Test Case Selection and Prioritization using Cuckoos Search Algorithm

Reetika Nagar[#1], Arvind Kumar[2], Gaurav Pratap Singh[#3], Sachin Kumar[4]

[#]Computer Science and Engineering Department
Aligarh College of Engineering and Technology
Aligarh, India
[1]reetikanagar@gmail.com,, [3]gauravparatpsingh2004@gmail.com
[2]Pitney Bowes Software
Noida, India
arvind.jki@gmail.com
[4]Computer Science and Engineering Department
Shivdan Singh Institute of Technology & Management
Aligarh, India
sachinbhati03@gmail.com

***Abstract*—Regression Testing is an inevitable and very costly activity that is implemented to ensure the validity of new version of software in a time and resource constrained environment. Execution of entire test suite is not possible so it is necessary to apply techniques like Test Case Selection and Test Case Prioritization for proper selection and schedule of test cases in a specific sequence, fulfilling some chosen criteria. Cuckoo search (CS) algorithm is an optimization algorithm proposed by Yang and Deb [13]. It is inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds. Cuckoo Search is very easy to implement as it depends on single parameter only unlike other optimization algorithms. In this paper a test case selection and prioritization algorithm has been proposed using Cuckoo Search. This algorithm selects and prioritizes the test cases based on the number of faults covered in minimum time. The proposed algorithm is an optimistic approach which provides optimum best results in minimum time.**

Keywords— Cuckoos Search; Levy Flight; Regression Test Selection; Test Case Prioritization; Artificial Intelligence

I. INTRODUCTION

Software maintenance [1, 2, 3] is an essential phase of software development life cycle. Software maintenance is becoming very expensive with the changing requirements and trends. During software maintenance activities performed are such as enhancement of software functionalities, error correction, optimization, etc.

The two types of maintenance testing are confirmation testing and regression testing. Confirmation testing also known as retesting is the process to ensure that the tests cases in which defects were found and test cases which failed in last execution are corrected and another test execution will undergo to pass after the defects and re-confirm that the failure and defects that has been fixed does not exist. It is important to ensure that confirmation testing should be performed in exactly the identical way in which initial testing was performed using the same data, inputs and environment conditions.

Regression testing is the verification process to determine that previous functioning of software remains after a change is made in the software and make sure that the previously tested code has not introduced any new errors. Regression testing is an expensive process, thus several techniques have been developed in an attempt to minimize regression testing cost. Regression testing techniques can be implemented more effectively using optimization algorithms to achieve good quality optimal results. Optimization algorithms main goal is to explore large search space to get best optimal solutions. Optimization algorithms can find quality solutions to difficult optimization problems in a reasonable amount of time, but there is no guarantee that optimal solutions can be reached. In this paper, a detailed analysis of regression testing, its techniques and optimization algorithms used to choose minimum set of test cases that covers all the faults and bugs in minimum time is given. In this paper, a detailed analysis of regression testing, its techniques and optimization algorithm used to choose minimum set of test cases that covers all the faults and bugs in minimum time is given. As for effective testing it becomes necessary to choose minimum set of test cases that covers all possible faults and bugs in minimum time. Regression testing techniques can be effectively used for all this activity. In this paper, we have proposed an algorithm for reduction of test cases using optimization algorithm – Cuckoo Search Algorithm.

978-1-4799-8433-6/15/$31.00 © 2015 IEEE

Sections of this paper have been organized in following manner. Section II discusses about regression testing and its techniques used in this paper. In Section III literature review about Cuckoos search is given. Section IV gives a brief idea about the optimization algorithm used. Section V discusses about proposed algorithm. In section VI implementation of proposed algorithm has been done using some problem or software system example. In Section VII results has been shown. In Section VIII conclusions and the future work is given.

