Metaheuristic Based Approach to Regression Testing

Shweta Mittal, Om Prakash Sangwan

Deptt. Of Computer Science,
Guru Jambeshwar University of Science & Technology
Hisar, India

***Abstract*— Regression testing is a very expensive and time consuming process as there may be insufficient resources to re-execute all the test cases in resource and time constrained environment. It acquires a lot of human effort, if done manually. Lot of techniques have been reported on how to select regression tests so that the number of test cases do not boast up too high as the software evolves. The techniques include regression test selection, test minimization and test prioritization. In this paper, various metaheuristic approaches have been studied to examine their potential benefits to regression testing. The paper also addresses the problem of choice of fitness metric and determination of the most suitable search technique to apply. Genetic algorithm performs well, although Greedy approaches are surprisingly effective, given the multimodal nature of the landscape. It may be accomplished that Cuscuta ordering which is inspired by intelligent behaviour of plants gives same results as given by the optimal and ACO ordering but better than unordered, random and reverse order. The study also reveals that ABC outperforms the other approaches i.e. GA, ACO, BCO and PSO in test suite optimization process as parallel behaviour of the bees is used to reach the solution generation faster.**

***Keywords*— Metaheuristic, Regression Testing, Test Case prioritization.**

I. Introduction

Software testing is one of the important process in Software Development Life Cycle. As specified by Glen Myers," Testing is a process of executing a program with the intent of finding an error". It typically consumes at least 50% of the total cost involved in software development [28]. Test suite is a collection of test cases that is used to test a software program to show that it behaves as expected. Test suite optimization is a process of generating effective test cases in a test suite that can cover the given SUT (System Under Test) within less time. Regression testing is usually performed during the maintenance phase whenever any changes are made to the software in order to ensure that no new defects are introduced and also no old defects have regressed. Various techniques have been proposed by researchers on regression test case selection, test case minimization and test case prioritization using metaheuristic techniques. The emerging area in this field is swarm intelligence. Bonabeau has defined the swarm intelligence as ''. . .any attempt to design algorithms or distributed problem-solving devices inspired by the collective behaviour of social insect colonies and other animal societies. . ." [30]. In this paper, we have explored the utilization of metaheuristic based approach to regression testing.

A. Regression Testing

Maintenance is required when some of the components of software need to be replaced, software is upgraded or enhanced to include additional features and to provide more services. Regression testing is defined as "the process of retesting the modified parts of the software and ensuring that no new errors have been introduced into previously tested code" [1]. Testers might re-run all test cases generated at earlier stages to ensure that the program behaves as expected. However, as the software evolves, test suite grows larger for which it is practically impossible to retest all the test cases. Effective regression testing is a trade off between number of regression test cases and the cost of regression testing.

There is a need of some approach for selecting the subset of test suite which can reveal the maximum faults within lesser time and the approach chosen will depend on the problem in hand. If the code of software is available, then white box testing techniques i.e. path coverage, code coverage, statement coverage , branch coverage etc. are used, while if only functionalities are available, then Boundary Value Analysis (BVA) , Equivalence Partitioning , all pair testing etc. black box techniques are used. Average Percentage of Fault Detected (APFD) is most frequently and widely used metric now-a-days which can be calculated as following [31]:

$$\mathbf{APFD = 1 - \frac{TF_1 + TF_2 + + TF_m}{n.m} + \frac{1}{2n}}$$

where, T - The test suite under evaluation
m - The number of faults contained in the program under test P
n - The total number of test cases in and
TF_i - The position of the first test in T that exposes ith fault.

Exhaustive testing a program's input is infeasible for any reasonably-sized program [7]. The number of possible tests for even simple software components is practically infinite and execution of all the test cases will require a large amount of human effort and time. The various techniques which researchers have considered for minimizing the cost of regression testing includes regression test selection, regression test minimization and regression test prioritization. Therefore, the objective of our paper is to find the efficient test cases from the test suite which covers the maximum faults or functionalities, minimize redundancy and execution time etc. by means of the available techniques so that the cost can be minimized.

Regression test selection techniques reduce the cost of regression testing by selecting an appropriate subset of test cases from the existing test suite based on information about the program, modified version and test suite [3]. We can further reduce the size and time required to execute a test suite by test suite minimization and test suite prioritization techniques. Test suite minimization techniques lower the cost by reducing a test suite to a minimal subset that maintains equivalent coverage of the original test suite with respect to a particular test adequacy criterion[2]. This can be achieved by removing the redundant test cases so that minimized test suite cover the same fault or functionalities but within the lesser time or resources or by removing the test cases for which the functionalities has been removed. The main disadvantage behind test case selection and minimization is that it may discard the necessary test cases.

