Implementing Test Case Selection and Reduction Techniques using Meta-Heuristics

Reetika Nagar[1], Arvind Kumar[2], Sachin Kumar[1], Anurag Singh Baghel[1]

[1]School of Information & Communication Technology
Gautam Buddha University, Greater Noida, India

[2]Pitney Bowes Software
Noida, India

reetikanagar@gmail.com, arvind.jki@gmail.com, sachinbhati03@gmail.com, anuragsbaghel@yahoo.com

***Abstract*—Regression Testing is an inevitable and very costly maintenance activity that is implemented to make sure the validity of modified 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 to select and prioritize a minimum set of test cases, fulfilling some chosen criteria, that is, covering all possible faults in minimum time and other. In this paper a test case reduction hybrid Particle Swarm Optimization (PSO) algorithm has been proposed. This PSO algorithm uses GA mutation operator while processing. PSO is a swarm intelligence algorithm based on particles behavior. GA is an evolutionary algorithm (EA). The proposed algorithm is an optimistic approach which provides optimum best results in minimum time.**

Keywords— Particle Swarm Optimization; Genetic Algorithm; Regression Test Selection; Test Case Prioritization

I. INTRODUCTION

Software maintenance [1, 2, and 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 environments 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 meta-heuristic approaches to achieve good quality optimal results. Its main goal is to explore large search space to get best optimal solutions. Meta-heuristic 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 meta-heuristics approaches used to choose minimum set of test cases that covers all the faults and bugs in minimum time is given. In this paper, we have proposed a hybrid algorithm for reduction of test cases using meta-heuristics – PSO and mutation operator of GA. 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.

Other sections of this paper have been organized in following manner. Section II discusses about regression testing and its techniques used in this paper. Section III gives a brief idea about the meta-heuristics algorithms used such as particle swarm optimization and genetic algorithm. Section IV discusses about proposed algorithm. In section V implementation of proposed algorithm has been done using some problem or software system example. In Section VI results has been shown. In Section VII conclusions and the future work is given.

II. REGRESSION TESTING AND ITS TECHNIQUES

One of the main reasons to introduce regression testing is to establish whether a change 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 using particle swarm optimization has been ordered in test case prioritization technique depending on their fault coverage efficiency in minimum time.

III. Meta-heuristic Algorithms Used

Meta-heuristic algorithms [9] 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. Algorithms used here have been explained below.

A. Genetic Algorithm

Genetic Algorithm (GA) is an optimization technique which can be used in various problems, including those that are NP-hard [10, 11].

Two main requirements which are to be satisfied before applying GA are:

1. An encoding is used to represent a solution in the solution space, and
2. An objective function i.e. a fitness function which evaluate the effectiveness and goodness of a solution either absolute or relative [12].

Operators of genetic algorithm are selection, crossover and mutation. These operators are used to generate the second generation population of solutions from first initial generation. In this proposed work genetic algorithm mutation operator has been used to select the best and superior population of test cases.

B. Particle swarm optimization

Particle swarm optimization (PSO) [13, 14] is one of the evolutionary computation techniques. Each particle in PSO is connected with a velocity. Particles fly through the search space with velocities which are dynamically adjusted according to their historical behaviors. Therefore, the particles have likelihood to fly towards the best search area over the course of search process.

The PSO algorithm keeps track of three global variables:

- Target value
- Global best (gBest) value shows which particle's data is currently closest to the Target
- Stopping value determines when the algorithm should stop if the Target isn't detected.

Each particle in PSO algorithm consists of:

- Data representing a possible solution
- A Velocity value indicating how much the Data can be changed
- A personal best (pBest) value indicating the closest the particle's Data has ever come to the Target.

In the proposed algorithm, PSO has been used for selection and prioritization of test cases to get optimal results in minimum execution time.

IV. 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 PSO and mutation operation of GA. This algorithm selects test cases from the given test suite that will cover all possible faults detected in minimum execution time. Here particles are used as agents to traverse the minimum set of test cases.

