step 3: GA stage.

**Selection Operator**

Roulette wheel choice procedure is generally utilized as a part of genetic algorithm since it can guarantee that the choice likelihood of every hub is relative to its wellness, i.e. the better a hub's wellness, the more probable it will be chosen

**Crossover Operator**

Hybrid occurs between two guardians which are freely chosen from the populace. Youngsters are made by the single-point hybrid operation. It can be characterized as takes after:

 

$$p_1^{new} = w.Pi + (1-w).P \quad (10)$$

And

$$p_2^{new} = w.P2 + (1-w).P \quad (11)$$

**Mutation Operator**

Mutation operator is mainly used to maintain node diversity premature along with avoiding the premature convergence for the population. The mutation operator which is adaptive by nature is defined as follows:

$$P_i^j(k+1) = P_i^j(k) + p.sP_i^j(k) \quad (12)$$

Where, $P_i^j(k)$ = The $i_{th}$ dimension of the $j_{th}$ node in the $k_{th}$ generation [22].

*D. Modified Firefly algorithm for Test Case Minimization*

*Step1. Population is initialized with m nodes.*

*Step2. Fitness of each node is computed and the fitness and position of every node is updated and a record is maintained.*

*Step3. Global best solution with minimum objective function is identified.*

*Step4. If the stopping criteria is satisfied then optimal solution is obtained else go to step 5.*

*Step5. If the stopping criterion in step 4 is not satisfied then the population is divided into two smaller populations based on the hybrid probability P.*

*(i) Nodes objective function value is evaluated.*

*(ii) Update position of the nodes.*

*(iii) Update velocity of all the nodes.*

*(iv) Also update the best solution for all the nodes.*

*(v) Perform selection based on fitness for each node.*

*(vi) Perform crossover and mutation for each node.*

*Step6. If all the necessary conditions are satisfied, update all the nodes else go to step*

## IV. RESULTS AND ANALYSIS

The Firefly algorithm presented in this paper was simulated on MATLAB. The proposed algorithm was empirically evaluated by executing it with testing data of CloudSim. Results obtained were compared with the PSO. The figure 2 below depicts the coverage ratio of PSO and Firefly algorithm. The results are encouraging ones. Also the cost of best cost for PSO and Firefly Algorithm are calculated. The computations are shown in figure 3 below.

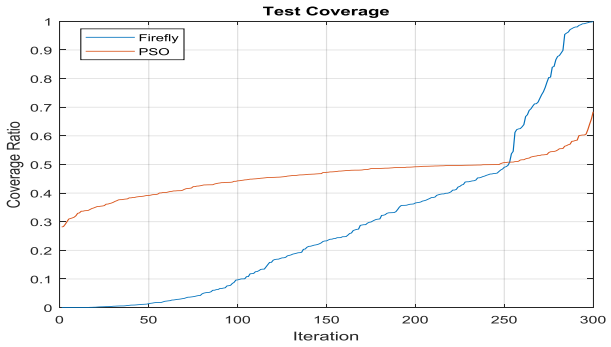


Figure 2.Comparison of Test Coverage of PSO and Firefly algorithm

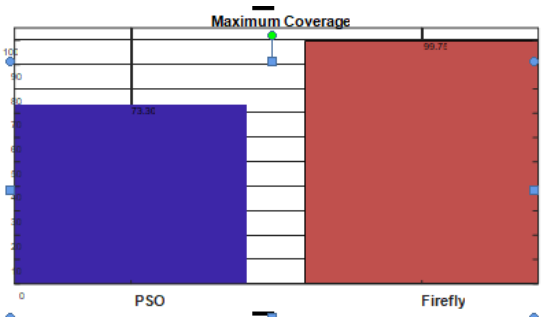


Figure 3.Comparison of MAX Test Case Coverage of Minimization using PSO and Firefly Algorithm

It was observed that best coverage was achieved after 803 iterations in case of PSO whereas Firefly algorithm gave best coverage after 401 iterations. The difference in performances of the two is very significant.

## V. CONCLUSION AND FUTURE SCOPE

This research work has demonstrated Firefly calculation based approach for test case minimization. The new calculation contemplates minimization of test cases as a streamlining issue with taking in test weights and test span. The Firefly calculation is picked as a result of its broadly useful nature and capacity to take a shot at a minimization issue. The new calculation is to be sure relevant and successful in limiting suites with huge suite examine decrease and permits to 99% test scope when contrasted with ~75% of PSO construct calculation relying on the experiment chose. The instrument can produce coordinated experiments in the wake of investigating different experiment situations. In Future we are planning to evaluate the work with more number of attributes. Also the time complexity of the execution of Firefly algorithm is to be considered as another important feature of evaluation

 

## REFERENCES


[1] M. Utting and B. Legeard, Practical model-based testing: a tools approach. 2010.

