Chapter 1. Introduction

1.1 General Idea

Real world optimization problems have become large, complex and dynamic. Computational Intelligence (CI) algorithms are gaining popularity due to their ability to find acceptable results in stipulated amount of time. CI algorithms are based on heuristics that are capable of achieving near optimal solutions within the given time and space constraints [16]. CI algorithms have been applied in various domains like computer virus detection, fault diagnosis, robotics and control, anomaly detection, pattern recognition and optimization [29].

In the vast domain of CI, we are interested in exploring the application of swarm based computing in software testing domain. The swarm intelligence is guided by the concept of information sharing by collective behaviour of unintelligent agents. The two keen properties of swarm based algorithms are flexibility and robustness. Flexibility is defined as the ability of the algorithm to adapt to the changing environment [1]. Robustness endows the colony with the ability to function even though some individuals may fail to perform their tasks. Swarm based algorithms have the ability to converge to the optimal solution faster and with fewer parameters. The swarm based algorithms like Biogeography Based Optimization, Ant Colony Optimization and Firefly have gained importance in various application domains like biomedical and social science, data mining, networking and military applications [2]. Another category of algorithm, named Music Inspired Algorithm: Harmony Search, have been used to explore the large search area in software testing domain.

Furthermore, Support Vector Machine, which are considered as statistical linear classifier have been used in software testing domain, for identification and prediction of infeasible test cases [45] and genetically modified support vector machine for predicting fault prone components [48].

Software testing is a new research domain envisioned for application of computational intelligence algorithms. Our research is inclined towards application of swarm based intelligent algorithm. Software testing cycle is one of the tedious process in the software development period wherein the software tester performs the job of analysing each code fragments using the test suite. If test suite is large and so is the software, the time period of the cycle is extended to multiple years. Past research has shown application of swarm based techniques like Cuckoo Search for test case prioritization using Line of Code [LOC] approach [27], River Formation Dynamics has been used in construction of test plans using test specifications by considering test cycle as a finite state machine [28], and firefly algorithm has been used in the generation of optimal test paths [19].

Regression Testing is a testing process being carried out during the maintenance phase of the software development life cycle [1]. It is a process which involves retesting the program to check the rippling effect of the changes being made in the program. For the success of the retesting process, it is crucial that test cases must provide complete coverage, hence it is essential to re-run all the test cases in the test suite. However, this re-run technique to provide complete coverage of the modified program is feasible when the length of the program and

execution time is very less. If the length of program and execution time is large, then retest-all approach for regression testing become non-pragmatic [3]. Hence, in order to improve the testing cycle in the maintenance phase by reducing the time consumption and providing testing that attacks the affected area in the program, it essential to depend on Test Suite Optimization [TSO].

1.2 Motivation

A test case is defined as a set of inputs to a program in order to extract possible weakness in the program. The weakness in the program can be array index out of bounds, faulty loop execution, wrong testing conditions, undefined variable, uninitialized variable, wrong memory indexing, parameter mismatch and linking errors [4].

Test suite optimization is a field that comprises of test suite minimization, test suite selection and test suite prioritization [5]. Test suite minimization is a process of reducing the test cases whose existence does not improve the efficiency of testing. Removing of test cases hurts the fault tolerance characteristic of the system, as complete testing bring in confidence in the system and increases its trustworthiness quotient [5]. But, test suite minimization works on the foundation of selection of those test cases that are not redundant and provide exhaustive testing.

Second subfield in TSO is test suite selection. Test suite selection is a process of selecting those test cases that provide complete coverage [6]. It is based on set theory, which states that a set A union set B defines the universal set. Test suite selection identifies a subset test cases, with sole motive to provide complete testing.

Third and last subfield of TSO is test suite prioritization. Test suite Prioritization is a process of assigning ranks or priority on varied scales to the test cases, so that at the time of testing proper sequence of test cases are defined to carry out structured fault detection and removal [7]. It is not focussed on removing any test cases rather it provide priority values to the test cases based on their fault detection capability and execution time. Test suite prioritization can be considered as a separate process in the regression testing cycle but is usually accompanied with test suite minimization and test suite selection [7].

1.3 Related Work

Test suite Optimization [TSO] process has been improved by various techniques in earlier researches. In [2] a greedy technique was proposed that worked on the principal that, those test cases are selected that cover maximum testing requirements which are mutually exclusive.

In [3] a nature inspired technique is used to modify the process of test selection and test prioritization. Ant Colony Optimization [ACO] technique work on the real life behaviour of ant. The ant colony optimization is a swarm based technique which explains how randomness characteristics in ACO helps in exploration of all possible solutions and choose an optimal solution [1]. There is single fitness function in ACO that helps in optimal solution identification.

In [4] an evolutionary algorithm was used to generate solutions based on crossover, selection and mutation. Genetic algorithm is based on single fitness function that is used to evaluate the problem domain whereas bio inspired algorithm employees a cost function for convergence and selection of fit individual in each generation. Moreover, bio inspired algorithm uses function that is indifferent from discontinuity and differentiability whereas other algorithm like hill climbing, simulated annealing, genetic algorithm are influenced by the correlation values between various fields used to compare test cases like execution time and fault covered.

Predicting software modules which are defect prone [47] and software reliability prediction [48] are some broad areas where SVM has been used. In our research, we used SVM for test case prioritization and compared its efficiency with Biogeography Based Optimization [BBO] and Extended Biogeography Based Optimization [EBBO] using some significant parameters such as identification of redundant test cases, untestable modules, worthless test cases and complete coverage connected component of test cases with the test suite.

Biogeography Based Optimization (BBO) is a swarm intelligence algorithm which has the ability to solve NP Hard problem using the models biogeography that are speciation (It is the evolution of new species), migration (movement of the species between the islands) and extinction (removing the species from the habitat). Extended BBO is a modified form of BBO wherein Simulated Annealing, a traditional optimization technique is used to achieve global optimization.

Research in the domain of software testing requires numerous data set which can be obtained from UCI Repository [50], repository of machine learning at University of Ottawa [51] and University of Southern California [52].

1.4 Problem Statement

BBO has shown good results in different domains such as feature extraction in Remote Sensing [29], face recognition [30], ground water detection [31] and travelling tournament problem [22]. BBO is a meta-heuristic algorithm that makes no assumption about the problem domain. Moreover, BBO can be used to optimize multi-dimensional real valued function and problems. BBO does not incorporate any gradient function like newton rule, perceptron rules or relaxation procedure, which are used in various statistical algorithm that were used in [5]. Our research focuses on exploration of BBO algorithm in the domain of test case prioritization and test case minimization.

Harmony Search is a music inspired computing algorithm which is slowly gaining importance in different domain. It has been successfully applied in test case generation [49]. Harmony search is also a meta-heuristic algorithm similar to BBO but it does the work in random fashion whereas BBO has a well set of multiple agents responsible to arrive at global optimized solution. We explore harmony search for test suite optimization which has not been done in earlier work [9].

Very recently BBO has been further enhanced in [22] called Extended BBO [EBBO] and is showing better results than original BBO. We also adapt EBBO for test suite optimization to evaluate its performance with respect to BBO. Therefore, problem of the thesis can be stated as:

Adapt BBO and Extended BBO for Test Suite Optimization and evaluate their performance with other Computational Intelligence (CI) Techniques.

1.5 Objective and Scope

Biogeography Based Optimization (BBO) is modified through mutation and migration operator to perform the function of optimization in Test suite so as to reduce the cumbersome task of Regression Testing. BBO performs its function, taking Fault detection rate as the objective function of the test cases. Furthermore, BBO is extended using simulated annealing to perform global optimization of Test Suite so as to reduce the amount of time consumed in software testing. Following parameters were modified in BBO and EBBO: Mutation probability, Size of Elite array, Generation Limit, Cost Function, Emigration and Immigration probability.

We exhibit their performance and compare them with other meta-heuristic algorithms viz. Ant Colony Optimization (ACO), Firefly, Support Vector Machine (SVM) and Harmony Search. In this ACO and Firefly belong to swarm based computing algorithm category and have been extensively used in the testing domain [20]. SVM being a linear classifier, is used in supervised learning and to predict the priority of the test cases.

The performance of BBO, EBBO and various other meta-heuristic algorithm are compared based on following parameters: Number of redundant test cases identified, Number of worthless test cases identified, Test cases that provide complete coverage, Priority range created, Percentage reduction in test suite and identification of untestable part of module.

We have chosen R programing language to implement BBO, Extended BBO and Harmony search, whereas we have implemented Firefly and ACO in MATLAB.

Thus scope of this thesis is to:

- Adapt BBO for Test Suite Optimization
- Adapt Extended BBO for Test Suite Optimization
- Empirical Study of BBO, Extended BBO for Test Suite Optimization in JBOSS application Server.
- Comparing the results of BBO, Extended BBO with Harmony Search, Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Firefly and Support Vector Machine (SVM).

1.6 Organization of Thesis

The rest of this paper is organized as follows.

Chapter 2 provide details about the past research done on improving regression test suite optimization.

Chapter 3 provide details Biogeography based optimization and Extended Biogeography Based Optimization in solving Travelling Tournament problem.

Chapter 4 provide details about Traditional approaches like Harmony Search, Ant Colony optimization, Firefly and Support vector Machine in solving Test Suite optimization (TSO) problem.

Chapter 5 shows the procedure as to how to adapt biogeography based optimization and Extended Biogeography based optimization in solving TSO

Chapter 6 provides information about the JBOSS server (dataset), simulation environment, Results and analysis and discussion

Chapter 7 provides the list of publication from this research in various conferences and journals.

Chapter 8 is about references, where all the research articles which contributed to this research are listed.

Chapter 2. Literature Survey

In this section we provide an overview of related work in the field of test suite optimization. This section discuss about the past algorithm that have been applied for test suite prioritization, test suite selection and reduction. These activities are essential for regression test suite design.

In [7] it proposed a test suite prioritization technique utilizing the structure of the program. The technique tries to prioritize the test suite subsequence and compared it with those technique that does the same randomly. Genetic Algorithm was used and it effectiveness was judged using J-depend and Gradebook open source tool. There is a limitation to this technique that is its efficiency depends on the attributes and also does not incorporate the elitism criteria.

In [8], greedy algorithm was used to identify a test case that satisfies the maximum number of testing requirements and at the same time has minimum overlap with other test cases. The algorithm was tested on Siemens suite and space program. The research was developed over the idea to have minimum overlap over other test cases but overlapping does not affect the testing issues. Moreover, in regression testing main idea is to have a minimum set of test cases that provides maximum coverage.

In [9], Bee colony algorithm was used to provide solution to the problem of regression testing. In this proposed work, bee colony algorithm was used for attaining maximum fault coverage in minimum execution. The algorithm works on some heuristic function or cost function that is to be minimized or maximized.

In [10], hybrid particle swarm optimization technique is used to carve out a set of test cases that have minimum execution time. The algorithm merges the best out of Particle swarm optimization that is global search and Genetic Algorithm's mutation operator. Though a sound technique, but in regression testing we look at regression test suite design in a pervasive manner that is to design an algorithm that does the work on test suite selection, prioritization and reduction. In order to arrive at complete test suite optimization, meta-heuristics algorithm achieve the state of the art.

In [32], mutant gene algorithm is used in coordination with genetic algorithm. The main idea of the work was to have minimum set of test cases and improve test case selection. The algorithm works well with main parameter being mutation value but as far as meta-heuristic algorithms are concerned there are multiple intelligent agent, hence the work focuses on global optimization.

In [11], on the fly test suite optimization technique was proposed using fuzzy logic for multi-objective test suite optimization. In fuzzy logic, the main idea is to generate a set of rules that affect the label attribute but it is not always the case in test suite problems, as in regression testing what we need is to attain maximum coverage using fault covered and execution constraints. Moreover, there are test cases which do not detect any fault in the module hence will not generate any rule and affect accuracy of the result but the idea of on the fly test suite optimization is better for regression testing domain. To solve such problem harmony search, a meta-heuristic algorithm can be used. The XOR operation in harmony search make it feasible to test those modules which were left behind by the test case.

In [12], regression testing is improved by ant colony optimization and dynamic dependency injection. The main idea is to have a set of test cases which possess the potential to detect any

bug that creeps in after the system starts functioning with real environment variables. Hence, the algorithm is modified by dynamically injection dependency on the best route identified by ant colony algorithm.

In [13], hierarchical method for test case prioritization was developed based on requirement in SRS. This approach assigns priority values to requirements based on a set of 12 factors viz. Customers assigned, requirements volatility, developers assigned, fault proneness etc. This method was based on requirements, resultant data set can be used with BBO and Harmony search for fault detection and severity.

In [14], a fuzzy logic based expert system is designed for obtaining a set of test cases based on multi-objective regression testing. The quality aspects on which test suite optimization is performed are performance that is fault detection rate, throughput and code coverage.

In [15], model based test suite prioritization is specified. The model based test suite prioritization has very small execution overhead as compared to execution of the system and henceforth helps in early fault detection.