In PSO Algorithm, during search of best solution, particles can converge to point between particle best and neighbor best in the search space, thus particle converge towards single point, ignoring global best conditions. To overcome this limitation GA mutation function has been used along with PSO algorithm to propose hybrid PSO algorithm. Proposed hybrid PSO algorithm is used to reduce the number of test cases covering all possible faults in minimum execution time. Regression testing selection and prioritization techniques are used in proposed algorithm to reduce the number of test cases. In the proposed algorithm, PSO (finding all possible faults of given test cases in minimum execution time) using knapsack method criteria [15] (target limit of execution time is fixed) has been implemented along with GA mutation function. 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. GA mutation function proposed is used for updating test case sequence and generating new sequences for particle traverse to get best optimal solution. Thus, hybrid PSO algorithm 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_n}.
2) The fitness function and stopping criteria of particles vary according to problem.
3) In proposed problem,
   - Fitness function of particle is 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,…,tc_n) covers some or all faults.
5) Set of faults is represented as F= {f_1, f_2,…,f_n}.
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 $1 \leq i \leq n$.
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 'p' number of particles, 'n' number of Test Case and 'm' possible number of faults to be detected.
11) Output is the sequence of traversing test cases and fulfilling stopping criteria.

B. Proposed Hybrid PSO Algorithm

```
1)  Generate population of 'p' particles and 'n' test cases
2)  Initialization of particles
     For each 'p' particle
     {For each 'n' test case
    {Initialize position}
     Get total execution time}
3)  Implementation
    {For each 'p' particle
    {For each 'n' test case
    Select and sort the test cases
    }
    }
4)  Updation
    For each 'n' particle
    {Calculate velocity of each particle.
    Update position or sequence of each particle
    }
5)  Examine
    Check each particle.
    If (any stopping criteria satisfies)
     {STOP
    Get optimal sequence of processing test cases for
    particles}
     Else
     Go to Step 3.
6)  Calculate fitness value of optimal sequence having reduced
    number of test cases
7)  End
```

C. Description of Proposed Algorithm

1) Generate test case population pool
- Define 'p' particles and 'n' test cases with numerical value of faults. For example, Let n=5 or any value. There are 5^5 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. 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) Initialization of particles –
- For each particle form data sequences of test case coordinates. Randomly chose test case sequence among population.
- Once the particle data, that is, possible solution is determined. Find its fitness value, that is, total execution time of pattern.

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 fitness value of each particle data determine particle pBest and gBest. pBest is the previous best value of particle. Initially pBest is considered to be the maximum execution time of all the test cases. If fitness value of particle is better, that is, less than its pBest then,

  Set pBest ← current fitness value

  Best is the global best value of particle. Initially gBest is considered to be null. If gBest is less than pBest then,

Set gBest ← pBest

- Once the global best value of all the particles is determined. Prioritize the particles using sorting particles by their gBest scores, best to worst.

4) Updating-

- Calculate velocity of particles- The velocity score is calculated using the global worst (after sorting, worst will be last in list), defining velocity as the measure of how bad each particle is doing (as opposed to how good). Velocity for different problems is determined differently considering standard velocity equation [16, 17].

$$\text{velocity} = \frac{(\text{V\_MAX} * \text{particles.get(i).gBest()})}{\text{worstResults}}$$

Where

worstResults is the worst fitness value ,that is, maximum execution time of particles of the considered problem.

- Updating Position- Particle data is updated by swapping index value within each particles own data set. The amounts of swapping depend on how bad it's doing (i.e. velocity).

5) Examine-

- If (any stopping criteria satisfies)

  Stop the algorithm and apply collection rule, that is, collect all updated position particles and determine best solution, that is, test case sequence. Else repeat step 3.

- For all the best data sequence or pattern of particles achieved determine their optimal solution and find its fitness value. Optimal route or sequence are achieved as-
  - Randomly chose one best particle sequence and find optimal route or sequence to traverse the test cases. Number of test cases to be traversed by a particle depends on following condition: (fault_count[i+1]>fault_count[i]). Fault_count[i+1] Fault_count[i+1] consists of faults of both index 'i' and 'i+1', achieved using GA mutation function by 'OR' operator. If the above condition satisfies add test cases 'i+1' index in particle data or test cases sequences.
  - Find execution time of best optimal solution

6) Calculate fitness value of optimal sequence having reduced number of test cases.
Fitness value of best test case sequence/route =

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

7) 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 hybrid PSO-GA algorithm

V. Implementation

The hybrid PSO algorithm has been implemented using JAVA. The algorithm takes input as different number of test cases, their faults count, number of particles for traversing the test cases and determining the best processing sequence of minimum test cases selected, maximum epochs and target value for estimating the stopping criteria while processing particle swarm optimization algorithm 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 different problems or software systems test suite, covering total faults varying from 1 to 10 and number of particles 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 different problems. Dataset of test case has been assumed randomly and has not been referred from any prescribed problem.

A. Problem Example

In 1st problem considered number test cases is 6, faults count is 5 and number of particles is 8.