II. Regression Testing and its Techniques

One of the main objectives to introduce regression testing is to determine whether modification in one part of the software affects other parts of the software. Let Q be a program [4], modified version of Q is Q′, and let 'T' be a test suite for Q. Regression testing consists of reusing 'T' on Q′, and determining where the new test cases are needed to effectively test code or functionality added to or changed in producing P′. Regression testing is used not only for testing the correctness of a program, but often also to trace the quality of its output [5]. It is a risk free process occurs at an optimal cost and in minimum time. The purpose behind regression testing is to increases the productivity and efficiency of software products and quality assurance applications. Regression testing has been categorized to different testing techniques as given below:

A. Regression Test Selection

Regression test selection [6] problem is about choosing a subset of test cases. In regression test selection technique test cases that are necessary to validate modified software are selected from existing test cases set. It is used to reduce the time required to retest the modified program by selecting test cases. There are different techniques for test case selection. Rothermel [7] identified the various categories in which Regression Test Selection Technique can be evaluated and compared. These categories are: (a) Inclusiveness; (b) Precision; (c) Efficiency; (d) Generality.

B. Test Case Prioritization

Test Case Prioritization (TCP) [8] technique prioritize the test cases. Prioritization of test cases is generally done to reduce the cost of regression testing. Test cases are prioritized so that those which are very essential, by some means, are made to run earlier in the testing phase. There exists a large variety of prioritization techniques. Mostly used techniques are coverage-based prioritization techniques, that is, prioritization in terms of the number of statements, path coverage, branch coverage and fault coverage. In this proposed work prioritization of test cases has been done based on their fault coverage capability in minimum time. Test cases which are selected by regression test selection technique has been ordered in test case prioritization technique depending on their fault coverage efficiency in minimum time using proposed algorithm.

III. Literature Review

In the past recent years many modern meta-heuristic algorithms have been developed with an aim to carry out global search, such as genetic algorithms [9], ant colony optimization (ACO), particle swarm optimisation (PSO) [10]. The efficiency of meta-heuristic algorithms can be credited to the fact that they imitate the optimal features in nature, especially the selection of the fittest in biological systems which have evolved by natural selection over millions of years. Two important characteristics of meta-heuristics are: intensification and diversification [11, 12, 13]. A variety of meta-heuristics has been used in different software engineering solutions [14, 15]. Recently, a new meta-heuristic search or optimization algorithm has been developed by Yang and Deb known as Cuckoo Search (CS).

Cuckoo search (CS) is one of the latest nature-inspired optimization algorithms [16, 17, 18]. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Levy flights [19], rather than by simple isotropic random walks. Recent studies show that CS is potentially far more efficient than PSO and genetic algorithms. Cuckoos are fascinating birds, not only because of the beautiful sounds they can make, but also because of their aggressive reproduction strategy. Some species such as the Ani and Guira cuckoos lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs. Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species). Further detail of cuckoo search has been given in next section.

The literature on cuckoo search is expanding rapidly. There has been a lot of attention and recent studies using cuckoo search with a diverse range of applications [20]. Walton et al. improved the algorithm by formulating a modified cuckoo search algorithm [21], while Yang and Deb extended it to multiobjective optimization problems.

IV. Optimization Algorithms Used

It has been identified that one of the software engineering areas with a more suitable and realistic use of artificial intelligence techniques is software testing [22, 23] and those techniques are known as meta-heuristic approaches. Meta-heuristic algorithms are basically used to solve almost any optimization problem. The main aim of meta-heuristic algorithms is to efficiently explore the search space and order to find an optimally best solution for any problem under consideration. A recently developed meta-heuristic optimisation algorithm, Cuckoo Search is being used under this study for selection and prioritization of different test cases in a test suite.

A. Cuckoo Search

Cuckoo Search [16, 24, 25] is more robust and generic as compared to other optimization technique. Its processing depends on following rules:

1) Each cuckoo lays one egg at a time, and dump its egg in randomly chosen nest;
2) The best nests with high quality of eggs will carry over to the next generations;
3) The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0, 1]$. In this case, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location.

In this case, the host bird can either throw the egg away or abandon the nest, and build a completely new nest. For simplicity, this last assumption can be approximated by the fraction p_a of the n nests are replaced by new nests (with new random solutions).