Test Suite Prioritization techniques order the test cases such that higher priority test cases according to some criteria are executed prior to the ones with the lower priority. For example, testers might wish to run test cases with the higher coverage prior to those having the lower coverage. So, none of the test case is discarded and the user can execute the test cases according to available time and resources. Test case prioritization techniques have its very good application in time constrained environment. Researchers have formally defined the test case prioritization problem as follows:

"Definition: The Test Case Prioritization Problem : Given: T, a test suite, PT, the set of permutations of T, and f, a function from PT to the real numbers.
Problem: Find T′ € PT such that (¥ T″) (T″ €PT)(T″≠ T′)[f(T′) ≥f(T″)].

In this definition, PT represents the set of all possible permutations of T and f is a function that applied to any such ordering yields an award value for that ordering."[4]

Test case prioritization is basically of two types: Version Specific and General test case prioritization. Version-specific test case prioritization, given a program P and test suite T, test suite will be prioritized only for use on a specific version of P whereas in general prioritization, it will be useful over the successive versions of P. Test case prioritization is very effective in time constrained environment in which we have limited time to re-execute all the test cases. Test suite prioritization in general is an NP complete problem and to reduce it into a polynomial time, authors have used time based constraint [19].

There can be many goals for prioritization:

- To maximize the rate of fault detection.
- Maximize the coverage
- Increase the confidence in the reliability of the SUT in less time.
- Increase the likelihood of revealing faults etc.

The different techniques which are commonly used in test suite prioritization are as follows:

- Random ordering: In this technique, test cases are randomly ordered in a test suite.

- Statement coverage : Test cases are prioritized in terms of the total number of statements covered. This can be achieved by counting the amount of statements covered by each test case and sorting them in the descending order .When multiple test cases cover the same number of statements, an additional rule is necessary to order the test cases(they can be ordered randomly) .

TABLE I
TEST SUITE STATEMENT COVERAGE

Test Case	Statements covered
T1	**S2,S3**
T2	**S1,S4,S6,S7**
T3	**S4,S5,S6**

Table 1 shows test suite statement coverage indicating the order of test cases as follows : T2,T3,T1

- Additional statement coverage: Highest priority is given to the test cases which cover the maximum number of statements, the next priority to the test cases with maximum statements not previously covered and so on. For test suite specified in Table 1, ordering will be T2,T1,T3.

- Branch Coverage: It is similar to the statement coverage with the difference that coverage is measured in terms of program branches, that is, test case which covers the maximum branches is given the higher priority and so on.

- Additional branch coverage: Similar to additional statement coverage, except the coverage is measured in terms of number of branches rather than the number of statements.

Besides this, there are several other methods like Fault Exposing Potential (FEP), Additional FEP etc. which we have not included in our study .Test suite minimization and Test suite prioritization can also be used in combination i.e. first minimized test suite is obtained by removing the redundant test cases or those which are not needed and then assigning priority to test cases in minimized test suite.

II. Metaheuristic Techniques

Metaheuristic is a higher-level procedure designed to generate a lower-level procedure or heuristic (partial search algorithm)[32] . These techniques provides us with the greater advantage of lesser execution time while making some negotiation on the solution obtained i.e. it provides the near optimal solution. They grant sufficiently good solution to an optimization problem especially with partial or inadequate information or limited computation capacity. Here, "heuristic" means search, that is, there is some learning component which is guiding the whole search process. Metaheuristic techniques are usually non-deterministic i.e. given the same initial population, they will lead to different results, but the results are in close proximity to each other. Techniques which compose metaheuristics range from simple local search procedure to complex learning processes[33]. Based on our literature work in this section, we have done the classification of metaheuristic techniques as shown in Figure 1.

.

A. Classification of Metaheuristic Techniques

The techniques can be classified into different groups depending on the criteria being considered, such as population based, iterative based, stochastic, deterministic, etc [5]. Population based approach maintain and improve multiple candidate solution, often using population characteristics to guide the search. Important groups of population based algorithms are Evolutionary Algorithms (EA) and Swarm Intelligence (SI) based algorithms.

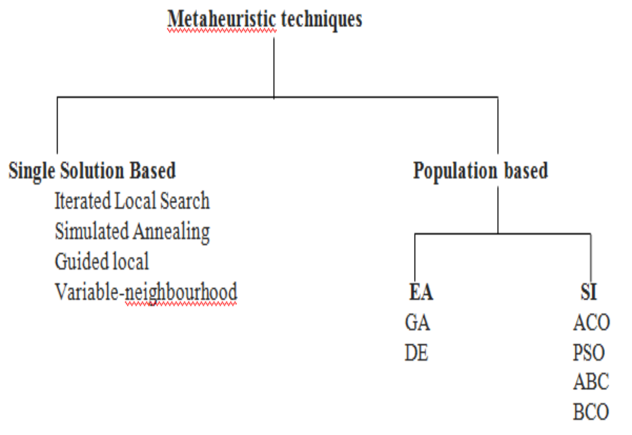

Fig. 1 Classification of Metaheuristic Techniques

where , GA: Genetic Algorithm , DE : Differential Evolution , ACO : Ant Colony Optimization, PSO: Particle Swarm Optimization, ABC: Artificial Bee Colony, BCO: Bee Colony Optimization .