In [34], stated that evolutionary algorithm is relatively new approach to find solutions for various optimization problems. When it comes down to multi-objective problems where multiple objective need to be assessed, evolutionary algorithm comes in handy. This is because, evolutionary algorithm have a tendency to provide comprehensive search over the problem, highly effective as solutions are filtered based on fitness functions and fast convergence as compared to other statistical approaches. Hence NSGA, Genetic algorithm are evolutionary approaches that have been applied to multi-objective optimization problems.

In [37], a new method of solution modification and reduction is presented by means of clusters and similarity among clusters. These clusters are formed based on decreasing distance function value. The cluster centroid value can serve as appropriate factor for inference about the data partitions and to appropriateness of division of data. This method of clustering has been used in Multi-Objective Biogeography Based Optimization.

In [39], used techniques like Bayesian predictor to select best individuals from the population which are served as inputs to multiple linear regression module. The linear regression module together with cross validation is able to forecast the tradeoff worthiness of test case and fault detection rate of test case. The technique though achieved the desired solution but suffers from drawbacks such as the technique of linear regression is a statistical learning technique which is highly influenced by the data and the graph of the data set. Moreover, the algorithm used in deriving the solution is highly dependent on the data set and is not generic. Furthermore, linear regression is a traditional technique used in optimization but it tends to stop at local optima and provide single solutions. As far as computational intelligence (CI) field is taken into consideration, the algorithms in CI are much faster and are not affected by non-differential behavior of the data points.

In [40], a new technique called Multi-Objective Genetic Algorithm (MOGA) has been proposed to find alternative solutions for optimal test suite selection and prioritization problems. The proposed technique was applied on complex system to suggest strategies for maximizing and minimizing the objectives of the complex system in the presence of defined constraints. The algorithm though successfully derived various Pareto fronts but did not

incorporate elitist criteria in generating best solutions. Moreover, the algorithm didn't able to provide diversity in the solution and the distribution of solution was highly uneven.

In [41], proposed Multi-Objective Differential Evolution (MODE) approach for solving Pareto front problems especially in software engineering system. The main idea behind application of differential evolution is high performance obtained after application of differential evolution on the data set where the attributes / decision variables have independence. MODE was applied on software effort prediction and software quality determination.

In [42], proposed a novel Pareto Differential evolution technique to solve vectorized optimization problems. The technique is motivated from application of differential evolution to multi-objective problems. The differential evolution has been used as a method for migration in the bio-geography based optimization. The diversity of the population in the archive array (non-dominated solutions) has been greatly enhanced.

In [43], demonstrated how some utility functions and a paradigm are sufficient to generate alternatives for the problem in hand. The paper provided an overview of modified and extended utility maximization theory within the environment that is context free and under a defined and constructed threshold. The maximization theory together with models that are statisitical or heuristics can aid in decision making, social theory and behavioral and cognitive sciences.

In [44], defined a new application area for multi-criterion decision making. The book defines a tradeoff and synergetic relation between the land use management and multi-criteria analysis. The field of spatial mining, spatial planning and spatial analysis already use multi-criteria analysis, hence field of software requirement management can serve as an apt case study for the application of multi-objective optimization approaches.

In [45], support vector machine classifier is used to identify infeasible test cases in the test suite. Infeasible test cases are those which terminate prematurely and are responsible in wastage of software resources. So in order to reduce wastage of resources like computation cost, memory usage, processing time and execution resources, we need to sort out those test cases that are feasible. The method of support vector machine and induced grammar have been used in identifying feasible test cases in the test suite. In our problem domain, SVM has not been able to identify infeasible test case based on supervised learning. So in [45], the method of induced grammar play the critical role in making SVM learn to detect infeasible test case.

In [46], traditional problem in early 90's were also multi-objective problems but due to lack of development of approaches to solve Multi-objective problems, multiple objective were combined to single objectives through linear combination and as a whole were optimized either through maximization or minimization. Genetic Algorithm has been modified by incorporating Pareto dominance in its selection process and applying niche pressure to spread the non-dominated solution over the Pareto optimal surface.

Chapter 3. Nature Inspired Algorithms for Complex Problem : Test Suite Optimization

In this chapter we develop motivation for BBO and EBBO by defining an algorithm for solving NP hard problems such as Travelling Tournament Problem [TTP]. Following it, we will elaborate Test Suite Optimization problem, which is categorized as a NP Hard problem in software engineering domain [11]. We will first describe about test suite followed by its various intricacies and finally describe with a flowchart as to what is a test suite optimization problem.

3.1 Biogeography Based Optimization (BBO)

Biogeography Based optimization (BBO) is a meta-heuristic bio inspired algorithm with lays on the foundation of speciation, migration of special between islands and extinction of species. The features variable in BBO are Suitability Index Variable (SIV) and Habitat Suitability Index (HSI). When a species migrate from an island, it doesn't mean that complete migration happens, only selected species are migrated. As a result of partial diffusion, some species gets extinct [20]. This is necessary in Biogeography Based optimization as species represent the independent variables of a cost function and each island represents a candidate solution to an optimization problem.

Islands with high value for Habitat suitability Index not only have a high emigration rate, but they also have a low immigration rate because of the high number of inhabitants in that island. Moreover, high HIS result in death of migrant due to high competition for resources. Islands with low value of habitat suitability index have high value of immigration rate as a result large species will immigrate to these islands [20]. This happens because of low population, hence there will be less competition but there will be increased diversity. When there will be high immigration on the island having low HSI, its HSI value will increase. In our work, λ means immigration probability and δ means emigration probability.

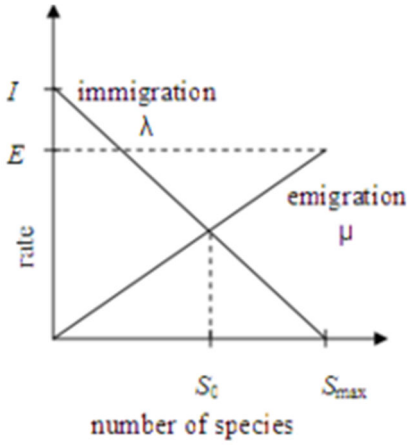


Figure 1: The variation of immigration and emigration probabilities in BBO

We are to follow a linear based BBO based on linearity in functioning of the complex system. Is there are m species of an individual, its immigration rate is α_k and its emigration rate is β_k.

$$\alpha(k) = I\left(1 - \frac{k}{Smax}\right) \qquad (1)$$

$$\beta(k) = \left(\frac{Ek}{Smax}\right) \qquad (2)$$

In the above equation I and E are the maximum immigration and emigration rates. S (max) is the maximum number of species [14]. The individual with large value of k will have lower immigration rates and higher emigration rates. It will be considered as a good individual. Now, those individuals which have lower fitness values will have higher values for immigration and will henceforth will accept values from members with higher fitness [14]. Due to this process, the algorithm approaches towards developing an optimal Pareto front.

In order, to show BBO's efficiency in solving various optimization problem we have taken Travelling Tournament Problem (TTP) [29]. Travelling Tournament Problem (TTP) is a type of mathematical optimization problem in which the problem is the sequence the series of team in a manner that

1. Each team plays every other team twice, once at home and once in the other's stadium.
2. No team plays the same opponent in two consecutive weeks.
3. No team plays more than three games in a row at home, or three games in a row on the road.

A matrix is provided of the travel distances between each team's home cities. All teams start and end at their own home city, and the goal is to minimize the total travel distance for every team over the course of the whole season [29].

3.1.1 Generic Algorithm of BBO

- Generate the population using random permutation. The population is created with a size equal to team size.
- Define the HSI for each habitat.
- Set the maximum generation limit
- For k=1 to maximum generation
- For i=2 to size(population)
- present_cost=present_cost-population(i).cost
- population(i).species_count=present_cost
- End for
- Determine the rate of evolution and rate of extinction
- Determine rate of growth and rate of decline of specification defined above.
- Use rate of decline to identify emigrating habitat.
- For j=1 to size of team
- Choose rate of growth as immigrating habitat.
- Perform following modification
    - Choose optimal SIV.
    - Calculate probability for immigrating habitat
    - Calculate probability for emigrating habitat
- Choose the best probability for optimal solution.
- If SIV of immigrating habitat gets selected, store j (its index) to replace in index array [ ].
- End for

- Value at jth index of immigrating habitat is stored in temporary variable and value of xth SIV at jth index of emigrating habitat gets selected and stored in jth index of immigrating habitat and we search where value of xth SIV exists in immigrating habitat excepts at jth location because it is the replaced ones, at that location store temporary variable value.
- End For

3.1.2 Discussion

In the above algorithm, we generate a population using randperm() function in MATLAB and define the HSI of habitats and maximum generation for which BBO will be executed. Determine the present cost randomly using random number generator. Populate the species count based on present cost. We then determine define the rate of growth, rate of decline, rate of evolution and rate of extinction [16].

We take rate of growth of habitat as the parameter to determine the immigrating habitat. We calculate the emigration and immigration probability of each habitat. We select the best probability for optimum solution. Store the SIV of the immigrating habitat [17]. Perform mutation in the array containing selected SIV's from various habitats.

3.2 Extended Biogeography Based Optimization [EBBO]

Remote sensing image classification in recent years has been a proliferating area of global research for obtaining geo-spatial information from satellite data. In Biogeography Based Optimization (BBO), knowledge sharing between candidate problem solutions or habitats depends on the migration mechanisms of the ecosystem [20].

In this chapter of thesis, an extension to Biogeography Based-Optimization is proposed for Travelling Tournament Problem by modifying the migration module of Biogeography Based optimization. The new migration operator of BBO incorporates logarithmic function and is hybridized with simulated annealing [30]. It is observed in recent literature that logarithm migration curves better represent the natural migration phenomenon as compared to the existing approach of using linear curves. The hybridization by simulated annealing is an essential step in modified BBO as simulated annealing method thrives to achieve global optimal solution and avoid local minima [30].

The motivation of this approach is to apply this realistic migration model in BBO, from the domain of natural computing [21]. The adopted approach calculates the migration rate using Elitist criteria. EBBO can be applied to test suite optimization (minimization, selection and prioritization). Results showed improvement over existing BBO's application on test suite optimization problem.

3.2.1 Generic Algorithm of EBBO

- Generate the population using random permutation. The population is created with a size equal to team size.
- Define the HSI for each habitat.
- set the maximum generation limit

- For k=1: maximum generation
- For i=2: size(population)
- present_cost=present_cost-population(i).cost
- population(i).species_count=present_cost
- End for
- Determine the rate of evolution and rate of extinction
- Determine rate of growth and rate of decline of specification defined above.
- Use rate of decline to identify emigrating habitat.
- For j=1:size of team
- choose rather of growth as immigrating habitat.
- perform following modification
    - Choose optimal SIV.
    - Calculate probability for immigrating habitat
    - Calculate probability for emigrating habitat
- Choose the best probability for optimal solution.
- If SIV of immigrating habitat gets selected, store j (its index) to replace in index array [ ].
- End for
- Value at jth index of immigrating habitat is stored in temporary variable and value of xth SIV at jth index of emigrating habitat gets selected and stored in jth index of immigrating habitat and we search where value of xth SIV exists in immigrating habitat excepts at jth location because it is the replaced ones, at that location store temporary variable value.
- End For
- Use the generated sequence as input to simulated annealing module in order to perform global optimization.

3.2.2 Discussion

In the above algorithm, we generate a population using randperm() function in MATLAB and define the HSI of habitats and maximum generation for which BBO will be executed. Determine the present cost randomly using random number generator. Populate the species count based on present cost [46]. We then determine define the rate of growth, rate of decline, rate of evolution and rate of extinction.

We take rate of growth of habitat as the parameter to determine the immigrating habitat. We calculate the emigration and immigration probability of each habitat. We select the best probability for optimum solution. Store the SIV of the immigrating habitat [46]. Perform mutation in the array containing selected SIV's from various habitats.

Adding to the original algorithm of BBO, the method of simulated annealing in order to achieve a set of solutions that are globally optimized. After the mutation process of BBO, we apply simulated annealing, for finding an optimal solution rather than best solution. Extended BBO with simulated annealing, performs better than BBO because, simulated annealing technique finds good solution if the search space is discrete, which is in TTP problem.

3.3 Test Suite Optimization

A Test suite is defined as a container of test cases about a scope or problem. The set of test cases are defined as the modules in the software under test. A test suite provides reports about the efficiency of the test cases and can hold any of the states, Active, In-process and Successful [24].

A test case can be augmented to any test suite and before the formation of the test suite, there is formulation of test plans which shows the execution cycle of the test suite [25]. A test suite consist of large number of test cases.

A test suite consist of test cases which are either functional or non-functional test cases.

As said above, a test suite is said to be successful if it completes the testing process by comparing each test case with the corresponding exit condition [26]. In order to define the completion of the testing process we need to define the Test Completion Criterion.

Examples of Test completion criterion are :

- A defined amount of coverage percentage has been achieved.
- Prioritization of Test cases
- Minimization of Test Cases.