<u>Implementation-</u>

Inputs: Number of particles: 8
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 of Faults of 1st Problem

S.No	Faults of Test Cases	Faults Count
0	11000	2
1	00010	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 00010

Test case 2th is 10000

Test case 3th is 10111

Test case 4th is 11001

Test case 5th is 10100

After execution of proposed hybrid PSO-GA 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 80.1641 unit time
- Best shortest route of traversing test cases is TC0, TC4, TC3 covering all possible faults in 47.5965 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 PSO-GA is approximately 50%.

VI. Results

The results obtained in section V 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 hybrid PSO algorithm has been represented graphically using MATLAB. Graphical representation of problems implemented in above section is given below:

A. *Problem Example*

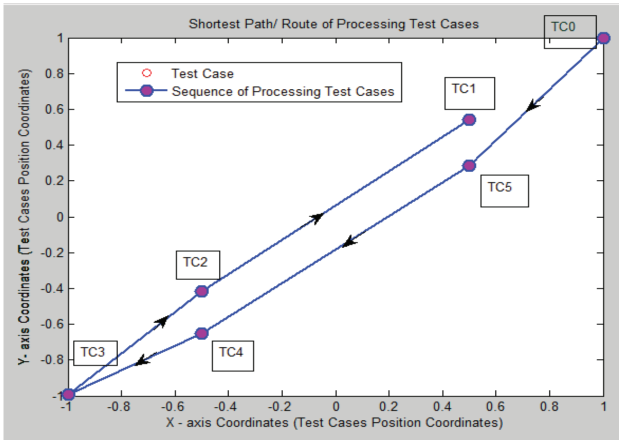

Fig. 1. Graphical Representation of Path/Route of Processing Test Cases for first problem

Execution time of processing test cases is 80.1641 unit time

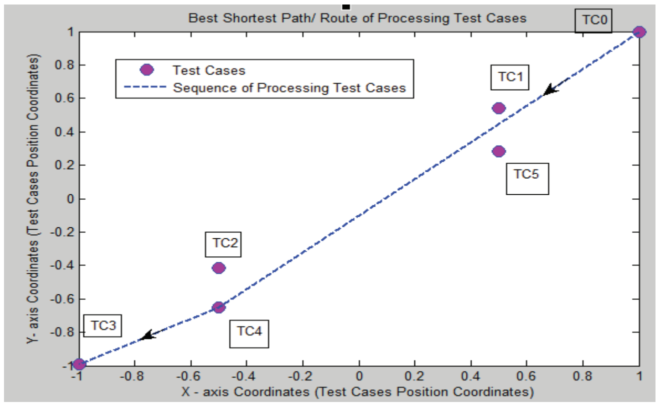

Fig. 2. Graphical Representation of Best Shortest Path/Route of Processing Test Cases for first problem

Execution time of processing test cases is 47.5965 unit time

VII. Conclusions and Future Scope

We have proposed a novel hybrid PSO algorithm for selection and prioritization of test cases from a given test suite using particle swarm optimization and genetic algorithms. This approach has been tested for several problems or examples. One of these problems 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 PSO-GA is approximately 50%. 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 meta-heuristic 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, 2009, pp. 5-5.

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

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

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

[12] H. Zakir Ahmed, “Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator”, International Journal of Biometrics & Bioinformatics (IJBB), vol. 3, no. 6, pp. 96-105, 2010.

[13] Kennedy, J., and Eberhart, R. C., “Particle swarm optimization”, in Proc. of IEEE International Conference on Neural Networks (ICNN), vol.4, pp.1942-1948, 1995.

[14] Kennedy, J., and Eberhart, R. C., “A discrete binary version of the particle swarm algorithm”, in Proc. Conf. on Systems, Man, and Cybernetics, Piscataway, NJ: IEEE Service Center, 1997, pp. 4104-4109.

[15] Liang,Y, Liu,L., Wang,D.,Wu, R., 2010, “Optimizing Particle Swarm Optimization to Solve Knapsack Problem”, ICICA, CICIS, vol.105, Springer, Berlin, pp. 437-443.

[16] R. Poli, J. Kennedy, T. Blackwell and A. Freitas, guest editors, Special Issue on “Particle Swarm Optimisation: the second decade” of the Journal of Artificial Evolution and Applications, February 2008.

[17] Riccardo Poli, James Kennedy, Tim Blackwell, “Particle Swarm Optimization”, Swarm Intelligence 1(1), pp. 33-57, 2007, Springer, LLC 2007