The most popular EA is Genetic Algorithm (GA). GA was invented by John Holland in 1960 which attempts to simulate the Darwinian theory of natural evolution[8]. As shown in Figure 2, an iterative process is executed, initialized by a randomly chosen population. The iterations are called generations and the members of the population are called chromosomes. The process terminates when a population satisfies some pre-determined condition (or a certain number of generations have been exceeded). On each generation, some members of the population are recombined, crossing over elements of their chromosomes. A fraction of the offspring of this union are mutated and, from the offspring and the original population a selection process is used to determine the new population. Crucially, recombination and selection are guided by the fitness function; fitter chromosomes having a greater chance to be selected and recombined. Lower probability for mutation and higher for crossover helps to achieve right balance between the exploration and exploitation. GA has been extensively used by researchers for regression test case prioritization [24,25].

```
1. Randomly initialize population(t)
2. Determine fitness of population(t)
3. Repeat
       i.  Select parents from population(t)
       ii. Perform crossover on parents creating
           population(t+1)
       iii. Perform mutation of population(t+1)
       iv. Determine fitness of population(t+1)
4. Until best individual is good enough.
```

Fig. 1 Pseudo code of GA

Differential Evolution is the variant of GA. In DE algorithm, all solutions have an equal chance of being selected as parents independent of their fitness values. The use of a mutation operation, which has the self-adaptability feature, a crossover operation and a greedy process for the selection, makes DE a fast converging evolutionary algorithm. Moreover , DE does not face any Hamming Cliff Problem.

The term swarm is used for the group of interactive agents (generally small in size) which possess some special behavior like bees swarming around their hive, ants following a trail, flock of birds travelling together forming a particular shape etc in the search of food. Swarms locally interact with each other and with the environment following a decentralized approach. Ant Colony Optimization (ACO), Bee Colony Optimization(BCO), Artificial Bee Colony Optimization (ABC), Particle Swarm Optimization(PSO) are examples of Swarm Intelligence.

ACO was initially proposed by Marco Dorigo in 1992. It is based on the behaviour of ants in search of food sources. Artificial ants leave a virtual trail accumulated on the path segment they follow. The path for each ant is selected on the basis of the amount of "pheromone trail" present on the possible paths starting from the current node of the ant. Over time, the pheromone trail starts to evaporate, thus reducing its attractive strength. Pheromone evaporation has the advantage of avoiding the convergence to a locally optimal solution. Updating the trail is performed when ants either complete their search or get the shortest path to reach the food source. The pseudo code of ACO is presented in Figure 3.

```
procedure ACO
  while( termination condition is not satisfied)
    generateSolutions()
    daemonActions()
    pheromoneUpdate()
  end while
end procedure
```

Fig. 3 Pseudo code of ACO

Various authors implemented ACO for test case selection and prioritization [14,34].

BCO is a population-based search algorithm which was developed in 2005 and it has been used for regression test case prioritization ([11],[20]). It mimics the food foraging behaviour of honey bee colonies. In this algorithm, there are 2 types of bees i.e. scout bees and forager bees. The scout bees move randomly in an area surrounding the hive in search of available food sources. They continue with the exploration process and return back to the hive until they are exhausted (as shown in step 2 of Figure 4). After returning to hive, they deposit the food harvested and perform the waggle dance. By observing the dance, the forager bees learn the steps performed by scout bees and move to the food sources as guided by the scout bees. The forager bees continue then with the path exploitation.

```
1. Initialize bee population.
2. For each Scout Bee Si
       Perform exploration process (until some
       condition met)
3. For each Forager Bee Fi
       Perform exploitation process (until some
       condition met)
4. Evaluate all solutions and find the best one.
```

