**International Journal of Electrical and Computer Engineering (IJECE)**
Vol. 5, No. 6, December 2015, pp. 1424~1432
ISSN: 2088-8708


# Test Case Reduction Using Ant Colony Optimization for Object Oriented Program

**Sudhir Kumar Mohapatra*, Srinivas Prasad****
* Research Scholar, SOA University, Bhubaneswar, Odisha, India
** Dept. of Computer Science &Engineering, GMRIT, Andhra Pradesh, India

**Article Info**

***Article history:***

Received May 6, 2015
Revised Jul 18, 2015
Accepted Aug 2, 2015

***Keyword:***

Ant colony optimization
Representative set
Software testing
Test suite reduction

**ABSTRACT**

Software testing is one in all the vital stages of system development. In software development, developers continually depend upon testing to reveal bugs. Within the maintenance stage test suite size grow due to integration of new functionalities. Addition of latest technique force to make new test case which increase the cost of test suite. In regression testing new test case could also be added to the test suite throughout the entire testing process. These additions of test cases produce risk of presence of redundant test cases. Because of limitation of time and resource, reduction techniques should be accustomed determine and take away. Analysis shows that a set of the test case in a suit should satisfy all the test objectives that is named as representative set. Redundant test case increase the execution price of the test suite, in spite of NP-completeness of the problem there are few sensible reduction techniques are available. During this paper the previous GA primarily based technique proposed is improved to search out cost optimum representative set using ant colony optimization.



***Corresponding Author:***

Sudhir Kumar Mohapatra**,**
Departement of Computer Science & Engineering,
SOA University, Odisha, India.
Email: sudhirmohapatra@hotmail.com

## 1. INTRODUCTION

Software testing and retesting is done frequently during the software development lifecycle and in particular in regression testing. In regression testing software grows and evolves, that create new test cases and added them to a test suite to exercise the latest changes to the software. Over many versions of the development of the software, test cases in the test suite can be redundant .The redundant test case may in respect to the testing requirements for which they were generated, because these requirements are now also satisfied by new test cases in the test suite that were newly added to cover changes in the later versions of software. Due to limitation of time and resource for retesting the software every time before a new version is release, it is really important to search for techniques that ensure manageable test suits size by periodically removing redundant test cases. This process is called *test suite minimization*. The test suite minimization problem [1] can be formally stated as follows:

**Given.** A test suite T of test cases $\{t_1,t_2,t_3,\ldots..,t_m\}$, a set of testing requirements $\{r_1,r_2,r_3\ldots.,r_n\}$ that must be satisfied to provide the desired test coverage of the program, and subsets $\{T_1,T_2,..,T_n\}$ of T, one associated with each of the $r_i$'s such that any one of the tests $t_j$ belonging to $T_i$ satisfies $r_i$.

**Problem.** Find a minimal cardinality subset of T that exercises all $r_i$'s exercised by the unminimized test suite T.

The $r_i$'s can represent either all of the program's test case requirements or those requirements related to program modifications. A representative set of test cases that satisfies the $r_i$'s must contain at least one test case from each $T_i$. Such a set is called a hitting set of the group of sets $T_l$, $T_2$, . . . , T. A maximum

reduction is achieved by finding the smallest representative set of test cases. However, this subset of the test suite is the minimum cardinality hitting set of the T,'s and the problem of finding the minimum cardinality hitting set is NP-complete [2]. Therefore, since we are unaware of any approximate solution to the problem, we develop a heuristic [3], [4] to find a representative set that approximates the minimum cardinality hitting set.

The development team if able to find out redundant test case and eliminate them from the test case then the test suite size can be reduced. while finding the representative set the team must ensure that all test requirements are satisfied by the reduced test suite, to make testing more efficient. That is, given the original test suite T={$t_1$, $t_2$, $t_3$, ..., $t_n$} and a set of test requirements R={$r_1$, $r_2$, $r_3$, ..., $r_m$}, the goal is to find a subset of the test suite T, denoted by a representative set RS, to satisfy all the test requirements satisfied by T. The process of finding the representative set is called test suite reduction [5]-[8].

