CHAPTER 8

GRAVITATIONAL BEE SEARCH ALGORITHM WITH FUZZY LOGIC FOR EFFECTIVE TEST SUITE MINIMIZATION AND PRIORITIZATION

8.1 INTRODUCTION

To prove their accuracy, software systems become more complex. This is one of the challenging jobs in software development life cycle. Great configurable systems gain more attention in latest software systems, there are different types of existing optional characteristics might involuntarily interact with each other code (Dale Blue et al, 2013). While verification like model based testing, formal verification might need expensive resources due to their complexity and size. Testing functional process is to eliminate and as well as selecting what to test from a possible immeasurable test space. Thus, what kind of testing is needed has to be carefully considered. The planning of test process is respect to the selection and definition of tests out of possible tests, with the aim of minimizing the risk of issues and removing the redundancy.

Before testing a code of software or program, testers have to form the objective criteria of the testing process. An objective criterion of the testing is assumed as a set of requirements has been defined and test cases are formed to satisfy the requirements. A set of test cases which meets all testing requirements is called as a test suite. The development of test suite is expensive and the thoroughly maintained test suite can become large to execute. Always it is not possible to run the whole test suite as it requires significant amount of time to be executed (Saeed Parsa et al, 2010).
8.2 BACKGROUND OF THE STUDY

Various researchers have given different types of methods to minimize the test. Minimizing the test suite is, to find a very minimal subset that can still handle as much as the original set. Almost all the existing test suite minimization methods could considerably minimize the size of the test suites. But the thing to consider is how efficient the minimized test suites can handle or meets as their unreduced test suites (Ankur Prakash Mudgal, 2013). Since the motive of test suite execution is to find the faults in given software, the test suit quality is measured by an evaluation of fault detection capacity of software (Saran Prasad et al, 2012). The exchange between the time required to manage and execute the test and their fault detection capacity should be considered when adopting the test suite minimizing methods. Thus, the focus of this study is to minimize the test suite by finding a set of test cases that achieves the same coverage as the original test suite. In this study, the minimization is attained using a Gravitational Bee Colony Search (GBC) algorithm which is a hybrid approach of artificial bee colony and gravitational search algorithms. Afterwards the clustering concepts of data mining are used to traverse through the minimized test suite and corresponding data featured Fuzzy operation is applied for prioritization for effective test process. The algorithm looks for an optimum solution by calculating fitness values based on coverage information. The search process is repeated until a minimized test suite is achieved.

8.3 PROBLEM STATEMENT

The test suites are unique corresponding to software system. To choose a subset of test cases from the available test suite for the given requirements is one of the main challenge. The test suite minimization problem is defined as follows:

Given a set of test cases \( T \) is defined as \( \{t_1, t_2, t_3, \ldots, t_n\} \), a set of testing requirements is defined as follows. \( R \{r_1, r_2, r_3, \ldots, r_m\} \), is the set of testing
requirements that must be covered to given code (program), trained by each and every test case in \( T \). The minimizing problem in test suite is to find a subset \( T_1 \) of \( T \) and it must be in minimal cardinality that trains the same set of requirement as those trained by the un-reduced test suite \( T \). \( PT \) is the set of permutations of the minimized test suite \( T_1 \), \( f \) a function from \( PT \) to the real number. Find \( T_1' \in PT \), such that (for all \( T_1'' \)), \( T_1'' \in PT \), \( (T_1'' \neq T_1') \), \( f(T_1') \geq f(T_1'') \).

Let \( F(T_j) \) be a function that yields the subset requirements in \( C \) covered by the individual test case \( T_j \). The solution of coverage test case \( t \) is as follows:

\[
\text{Coverage (t)} = 100 \times \left| \frac{\bigcup_{j=1}^{k} F(T_j)}{k} \right|
\]  

(8.1)

Where \( U_{tj=1} \{F(T_j)\} \) is the functional requirement’s union subsets covered by the nominated test cases which means \( T_j \) for which \( (t_j = 1) \).

The function to be minimized (execution cost) signified the volume of time needed to execute the nominated test suite. Properly, each and every test case \( T_j \in T \) has a cost score \( c_j \). The overall cost of a solution \( t \) is as follows.

\[
\text{Cost} = \sum_{j=1}^{t} c_j
\]  

(8.2)

Finally, the proposed Gravitational Bee Colony Search (GBC) algorithm is utilized to provide a better test suite with the aim of functions cost and coverage.

### 8.4 PROPOSED METHODOLOGY

This hybrid approach is attained with the combination of Gravitational search algorithm and Artificial Bee colony optimization. The below sections briefly discuss about each algorithms.
8.4.1 ARTIFICIAL BEE COLONY ALGORITHM (ABC)

ABC algorithm is a popular swarm based Meta heuristic algorithm which imitates the real honey bee’s searching behaviour. ABC is a well-known optimization technique which is based on population search, where the bees modify the position of the food. Locating the position of food with high nectar volume is the major goal of the bees. ABC algorithm contains three types of bees such as onlooker bees, employee bees and scout bees (Jeya Mala. D et al, 2009). Based on employee bee’s dance, the onlooker bees choose their food source. The employee bees go to look for their food source and return back to the hive and perform a dance on this region. When employee bees find more abandoned food source they becomes scout bee and again look for a new food source. The nectar source is chosen using a nest mate in sequence who found the food source previously. The bees dance in the “hive”, on nectar’s sources which are used to convince other bees to follow them.