. Fig. 4 Pseudo code of BCO

The idea of PSO was given by John Kennedy and Eberhert in 1995. PSO is a population-based stochastic optimization technique. It consists of swarm of particles i.e. fishes, birds etc. moving in a search space of possible solutions for a problem. Every particle has a position vector representing a candidate solution to the problem and a velocity vector. Position here corresponds to rank of test case in a test suite and the velocity to execution time or coverage respectively. Moreover, each particle contains a small memory that stores its own best position seen so far and a global best position obtained through communication with its neighbour particles. The pseudo code of PSO is presented in Figure 5.

```
1.  Generate population of 'n' number of particles
2.  For each n particle
        i.  Initialize position
        ii. Initialize velocity
    End for
3.  Identify best particle from the population.
4.  For each n particle
        i.  Update position and velocity of particle
            w.r.t. best particle
        ii. If (updated position & velocity > old
            position and velocity)
              Keep the new position and velocity
            Else
              Revert to old position
5.  Check each particle
        If (any particle fulfills stopping
    criteria)
            STOP.
        Else
            Go to Step 4.
```

Fig. 5 Pseudo code of PSO

ABC algorithm was proposed by Karaboga in 2005 to solve numerical optimization problem. It is a non-pheronome, population based metaheuristic technique. Each employed bee is associated with a particular food source. Also, the number of employed and onlooker bees are equal. As explained in Figure 6, Bees randomly search for food positions with the higher amount of nectar. Once the bees find such position, they go back to the hive and communicate about the food source position. The most important part of the hive is dancing area where the bees exchange information. The related dance is called waggle dance. If food source position is close to the hive, the bees perform waggle dance else bees perform the round dance. Waggle dance is eight like figure. The bees have capability of memorizing the location of food sources with the higher nectar amount. Once such food source is discovered, the bees start exploiting it. Hence, they become the employed forager. ABC is based on the foraging behaviour of honey bees. There are three types of bees:

- Employee bees: Each employee bee is associated with a particular food source. The bees go to their associated food source and determine its neighbor source, then evaluate its nectar amount and dances in the hive.
- Onlooker bees: Each onlooker bee watches the dance of employed bees and chooses one of their sources depending on the dances, and then goes to

that source. After choosing a neighbor around that, she evaluates its nectar amount.

- Scout bees: If the onlooker bee finds that the selected test cases are not efficient, then the scout bee generates a new population of test cases and replaces the test cases in the temporary-test case-list with random new test cases.

The whole process is repeated until termination condition is met, that is, maximum coverage criteria is obtained or maximum number of cycles is executed. ABC has been widely used for regression test case optimization ([13],[16],[26],[27]) and for optimizing numerical benchmark functions ([5],[9],[10]).

```
1. Initialize
2. REPEAT
        i.   Move the employed bees onto their food
             sources and determine their nectar
             amounts.
        ii.  Move the onlookers onto the food sources
             and determine their nectar amounts.
        iii. Move the scouts for searching new food
             sources.
        iv.  Memorize the best food source found so
             far.
3. UNTIL (requirements are met).
```

Fig. 6 Pseudo code of ABC

Till now there are strong observations and formulation about the animal's intelligent behaviour such as ant colony, bee colony. They involve foraging for food not by simple but by collective intelligence behaviour. Not only animals as described above forage for food intelligently but the same have been done by the plants too. A key point of observation is that Cuscuta somehow knows its starvation. If the host is found unsuitable, the Cuscuta species continue its search but once selection is made the Cuscuta species coil around its selected host in a specific manner (anticlockwise) to transfer resources from the host plant. Considering this dynamics, it can be said that Cuscuta search for its food from its current need and will continue to attack till its starvation get complete. As soon as its starvation is completed a new branch will evolve. The evolution of new branch is considered as the completion of search i.e. no left starvation. The new branch will again repeat the same process until all plants nutrients are been taken by Cuscuta i.e. at short of dead host[29].

III. Literature Review

Lot of techniques have been proposed by various researchers on how to apply metaheuristic techniques to regression testing. W. Eric Wong et al. proposed hybrid technique which combines modification, minimization and prioritization-based selection using a list of source code changes and the execution traces from test cases run on previous versions [6]. He reported his experience with a tool called ATAC which showed that both minimization and prioritization serve as good, cost-effective alternatives for testers who need to conduct quick regression testing under time pressure and budget constraints. Size reduction, precision and recall are the metrics used to examine the goodness of the proposed approach.

G. Rothermal et al. described several techniques for using test execution information to prioritize test cases, including techniques that order test cases based on their total coverage of code components, coverage of code components not previously covered, and their estimated ability to reveal faults in the code components that they cover [4]. The performance of these techniques was compared with untreated, randomly and optimally ordered test suites. Analysis of the data showed that each of the prioritization techniques studied improve the rate of fault detection of test suites, and this improvement occurred even with the least expensive of those techniques.

Some researchers applied greedy techniques for test case prioritization problem. Zheng Li et al. compared the greedy techniques i.e. Greedy, Additional Greedy and 2-optimal with the two metaheuristic techniques i.e. Hill Climbing and Genetic Algorithm [18]. The results showed that Genetic Algorithms perform well, although Greedy approaches are surprisingly effective, given the multimodal nature of the landscape.