The organization of this paper is as follows. In section 2 related works is discuss followed by section 3 which contain test case reduction problem using ant colony optimization. The proposed model is discussed in section 4 and experimental result in section 5. In last section the findings of the paper are summarized.

## 2. REVIEW RELATED WORK

The Greedy algorithm [9], [10] removes the test case continuously. The algorithm stop when a representative set i.e RS which covers the entire requirement is derived. In Chen and Lau [11] algorithm choose all important test case first then apply greedy algorithm over the remaining test case for rest of test case selection [12] from that. In [5] Jeffrey and Gupta produce representative set for test suite reduction using selective redundancy. Harrold, Gupta and Soffa [1] find representative test cases for each subset and include them in the representative set. In [14] the authors use irreplaceability to evaluate the importance of tests and present an algorithm that ultimately produces reduced test suites with a substantially decrease in the execution cost. Using genetic algorithm in paper [13], [15]-[16] the authors are able to minimize test case which cover the entire requirement that can be covered by all the test cases. In [17], [18] Prasad and Mohapatra has proposed a genetic algoithm technique to find representative set. ACO use in wireless network gives a clear picture of using it in optimization problem by Dac-Nhuong Le [19] and Mina Jafari, Hassan Khotanlou [20].

## 3. TEST CASE REDUCTION PROBLEM USING ANT COLONY OPTIMIZATION

A test requirement matrix called as TR table is first created from the requirement and test case of asoftware. Test requirement table (TR) is a two dimensional 0-1 value table of size ( m * n). The test suite T={ t1, t2, t3 …..,tm} is represented in row and the requirement R={r1, r2,…..,rn} is represented in the column. That is each row of the table represent requirements fulfill by a particular test case. Entry into the TR table is determined by:

$$TR(i,j) = \begin{cases} 0 & if\ ti\ \ cannot\ satisfy\ \ rj \\ 1 & if\ ti\ satisfies\ rj \end{cases} \quad (1)$$

In Table 1 a test suite of four test case and their five requirements are given. Each test case is representing in row where as the requirement fulfilled by the test case are marked as 1 in the requirement column otherwise 0.

Table 1. An example of test case and requirements fulfill by it with execution time

| Test case | Requirements to be satisfied | | | | | Time |
|---|---|---|---|---|---|---|
| No | r1 | r2 | r3 | r4 | r5 | |
| t1 | 1 | 1 | 1 | 0 | 0 | 2 |
| t2 | 0 | 1 | 1 | 1 | 0 | 5 |
| t3 | 1 | 0 | 0 | 0 | 1 | 2 |
| t4 | 0 | 0 | 1 | 0 | 1 | 2 |
| t5 | 1 | 0 | 1 | 0 | 1 | 1 |

As per the TR table with m rows and n columns, it is essential to select a subset of rows to cover all of the columns in the matrix with minimal execution time. Suppose the vector element represents the row i in the vector x is selected and xi=0 means not, therefore, the set coverage problem can be represented as standard optimization problem:

$$\text{Min } z(x)=\sum_{i=0}^{n} C_i X_i$$
$$\text{S.t } \sum_{i=0}^{n} a_{ij} x_i \quad i \geq 1 \text{ ,i=1,2,3,4,……} \qquad (2)$$
(Ensure that every column is covered by at least one row)
$$xj \in \{0,1\} \text{ , j=1, 2, 3,……..}$$

The test suite reduction problem is converted to set coverage problem, and then converted to standard optimization problem. The idea of proposed algorithm start from this. It is an optimization algorithm that can use Ant colony optimization to solve this reduction problem.

The test case reduction problem can be model as a complete graph network $G(V, E)$ with E represents all the test case. The edges $e_{ij}$ represents a path from one test case to other. From table no 1 the generated complete graph is:

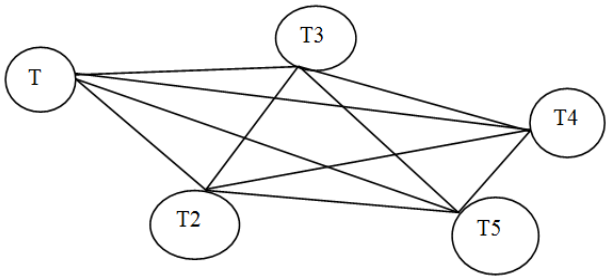


Figure 1. The test case reduction problem model by a complete graph

Ant algorithms use a group of artificial ant for optimum answer, out of all ant individual ant derived an entire answer in some steps. Temp answer is that the answer derived by ant k in some n steps. At every step every ant k computes a group of possible expansions to its current state and moves to at least one of those in chance.Each ant k moves from one vertex i to another vertex j with a transition probability rule p*kij*(t), which is described by the formula:

$$P_{ij} = \begin{cases} \dfrac{\left(\tau_{ij}^{\alpha}\mu_{ij}^{\beta}\right)}{\sum_{j\in N_i}\left(\tau_{ij}^{\alpha}\mu_{ij}^{\beta}\right)}, for\ j \in N_i \\ 0, \quad for\ j\ which\ not\ \in N_i \end{cases} \qquad (3)$$

The temp solution of the problem is a part of solution and the partial solution is a subset of vertices, which constitute a solution of the problem. Parameters α and β which is used in the transition probability rule pkij(t) expressed by (3), indicate about this, how important the pheromone trail τij and the attractiveness μij are during transition from one to another state. Values of these parameters α and β should be set by experiment and tuned to the test case reduction problem with minimum covering cost.

After a solution has been found each ant deposits a pheromone with a quantity Δτ on all vertices, which constitute the solution Vs, in accordance with the pattern:

$$\tau_{ij}(t) = \tau_{ij}(t) + \Delta\tau \qquad (4)$$

Thus these vertices which were included into a solution have received an additional quantity of a pheromone and can be chosen to a solution that would be constructed next with a higher probability than others vertices from the set VT.

An evaporation mechanism is incorporated into an ant algorithm in order to avoid a too fast convergence to a sub-optimal solution. An intensity of evaporation is controlled by a parameter ρ and a quantity of a pheromone on each vertex from the set VT is update at the end of each cycle in accordance with the pattern:

$$\tau_{ij(t)} = (1-\rho)\tau_{ij}(t), \rho \in [0,1] \qquad (5)$$

Thus a diversity of a solution is granted. Values of a parameter ρ should be set by experiment.

A quantity of deposited pheromone $\Delta\tau$ depends on a quality of solution Q and if the better is a solution than the more pheromone is deposited and in general can be stated as formula:

$$\Delta\tau = f(Q) \quad (6)$$

and in particular can be expressed by some specific formula, which take into account the covering cost.

### 3.1. Algorithm

```
begin
    while (All the requirement Covered) do
        for (k:=1 to n Ants) do
            while (A solution is not Complete ) do
            Update Available Vertices;
            Choose next vertex i with probability p(i) and consistency checking;
            Add to a Temp Solution;
            Update Temp Solution;
                If (requirement Covered)
                Return Best Solution Founded
                Terminate;
                end
            end
        Update Current Best Solution;
        end
    Update Best;
    Use an evaporation mechanism;
    Update Pheromone;
    end
    Return Best Solution Founded;
end
```

### 3.2. Theoretical Example

From Figure 1 let 5 ant start from five vertex represented in the figure, here no of vertex(n)= no of ant(k). In first execution let all the ant derived the following solution in two steps.

Table 2. An example of test case, requirements and it cost

| Ants | Test Case | Fault detected | Execution time |
|---|---|---|---|
| A1 | {T1-T2} | 4 | 7 |
| A2 | {T2-T3} | 5 | 7 |
| A3 | {T3-T5} | 3 | 3 |
| A4 | {T4-T1} | 4 | 4 |
| A5 | {T5-T4 } | 3 | 3 |

In the resut it is clearly visible that ant A2 covers all the requirement with execution time of the two test case 7 Sec. If it is a time constrained reduction then the algorithm can execute further for getting a result with less execution time otherwise we can stop our execution as all the requirement is fulfilled by ant A2.

## 4. OUR PROPOSED MODEL

Figure 2 describe the procedure of execution of ACO-Reduce algorithm. Before applying the five test suite reduction techniques, we collected the test case-requirement matrices from the previous execution of the test case T over program P. In case of regression testing the test cases T is reduce using ACO-Reduce and give reduce test cases T'. These test cases are run on the modified program P' in the maintenance stage.