[2] P. R. Srivastava, M. Ray, J. Dermoudy, B. Kang, and T. Kim, "Test Case Minimization and Prioritization Using CMIMX Technique *," vol.333031, pp. 25–26.

[3] Z. Li, M. Harman, and R. M. Hierons, "Search algorithms for regression test case prioritization," IEEE Trans. Softw. Eng., vol. 33, no. 4, pp.225–237, 2007.

[4] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies," IEEE Trans. Softw. Eng.,2002.

[5] P. Parashar, A. Kalia, and R. Bhatia, "How Time-Fault Ratio helps in Test Case Prioritization for Regression Testing," no. 1, 2016.

[6] R. Singh and M. Santosh, "Test Case Minimization Techniques : A Review 1,2," Int. J. Eng. Res. Technol., vol. 2, no. 12, pp. 1048–1056, 2013.

[7] B. S. Ahmed, "Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing," Eng. Sci. Technol. an Int. J., 2016.

[8] S. Biswas, M. S. Kaiser, and S. A. Mamun, "Applying Ant Colony Optimization in software testing to generate prioritized optimal path and test data," in 2nd International Conference on Electrical Engineering and Information and Communication Technology, iCEEiCT 2015, 2015.

[9] E. Engström, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," Information and Software Technology. 2010.

[10] S. Sharma and A. Singh, "Model-based test case prioritization using ACO: A review," in 2016 4th International Conference on Parallel, Distributed and Grid Computing, PDGC 2016, 2016.

[11] Vandana and A. Singh, "Multi-objective test case minimization using evolutionary algorithms: A review," in Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017, 2017, vol. 2017–Janua.

[12] M. Rani, "Review of Regression Test Case Selection Techniques," vol. 3, no. 5, pp. 1029–1034, 2014.

[13] S. Yoo and M. Harman, "Regression Testing Minimisation, Selection and Prioritisation : A Survey," Test. Verif. Reliab, vol. 00, pp. 1–7, 2007.

[14] T. L. Graves, M. J. Harrold, J. Kim, A. Porters, and G. Rothermel, "An empirical study of regression test selection techniques," in Proceedings of the 20th International Conference on Software Engineering, 1998, pp. 188–197.

[15] H. Srikanth, L. Williams, and J. Osborne, "System test case prioritization of new and regression test cases," in 2005 International Symposium on Empirical Software Engineering, ISESE 2005, 2005, vol. 00, no. c, pp. 64–73.

[16] P. McMinn and M. Holcombe, "The state problem for evolutionary testing," … Evol. Comput. 2003, 2003.

[17] C. Catal and D. Mishra, "Test case prioritization: A systematic mapping study," Softw. Qual. J., vol. 21, no. 3, pp. 445–478, 2013.

[18] B. Korel, L. H. Tahat, and M. Harman, "Test prioritization using system models," in IEEE International Conference on Software Maintenance, ICSM, 2005, vol. 2005, pp. 559–568.

[19] P. Gaur and R. S. Singhal, "A critical review on test case prioritization and optimization using soft computing techniques," International Journal of Control Theory and Applications. 2016.

[20] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, and M. Steinbrecher, Computational Intelligence: A Methodological Introduction. 2016.

[21] A. Alert and L. Grunske, "Test data generation with a Kalman filter-based adaptive genetic algorithm," J. Syst. Softw., 2015.

[22] H. Duan, Q. Luo, G. Ma, and Y. Shi, "Hybrid Particle Swarm Optimization and Genetic Algorithm for Multi-UAV Formation Reconfiguration," Ieee Comput. Intell. Mag., 2013.


## Authors Profile

*Mr. Ajmer Singh* pursed Bachelors and Masters of Technology from Kurukshetra University, india. He is currently pursuing Ph.D. and currently working as Assistant Professor in Department of Computer Science and Engineering, DCRUST Murthal, India..He is a member of IAENG. He has 10 years of teaching experience and 4 years of Research Experience.

*Mr. Rajvir Singh* is B.Tech and M.Tech in Computer Science and engineering. He is pursuing PhD in Software testing domain. and currently working as Assistant Professor in Department of Computer Science and Engineering, DCRUST Murthal, India..He is a member of IAENG. He has 10 years of teaching experience and 4 years of Research Experience.

*Vandana* is a M.Tech Scholar at DCRUST Murthal, Her area of interest include soft computing technique and software testing

 