Arvinder Kaur et al. proposed the Bee Colony Optimization Algorithm for Fault Coverage Based Regression Test Suite Prioritization[11]. The BCO algorithm has been formulated for fault coverage to attain maximum fault coverage in minimal units of execution time of each test case, using two examples whose results are comparable to optimal solution. Average Percentage of Fault Detection (APFD) metrics and charts has been used to show the effectiveness of proposed algorithm and was implemented in CPP compiler. Arvinder Kaur et al. further used Particle Swarm Optimization with Cross-Over Operator for prioritization in regression testing which did prioritization of test cases on different selection criteria other than the fault coverage and code coverage, as the algorithm uses the phenomena of convergence (PSO) while diversifying search space (GA operator) for regression testing [12]. The APFD & APCC values were comparable w.r.t. optimal result, that proves algorithm prioritized efficiently. Some researchers have also combined PSO with the mutation operator of GA [17]. The Genetic Algorithm (GA) operators provide optimized way to perform prioritization in regression testing and on blending it with Particle Swarm Optimization (PSO) technique makes it effective and provides fast solution. The Genetic Algorithm (GA) operator that has been used is mutation operator which allows the search engine to evaluate all aspects of the search space. For the problem taken, the algorithm provides 75.6% of fault coverage but sometimes, mutation can result in quite long execution time.

Bharti Suri et al. used another metaheuristic technique i.e. Ant Colony Optimization for test case selection and prioritization[14]. Random nature of ACO helps to explore the possible paths and choose the optimal from them. The results obtained were in close proximity to the optimal results. However, the best results are not found for all the cases.

Arvinder Kaur et al. implemented GA for regression test suite prioritization within time constrained environment on the basis of total fault coverage[24]. APFD metric is used to evaluate the performance of the algorithm. The algorithm was implemented only for integer input and the future work is to implement it for string input variable. S. Raju and G. V. Uma implemented Factors Oriented Test Case Prioritization Technique in Regression Testing using Genetic Algorithm [25]. In the proposed technique, Prioritization Factors (PF) were used. These factors may be concrete, such as test case length, code coverage, data flow, and fault proneness, or abstract, such as perceived code complexity and severity of faults, which prioritizes the system test cases based on the six factors: customer priority, changes in requirement, implementation complexity, completeness, traceability and fault impact. APFD and PTR metric were used to evaluate the fitness. Based on the performance measure obtained, the proposed method is effectively prioritizing the test cases.

Abraham Kiran Joseph et al. presented a hybrid approach, a combination of PSO and ABC for test case optimization [15]. The objectives considered in this research work were statement coverage and fault coverage within a minimum execution time. Based on the proposed hybrid approach, an optimal result for test case execution is obtained. The performance of the proposed method was evaluated and was compared with other optimization techniques such as PSO and Ant Colony Optimization (ACO). It was observed from the experimental results that the proposed PSABC based test case prioritization based approach provides better results as compared to ACO, PSO and ABC.

Researches for the purpose of regression testing using fuzzy logic are very rare. Harsh Bhasin et al. proposed a new approach to prioritization of test cases using fuzzy logic[23]. It was found that fuzzy expert system provides better results than the other decision making systems. Aftab Ali et al. used fuzzy logic for test suite optimization to optimize test cases based on fault detection, execution time and coverage [21]. The proposed expert system finds a trade off among the quality aspects, technique used and level of testing. The implementation of algorithm and its comparison with other CI techniques is left for the future work. Deepak Rai et al. used Honey Bee Mating Optimization Algorithm with Fuzzy Rule Base for regression test suite optimization. Reduction in the test suite using the proposed algorithm is approximately 50%, but it is little bit low than ACO and BCO[20]. Deepak Rai et al. further estimated the regression test case selection probability using Fuzzy Rules[22]. In the paper , three inputs were considered i.e. fault coverage, execution time and code coverage and the output taken was selection probability. The author had used triangular membership function for fuzzification and COG(Center of Gravity) for defuzzification. The results obtained were very close to the optimal results.

The Artificial Bee Colony optimization problem was introduced by Dervis Karaboga in 2005 for solving numerical optimization problem [5]. From the simulation results it was concluded that the proposed algorithm has the ability to get out of a local minimum and can be efficiently used for multivariable, multimodal function optimization. The results produced by ABC, Genetic Algorithm (GA), Particle Swarm Algorithm (PSO) and Particle Swarm Inspired Evolutionary Algorithm (PS-EA) were compared and the results showed that ABC outperforms the other algorithms. Dervis karaboga et al. compared the performance of ABC algorithm with that of differential evolution (DE), particle swarm optimization (PSO) and evolutionary algorithm (EA) for multi-dimensional numeric problems[9]. Results showed that ABC algorithm performs better than the mentioned algorithms and can be efficiently employed to solve the multimodal engineering problems with high dimensionality.