For a maximization problem, the quality or fitness of a solution can simply be proportional to the value of the objective function. Other forms of fitness can be defined in a similar way to the fitness function in genetic algorithms. For simplicity, we can use the following simple representations that each egg in a nest represents a solution, and a cuckoo egg represent a new solution, the aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests. In this proposed work nests are the test cases and cuckoo egg means a new fault has been had detected which helps to select the best and superior population of test cases.

B. *Levy Flight*

Levy flight [13, 19, 26] style is a straight flight paths punctuated by a sudden 90^o turn. Levy flight in general is the pattern of moving from one place/nest to another. When generating new solutions $\mathbf{x}^{(t+1)}$ for, say, a cuckoo i, a Levy flight is performed

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \alpha \oplus \text{Levy}(\lambda) \ldots\ldots (1)$$

where, $\alpha > 0$ is the step size which should be related to the scales of the problem of interests. In most cases, we can use $\alpha = 1$. The above equation is essentially the stochastic equation for random walk. In general, a random walk is a Markov chain whose next status/location only depends on the current location (the first term in the above equation) and the transition probability (the second term). The product $\oplus$ means entry wise multiplications. This entry wise product is similar to those used in PSO, but here the random walk via Levy flight is more efficient in exploring the search space as its step length is much longer in the long run.

V. PROPOSED ALGORITHM

In this paper, an algorithm has been proposed to minimize the overall cost of regression testing by test cases reduction. The proposed algorithm is based on Cuckoo Search. This algorithm selects test cases from the given test suite that will cover all possible faults detected in minimum execution time. Here test cases are used as cuckoos and faults of test cases are represented as eggs of cuckoos.

Regression testing selection and prioritization techniques are used in proposed algorithm to reduce the number of test cases. In the proposed algorithm, cuckoo search via levy flight (finding all possible faults of given test cases in minimum execution time) using knapsack method criteria (target limit of execution time is fixed) has been implemented. In the proposed algorithm knapsack criteria is considered in reverse order, that is, instead of increasing execution time to the defined limit it has to be reduced to value less than equal the defined limit or target, when execution time of test cases sequence is lower or equal to target algorithm terminates. Thus, Cuckoo Search via Levy Flight algorithm for finding minimum set of test cases has been proposed.

A. *Assumptions*

Following assumptions has been considered for the proposed algorithm:

1) Initial population pool of test cases is randomly generated. Population is represented as TS= $\{tc_1, tc_2, .., tc_t\}$.
2) The fitness function and stopping criteria vary according to problem.
3) In proposed problem,
   - Fitness function of test cases is fault count (ff_i) and execution time.
   - Stopping Criteria is total possible faults covered in minimum time or algorithm processed for given maximum number of iterations.
4) In the original test suite each test case ($tc_1, tc_2, \ldots, tc_t$) covers some or all faults.
5) Set of faults is represented as F= $\{f_1, f_2, \ldots, f_m\}$.
6) This problem is considered to be execution time constrained (knapsack problem). Here knapsack criteria is considered in reverse order, instead of increasing execution time value till the TARGET limit , execution time value has to be reduced to value less than equal to the TARGET limit.
7) T_i is the execution time for each test case tc_i, where $1 \leq i \leq t$.
8) Test cases are represented in binary form by 'm' bits (m is the total number of faults).
9) Each bit of the test case depends upon the ability to detect faults. Bit '1' denotes fault is detected and '0' denotes no fault is detected.
10) Input is 'n' number of host sequence nodes, 't' number of test cases and 'm' possible number of faults to be detected.
11) Here number of host sequence nodes is equal to number of test cases.
12) Output is the sequence of traversing test cases and fulfilling stopping criteria.