Significance of Test completion criterion are :

- It is essential to stop the testing process.
- It is essential is to achieve certain level of quality of the project.
- It is also help in defining the amount of resources involved in testing cycle.

There are various approaches to testing viz.

- Approaches which are consultative in nature.
- Model based Approaches: In this approach the test process of the software are based on UML diagrams like State Chart Diagram, Activity diagram etc.
- Risk Based Approaches: here the test cases are prioritized based on the risk determined during the requirement elicitation phase.
- Traditional and Sequential approaches : These approaches are based on some model like waterfall model, which are sequential in nature
- Heuristic Approaches: These are intelligence based approaches that are based on some fitness functions, probabilities, crossover, mutation and selection. Since they are unaware of the problem nature, they are well suited in solving the problem due to their adaptive behavior. In our thesis we focus on heuristic approaches to solve Test suite optimization problem.

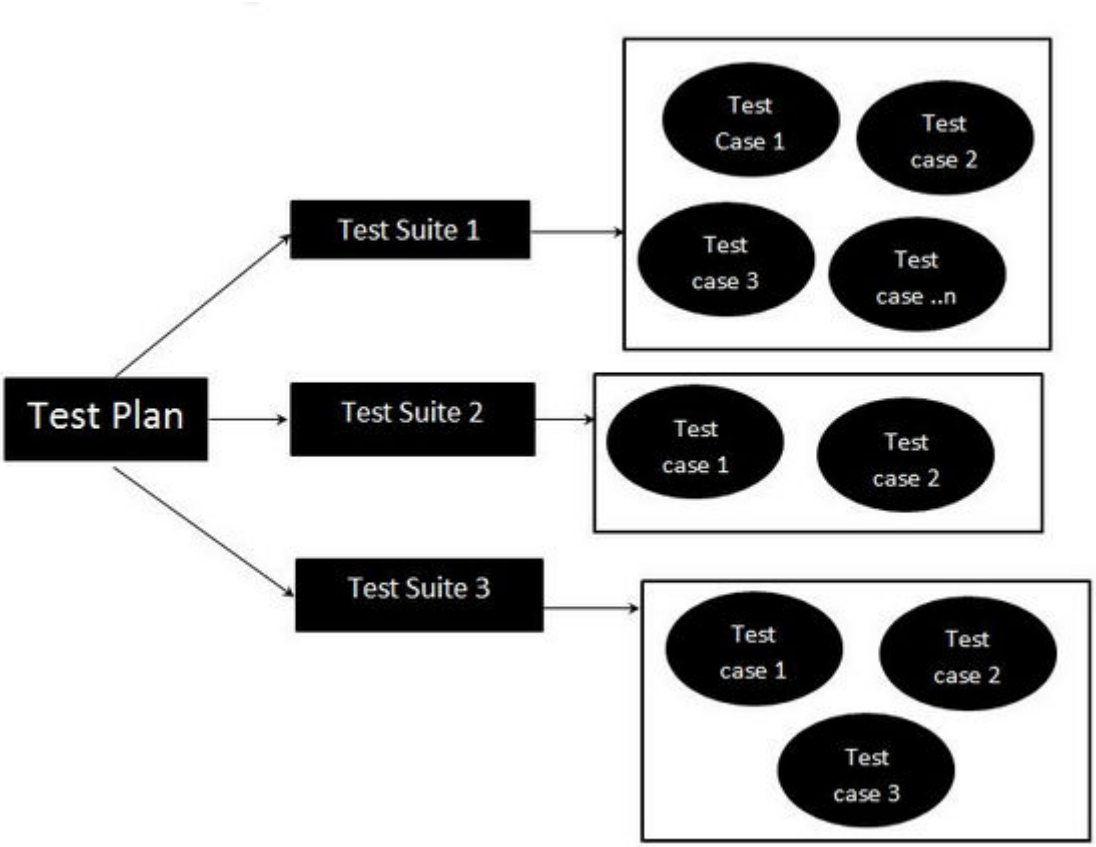


Figure 2. Test Suite Diagram

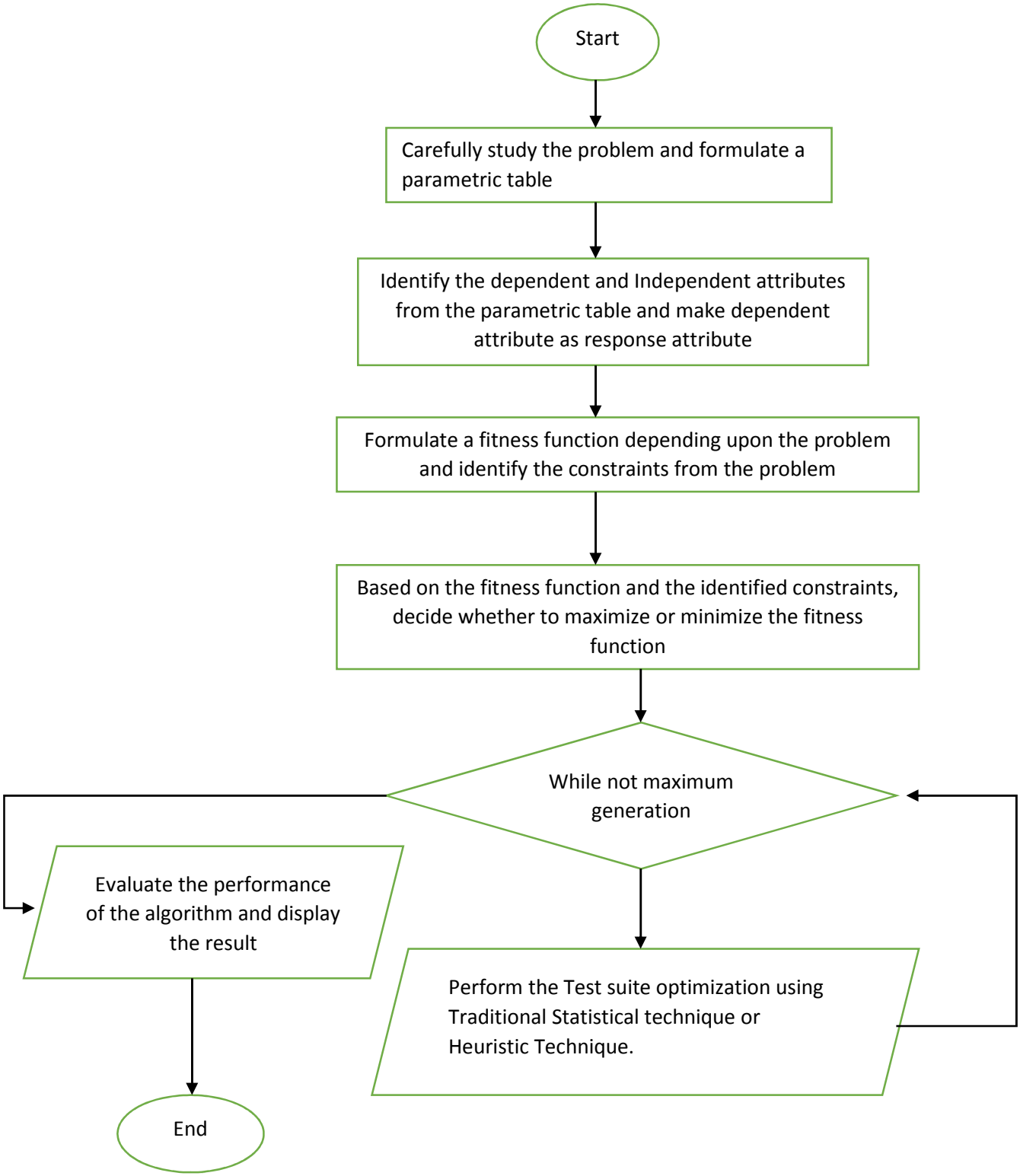

Start
Carefully study the problem and formulate a parametric table
Identify the dependent and Independent attributes from the parametric table and make dependent attribute as response attribute
Formulate a fitness function depending upon the problem and identify the constraints from the problem
Based on the fitness function and the identified constraints, decide whether to maximize or minimize the fitness function
While not maximum generation
Evaluate the performance of the algorithm and display the result
Perform the Test suite optimization using Traditional Statistical technique or Heuristic Technique.
End

Figure 3 : Generic Flow Chart of Test Suite Optimization Problem

Test suite optimization problem is considered as a NP Hard problem solvable in exponential time by some traditional statistical algorithm [34]. In order to make Test Suite Optimization problem solvable in polynomial time we employee meta-heuristics algorithms like Ant Colony Optimization, Particle Swarm Optimization etc. The process of solving test suite optimization problem using heuristics is :

- **Careful study of the problem and formulation of parametric table:** In this step, we analyze the problem domain and the software test cases that are to be optimized. We perform the needed job by formulating the parametric table.

  In a parametric table, we state all the attributes that are influencing the problem under test. The table is in the form of matrix, with rows, defining the test suites and columns define the modules of the software.

- **Categorize the attributes:** In this process, we classify the attributes in the parametric table as dependable attributes and independent attributes. Based on, whether we are into supervised learning or unsupervised learning we identify a label attribute or response attribute [33].

- **Formulate the fitness functions and constraints:** a fitness function is a formula, defined based on the dependent and independent attributes. The fitness function is tested based on constraints, derived from the problem domain [35].

- **Type of Optimization Operation:** In this stage of optimization process, we employee the fitness function , to be operated over the generation and each generation either maximize the fitness function or minimize the fitness function. The number of generations and the change in fitness function value are important parameters, in order to prevent saddle point, local minima, over fitting and under fitting [38].

- **Optimization Process and Evaluation :** At this stage, we use either statistical technique like Support Vector Machine or heuristic technique like Neural Network, Fuzzy Logic etc, for solving the optimization problem. The results are evaluated using some known metrics like Mean Square Error (MSE), Root Means Square Error (RMSE) etc. followed by display of the result.

In the problem of Test suite optimization we have defined Fault Detection Rate [FDR] as the fitness function and complete test coverage, redundant test case and unworthy test case as the constraints against which we minimize and prioritize the test cases in the test suite.

Following are the important points about Test suite optimization necessary for understanding this thesis :

- Regression Test Selection [RTS] is an important problem under Test suite optimization. RTS involves, identification of reusable test cases, retestable test cases and obsolete test cases. RTS also involves formation of new test case from those test cases which are unable to test a particular module.
- The process formation of test plans, test strategy and test suite should be carried out in parallel with requirement, design and coding stage so that test suites can be designed accurately and reflect the need of the customer [18].
- RTS also enable the software tester to include only those test cases which are efficient and useable. RTS enable robustness in the test suite by clearing out unworthy or redundant test cases from test suites [17].
- Identification of appropriate metric and performing analysis of the results helps in defining the effectiveness of the regression test suite. The declaration of priorities of the test case based on FDR state the effectiveness of test suite and algorithm used in deriving those results [23].

In the next chapter we try to adapt various meta-heuristic algorithm like Harmony search, ACO and firefly in solving TSO problem. We also juxtaposes Support Vector Machine, a statistical technique in solving TSO, so as to bring in to light clear comparison. We then, discuss in detail about adapting BBO and EBBO for TSO.

Chapter 4. Early Meta-Heuristics Algorithms and Support Vector Machine for Test Suite Optimization

Meta-heuristic Algorithms are procedures that have the tendency to provide near good solution to any optimization problem. These algorithms have the power to attain a solution point under the influence of limited information and computational capacity [8]. Meta-heuristic algorithms are far better than traditional algorithm as they does not possess pre-notion about the problem and try to learn from the solution during the progress of finding an optimal solutions [14]. The fitness functions on which meta-heuristic algorithms work are generally indifferent to differentiability and smoothness in curves.

Test suite optimization involves migration, selection and hybridization which are aptly brought into picture by meta-heuristic algorithm. The meta-heuristic algorithm taken into consideration are Biogeography based optimization, Modified Biogeography based optimization using simulated annealing, Harmony Search, Ant Colony Optimization and Firefly.

Based on the time frame and application domain in which these swarm based algorithm were applied in the field of software testing, we classify the meta-heuristic algorithms under two categories viz. Early approach and Recent approach.

The category of early approach incorporates Firefly algorithm, Harmony Search, Support Vector Machine (SVM) and Ant Colony Optimization (ACO).

Recent approach incorporates Biogeography Based Optimization, Simulated Annealing modified Biogeography Based Optimization and Harmony Search.

This chapter describes the algorithm of Harmony Search, Ant Colony Optimization, Firefly and Support Vector Machine which are adapted to Test Suite Optimization. Following it, chapter 5 describes the algorithm and pseudo code of BBO and EBBO which are adapted to Test Suite Optimization.

The process of TSO is carried out using real world test suite JBOSS application server. The application server has 10 modules for which 10 test cases were formulated. Based on FDR values and module coverage of test cases we rank the test cases. The ranking and FDR values assist in test suite minimization and test suite prioritization. More detail of which is described in chapter 6.