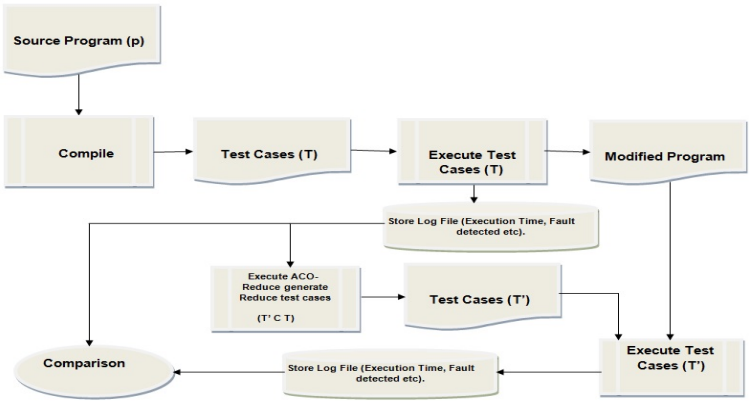


Figure 2. Model for execution of ACO-Reduce procedure

## 5. EXPERIMENT RESULTS

This technique is implemented using MATLAB which take the test suit pool T X R as input and using the above ant colony optimization technique find out a representative set, which is named as ACO-Reduce. The following four existing technique are also applied using MATLAB for comparison of result between our approach and the existing approach.

1) Harrold et al.'s heuristic

To make consistency with other researches [22], [23], we use 'GA' to denote Harrold et al.'s Heuristic [21]. The aim of this heuristic algorithm is to find the minimum size of representative set of the test suite. The basis of this algorithm is to find essential test cases, which are defined as those test cases that when removed, some test requirements can never be satisfied.

2) Chen and Lau's GRE heuristic

This heuristic algorithm is proposed by Chen and Lau in [22]. 'GRE' is used to denote their heuristic [22], [23]. The algorithm is based on the mix of three strategies: the greedy strategy, the essential strategy, and the 1-to-1 redundancy strategy.

3) Mansour and El-Fakin's approach

Genetic algorithms are based on the mechanism of natural evolution, where reproduction and selection operations are applied to populations over successive generations for evolving optimal solutions [24]. Mansour and El-Fakin [25] adapt the hybrid genetic algorithm to solve the test suite reduction problem. In this paper, we use 'MEF' to denote Mansour and El-Fakin's approach.

4) Black et al.'s approach

One recent strategy for test suite reduction is proposed by Black et al. [26], in which, two integer linear programming (ILP) models are provided. In this paper, we use 'BAA' to denote Black et al.'s approach.

All the implemented techniques were executed on a PC with an Intel Pentium 2.26 GHz CPU and 512 M memory running the Windows 2000 Professional operating system. Table 3 shows the details of subject programs and the collected test case-requirement matrices. Column 1 lists all the subject programs. Column 2 lists the number of lines of code (LOC) of each subject program. Column 3 lists the size of the corresponding subject program's test suite pool where T denotes the number of all the test cases and R denotes the number of test requirements. Five programs were studied, ranging from 1425 to 3095 lines of code(LOC). These five Java programs in our experiment are binary search tree (BST) with all operation and application , power equalizer (PEQ), transmission control (TC), stack implementation for job (STACK), stock index prediction (STOCK). The feature of these programs has been given in Table 3.

Table 3. Summary of programs used in experimentation

| Program | Source file (LOC) | Test suite pool (T X R) |
|---|---|---|
| **BST** | 1864 | 1694 X 983 |
| **PEQ** | 1456 | 674 X 124 |
| **TC** | 2987 | 2287 X 157 |
| **STACK** | 1425 | 719 X 70 |
| **STOCK** | 3095 | 3970 X 128 |

The result analysis is done basing upon the scalability and size of the representative set [27]. All the test case reduction technique is scale with the complexity of the test suit. To measure scalability these algorithm are implemented with test suit of different complexity and record their time. A complexity of test suit is

$$Complexity(t)=\log 10\ (m \times n) \quad (7)$$

In Equation (7), m is the number of test cases in the test suite (t), and n is the maximum number of test requirements that can be satisfied by t. Again we compare the size of representative set produce by all the algorithm using our selected program.