B. *Proposed Algorithm*

1) Generate population of 't' test cases
2) Initialize a population of 'n' host sequence nodes
3) **while** (T <Max Execution Time) or (stop criterion)
   Get a cuckoo (test cases) randomly by Levy flights
   Evaluate its fitness (ff_i)

Choose a sequences node among 'n' (say, j) randomly
end
A fraction (p_a) of worse sequence node are abandoned and new ones are built;
Keep the best solutions (test cases sequences maximum fault count);
Rank the solutions and find the current best
end while
Post process results and visualization
end
End of algorithm Proposed

C. Description of Proposed Algorithm

1) Generate test case population pool

- Define 't' test cases with numerical value of faults. For example n=5, 6 or any value. There are 5^5 or 6^6 possible combinations, but this algorithm can find them in less than 5^3, and sometimes less than 5^2!
- Define the maximum number of faults to be detected in test cases, that is, maximum number of eggs cuckoos could lay. For example, 5 fault. This means the numerical value of faults can vary from 0 to 2^5-1. Out of bound faults are not considered.
- Convert numerical value of test cases fault to binary value and count number of '1'. Number of '1' is the number of faults.
- Perform union function on all the test cases, that is, add all the test cases using 'OR' operator.
- Evaluate the final outcome of all the test cases to determine the presence and absence of faults, where '1' represents presence of faults and '0' represents absence of faults.
- Eliminate the absent fault from all the given test cases faults. So that all possible faults can be found effectively.

2) Initialize a population of 'n' host sequence nodes –

- Each test case is itself a host sequence node.
- Host sequence nodes are like host nest. It represents collection of faults. Test cases nodes having new and optimal solution are considered and others are abandoned.

3) Implementation

- Process repeats till any stopping criteria satisfy. For the considered problems stopping criteria are the target value and maximum number of iterations or epochs. In the proposed algorithm maximum numbers of epochs for processing the algorithm are randomly fixed before processing the algorithm.
- Using the levy flight try to find a new solution path for processing minimum set of test cases covering all possible faults. If a new fault is determined update its execution time and test case processing path.
- Find fraction, that is, execution time of test cases sequences and best sequences which is covering

  minimum test cases with all faults with least execution time is kept.
- Rank all the best solution and find the optimally best result. Find execution time of best optimal solution.

4) Calculate fitness value of optimal sequence having reduced number of test cases.

Fitness value of best test case sequence/route =

$$\text{Number of epoch} + \frac{1}{(\text{Path Length} + \text{Execution Time})}$$

5) Test cases selected are determining all possible faults of test suite in minimum execution time making testing time and cost effective. Hence test case selection and reduction is done. End of proposed algorithm

VI. IMPLEMENTATION

The proposed algorithm has been implemented using JAVA. The algorithm takes as input the different number of test cases, their faults count and determining the best processing sequence of minimum test cases selected, maximum epochs and target value for estimating the stopping criteria while processing cuckoo search technique in proposed algorithm. The basic evaluation of test case selection and prioritization techniques using the proposed algorithm is to cover all possible faults of the system in minimum number of test cases within minimum optimal time.

Let us consider an example, covering total faults varying from 1 to 10 for executing the algorithm. The value of maximum epochs and target max has been fixed randomly for the design algorithm as 50 and 86.63 respectively. But value of maximum epochs, target may vary according to user requirement. In this section we show the working and efficiency of proposed approach using an example. Dataset of test case has been assumed randomly and has not been referred from any prescribed problem.

A. Example

In the considered example number of host sequence nodes is 6, number of test cases is 6 and faults count is 5.

Implementation-

Inputs: Number of host nodes: 6
Number of test cases: 6
Number of faults: 5

Range of decimal number for representing faults: 0 to 31

Generation of test case population: Randomly generated faults of each test case: 24, 2, 16, 23, 25, 20

Binary representation of faults (here test case 1st is represented as 0th and so on):

TABLE 1
BINARY REPRESENTATION

S.No	Faults of Test Cases	Faults Count
0	11000	2
1	10	1
2	10000	1
3	10111	4
4	10101	3
5	10100	2

Union of all test cases is 11111

Possible combination of faults is 11111

Possible combination of faults of all test cases is

Test case 0th is 11000

Test case 1th is 10

Test case 2th is 10000

Test case 3th is 10111

Test case 4th is 11001

Test case 5th is 10100

After execution of proposed Cuckoo search algorithm following result has been achieved:

- Shortest path/route of traversing or processing test cases is TC0, TC5, TC4, TC3, TC2, TC1 covering all possible faults in 70.1641 unit time
- Best shortest route of traversing test cases is TC0, TC4, TC3 covering all possible faults in 37.5965 unit time

Fitness value of best shortest test case sequence/route is 2.02016. It is observed that reduction in the test suite of the considered problem using cuckoo search is approximately 40%.