4.1 Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a meta-heuristic algorithm used to solve NP Hard optimization problems. ACO is inspired by the pheromone secretion of ants, which forms the numerical information that will provide solution to the problem. ACO involves formation of acyclic graph based on circuitous communication between ants via pheromone trails [16]. The pheromone content information is used to construct the solution using probability statistics.

Moreover, during the ACO's search procedure the ants try to adapt themselves showing learning experience. ACO has found it application in various domains such as Network, in

which it is used in NS2 simulator, routing packets, Mixed Pixels Resolution in Remote sensed images, Bioinformatics in which it is used in DNA sequencing and DNA matching. Moreover, ACO has been used in solving traditional colour problem, 3-SAT problem and travelling salesman problem.

Application of ACO in software engineering is novel and is treading fast as complex problem in software engineering like software quality estimation, software requirements prioritization, software project time line design and test suite prioritization can be solved in less time with less complexity [46]. ACO has been applied in the process of generation of test suites for state based software testing. A state based dynamic graph of the software that us under test was assessed using a group of ants to generate optimal test suites [17]. Prioritization of test cases/suites is done so that maximum faults can be recovered in minimum time [18]. Since nature inspired algorithm are efficient in time constraint optimization problems, ACO serves better in prioritization of test cases. Below, we have provided an algorithm of ACO for solving Test Suite Optimization (TSO) problem. In solving TSO, the word pheromone is synonymous to FDR (fault detection rate).

4.1.1 Algorithm of ACO for solving TSO

i. Formulate a parametric table, with rows defining the test cases and columns defining the modules of JBOSS.
ii. Define the input function trajectory
iii. Define the output function trajectory
iv. Define initial pheromone value
v. Place each ant on initial state with empty memory.
vi. While not the termination condition
    do
    a. Construction of Ant Solution : the ant move from its initial position to the prospective position based on the probability values which depend on attractiveness and pheromone level.
    b. Apply guided local search (Heap Search, Binary Search etc).
    c. Check the best tour :
    d. If the transition of ants showed improvement, update it.
    e. Update Trails :
    f. Evaporate a fixed amount of FDR from each path
    g. For each ant, do FDR update
    h. Reinforce the ant cycle with elitist ants
    i. Create new population based on FDR values using :
    j. Structured crossover.
    k. Structured mutation.
vii. End While
viii. Rank the test cases based on their FDR values and coverage of modules that is the test cases with maximum number of 1's get the highest rank (1).
ix. Test cases with FDR value 0 are obsolete test cases and should be removed from further study.

4.1.2 Discussion on ACO Algorithm

In ACO, an ant is a simple agent for computation, tracking whose movement results in optimization of the problem. ACO has the capability to arrive at the solution iteratively. In each iteration, there is change in position from, say x to y, where y is closer to the optimal solution. The movement of any ant from state x to state y depends on :

- Attractiveness (xy) : priori probability of that move based on some heuristic.
- Trail Level $\tau(xy)$: posteriori probability is the indication of desirability of the move.

During each generation in test suite optimization, we try to update FDR of each test case, with simultaneously keeping older values, forming a trail and new values are responsible for attractiveness. Hence the movement of a test case from FDR x to FDR y is stated through following probability formula:

$$p^k(xy) = \frac{t^\alpha(xy)\eta^\beta(xy)}{\sum t^\alpha(xy)\eta^\beta(xy)}$$

$t^\alpha(xy)$ is amount of pheromone (FDR value) in the trail array whereas $\eta^\beta(xy)$ is amount of pheromone that determine the attractiveness quotient. α And β are parameter values for controlling the influence.

4.2 Firefly Algorithm

Firefly is a meta-heuristic algorithm used to solve combinatorial optimization problem. Firefly algorithm is based on flashing of light of fireflies, which is the reason for attraction and forms the numerical information for solving the problem. Mapping of firefly algorithm to the domain of software engineering is one of the task of this research. All the test cases in a test suites are the fireflies and they all possess the same traits, hence are unisex [19].

In, fireflies' attractiveness is influenced by brightness that is those test cases in a test suite which show similar behaviour to test cases in other test suite, for instance two test cases of different suites are responsible for finding the fault in a module of JBOSS, hence these two test cases need to be together. The brightness of the test suite are identified by the modules they test and fault detection rate of the test suite, which in our problem space serves as objective function.

When two test cases in different suite or same suite show similar properties like fault detection rate or identifying number of redundant test cases or any properties listed in table 5 are not attracted to each other. The firefly algorithm showed mediocre result in test suite optimization problem. Firefly algorithm has been used in optimizing test case paths using objective function and guidance matrix, henceforth, it generates critical and optimal paths [19]. Below we have provided an algorithm of Firefly for solving Test Suite Optimization (TSO) problem.

4.2.1 Algorithm of Firefly for solving TSO

i. Begin

ii. Formulate a parametric table, with rows defining the test cases and columns defining the modules of JBOSS.
iii. Define the Objective function of the problem.
iv. Determine the light Intensity I_i at x_i which is determined using $f(x_i)$.
v. Determine the light absorption coefficient ψ
vi. While not stopping condition of max generation (i)
    a. For j= 1: number of fireflies
       If $I_i > I_j$
            Shift the firefly i towards j in some d dimension via Levy Flights
       End if
    b. The value of Attractiveness varies with distance via $e^{-\psi r}$.
    c. Test the new solutions based on fitness functions and update the light intensity array
    d. End j
vii. Perform ranking of Fireflies and the best current firefly.
viii. End while
ix. Analyse and visualize the results.
x. End
xi. Rank the test cases based on their FDR values and coverage of modules that is the test cases with maximum number of 1's get the highest rank (1).
xii. Test cases with FDR value 0 are obsolete test cases and should be removed from further study.

4.2.2 Discussion on Firefly Algorithm

In this algorithm, firstly, we develop an initial population of the problem. The tuples of the problems are called the fireflies. We define an objective function, also called fitness function based on the problem domain. The parameter light intensity is vector, which is a measure of brightness of fireflies. The change in attractiveness is defined by $e^{-\psi r}$. All the fireflies are unisexual, hence any firefly can be attracted towards all the fireflies. The firefly or the test case with high FDR will attract those test cases with comparatively low FDR. Those test cases which have FDR much higher, move randomly in the generation and are acted upon by other fireflies/test cases in next generation. Based on Firefly behaviour we can modify the existing test suite in order to develop those test suite which provide complete coverage of the software under test.

4.3 Support Vector Machine

Support Vector Machine (SVM) is linear classifier which does not depend on probability values of the attributes along the classifying surface. A standard SVM model comprises of a set of input attributes and perform prediction and classification along the linear hyperplane.

More elaborately, SVM model involves formation of hyperplane or a sets of hyperplanes in higher (infinite) dimensional space. This hyperplane helps in classification, regression and prediction. A properly constructed hyperplane serve as an excellent classifier. How do we say whether the formed hyperplane is an excellent classifier or not? If the separation distance of

the hyperplane and the training points is the largest, we can state that, SVM has performed good classification [45]. This is so, because, larger the functional margin, lower is the error of classifier.

Many optimization problems in the real world are not linearly separable problems, so in order to solve such problems, we try to map the lower dimensional surface be mapped to higher dimensional surface sufficient to separate the training data points.

In the software testing domain, the modules in the software are by default independent attributes until and unless percentage of coupling and cohesion are specified. In JBOSS server, all modules are independent and thus construct an N dimensional problem. SVM developed for test suite optimization work on the concept of supervised learning and predict the priority of the test cases based on their influence on mutant test cases in various modules in JBOSS. The formulation of SVM for TSO is

$$\min(i)\ |\langle W|X^i\rangle + b| = 1; where\ i\ defines\ N\ dimensions$$

For computation involved in mapping from lower dimension to higher dimension, SVM provide lower computation through dot product of the attributes in their original space by defining appropriate kernel for the problem [45]. Hyperplanes, in the higher dimensions are group of points whose inner product with a vector in same space is constant. Following this, we have provided an algorithm of SVM for solving TSO problem.

4.3.1 Algorithm of SVM for solving TSO

The SVM is linear based classifier that can be modified to perform non-linear classification by incorporating evolutionary mechanism of classification using Genetic Algorithm and swarm based intelligence using Particle swarm optimization. Below, we have defined an algorithm of simple SVM. The algorithm defined, stops when the error is in the range [0, 1).

i. Formulate a parametric table, with rows defining the test cases and columns defining the modules of JBOSS.
ii. Identify the dependent attribute which will be a response attribute
iii. While violating points are present do
iv. Identify the violator
v. Candidates_SVM= union(Candidates_SVM, violator)
vi. If rand $\alpha(p) < 0$ because of addition of c to S then
    i. Candidates_SVM=Candidates_SVM/p
    ii. Repeat loop till all violator points are not cropped.
vii. End if
viii. End while.
ix. Rank the test cases based on their FDR values and coverage of modules that is the test cases with maximum number of 1's get the highest rank (1).
x. Test cases with FDR value 0 are obsolete test cases and should be removed from further study.

4.3.2 Discussion

- Candidates_SVM is an array containing points after they are pruned if they are violator points.
- S is the support vector set.
- SVM takes the data set as input with predefined priority as the class label and generates new priority based on its learned algorithm.

4.4 Harmony Search

Test suite minimization and prioritization processes are essential because they result in early fault detection and also reduces the software cost. Minimization process reduces the complexity of the testing cycle resulting in fast regression testing of the software [23]. Traditional nature inspired techniques like ACO, Firefly has been used in TSO but they lack the strength of harmony search which is exploration of the search space.

Harmony Search is a nature inspired meta-heuristic technique which uses harmonic waves in the environment and response of human brain towards them to generate a set of solution that can be used to optimize the problem at hand [24]. Harmony search algorithm consist of a harmony memory (HM) having harmony memory size of problem domain (2-D vector). HM is responsible for storing the candidate solution which are randomly generated by the search space. We have provided an algorithm of Harmony search, which has been modified for solving TSO problem.

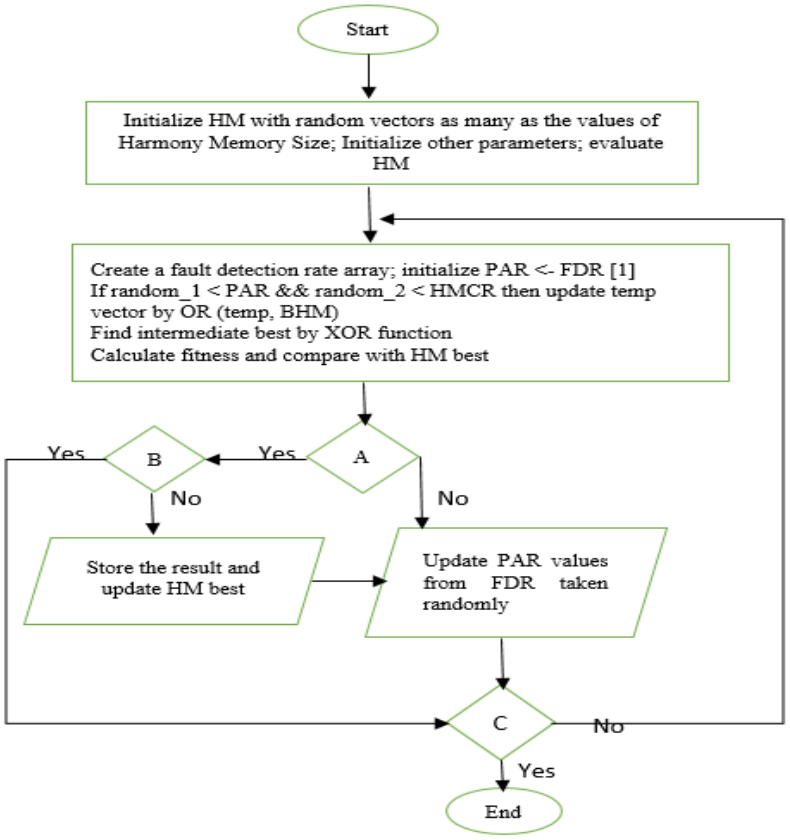


Figure 4. Flow chart of Harmony Search for TSO, A: Is fitness > HM Best, B: Complete Coverage, C: Max Iteration

4.4.1 Pseudo code of Harmony Search for solving TSO

i. Formulate a parametric table, with rows defining the test cases and columns defining the modules of JBOSS.
ii. {Data} k represent fault covered and execution time of each test case, k= {1, 2, N}
iii. {data2} k represents in Boolean table, effect of test cases on different modules
iv. Define trial array of size binary matrix (row) and binary matrix (columns)
v. Define Harmony Memory Best as a column vector
vi. Define Harmony Memory and Binary Harmony Search
vii. set Intermediate solution array : Column Vector
viii. set Result array : Size of Harmony Memory
ix. set var1 : data{FC}, var2 : data{ET}
x. set Fault Detection Rate matrix : Row vector [var1/var2]
xi. set random1 between [0,1] and random2 between [0.5,1]
xii. while(generation limit not achieved)
    a. do
    b. while(i < binary row length)
    c. do
    d. if(random1 < HMCR)
    e. then update trial
    f. if (random2 < PAR)
    g. then set temp variable : random number [1, row length]
        i. while (k < binary matrix column length)
        ii. do
            1. update trial : OR (trail and Binary Harmony Memory)
        iii. end
    h. end end
    i. while (not binary column length)
    j. do
    k. intermediate best : xor (trial , Harmony Memory Best)
    l. end
    m. fitness value : sum(intermediate best)
    n. if (fitness value == binary Column length)
    o. then print("the best test case")
xiii. end
xiv. else
xv. then
xvi. fitness of trial array : sum(trial)
xvii. fitness of Harmony Memory Best array : sum(Harmony Memory Best array)
xviii. if (fitness of trial array > fitness of Harmony Memory Best)

then result <- trial array

xix. else

xx. then

result<- Harmony Memory Best

xxi. end

xxii. end

xxiii. end

xxiv. increment iteration ; end

xxv. Rank the test cases based on their FDR values and coverage of modules that is the test cases with maximum number of 1's get the highest rank (1).

xxvi. Test cases with FDR value 0 are obsolete test cases and should be removed from further study.