D. Jeya Mala presented ABC Tester for test suite optimization and the superiority of the proposed approach over the existing GA based approach was found [13]. Problems with GA include no memorization, non linear optimization, risk of suboptimal solution and delayed convergence. It can be concluded that the proposed approach used fewer iterations to complete the task, more scalable i.e. it requires less computation time to complete the task and best in achieving near global optimal solution. Bahriye Akay et al. introduced the modified ABC algorithm for real parameter optimization[10]. A scaling factor that tunes the step size adaptively was introduced. It can be concluded that the standard ABC algorithm can efficiently solve basic functions while the modified ABC algorithm produces promising results on hybrid functions compared to state-of-the-art algorithms.

Mukesh Mann and Om Prakash Sangwan applied Cuscuta Search Algorithm for prioritizing test cases in an order of maximum fault coverage with minimum test suite execution and compared its effectiveness with different prioritization ordering [29]. Taking into account the experimental results, it can be concluded that Cuscuta ordering gives same results as given by the optimal and ACO ordering but better than No order, Random order and Reverse order.

TABLE 2
COMPARATIVE STUDY OF METAHEURITIC TECHNIQUES

Sr. No.	Author/Paper	Technology used	Data Set	Results
1.	Arvinder Kaur, Shivangi Goyal[11]	BCO for Test Suite Prioritization	College Program ,Hotel Reservation	Comparable to optimal order
2.	Arvinder Kaur,Divya Bhatt[12]	PSO+crossover for prioiritization	Student, Hotel,Triangle,etc.	Comparable to optimal order
3.	D. Jeya Mala, V. Mohan [13]	ABC for test suite optimization	Academic & Industrial Test	Path Coverage higher than GA
4.	Bharti Suri, Shweta Singal[14]	ACO for selection and prioritization	C++ code	Not good in all cases, but close to optimal
5.	D. Karaboga, B. Basturk [9]	ABC	Numerical Benchmark function	ABC better than DE, EA and PSO for multimodal functions
6.	Arvinder Kaur, Shubhra Goyal[24]	GA for prioritization	College Program, Hotel reservation	Comparable to optimal order
7.	Soma Sekhara Babu Lama et al.[26]	ABC for optimization	Triangle Classification problem	ABC Better than GA, ACO and Tabu Search
8.	Adi Srikant et al.[27]	ABC for optimization	Quadratic Equation	Better than GA, old ABC, ACO
9.	Arvinder Kaur, Divya Bhatt[17]	PSO+ GA mutation for prioritization	Java code	Good fault coverage
10.	Abraham Kiran Joseph et al.[15]	ABC+ PSO for optimization	Random Example	PSABC better than ABC, ACO and GA
11.	Sunrender Singh Dahiya et al.[16]	ABC for test suite optimization	Triangle classifier, Binary Search, etc.	Not suitable for large i/p domains and for many constraints

Metaheuristic techniques implemented by various researchers for regression testing have been compared and presented in the Table 2 on the basis of technology used, data sets and the results obtained. From the table shown, it can be concluded that ABC performs well in most of the papers described above for test suite optimization. It can also be verified that ABC provides better results than GA, ACO and BCO. Although , the results of ACO, BCO and GA were close to that of the optimal order.

It can be accomplished that BCO, PSO, ACO, Additional Greedy, GA , Hill Climbing can be used for regression test suite prioritization . For regression test suite selection, ACO, Tabu Search and GA were used.

- With GA, we have the disadvantage of obtaining a local optima or premature convergence etc.
- The drawbacks of Ant Colony Optimization (ACO) include higher length test sequences and repetition of nodes within the same sequence without any advantage on test adequacy criteria. Two ants started at an initial node, and during random selection of next node, they will go to the same next node. Since the process is random; one cannot expect such behavior.
- The main drawback behind neural network based approach is their black box structure i.e. it is difficult to interpret and also they have a slow convergence speed.

As ABC is non-pheronome based approach, there is no need for updating the pheromone. Also, the parallel behaviour of the employed, onlooker and scout bees make the search process much faster, so it has a very high processing speed. The global search method carried out by the scout bee is combined with the local search carried out by the onlooker and employed bee. So, there is a right balance between exploration and exploitation.