VII. RESULTS

The results obtained in section VI for the considered problem can be represented in the form of an directed graph G(V,E) where V ≡ TC i.e. test cases are represented by the vertices in the graph. E is the path to traverse test cases and get an optimal sequence of test cases in the graph. The results obtained after implementing and executing cuckoo search has been represented graphically using MATLAB.

Graphical representation of problems implemented in above section is given below:

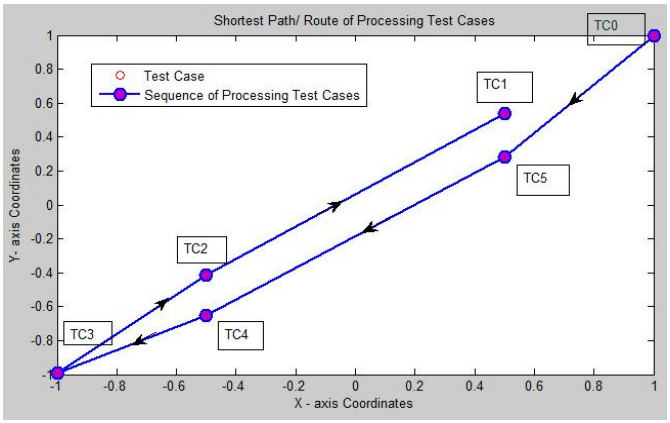

Fig. 1. Graphical Representation of Path/Route of Processing Test Cases for problem 1st

Execution time of processing test cases is 70.1641 unit time

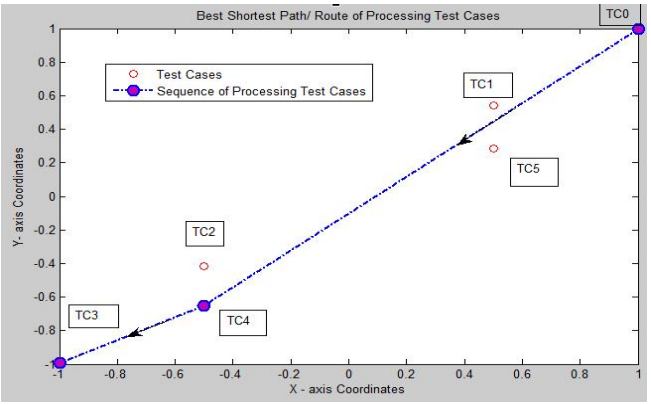

Fig. 2. Graphical Representation of Best Shortest Path/Route of Processing Test Cases for problem 1st

Execution time of processing test cases is 37.5965 unit time

VII. CONCLUSIONS AND FUTURE SCOPE

We have proposed algorithm for selection and prioritization of test cases from a given test suite using cuckoos search via levy flight algorithm. This approach has been tested for several problems or examples. One of these examples has been described in this paper. The algorithm implemented using this approach identifies and reduces the test cases effectively. It is observed that reduction in the test suite of the considered problem using Cuckoos Search is approximately 40%. Another observation is that to cover all possible faults all paths need not be explored and fitness value of best shortest test case sequence/route is 2.02016. The proposed algorithm provides better results and positive

feedback and lead to better solutions in minimum optimum time.

In future research automation of the proposed algorithm is to be done. Further it can be implemented on large and complex problems or software. We also aim to reduce test suite size using other optimization algorithms and compare them.

REFERENCES

[1] Harrold, M.J., "Reduce, reuse, recycle, recover: Techniques for improved regression testing", IEEE International Conference on Software Maintenance, Edmonton, AB, Canada, pp. 5-5, 2009.

[2] Roger S. Pressman, Software Engineering: a practitioner's approach, 6th edition, McGraw-Hill, 2004.

[3] Ian Sommerville, Software Engineering, 7th edition, Addison Wesley, 2004.