4.4.2 Discussion on Harmony Search for solving TSO

This section explains in steps the process of applying harmony search on the data set consisting of two vectors, first data set consist of test cases with fault covered and execution time while the second data set is in Boolean form which is obtained after applying the test cases on various modules of application.

- Create a database using MySQL (preferably due to its connectors), consisting of the desired data set. First data set is named as HM and second as BHM (Binary Harmony Memory).
- Initialize the harmony memory (HM) equal to size of dataset.
- Define the Harmony Memory Consideration Rate (HMCR)
- Define the generation limit for the algorithm.
- Create a fault detection rate array based on fault covered and execution time.
- Entities in fault detection rate array are selected as values for pitch adjustment rate for different generation.
- Initialize pitch adjustment rate with first individual of fault detection rate array.
- Define two random variable with value ϵ [0, 1]. Alter the values with each generation.
- Check if random variable 1 < HMCR then update the temporary array with BHM's individual with index 1
- If random variable 1 < PAR with update temporary array by OR function being applied to temporary array and Binary harmony memory.
- Determine the intermediate best individual and compare it Harmony Memory's best individual.
- If fitness (intermediate best) > Harmony Memory Best and provides complete coverage then we stop searching in further generation.
- If fitness (intermediate best) > Harmony Memory Best and does not provide complete coverage then the result array is updated with values from intermediate best else result array is updated with harmony memory best.
- Lastly, harmony memory best is updated with the individual from result array and next generation is initiated.
- If maximum generation limit is achieved then the algorithm is stopped from further execution.

Chapter 5. Recent Approaches to Test Suite Optimization

In this chapter we will cover two meta-heuristic algorithms viz. Biogeography Based Optimization and Extended Biogeography Based Optimization. These algorithm are a part of computational intelligence technique and performs optimization which is inspired by the nature.

The process is carried out using real world test suite JBOSS application server. The application server has 10 modules for which 10 test cases were formulated. Based on FDR values and module coverage of test cases we rank the test cases. The ranking and FDR values assist in test suite minimization and test suite prioritization. More detail of which is described in chapter 6.

5.1 Biogeography Based Optimization (BBO)

BBO a meta-heuristic algorithm was developed by [46] for solving mathematical optimization problems. In this section we will modify and adapt BBO for Test suite optimization. TSO, imbibes in itself, test suite selection, prioritization and minimization.

In the current scenario of TSO, habitability in BBO is defined as: SIV's are the independent variable and HSI's are the dependent variable. In order to solve multi-objective nature of test

suite optimization, we try to map the problem domain to single objective. We tried to use Fault detection rate as cost function which can be maximized or minimized as per the problem.

The prominent factors that affect the working of the algorithms are:

- The migration operator
- The mutation operator
- The probabilities of immigration and emigration

Firstly, the migration operator works based on the fitness values of the individuals. When we do sorting of the individuals we are taking into account particles from different habitat. Based on the curvature formed from immigration and emigration probabilities also affect the working of Biogeography Based Optimization algorithm.

5.1.1 Algorithm of Migration modified for TSO

i. For i=1: number of individual (N)
ii. Do
iii. Select _individual < probability_immigration
iv. If random < probability_immigration
v. For j=1:number of individuals (N)
vi. Select_individual < probability_emigration
vii. If random < probability emigration
viii. Random selection of a variable X(j,k) from X(j).
ix. Replacing X(i,k) with X(j,k).
x. End if
xi. End for
xii. End if
xiii. End for

Secondly, the mutation operator is often used in an optional manner, in cases like when random probability is less than mutation probability or when the archive array is not showing diversity in the population. The mutation operator is inversely proportional to the number of species and probability of the habitat. If individual has lower value for probability of speciation then it has higher value of probability of mutation and hence it can be considered as a better individual. Similarly, the individual with higher values for speciation will have lower mutation probability rates and hence it will not easily mutate with other individuals.

Hence mutation operator brings in diversity in the population and provides assurance of convergence with development of Pareto fronts and not just one optimal solutions.

5.1.2 Algorithm of Mutation modified for TSO

i. For i=1: number_of_individual (N)
ii. Do
iii. If random < mutation_probability
iv. Replace X(i) with a random SIV
v. End if
vi. End for

5.1.3 Pseudo Code of BBO for solving TSO

i. Formulate a parametric table, with rows defining the test cases and columns defining the modules of JBOSS.
ii. $\{Data\}_k$ represent fault covered and execution time of each test case, k= {1, 2, N}
iii. $\{data2\}_k$ represents in Boolean table, effect of test cases on different modules
iv. Identify maximum and minimum domain of values in fault covered field in {data}
v. Define an Elite array for selection best individual in each generation and set its size
vi. Define the cost array using {data} value
vii. Cost array <- Fault detection rate
viii. Mutation probability (σ) <- values [0, 1]
ix. While (not the maximum generation limit achieved)
    a. Define a Random number
    b. For each $\{data\}_k$, set the emigration probability δ_k proportional to fitness of $\{data\}_k$, $\delta \in [0, 1]$
    c. For each $\{data\}_k$, set the immigration probability λ_k , λ_k <- 1- δ_k
    d. {temp}<-{data}
    e. I: For each individual {temp}
    f. II: For each columns in Boolean table
        i. Use immigration probability to identify the individual which has be emigrated
        ii. Selection is based on roulette wheel (can be also with tournament selection)
    g. End II
    h. End I
    i. Based on σ mutate {temp} vector
    j. Define a new index for OR function from {temp} vector
    k. I: For each j in population size
    l. II: For each l in column of binary table
    m. New_binary_table <- OR({data2}[index][l], {data2}[j][l])
    n. End I
    o. End II
    p. Sort the new binary table
    q. Identify the elite solution in for next generation
x. End
xi. Rank the test cases based on their FDR values and coverage of modules that is the test cases with maximum number of 1's get the highest rank (1). This formulates a matrix for Test case prioritization
xii. Test cases with FDR value 0 are obsolete test cases and should be removed from further study results in test suite minimization. Also redundant test cases results in test suite minimization.

5.1.4 Discussion and Flow chart of BBO for TSO

Test Suite Minimization (T S M) is a technique to reduce the size of the test suite while maintaining the property of complete coverage. Some of the Test cases are redundant and only

increases the testing cost. Retaining such test cases does not bring about any significant change in the test process, rather running all the test cases consumes large amount of time. TSM finds the minimum set of test cases from the original test suite which satisfies all testing requirement. TSM also improves the testing process by reducing the time consumption. Biogeography based optimization (BBO) is a nature inspired algorithm that aims at improving the candidate solution based on some fitness function [21]. BBO is a meta-heuristic algorithm, since it includes many variation and does not make any assumptions about the problems. Due to this, BBO can be applied to wide class of problems, but Why Nature Inspired Algorithm? This is because nature inspired algorithms can be used to optimize multidimensional real-valued functions and problems [21]. Moreover, they doesn't incorporate gradient functions like gradient descent, newton rules, perceptron rules or relaxation procedures. Furthermore, they doesn't require the problem function to be differentiable, hence can be applied to discontinuous functions.

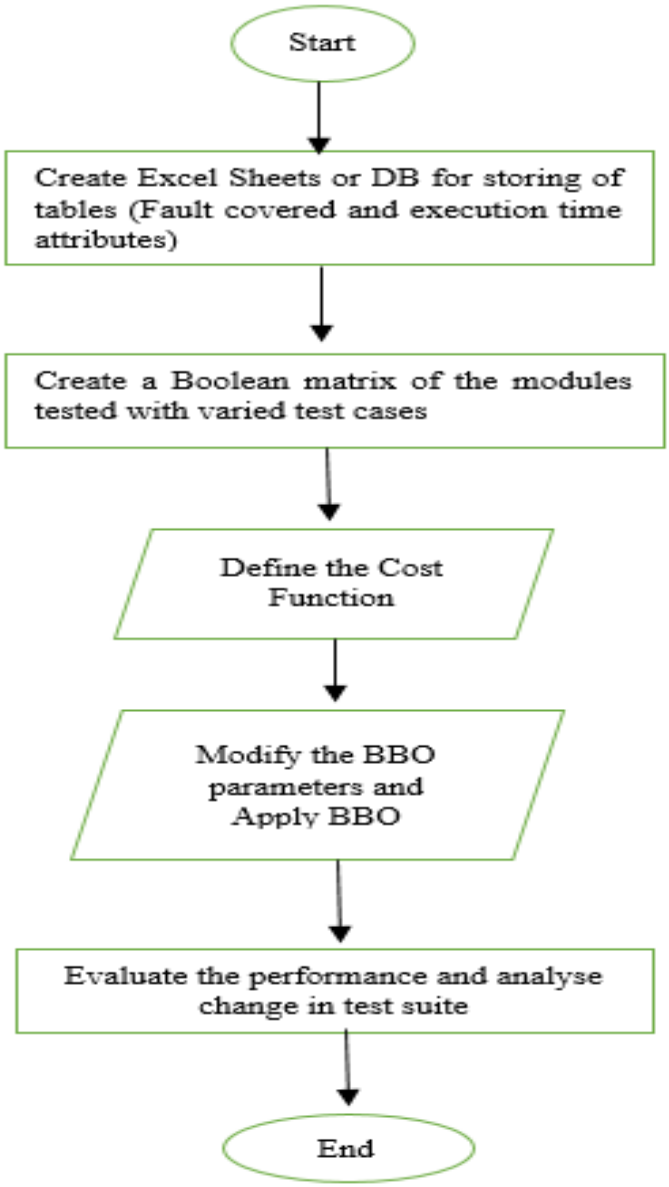


Figure 5. Flow chart of BBO

Flow chart Definition

- Create a database file containing the data set of the application on which testing has been performed. It is essential that you perform mutation testing on the data set. The result of the mutation test cases can be classified in a binary manner.
- Create a file consisting of m x n matrix where m represents the test cases and represents modules which are tested.
- Define the cost function of the BBO algorithm. In Test suite optimization the cost function is the fault detection ability. A fault detection ability is defined as the fault covered per unit execution time.
- Since Biogeography Base optimization is an open problem optimization algorithm with its generic set of parameters which need to be change as per the problem domain.
- BBO parameters to be altered as per problem domain
- Cost Function: Fault Detection Rate, that is Number of Faults covered per Execution Time.
- Mutation Probability: It is used to introduce diversity in the solution. The value of mutation is essential as high values will turn search into random search.
- Generation Limit: It defines the number of iteration of the BBO algorithm.
- Elite Solution List: It is multidimensional real-valued matrix which selects best candidate solution that will be preserved for next generation.
- Elite Cost List: the cost associated with the elite solution will be stored in minimum cost list.
- Minimum Domain and Maximum Domain: it defines the range of values in the attribute that behaves as Decision attribute in BBO Algorithm.
- Sorting: The sorting order should be either descending or ascending depending on the problem.
- Apply the Biogeography based optimization algorithm as stated in (Simon D.).
- Evaluate the performance by identifying the minimized set of test cases and priorities that can be assigned to the test cases.
- Furthermore, the result also defines the minimum cost, the stabilisation in the cost function after 10 generation.
- The algorithm can be applied to large data set consisting of 1000 X 1000 testing matrices.

5.2 Extended Biogeography Based Optimization

We now define the procedure of adapting Extended BBO for Test Suite Optimization and explain the same by defining an algorithm for Extended BBO and flow chart. We also touch upon simulated annealing, which has been used for global stochastic optimization.

5.2.1 Simulated Annealing

Simulated Annealing (SA) is a stochastic optimization technique which works towards global optimization in the search space. Simulated Annealing technique has been inspired by thermodynamic activity of the container under a given set of temperature, pressure and volume [22]. SA technique is initiated with initial solution followed by partial search over the state space. The selection and rejection of moves during the search is hold over by metropolis algorithm.

$$SA_{i+1} \leftarrow IF \ X \ SA_j$$

IF is the Improvement Factor which is in this case is a fitness function value. The fitness function for the TSO problem domain is fault detection rate. Since the fault detection rate judge the efficiency of the test cases, hence simulated annealing serves as global optimizer in BBO used to find global optimal solution.