IV. POTENTIAL BENEFITS OF METAHEURISTIC TECHNIQUES TO REGRESSION TESTING

In order to automate the testing process and to provide a near optimal solution in a lesser time, metaheuristic search techniques are used in the domain of regression testing. Initial population of ants, bees or genes etc. represents a test case (or test suite) and it evolves towards the better solution until the stopping criteria is met. The goal is to find an optimal solution to minimize the cost of regression testing to obtain maximum path coverage, branch coverage , statement coverage, fault coverage , minimum execution time or any combination of above. With different objective functions, techniques will have different complexity and search-space characteristics The different metrics that can be used for evaluating the benefit of various techniques are APFD (Average Percentage of Fault Detected), APCD (Average Percentage of Code Detected), APSD (Average Percentage of Statement Detected), APBD (Average Percentage of Branch Detected) etc.

V. Conclusion And Future Work

Effective regression testing is a trade-off between the number of regression tests needed nd the cost. The greater the number of regression tests, the more complete is the program revalidation. However, this also requires a huge budget and greater resources which may not be affordable in practice. In this paper, several techniques have been described for minimizing the cost of regression testing and their relative abilities are examined. Analysis indicate that the Greedy Algorithm performs worse than Additional Greedy, 2-Optimal, and Genetic Algorithms overall. Also, the 2-Optimal Algorithm overcomes the weakness of the Greedy Algorithm and Additional Greedy Algorithm. It can also be accomplished that ABC outperforms the other approaches i.e. GA, ACO, BCO and PSO in test suite optimization process. As a future work, different versions of ABC have to be applied for minimizing the cost of regression testing and analytical study can be conducted in finding the best ABC version to achieve near global optimal solution. Also, the performance of ABC can be compared with other metaheuristic techniques for efficiency evaluation. Prioritization technique based on Cuscuta search algorithm has been proposed to find the near optimal solution which gives the same results as given by the optimal and ACO ordering but better than unorderd, random and reverse order. Cuscuta search can be implemented for regression test case prioritization and its comparison with existing metaheuristic techniques can be done. Various tools that can be used for implementation are MATLAB , Weka , Java IDE etc.

References

[1] K.K.Aggarwal and Yogesh Singh, "Software Engineering Programs Documentation, Operating Procedures", *New Age International Publishers*, Revised Second Edition – 2005.

[2] G. Rothermel, M.J. Harrold, J. Ostrin, and C. Hong, "An Empirical Study of the Effects of Minimization on the Fault Detection Capabilities of Test Suites", *Proc. International Conference on Software Maintenance*, pp. 34-43, Nov. 1998.

[3] D. Binkley,"Semantics Guided Regression Test Cost Reduction", *IEEE Transactions on Software Engineering*, Vol. 23, No. 8, pp. 498-516, Aug. 1997.

[4] Gregg Rothermel, Roland H. Untch, Chengyun Chu andMary Jean Harrold, "Prioritizing Test Cases For Regression Testing", *IEEE Transactions on Software Engineering*, Vol. 27, No. 10, October 2001.

[5] Dervis Karaboga and Bahriye Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC)algorithm", *J Glob Optim* 39:459–471DOI 10.1007/s10898-007-9149-x, 2007.

[6] W. Eric Wong, J. R. Horgan, Saul London and Hira Agrawal, "A Study of Effective Regression Testing in Practice", *8th IEEE International Symposium on Software Reliability Engineering* (ISSRE'97), pp 264-274, Albuquerque, NM , November 1997.

[7]P. McMinn, "Search-based Software Test Data Generation: A Survey", *Software Testing, Verification and Reliability*, Vol.14, No. 2, pp. 105- 156, 2004.

[8] J.H. Holland, "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence", *Univ. of Michigan*, 1975.

[9] D. Karaboga and B. Basturk , "On the performance of artificial bee colony (ABC) algorithm", *Applied Soft Computing*, Vol. 8, No. 1, January 2008.

[10] Bahriye Akay and Dervis Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization", *International Journal on Information Sciences*, Vol. 192, June 2012.

[11] Arvinder Kaur and Shivangi Goyal, "A Bee Colony Optimization Algorithm for Fault Coverage Based Regression Test Suite Prioritization", *International Journal of Advanced Science and Technology* ,Vol. 29, April, 2011.

[12] Arvinder Kaur and Divya Bhatt, "Particle Swarm Optimization with Cross-Over Operator for Prioritization in Regression Testing", *International Journal of Computer Applications* (0975 – 8887) Vol. 27, No.10, August 2011.

[13] D. Jeya Mala and V. Mohan, "ABC Tester - Artificial Bee Colony Based Software Test Suite Optimization Approach", *International Journal of Software Engineering*,Vol.2 ,No.2 July 2009.