[4] Sebastian Elbaum, Praveen Kallakuri, Alexey G. Malishevsky, Gregg Rothermel, Satya Kanduri, "Understanding the Effects of Changes on the Cost-Effectiveness of Regression Testing Techniques", Journal of Software Testing, Verification, and Reliability, vol. 13, no. 2, pp. 65-83, June 2003.

[5] G. Duggal, B. Suri, "Understanding Regression Testing Techniques", COIT, India, 2008.

[6] G.Rothermel and M.J. Harrold, "A Safe, Efficient Regression Test Selection Technique", ACM Trans. Software Eng. and Methodology, vol. 6, no. 2, pp. 173-210, April 1997.

[7] G. Rothermel and M. J. Harrold, "A framework for evaluating regression test selection techniques", in Proceedings of the 16th International Conference on Software Engineering (ICSE 1994), pp. 201-210, IEEE Computer Society Press, 1994.

[8] G. Rothermel, R.H. Untch, C. Chu and M.J. Harold, "Test Case Prioritization", IEEE Transactions on Software Engineering, vol. 27,no. 10, pp. 928-948, October 2001.

[9] Goldberg, D. E.,Genetic Algorithms in Search, Optimisation and Machine Learning, Reading, Mass., Addison Wesley, 1989.

[10] Kennedy, J. and Eberhart, R. C., "Particle swarm optimization", Proc. of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995.

[11] Blum, C. and Roli, A., "Metaheuristics in combinatorial optimization: Overview and conceptural comparision", ACM Comput. Surv., 35, pp. 268-308, 2003.

[12] Gazi, K., and Passino, K.M., "Stability analysis of social foraging swarms" , IEEE Trans. Sys. Man. Cyber. Part B - Cybernetics, 34, pp. 539-557, 2004.

[13] Yang, X. S. and Deb, S., "Cuckoo search via Levy flights", Proceeings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009, India), IEEE Publications, USA, pp. 210-214, 2009.

[14] Kumar, A., Nagar, R and Baghel, A.S., "A Genetic Algorithm Approach to Release Planning in Agile Environment", published in IEEE International Conference Information Systems and Computer Networks (ISCON), pp. 118-122, 2014.

[15] Nagar, R., Kumar, A., Kumar, S. and Baghel, A.S., "Implementing Test Case Selection and Reduction Techniques using Meta-Heuristics", published in IEEE's "5th International Conference Confluence 2014: The Next Generation Information Technology Summit, pp. 837-842, 2014.

[16] Yang XS, Deb S, "Engineering optimization by cuckoo search", Int J Math Modell Num Opt 1(4):pp. 330–343, 2010.

[17] Yang XS, Deb S (2012) Multiobjective cuckoo search for design optimization. Comput Oper Res. Accepted October (2011). doi:10.1016/j.cor.2011.09.026.

[18] Yildiz AR (2012) Cuckoo search algorithm for the selection of optimal machine parameters in milling operations. Int J Adv Manuf Technol. doi:10.1007/s00170-012-4013-7.

[19] Pavlyukevich I , Levy flights, non-local search and simulated annealing. J Comput Phys 226:1830–1844, 2007.

[20] Walton S, Hassan O, Morgan K, Brown MR, "Modified cuckoo search: a new gradient free optimization algorithm", Chaos Solitons Fractals 44(9):710–718 , 2011.

[21] E.G. Talbi. (2009), Metaheuristics from design to implementation. [Online].Available: www.wiley.com.

[22] Dorit S. Hochba, "Approximation Algorithms for NP-Hard Problems", ACM SIGACT, vol. 28 no. 2, pp. 40-52, June 1997.

[23] J. Holland, "Adaption in Natural and Artificial Systems", Ann Arbor, MIT Press, University of Michigan, 1975.

[24] Payne, R. B., Sorenson, M. D., and Klitz, K., "The Cuckoos", Oxford University Press, 2005.

[25] Yang, X. S., & Deb, S., "Cuckoo search: recent advances and applications", Neural Computing and Applications, 24(1), 169-174. Springer, 2014.

[26] Barthelemy, P., Bertolotti, J., Wiersma, D. S., "A L´evy flight for light", *Nature*, 453, 495-498, 2008.