8.4.2 GRAVITATIONAL SEARCH ALGORITHM (GSA)

GSA is a newly materialized population based stochastic optimization approach which is based on mass interaction and gravity. This method presents an approach that navigates over a multi-dimensional search and imitates the mass interactions (Shanhe Jianga et al, 2014). In GSA, the particles are considered with their performances which are measured by their masses. In this process every particle is defined as a nominee solution to the given problem.

Particles are attracting each other by their gravity power, and this force can make a global movement of the all particles with higher masses. Therefore, masses in general, work with each other by a direct form of interaction with gravitational force. Subsequently, the higher masses have greater fitness values, indicates the good solution to the given test suite searching problem. Providing
the target function produces better results, here the search agents defined as a collection of bodies.

8.4.3 GRAVITATIONAL BEE COLONY ALGORITHM (GBC)

In this research we propose a new hybrid test case selection method which is the combination of Gravitational Search Algorithm (GSA) and Artificial Bee Colony (ABC) algorithm which is named as Gravitational Bee Colony (GBC). This proposed work is to reduce minimize the test suite and time by finding a set of test cases that meets the same coverage or better than the original test suite based on some criterion based on GSA and ABC. The proposed technique uses the initial test suites generated by Breakpoints Matrix (Auto Test generator) tool and it also covers the faults that are detected already.

Bees are utilized as agents; one part of the bees will start foraging on randomly chosen test cases. After this process the new test cases will be added to the discovered path by the bees and that increases fault detection capacity. After adding a new test case, the bees come back to their hive, and exchange the obtained information by using GSA. The Gravitation law is used to process the exchanged information. The distance between the two bees is calculated by Motion Law with the help of bee’s current and previous velocity. The current velocity of the bee is equivalent to the bee’s sum of coefficient previous velocity. The new set of test cases are generated after the gravitational law and motion law and utilized by new bees to search. This procedure repeats until any of the bees has find a new set of test cases which covers additional faults and start performing waggle dance.

Let \( F(T_j) \) be a function that gives the subset requirements in \( C \) covered by the individual test case \( T_j \). The solution of coverage test case \( t \) is as follows:
\[ Coverage (t) = 100 \times \frac{\left| U_{t_j=1} \{ F(T_j) \} \right|}{k} \] (8.3)

Where \( U_{t_j=1} \{ F(T_j) \} \) is the union subset of functional requirement covered by the suggested test cases which means \( T_j \) for which \( t_j = 1 \). The function to be minimized (execution cost) signified the volume of time needed to execute the nominated test suite. Properly, each and every test case \( T_j \in T \) has a cost score \( c_j \). The overall cost of a solution \( t \) is as follows:

\[ Cost (t) = \sum_{t_j=1} c_j \] (8.4)

The search space is calculated as a set of \( m \) number of bee is defined as follows:

\[ x_i = (x_i^1, \ldots, x_i^d, \ldots, x_i^D) \] (8.5)

Where dimension \( d \) of bee \( i \) has been shown by \( x_i^d \) and the distance between two bees are defined as below:

\[ F_{ij}^d = \frac{G(t)XM_i(t)XM_j(t)}{R_{ij}(t) + \epsilon} \left( x_j^d(t) - x_i^d(t) \right) \] (8.6)

Where Gravitational Force \( F_{ij}^d \) in \( G(t) \) is gravitational constant at time \( t \), \( R_{ij} \) is the distance between two bees \( i \) and \( j \) and \( \epsilon \) is defined as very small number.

\[ R_{ij}(t) = \|X_i(t),X_j(t)\|_2 \] (8.7)
Sum of all forces of bees that are forced by other bees of the system is defined as below:

\[ F_{ij}^d(t) = \sum_{j=1, j \neq i}^{m} r_j \int_{ij}^{d} (t) \]  (8.8)

By Newton’s second law, each bee get accelerating in direction of dimension \( d \), the acceleration of bee \( i \), in direction of dimension \( d \), at time \( t \) is shown by \( a_{id}^d(t) \) and is achieved from following equation.

\[ a_{id}^d(t) = \frac{F_{ij}^d}{M_i(t)} \]  (8.9)

The new position of the bee \( i \), at the dimension \( d \) is also same as the sum of its current position and its velocity \( V \) is defined as follows:

\[ V_i^d(t + 1) = r_i X V_i^d(t) + a_{id}^d(t) \]  (8.10)

\[ X_i^d(t + 1) = r_j X X_i^d(t) V_i^d + (t + 1) \]  (8.11)