[14] Bharti Suri and Shweta Singhal, "Implementing Ant Colony Optimization for Test Case Selection and Prioritization", International Journal on Computer Science and Engineering (IJCSE), ISSN : 0975-3397 Vol. 3, No. 5 ,May 2011 .

[15] Abraham Kiran Joseph and G. Radhamani, "A Hybrid Model of Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) Algorithm for Test Case Optimization", International Journal on Computer Science and Engineering (IJCSE), Vol. 3, No. 5 May 2011.

[16] Surender Singh Dahiya,Jitender Kumar Chhabra and Shakti Kumar, "Application of Artificial Bee Colony Algorithm to Software Testing", 21st Australian Software Engineering Conference, IEEE Computer Society, 2010.

[17] Arvinder Kaur and Divya Bhatt, "Hybrid Particle Swarm Optimization for Regression Testing", International Journal on Computer Science and Engineering (IJCSE) ,Vol. 3, No. 5 May 2011.

[18] Zheng Li, Mark Harman, and Robert M. Hierons, "Search Algorithms for Regression Test Case Prioritization", IEEE Transactions on Software Engineering, Vol. 33, No. 4, April 2007.

[19] Y. Singh, A. Kaur. and B. Suri, "A New Technique for Version – Specific Test Case Selection and Prioritization for Regression Testing", Journal of Computer Society of India, 36(4), pp. 23-32, 2006.

[20] Deepak Rai and Kirti Tyagi, "Regression Test Case Optimization Using Honey Bee Mating Optimization Algorithm with Fuzzy Rule Base", World Applied Sciences Journal, Vol. 31, No. 4, pp. 654-662, 2014.

[21] Aftab Ali Haider, Shahzad Rafiq and Aamer Nadeem, "Test Suite Optimization using Fuzzy Logic", International Conference on Emerging Technologies, 2012.

[22] Deepak Rai and Kirti Tyagi, "Estimating the Regression Test Case Selection Probability using Fuzzy Rules", International Conference on Recent Trends in Information Technology (ICRTIT) , 2013. [23] Harsh Bhasin, Shailja Gupta and Mamta Kathuria, "Regression Testing Using Fuzzy Logic", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 4 , No. 2 ,pp. 378 – 380, 2013.

[24] Arvinder Kaur and Shubhra Goyal , "A Genetic Algorithm for Fault based Regression Test Case Prioritization", International Journal of Computer Applications (0975 – 8887) Vol. 32, No.8, October 2011.

[25] S. Raju and G. V. Uma, "Factors Oriented Test Case Prioritization Technique in Regression Testing using Genetic Algorithm", European Journal of Scientific Research ISSN 1450-216X Vol.74, No.3 , pp. 389-402, 2012.

[26] Soma Sekhara Babu Lam, M.L. Hari Prasad Raju, Uday Kiran M, Swaraj Ch. and Praveen Ranjan Srivastav, "Automated Generation of Independent Paths and Test Suite Optimization Using Artificial Bee Colony", International Conference on Communication Technology and System Design, 2011.

[27] AdiSrikanth, Nandakishore J. Kulkarni, K. Venkat Naveen, PuneetSingh, and Praveen Ranjan Srivastava, "Test Case Optimization Using Artificial Bee Colony Algorithm", ACC 2011, Part III, CCIS 192, pp. 570–579. Springer-Verlag Berlin Heidelberg , 2011.

[28] Chartchai Doungsa-ard, Keshav Dahal , Alamgir Hossain and Taratip Suwannasart, "An automatic test data generation from UML state diagram using genetic algorithm ", International Conference on Software Engineering Advances, August 2007.

[29] Mukesh Mann and Om Prakash Sangwan, "Test case prioritization using Cuscuta search", International Academy of Ecology and Environmental Sciences , Network Biology, 2014.

[30] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, NY, 1999.

[31] R. Krishnamoorthi , S.A Sahaaya, A. Mary , "Regression Test Suite Prioritization using Genetic Algorithms", International Journal of Hybrid Information Technology, Vol. 2, No. 3, pp. 35-52, 2009.
[32] Bianchi, Leonora; Marco Dorigo; Luca Maria Gambardella; Walter J. Gutjahr . "A survey on metaheuristics for stochastic combinatorial optimization". International Journal on Natural Computing , Vol. 8, No. 2, pp. 239–287, 2009.
[33] C. Blum and A. Roli , "Metaheuristics in combinatorial optimization: Overview and conceptual comparison", ACM Computing Surveys, Vol. 35 , Issue 3, pp. 268–308, September 2003 .
[34] Mukesh Mann and Om Prakash Sangwan, "Generating and prioritizing optimal paths using ant colony Optimization", Computational Ecology and Software, Vol. 5, No. 1,pp. 1-15,January 2015