5.2.2 Pseudo Code of Extended BBO for solving TSO

i. Formulate a parametric table, with rows defining the test cases and columns defining the modules of JBOSS.
ii. $\{Data\}_k$ represent fault covered and execution time of each test case, k= {1, 2, N}
iii. $\{data2\}_k$ represents in Boolean table, effect of test cases on different modules
iv. Identify maximum and minimum domain of values in fault covered field in {data}
v. Define an Elite array for selection best individual in each generation and set its size
vi. Define the cost array using {data} value
vii. Cost array <- Fault detection rate
viii. Mutation probability (σ) <- values [0, 1]
ix. While (not the maximum generation limit achieved)
    a. Define a Random number

b. For each {data} $_k$, set the emigration probability δ_k proportional to fitness of {data} $_k$, $\delta \epsilon [0, 1]$
c. For each {data}$_k$, set the immigration probability λ_k , λ_k <- 1- δ_k
d. {temp}<-{data}
e. I: For each individual {temp}
f. II: For each columns in Boolean table
    i. Use immigration probability to identify the individual which has be emigrated
    ii. Selection is based on logarithmic fitness curves
    iii. Hybridization of the solution by simulated annealing
g. End II
h. End I
i. Based on σ mutate {temp} vector
j. Define a new index for OR function from {temp} vector
k. I: For each j in population size
l. II: For each l in column of binary table
m. New_binary_table <- OR({data2}[index][l], {data2}[j][l])
n. End I
o. End II
p. Sort the new binary table
q. Identify the elite solution in for next generation

x. End
xi. Rank the test cases based on their FDR values and coverage of modules that is the test cases with maximum number of 1's get the highest rank (1). This formulates a matrix for Test case prioritization
xii. Test cases with FDR value 0 are obsolete test cases and should be removed from further study results in test suite minimization. Also redundant test cases results in test suite minimization.

5.2.3 Discussion and Flowchart of Extended BBO

This section provides the steps necessary to apply modified BBO for test suite optimization. The data set for test suite optimization consists of a binary matrix showing response of module to test cases and a matrix showing the performance of the test cases based on fault covered and execution time.

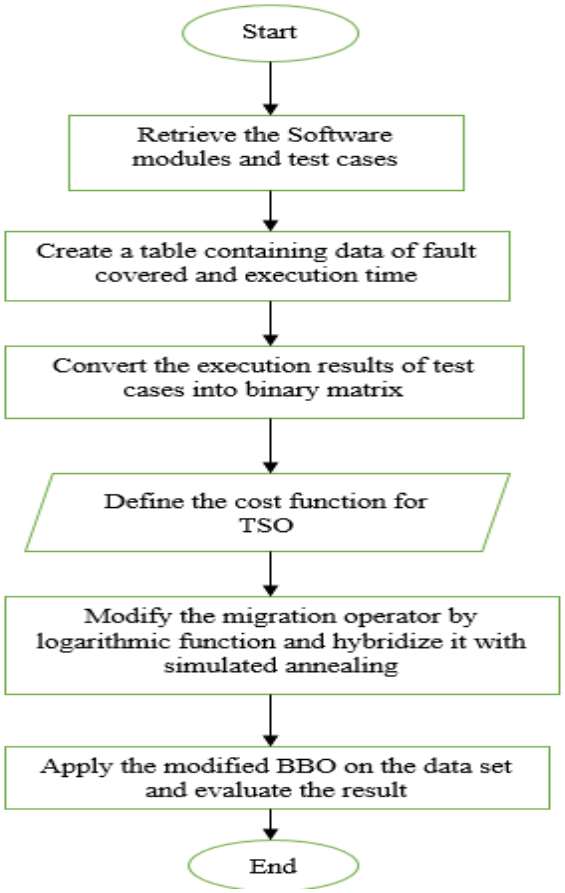


Figure 6: Flow Chart of Extended BBO

- JBOSS application server's modules were mutated and test cases that were formalized for testing were applied. A set of 10 test suites were formed for testing of 10 modules.

- Generated a matrix consisting of results of test cases when they are applied to the modules that is whether test cases have detected the error in the modules or not along with the execution time a particular test suite takes.
- Generated a matrix consisting of performance of the test cases that is fault covered and execution time. This matrix will serve as input in the derivation of cost function for test suite optimization using soft computing approach.
- Define the cost function that is to be maximized. The cost function with respect to the problem domain is fault detection rate. The efficiency of the test cases is determined by fault detection rate that is test case with high fault detection rate will be more efficient.
- In modified BBO, the migration operator is changed with logarithmic function and is hybridized with simulated annealing. Simulated annealing approach work towards global optimization.
- Apply the modified BBO on the data set and evaluate the performance by evaluating the binary matrix that will be changed by modified BBO.

Chapter 6. Dataset, Simulation Environment, Results and Discussion

6.1 About JBOSS Application Server

JBOSS is an open source application server which is based on JAVA-EE version. The job of JBOSS is not only to execute web based programs in JAVA but also implements JAVA-EE part inside the JAVA code. Since JBOSS, closely represents JAVA platform, which makes this server platform independent or cross platform, which means it can be executed in any operating system that supports JAVA.

In order to start JBOSS correctly, you need to define environment variables and perform a simple startup of JAVA VM to validate the working of JAVA and JBOSS.

To test the installation of JBOSS, move to bin directory and issue the following command :

“Run.bat”. The JBOSS directory contains various modules against whom test cases were formulated. The name of those modules are mentioned below.

JBOSS server consist of 10 modules viz, web module, meta-INF module, XML module, Processing Module, Integration module, Execution module, Platform module, resource management module, API module and Library Module.

6.2 Simulation Environment

The section provides information about the various tools used in deriving the results in this research. The parameters based on which the comparison was done are redundant test cases identified, worthless test case identified, complete coverage test cases identified, priority range creation, reduction in test suite and identification of untestable part of module. The data set was generated after the application of 10 test suites on 10 modules of JBOSS application server. Different parameters of meta-heuristic algorithms were taken in to consideration viz. mutation

probability, harmony memory consideration rate, pitch adjustment rate, elite solution, generation limit, cost function, emigration probability, and immigration probability [25].

6.2.1 What is Rapid-Miner?

Rapid-Miner is a software platform developed to fulfill the requirement for Machine learning, Data Mining, Text Mining, Image processing and Web Mining. The application domain that is served by Rapid-Miner is Business Analytics, prediction Analytics, Forgery Detection, Deep Data Analysis etc.

The tool provides the user set of modules and downloadable libraries that assist in research in computer science, provide education to facilitate research, training the model is essential to develop learned system and rapid application development due to fast integration with front end language IDEs.

Rapid Miner provide environment for visualization, optimization and training which are essential to bring about Data Mining.

6.2.2 What is R Programming?

R is a programming language that provides support for statistical computing and analysis. R programs work on R studio, IDE for R programming. The tool has the capability to plot any graph, which helps in better analysis of data. It is most widely used in Data Mining Community and Prediction Analytics Community.

R programming is an implementation of S programming along with some lexical semantics. R programming is created by Ross Ihaka and Robert Gentleman at University of Auckland. R programming has been compared with other statistical tools like Stata, SPSS and SAS.

R Programming works well in comparison with SPSS, Stata and SAS. R programming being open source is a threat to proprietary software like SAS, which is very costly.

Table 1: Simulation Environment for the Research work

S. No.	Simulation Variables	Value
1.	Programming Language	MATLAB, R
2.	Data Base	MySQL
3.	Data Analysis Tool	RAPIDMINER
4.	Data Set Repository	JBOSS and GENTOO

6.3 Results and Analysis

The section shows the results obtained after we have carried out the result in R studio using R programming. The plotting of the transition was done using GNU-Plot. The research was carried out in a system with 2 GB RAM, 256 GB HDD and 1.6 GHz core i3 Intel Processor.

Table 2: Parameters modified during application of Meta-heuristic technique for TSO

S.no	Parameters	BBO	Modified BBO	Harmony Search
1.	Mutation Probability	0.04	0.04	
2.	HMCR			0.7-0.95
3.	PAR			0.5-1.0
4.	Elite Solution	5	5	Row Vector
5.	Generation Limit	200	200	200
6.	Cost Function	Fault Detection Rate	Fault Detection Rate	Harmony Memory Best individual
7.	Emigration Prob.	0.133-0.944 (linear)	0.133-0.944 (logarithmic)	
8.	Immigration Prob.	0.05-0.866	0.05-0.866	

test cases	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	Priority
1	1	0	1	0	0	1	0	0	1	0	4
2	1	1	1	0	0	1	0	0	1	0	3
3	0	0	0	0	0	0	0	0	0	0	0
4	1	0	1	0	0	1	0	1	1	1	2
5	1	0	1	0	0	1	0	0	1	1	3
6	1	0	1	1	0	1	0	1	1	0	2
7	1	0	1	0	0	1	0	0	1	1	3
8	1	0	1	0	0	1	0	0	1	1	3
9	1	0	1	1	0	1	1	1	1	1	1
10	1	0	1	0	0	1	0	0	1	0	4

Figure 7. Test Suite Priority after BBO

The above figure shows the priorities of the data set. After simulation of the algorithm on the data set for 50 generation, we obtained such data set which BBO optimized. The priorities are ordered in ascending order with 1 as the highest priority and 10 as the minimum priority. It can be seen from the result that BBO has preserved the solution from high priority to medium priority. The result showed that test case 9 is much more efficient then rest of the rest cases. Test case 4 has least efficiency in testing the concerned application. Test case 3 has been reduced to zero which showed that test case 3 can be excluded during regression test suite selection. The above figure also showed that test cases 2, 5, 7 and 8 are redundant. Similarly, test cases 4 and 6 are redundant.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
1	1	0	1	0	0	1	0	0	1	0
2	1	1	1	0	0	1	0	0	1	0
3	0	0	0	0	0	0	0	0	0	0
4	1	0	1	0	0	1	0	1	1	1
5	1	0	1	0	0	1	0	0	1	1
6	1	0	1	1	0	1	0	1	1	0
7	1	0	1	0	0	1	0	0	1	1
8	1	0	1	0	0	1	0	0	1	1
9	1	0	1	1	0	1	1	1	1	1
10	1	0	1	0	0	1	0	0	1	0

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
1	1	0	1	0	0	1	0	0	1	0
2	1	1	0	0	0	0	0	0	0	0
3	1	0	0	1	0	0	0	1	0	0
4	1	0	0	0	0	0	0	1	1	1
5	1	0	0	0	0	0	0	0	1	1
6	1	0	0	1	0	0	0	1	0	0
7	1	0	0	0	0	1	0	0	0	1
8	1	0	0	0	0	0	0	0	0	1
9	1	0	0	1	0	0	1	1	0	1
10	1	0	0	0	0	0	0	0	0	0

Figure 8. Left figure shows test cases 2 and 9 after BBO while right shows test cases 2 and 9 before BBO

The above figure demonstrates that test cases 2 and test cases 9 are sufficient to provide complete coverage to regressing testing. It is evident from the figure 6 that module V5 has not be sufficiently evaluated during mutation testing hence no migration to that island has been made during Biogeography Based Optimization. It is seen that all modules {V1, V3, V4, V6, V7, V8, V9, V10} are completed covered by the test case 9 after migrating best variable from all the islands. It is also seen that all modules {V1, V2, V3, V6, V9} are covered by test case 2 after migrating best variable from all the islands.

	FC	ET
1	4	7
2	2	3
3	3	5
4	4	5
5	3	3
6	3	6
7	3	3
8	2	2
9	5	6
10	1	1

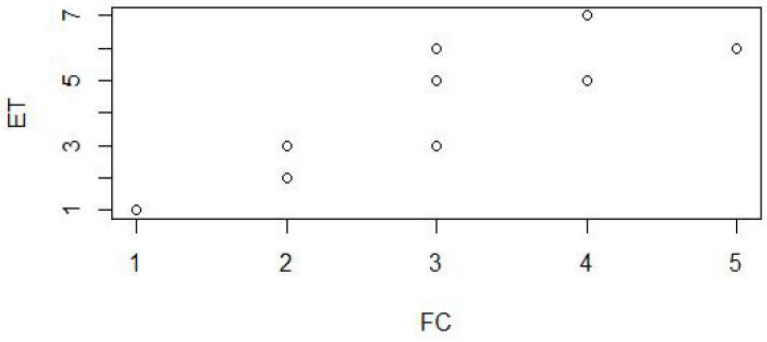


Figure 9. Test data of JBOSS, FC: Fault Covered, ET: Execution Time

The above figure 5 shows a sample test data which was used to during the research. FC stands for Fault Covered and ET stands for Execution Time. The highest domain value in FC is 4 and lowest domain value in FC is 1. The highest domain value in ET is 7 and lowest domain value is ET is 1.