Where \( r_i \) and \( r_j \) are defined as the random numbers with the interval between (0, 1) which have been utilized to retain the random features of the search. The prerequisite test suite ‘\( T \)’ of ‘\( n \)’ test cases are defined then the result is subset ‘\( S \)’, which comprises of \( m \) test cases (\( m \leq n \)), such that test cases are nominated the consideration of maximum fault coverage capacity in minimum execution time \( t \). Finally, the bee’s fitness score \( FS_{abs} \) is defined as follows.
\[ FS_{abs} = fit_i = \begin{cases} \frac{1}{1 + f_i} & \text{if } f_i \geq 0 \\ 1 + \text{abs}(f_i) & \text{if } f_i \leq 0 \end{cases} \] (8.12)

Where \( \text{abs} \) defined as absolute value of individual. Finally, this algorithm objective functions at minimizing a test suite is defined as follows.

\[ J = \sum_{j=1}^{k} \sum_{i=1}^{n} \| x_i^{(j)} - c_j \|^2 \] (8.13)

Where \( \| x_i^{(j)} - c_j \|^2 \) is a selected distance measure between a bees \( x_i \), \( c_j \) is defined as an indicator of the particular distance of the n bees from their individual coverage capacity. Finally, the proposed Gravitational Bee Search (GBS) algorithm is utilized to provide a better test suite with the aim of functions time and size. After identifying the objective functions of test suites, the prioritization by clustering concept of data mining and the corresponding data featured fuzzy logic is applied for efficient test suite execution. Here the three weights related to the prioritization are defined as high, medium, low

\[ w_r = \begin{cases} 1 & r \in \text{TesterRelevant}_r \\ 0.5 & r \in \text{TesterPartialRelevant}_r \\ 0 & r \in \text{TesternonRelevant}_r \end{cases} \] (8.14)

Where \( \text{TesterRelevant}_r, \text{TesterPartialRelevant}_r, \) and \( \text{TesternonRelevant}_r \) are those requirements nominated by the GBC as partially or relevant or non-relevant. However, the weight \( w_r \) can be considered as based on the Gravitational Force.
8.5 ANALYSIS RESULTS AND DISCUSSION

This section presents the compare analysis and results of the proposed test suite minimization method GBC with Fuzzy with ABC integrated with Fuzzy C-Means (FCM) and PSO (ABCFCM+PSO) (Abraham Kiran Joseph et al, 2014), Bee Colony Optimization (BCO) and Genetic Algorithm (GA) (BCO+GA) (Bharti Suri et al, 2009) in terms of test suite execution time, statement coverage, path coverage, fault coverage in how many number of iterations. Table 8.1 show the proposed test suite minimization method’s results.

Table 8.1 Test Cases Results

<table>
<thead>
<tr>
<th>Test Case/Faults</th>
<th>Test suite 1</th>
<th>Test suite 2</th>
<th>Test suite 3</th>
<th>Test suite 4</th>
<th>Test suite 5</th>
<th>No. of Faults Covered</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Test 2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Test 3</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Test 4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>Test 5</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Test 6</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>Test 7</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Test 8</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>

It is clearly observed from the below Figure 8.1 that the proposed method using GBC with clustered data featured fuzzy gives the promising statement coverage when compared with ABCFCM+PSO, and BCO+GA optimization methods.
It is clearly observed from the below Figure 8.2 that the proposed method utilizing GBC with Fuzzy gives the promising fault coverage when compared with ABCFCM+PSO, and BCO+GA optimization methods.

It is evidently proved from the below Figure 8.3 that the proposed method utilizing GBC with Fuzzy gives the good path coverage when compared with ABCFCM+PSO and BCO+GA optimization methods.
It is clearly shown from the Figure 8.4 that the proposed method utilizing GBC with Fuzzy gives the better results in term of time than ABCFCM+PSO, and BCO+GA optimization methods. The proposed work shows the minimum time to process all the test suites.
It is clearly observed from the Figure 8.5 that the proposed method utilizing GBC with Fuzzy gives the best results in term of number of iteration when compared with ABCFCM+PSO, and BCO+GA optimization methods. The proposed work shows the minimum iteration to process all the test suites.

![Figure 8.5 Iteration](image)

8.6 CONCLUSION

This study mainly focuses on minimize the test suite by identifying a set of test cases that attains the same coverage as the original test suite. In this study, the minimization is attained using a hybrid approach called Gravitational Bee Search algorithm with clustered data featured fuzzy logic (GBC with fuzzy) which is the combination of ABC and GSA and afterwards the clustering concepts of data mining is used to traverse through the minimized test suite and corresponding data featured fuzzy is applied for prioritization. This algorithm looks for an optimum solution using coverage to calculate fitness values. The search process is repetitive until a minimized test suite is found. This developed method efficiently finds and minimize the test suites. Additionally, the clustered
data featured Fuzzy operation is applied for test case prioritization. The proposed method gives better results in term of all the above mentioned objectives and gives positive response in the evaluation process.