### 5.1. Scalability

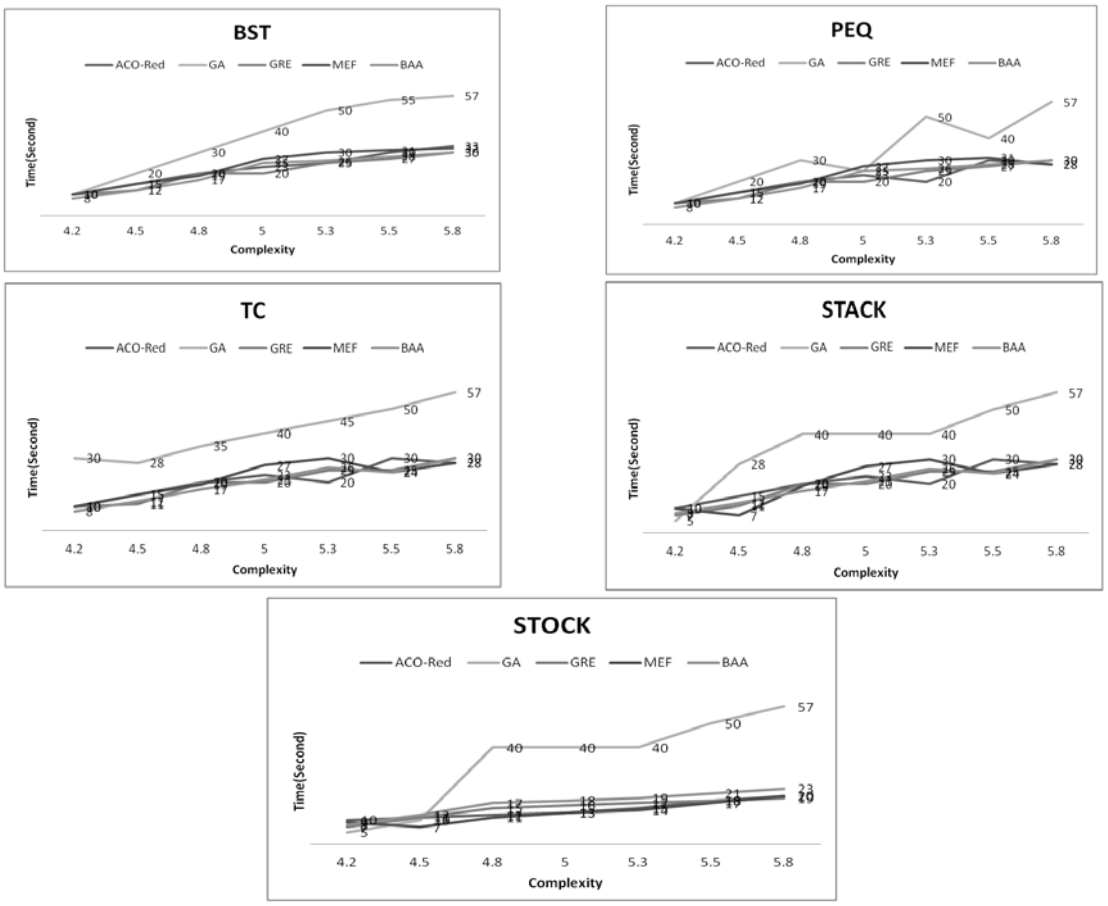


Figure 3. Scalability of ACO-Red, GA, GRE, MEF, BAA for 5 program

From Figure 3, we can observe ACO-Reduce needs the minimum time to calculate representative sets, while MEF and BAA takes approximately same time. GA algorithm takes more time than other in all of the comparison with different size of the test cases. Thus, the time efficiency of these four algorithms can be summarized as tACO-Red ≤ t GRE ≤ tMEF ≤tBAA ≤tGA . The complexity of the programs are calculated by Equation (7).