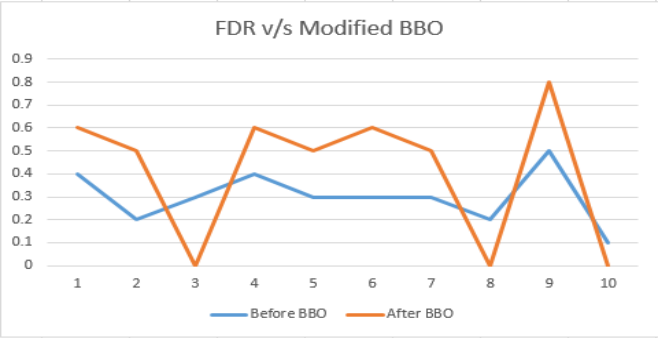


Figure 10. Fault Detection Rate (FDR) after modified BBO

Table 3. FDR Before and After BBO

S.no	Before BBO	After BBO
1.	0.4	0.6
2.	0.2	0.5
3.	0.3	0
4.	0.4	0.6
5.	0.3	0.5
6.	0.3	0.6
7.	0.3	0.5
8.	0.2	0
9.	0.5	0.8
10.	0.1	0

The above figure 8 and Table 3 shows Fault Detection Rate of the test cases of JBOSS Java Application Server. The Fault detection rate is used as cost function in BBO. The highest values in domain FDR is 1 and lowest values in domain FDR is 0.5. This FDR array is used as cost function for convergence in generation.

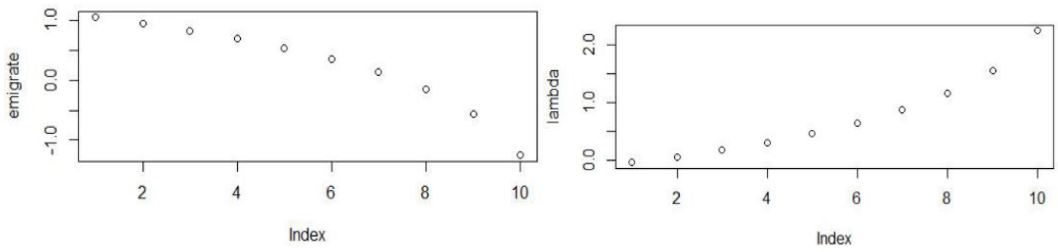
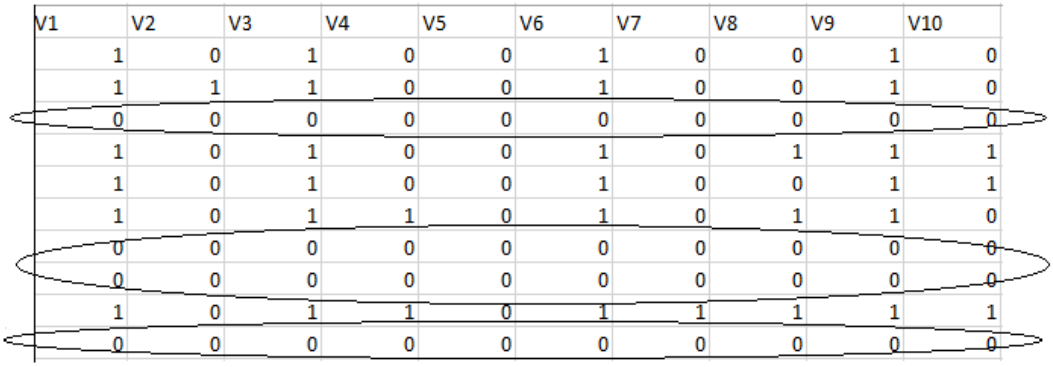


Figure 11. Emigration and immigration probability transition in modified BBO



V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
1	0	1	0	0	1	0	0	1	0
1	1	1	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	1	1	1
1	0	1	0	0	1	0	0	1	1
1	0	1	1	0	1	0	1	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0

Figure 12. Test suite minimization after Extended BBO

After careful evaluation of figure 4 and figure 8 it is evident that Extended BBO has higher tendency towards test suite minimization juxtaposes with the tendency to achieve complete coverage. Test cases 7, 8 and 10 have been found worthless compared with Test case 3 that was found worthless by Biogeography Based Optimization. Below graph plot drawn using GNU-Plot shows the fluctuation of emigrate probability in Extended BBO algorithm. Emigration probability is defined by a function, which should be something that can produce array of probability equals to the population size which can aptly exemplify the problem domain behaviour. The Extended BBO was developed using logarithmic function which showed apt reduction in test suite for regression testing with keeping all the test cases needed for complete coverage. The above figures demonstrate the function that was used to provide extended BBO to obtain best test case minimization for regression testing. We have used Logarithmic function to generate emigration and immigration probability due to its monotonicity, differentiability and smoothness.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
1	0	1	0	1	1	0	1	1	0	1

Figure 13. Harmony search module 5 was tested by test suite 1

The above figure shows that module 5 testing is possible after different combination of test suite. The original test suite does not have any test case which can test module 5 but after applying harmony search for 200 generation it is possible to test module 5. Such functionality was not available in BBO and modified BBO.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
1	1	0	1	0	0	1	0	1	1	1
2	1	0	1	0	0	1	0	0	1	1
3	1	0	1	0	0	1	0	1	1	1
4	1	0	1	0	0	1	0	1	1	1
5	1	0	1	0	0	1	0	1	1	1
6	1	0	1	0	0	1	0	0	1	1
7	1	0	1	1	0	1	0	1	1	0
8	1	0	1	0	0	1	0	0	1	1
9	1	0	1	0	0	1	0	0	1	1
10	1	1	1	0	0	1	0	0	1	0

Figure 14. The individual in square box are redundant test cases created by harmony search

In the above figure the individuals in square box are identified as redundant test cases by harmony search, when it is executed for 200 iterations. Moreover, it is evident from the figure that harmony search is inefficient in finding worthless test cases and module 7 which was testable in original test suite has been categorized as untestable by harmony search. In spite of some above identified disadvantages, the final result shows complete coverage by each test case and module 5 which was untestable in original test suite, can be made testable after multiple combinations.

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	Priority
1	0	1	0	0	1	0	1	1	1	1
1	0	1	0	0	1	0	0	1	1	2
1	0	1	0	0	1	0	1	1	1	1
1	0	1	0	0	1	0	1	1	1	1
1	0	1	0	0	1	0	1	1	1	1
1	0	1	0	0	1	0	0	1	1	2
1	0	1	1	0	1	0	1	1	0	1
1	0	1	0	0	1	0	0	1	1	2
1	0	1	0	0	1	0	0	1	1	2
1	1	1	0	0	1	0	0	1	0	2

Figure 15. Test Case Priority generated by Harmony Search

Above figure shows priority of test cases, as determined by harmony search after having parameters defined as in table 1. The above figure demonstrate the priority based on fault detection ability of the test cases. It is evident from the figure that harmony search has determined 2 test cases suitable for regression testing of JBOSS application server but the union of the two test cases does not provide complete coverage hence it demonstrate a weak point on the part of harmony search when compared with BBO. Although, after execution harmony search for 500 discrete iterations, it showed that 100% coverage with these test suite can be achieved, which is not shown by BBO and Extended BBO. Harmony search algorithm is more random and unpredictable as compared to BBO and firefly hence it is high in time consumption and complexity.

Adding further to the results of Extended BBO and Harmony Search, SVM, a classifier when applied for test suite optimization through prioritization showed some results which were significant.

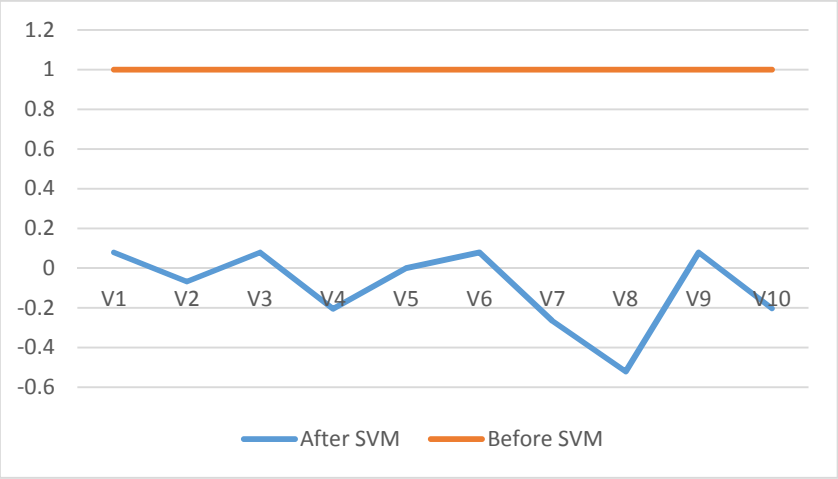


Figure 16. Change in Attribute's weight after SVM

In the above figure 13, we try to show how SVM has reduced the influence value of the modules in predicting of the priorities of the test cases in the test suite. {V1……..V10} are the different modules in JBOSS application server that were applied to 10 test cases in test suite. In the above figure 13, test case "V5" has its significance value of 0, which shows that the test case "V5" is an optional test case that is modules which are tested by "V5" can be tested by single or union of test cases in the test suite.

test cases	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	SVM_Priority
1	1	0	1	0	0	1	0	0	1	0	4
2	1	1	1	0	0	1	0	0	1	0	4
3	0	0	0	0	0	0	0	0	0	0	3
4	1	0	1	0	0	1	0	1	1	1	2
5	1	0	1	0	0	1	0	0	1	1	4
6	1	0	1	1	0	1	0	1	1	0	2
7	1	0	1	0	0	1	0	0	1	1	4
8	1	0	1	0	0	1	0	0	1	1	4
9	1	0	1	1	0	1	1	1	1	1	1
10	1	0	1	0	0	1	0	0	1	0	4

Figure 17. Test Suite prioritization using SVM

SVM has been trained using supervised learning methodology wherein SVM classified the test cases and generated the priority using the priority index defined by BBO. From the above figure 14, it is clear that Extended BBO has achieved much better results than SVM. This is so, because the test case 3, which was unable to detect any mutant in the modules, according to SVM, its priority is 3, which is superfluous whereas according to BBO in figure 4, it has a priority of zero.

Identification of test cases with zero priority, results in removal of test cases. Thus extending the work towards test suite minimization. In reference to the above figure 14, we are able to

observe that the results of test case 1 and test case 2 are different and still SVM gave same priority value to both the test cases. Moreover, SVM has not been able to work in similar manner as BBO and Harmony search. The exploiting of the search space and exploration of the problem domain by Harmony search and BBO has provided significant contribution in test suite minimization and prioritization through its results.

Furthermore, test case 10 and test case 7or 8 are different in the aspect of identification of faults in the modules, still SVM gave these test cases same priorities. ON comparing the results of SVM and BBO from figure 14 and figure 4, we see that 50 % of the results given by BBO matches with SVM.

On the positive side, we see that though according to SVM test case 2, 5, 7 and 8 have priority 4 and according to BBO, these test cases have priority 3, still both the algorithm BBO and SVM identify these test cases as redundant test cases.

Table 4. Performance Value of SVM (Classification)

S.no	Performance Metric	SVM Performance
1.	RMSE	0.983
2.	Sq. Error	2.398

In the above table 3, we state the performance of SVM, when test suite problem was taken as classification problem. Root Mean Squared Error (RMSE) of Simple SVM is 0.983 which shows that we have achieved the objective of error value to be in range [0, 1).

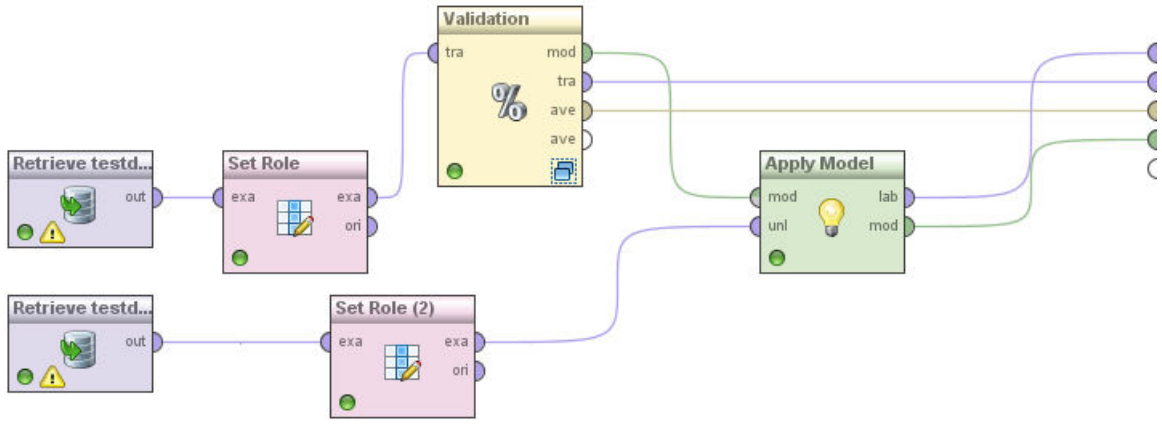


Figure 18. SVM Simulation model in Rapid Miner

Above figure 15, shows how SVM simulation was carried out using Rapid Miner data mining tool. Firstly, the data is retrieved, followed by identification of class label, in case of supervised learning. After the class label is identified, the data is sent to cross-validation operator, which perform 9:1 validation, which means 9 parts of training set and 1 part of testing set. In this operator we use SVM and performance metric operator, in order to apply SVM for testing and training and performance metric operator calculate the error in RMSE.

6.4 Discussion

Table 5. Comparison between various Heuristic Intelligence algorithms applied in TSO

S.no	Characteristics	BBO	Extended BBO	ACO	Harmony Search	Firefly	SVM
1.	Redundant test cases identified (high is better)	3	0	1	8	2	0
2.	Worthless test case identified (high is better)	1	4	0	0	0	0
3.	Complete coverage test cases identified (low is better)	2	2	4	0	5	3
4.	Priority range created	0-4	0-4	1-4	1-2	1-3	1-4
5.	Reduction in test suite	75 %	80%	62.5	80%	64%	60%
6.	Identify untestable part of module	No	Depend on probability function	No	Yes, but depend on HMCR and PAR	No	No

In this research work, a set of five meta-heuristic algorithms have been compared using following parameters : redundant test cases identified, worthless test case identified, complete coverage test cases identified, priority range creation, reduction in test suite and identification of untestable part of module. Using these six parameters for bringing comparison between BBO, Modified BBO, ACO, Harmony Search and Firefly. It can be analysed from results and analysis section that no single meta-heuristic algorithm can be rolled out, that will serve as a panacea for test suite optimization problem.

The high tendency of harmony search in exploration of problem space for solution resulted in figure 10 and Figure 11. Such exploration properties are not inherent to BBO. The exploitation power of BBO has resulted in formation of test suite priority and identification of test cases for complete coverage. This happened because of emigration, immigration and habitat suitability index that is incorporated in BBO. Harmony Search has the power to define a single solution out of the batch whereas Modified BBO has the power to form a colony of solutions with priorities. So, based on the requirement in problem domain regarding test suite optimization

these two meta-heuristics serves the purpose as they stand out when compared to ACO, Firefly and Simple BBO.

Firefly failed to impress in test suite optimization problem as it has tendency to get stuck in local minima. Moreover, in test suite optimization problem, large search procedure is required which is not present in firefly. Furthermore, firefly algorithm does not have learning habit and memory like harmony search and BBO, as a result of which it is unable to store intermediate best test suite formed, resulting in decrease in performance. But firefly, works better than ACO because it can work better with noisy test suites and stochastic objective functions like fault detection rate in test suite optimization problem.

Support Vector Machine (SVM), a linear classifier was developed to perform Test Suite Prioritization. In order to validate the functionality of SVM in solving TSO problem, we compare the results with BBO and EBBO. For table 5, it is clear that SVM partially fails in determining redundant test cases. SVM fails to identify worthless test cases and the reduction in the test suite is also 60%, which is very low as compared to BBO and EBBO. This is because, kernel design for test suite optimization is a very cumbersome task and SVM being a statistical technique suffers from over fitting and stopping at saddle point, which is local minima. Such conditions don't arrive in BBO, which is a heuristic algorithm.

Chapter 7. Conclusion and Future Work

Regression Testing is very crucial activity to bring about success in software development after some minor or major defect has been encountered during execution. It is generally carried out as a maintenance phase activity, which is considered as the most expensive phase, if premeditation about the phase has not been carried out.

There has been large algorithm been proposed and used in the refining the process of regression testing but all these algorithm suffers from constraint problem, elitism problem, loss of coverage and determining the redundant test cases in the test suite which might have been generated during mutation and crossover. Bio-geography Based optimization is meta-heuristic technique being implemented and used widely in various field like remote sensing, job scheduling, classification and selection.

This novel technique has been modified in this paper and compared with other meta-heuristic techniques and SVM. BBO algorithm produced sufficient reduction in test cases after its execution but when we applied Extended BBO but altering the probability of emigration and immigration in generations we obtained, state of the art result. Both Extended-BBO and BBO were compared with Harmony Search for Test case selection in which we got a result of a test case that provided 80% coverage as compared to harmony search test case which provided 60% test case.

Moreover, Extended BBO identified four worthless test cases thereby reducing the size of test suite by 40%, similar behaviour has not been seen in ACO, Firefly, SVM and BBO. Extended-BBO can be extended to multi-objective test suite optimization by altering the migration module, mutation module, incorporating minimization and maximization module for objective function and adding a module for non-dominated ranking system.

Chapter 8. List of Publications from this Research

Chapter 9. References

[1] Singh, Yogesh, et al, "Test case prioritization using ant colony optimization." *ACM SIGSOFT Software Engineering Notes* 2009; 1-7.

[2] Parsa, Saeed et al, " On the Optimization Approach towards Test Suite Minimization", *International Journal of software Engineering and its applications* 2010; 15-28.

[3] Mala, D. Jeya et al. "ABC tester—artificial bee colony based software test suite optimization approach." *International Journal of Software Engineering* 2009; 15-43.

[4] Kaur, Arvinder et al, " A Bee Colony Optimization Algorithm for Fault Coverage Based Regression Test Suite Priortization", *International Journal of Advanced Science and Technology* 2011; 3037-3046.

[5] Yoo, Shin et al, " Using Hybrid algorithm for Pareto Efficient multi-Objective test suite Minimization", The journal of Systems and Software 2010; 689-701.
[6] Singh, Yogesh, Software Testing, Cambridge University Press, 2011.
[7] Agarwal, KK, Software Engineering, New Age International, 2008.
[8] Yoo, S, et al, "Regression testing minimization, selection and prioritization: a survey", *Journal of Software: Testing, Verification and reliability* 2012; 67-120.
[9] Ali Haidar, Aftab et al, " On the Fly Test Suite Optimization with Fuzzy Optimizer", *2013 11th International Conference on Frontiers of Information Technology,* pp. 101-106, 2013.
[10] Vivekanandan, K, et al, "Improving Regression Testing through Modified Ant Colony Algorithm on a Dependency Injected Test Pattern", *International Journal of Computer Science Issues* 2012; 593-600.
[11] Subramanian, Selvakumar, et al, "A Tool for Generation and Minimization of Test Suite by Mutant Gene Algorithm", *Journal of Computer Science* 2011; 1581-1589.
[12] Kaur, Arvinder, et al, "Hybrid Particle Swarm Optimization for Regression Testing", *International Journal on Computer Science and Engineering* 2012; 1815-1824.
[13] Kumar, Harish, et al, "A hierarchical System Test Case Prioritization Technique Based on Requirements", *13th Annual International Software Testing Conference* 2013;1-9.
[14] Krishnamoorthi, R., et al, "Regression Test Suite Prioritization using Genetic Algorithms",
[15] Tahat, L. et al, "Regression test suite prioritization using system models", *Software Testing, Verification and Reliability* 2012; 481-506.
[16] Dorigo, M, et al, "Ant Colony Optimization", *Encyclopedia of Machine Learning, Springer* 2010;36-39.
[17] Solanki, Kamna, "An Empirical Literature Study of Various Test Case Prioritization Techniques", *Software Engineering and Technology* 2014; 169-173.
[18] Li, Huaizhong, et al, "Software Test Data Generation using Ant Colony Optimization", *International Journal of Computer, Control, Quantum and Information Engineering* 2007; 126-129.
[19] Srivastava, P.R, et al, "Optimal test sequence generation using firefly algorithm", *Swarm and Evolutionary Computation* 2013; 44-53
[20] Yang, X. S., "Firefly algorithms for multimodal optimization", *Stochastic Algorithms: foundation and applications, Springer Berlin Heidelberg* 2009,169-178.
[21] Simon, Dan, "Biogeography Based Optimization", *IEEE Transactions on Evolutionary Computation* 2008; 702-713.
[22] Gupta, Daya, et al, "Enhanced heuristic approach for Travelling Tournament Problem based on extended species abundance models of biogeography", *ICACCI* 2014;1118-1124.
[23] Singh, Rajvir, et al, "Test Suite Minimization using Evolutionary Optimization Algorithms: Review", *International Journal of Engineering Research and Technology* 2014; 2086-2091.
[24] Geem, Z.W, et al, "A New Heuristic Optimization Algorithm: Harmony search", *Simulations* 2001; 60-68.
[25] Whitley, D. et al, "Evaluating evolutionary algorithms", *Artificial Intelligence* 1996; 245-276.
[26] Cohen, M.B, et al, "Constructing test suites for interaction testing", *25th International Conference on Software Engineering* 2003; 38-48.
[27] Srivastava, P.R et al, "Test Case Prioritization using Cuckoo Search", *IGI Global* 2012, 113-115.
[28] Rabanal, Pablo, et al, "Testing restorable systems: formal definition and heuristic solution based on river formation dynamics", *Formal Aspects of Computing* 2013;743-768.

[29] Goel, Lavika, "Land Cover Feature Extraction using Hybrid Swarm Intelligence Techniques-A Remote Sensing Perspective", *International Journal on Signal and Image Processing* 2010; 14-16.
[30] Gupta, D, et al, "An Efficient Biogeography based Face Recognition Algorithm", *2nd International Conference on Advances in Computer Science and Engineering* 2013; 64-67.
[31] Du, D. and Simon, D., "Complex System Optimization Using Biogeography-Based Optimization", *Mathematical Problems in Engineering,* pp. 1-19, 2013.
[32] Simon, D., "Biogeography Based Optimization", *IEEE Transactions on Evolutionary Computation*, pp. 702-713, 2008.
[33] Chaves, P., et al, "Operation of storage reservoir for water quality by using optimization and artificial techniques", *Mathematics and Computers in Simulation*, pp. 419-432, 2004.
[34] Deb,Kalyanmoy, et al, "A fast and elitist multiobjective genetic algorithm:NSGA-II",*IEEE Transactions on Evolutionary Computation*, pp. 182-197,2002.
[35] Tauxe, George W., et al, "Multiobjective dynamic programming: A classic problem redressed" *Water Resources Research*, pp. 1398-1402, 1979.
[36] Yeh, T-C, et al, "Stochastic analysis of unsaturated flow in heterogeneous soils: 1. Statistically isotropic media", *Water Resources Research*, pp. 447-456, 1985.
[37] Thampapillai, Dodo J., et al, "Trade-offs for multiple objective planning through linear programming", *Water Resources Research*, pp. 1028-1034, 1979.
[38] Croley, Thomas E., et al, "Multiobjective risks in reservoir operation",*Water Resources Research,* pp. 807-814, 1979.
[39] Oliveira, Rodrigo, et al, "Operating rules for multireservoir systems", *Water Resources Research*, pp. 839-852, 1997.
[40] Reddy, M. Janga, et al, "Multi-objective particle swarm optimization for generating optimal trade-offs in reservoir operation", *Hydrological Processes*, pp. 2897-2909, 2007.
[41] Reddy, M. Janga, et al, "Optimal reservoir operation using multi-objective evolutionary algorithm", *Water Resources Management*, pp. 861-878, 2006.
[42] Zitzler, Eckart, et al, " Multiobjective evolutionary algorithms:a comparative case study and the strength Pareto Approach" *IEEE Transactions on Evolutionary Computation*,pp. 257-271, 1999.
[43] Wang, Xu, et al, " Multi-Objective Algorithm based on Biogeography with Chaos", *International Journal of Hybrid Information Technology,* pp. 225-234, 2014.
[44] Vishwanathan, SVN, et al, "SSVM : A Simple SVM Algorithm", *Proceedings of International Joint Conference on Neural Network,* pp. 2393-2398, 2002.
[45] Gove, R, et al, "Identifying Infeasible GUI Test Cases Using Support Vector Machines and Induced Grammers", *IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops,* pp. 202-211, 2011.
[46] Dorigo, M. et al, "An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem", *INFORMS Journal of Computing,* pp. 237-255, 2000.
[47] Elish, Karim O., et al, "Predicting defect-prone software modules using support vector machines", *Journal of Systems and Software,* pp. 649-660, 2008.
[48] Yang, Bo. , et al, "A study on software reliability prediction based on support vector machine", *IEEE Conference on Industrial Engineering and Engineering Management,* pp. 1176-1180, 2007.
[49] Huang, Ming, et al, "Application of Improved Harmony Search Algorithm in Test Case Selection", *Journal of Software,* pp. 1170-1176, 2014.
[50] UCI Repository, [web link] http://archive.ics.uci.edu/ml/.
[51] University of Ottawa, [web link] http://promise.site.uottawa.ca/SERepository/
[52] University of Southern California, [web link] http://csse.usc.edu/new/