### 5.2. Representative Set Size

Figure 4 depicts the sizes of the representative sets generated by the five test suite reduction techniques for the different subject programs. In every single diagram in Figure 4, the horizontal axis denotes the test suite's size whereas the vertical axis denotes the size of representative set generated by the 5 test suite reduction techniques. Details of our experiment with the five programs using five different reduction algorithm are summarize in the Table 4.

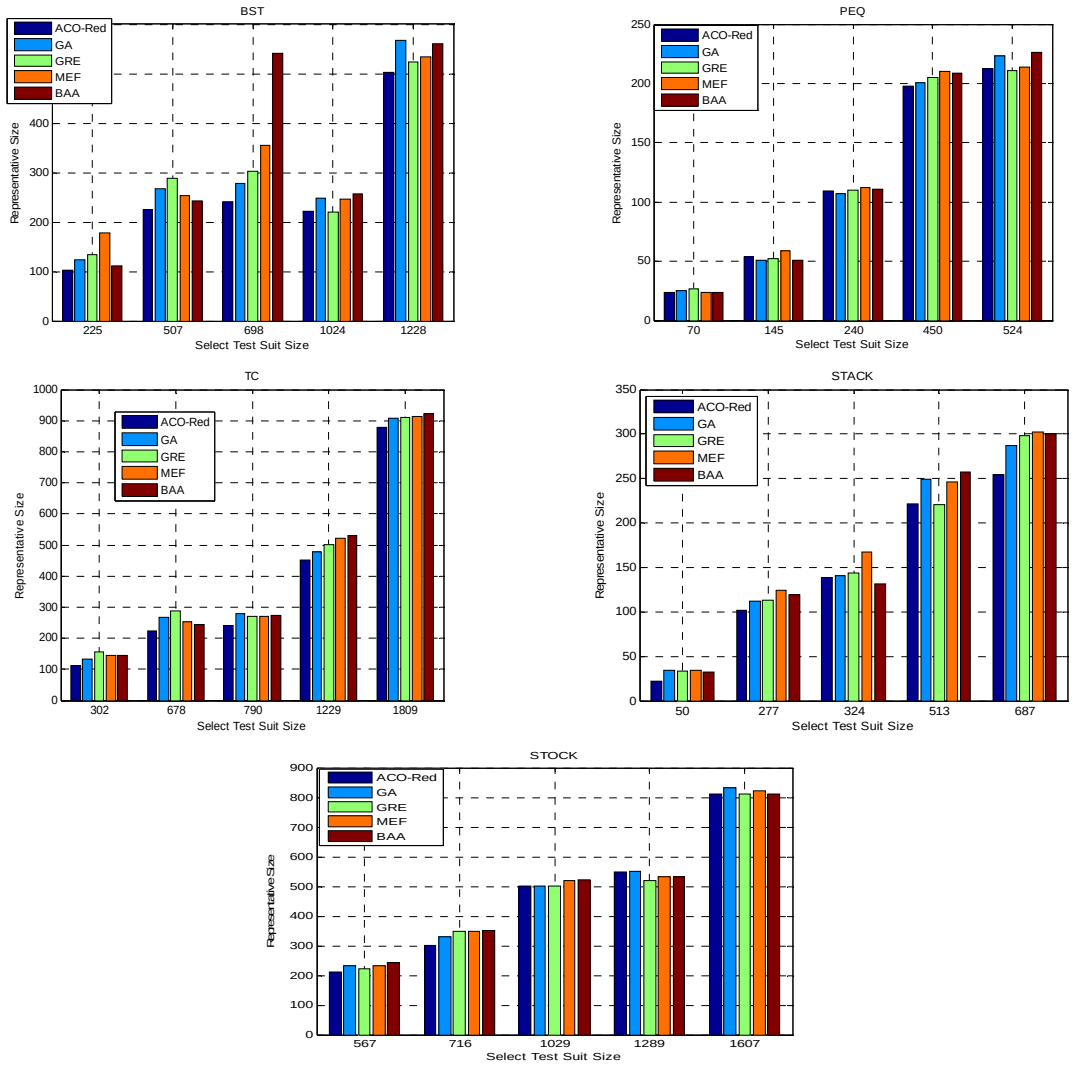


Figure 4. Sizes of representative sets of ACO-Red, GA, GRE, MEF, BAA for 5 program

Table 4. Summary of experiment done using five programs for different reduction algorithm

| Program / Reduction Technique | BST Test Case | BST RS | BST % Reduction | PEQ Test Case | PEQ RS | PEQ % Reduction | TC Test Case | TC RS | TC % Reduction | STACK Test Case | STACK RS | STACK % Reduction | STOCK Test Case | STOCK RS | STOCK % Reduction |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ACO-Reduce** | 1694 | 720 | 58 | 674 | 456 | 33 | 2287 | 1567 | 32 | 719 | 537 | 26 | 3970 | 2672 | 33 |
| **GA** | 1694 | 756 | 56 | 674 | 456 | 33 | 2287 | 1608 | 30 | 719 | 589 | 19 | 3970 | 2684 | 33 |
| **GRE** | 1694 | 810 | 53 | 674 | 467 | 39 | 2287 | 1582 | 31 | 719 | 537 | 26 | 3970 | 2701 | 32 |
| **MEF** | 1694 | 721 | 58 | 674 | 487 | 28 | 2287 | 1601 | 30 | 719 | 565 | 22 | 3970 | 2672 | 33 |
| **BAA** | 1694 | 734 | 57 | 674 | 478 | 30 | 2287 | 1609 | 30 | 719 | 581 | 20 | 3970 | 2679 | 33 |

## 6. CONCLUSION

In this paper an algorithm for test cases reduction is presented and implemented. It is compared with our other technique [17], [28]. It finds out representative set of the test case from the given set of test case. It

uses a simple ACO method to reduce the test case in regression testing. Moreover, the generated test suite is minimized greatly. Therefore it can reduce test cost of regression testing and improve the efficiency of the software with the optimized test suite. We have evaluated the effectiveness of our proposed regression test case reduction technique using several moderate sized objected oriented Java programs. It is observer from the experiment that the ACO-Reduce algorithm show promising results in terms of execution time as compared with other reduction algorithm. ACO-Reduce algorithm reduce the test case 10% effectively then other algorithm and its execution time is faster when compare with other algorithms.

## REFERENCES


[1] M. J. Harrold, R. Gupta, and M. L. Soffa, "A Methodology for Controlling the Size of a Test Suite", *ACM Trans. Software Eng. And Methodology,* Vol. 2, No. 3, pp. 270-285, 1993.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms", second ed. MIT Press, 2001.

[3] GUPTA R., "A reconfigurable LIW architecture and its compiler", Tech. Rep. 87-3, Dept. Computer Science, Univ. Pittsburgh, Pittsburgh, Pa., 1987.

[4] Gum A. R., and Soffa M. L., "Compile-time techniques for improving scalar access performance in parallel memories", *IEEE Trans. Parallel and Distributed Systems 2,* pp. 138-148, 1991.

[5] D. Jeffrey, and N. Gupta, "Improving Fault Detection Capability by Selectively Retaining Test Cases During Test Suite Reduction", *IEEE Trans. on Software Engineering,* Vol. 33, No. 2, pp. 108-123, 2007.

[6] J. W. Lin, and C. Y. Huang, "Analysis of Test Suite Reduction with Enhanced Tie-Breaking Techniques", *Information and Software Technology,* Vol. 51, No. 4, pp. 679-690, 2009.

[7] M. R. Garey, and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman and Company, 1979.

[8] R. M. Karp, "Reducibility among Combinatorial Problems", *Complexity of Computer Computations, Plenum Press,* pp. 85-103, 1972.

[9] V. Chvatal, "A Greedy Heuristic for the Set-Covering Problem", *Mathematics Operations Research,* Vol. 4, No. 3, pp. 233-235, August 1979.

[10] S. Yoo, and M. Harman, "Regression Testing Minimization, Selection and Prioritization: a Survey", *Software Testing, Verification and Reliability,* Vol. 22, No. 2, 2012.

[11] T. Y. Chen, and M. F. Lau, "A New Heuristic for Test Suite Reduction", *Information and Software Technology,* Vol. 40, No. 5-6, pp. 347-354, 1998.

[12] J. A. Jones, and M. J. Harrold, "Test-Suite Reduction and Prioritization for Modified Condition/Decision Coverage", *IEEE Trans. on Software Engineering,* Vol. 29 No. 3, pp. 195-209, 2003.

[13] Ma X. Y., He Z. F., Sheng B. K., Ye C. Q., "A genetic algorithm for test-suite reduction", In*: Proc. the International Conference on Systems, Man and Cybernetics,* pp. 133–139, 2005.

[14] Chu-Ti Lin, Kai-Wei Tang, Cheng-Ding Chen, and Gregory M. Kapfhammer, "Reducing the Cost of Regression Testing by Identifying Irreplaceable Test Cases", *In Proc. Of the 6th ICGEC '12.*

[15] Y. Zhang, J. Liu, Y. Cui, X. Hei, "An improved quantum genetic algorithm for test suite reduction", *IEEE International Conference on Computer Science and Automation Engineering (CSAE),* 2011.

[16] Dan Hao, Tao Xie, Lu Zhang, XiaoyinWang, Jiasu Sun, Hong Mei, "Test input reduction for result inspection to facilitate fault localization", *Automated Software Engineering*, Vol. 17, No. 1, pp 5-31, 2010.

[17] S. K. Mohapatra, S. Prasad, "Minimizing Test Cases to Reduce the Cost of Regression Testing", *Proceedings of the 8th INDIACom,* 2014.

[18] S. K. Mohapatra, S. Prasad, "Evolutionary search algorithm for Test Case Prioritization", *2013 International Conference on Machine Intelligence Research and Advancement.*

[19] Dac-Nhuong Le, "GA and ACO Algorithms Applied to Optimizing Location of Controllers in Wireless Networks", *International Journal of Electrical and Computer Engineering (IJECE),* Vol. 3, No. 2, pp. 221-229, 2013.

[20] Mina Jafari, Hassan Khotanlou, "A Routing Algorithm Based on Ant Colony, Local Search and Fuzzy Inference to Improve Energy Consumption in Wireless Sensor Networks", *International Journal of Electrical and Computer Engineering (IJECE),* Vol. 3, No. 5, pp. 640-650, 2013.

[21] M. J. Harrold, R. Gupta, M. L. Soffa, "A methodology for controlling the size of a test suite", *ACM Transactions on Software Engineering and Methodology*, Vol. 2 No. 3, pp. 270–285, 1993.

[22] T. Y. Chen, M. Lau, "A new heuristic for test suite reduction", *Information and Software Technology*, Vol. 40, No. 5-6, 347–354, 1998.

[23] T. Y. Chen, M. Lau, "A simulation study on some heuristics for test suite reduction", *Information and Software Technology*, Vol. 40, No. 13, pp. 777–787, 1998.

[24] D. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, 1989.

[25] N. Mansour, K. El-Fakih, "Simulated annealing and genetic algorithms for optimal regression testing", *Journal of Software Maintenance*, Vol. 11, No. 1, pp. 19–34, 1999.

[26] J. Black, E. Melachrinoudis, D. Kaeli, "Bi-criteria models for all-uses test suite reduction", *In: Proceedings of 26th International Conference on Software Engineering, IEEE Computer Society,* Washington, DC, USA, pp. 106–115, 2004.

[27] H. Zhong, L. Zhang, and H. Mei, "An Experimental Study of Four Typical Test Suite Reduction Techniques", *Information and Software Technology,* Vol. 50, No. 6, pp. 534-546, 2008.

[28] S. K. Mohapatra, S. Prasad, B. P. Kar, "Test Suit Reduction By Finding Cost Optimal Representative Set", *International Journal of Advanced Technology & Engineering Research (IJATER),* Vol. 4, No. 3, 2014.

## BIOGRAPHIES OF AUTHORS

Sudhir Kumar Mohapatra an M.Tech(Computer Science) holder from Utkal University is currently persuing P.hD from SOA University,Odisha, India in the department of Computer Science & Engg.

Srinivas Prasad has done his PhD in Computer Science ,UU, Orissa. He has 20 years of experience in industry as well as institution. Currently he is working as professor and Heads of Department in Dept. of Computer Science &Engineering, GMRIT, Andhra Pradesh, India.