Regression Test Case Optimization Using Honey Bee Mating Optimization Algorithm with Fuzzy Rule Base

Deepak Rai and Kirti Tyagi

M. Tech Computer Science and Engg, Ajay Kumar Garg Engineering College, Ghaziabad, UP, India
Department of Computer Science and Engg, Ajay Kumar Garg Engineering College, Ghaziabad, UP, India

Abstract: Maintenance of the software is concerned with the changes and modifications implemented to the software. It needs to be validated that the modifications has not led to the degradation in the quality of the software. Software regression testing is required to instill confidence that changes are valid. Regression testing is very expensive. It requires the optimization of regression test cases. Recently, evolutionary and meta-heuristic algorithms are coming into existence as search and optimization tools. This paper proposes a regression test case optimization technique based on honey bee mating optimization and fuzzy rule base, which reduces the size of the test suite by selecting the test cases from the existing test suite. The test cases which are necessary for validating the recent changes in the software and have the ability to find the faults and cover maximum coding under testing in minimum time are selected. An algorithm is designed which takes as input the test suite containing several test cases and based on their execution time and fault coverage, reduces the number of test cases. The test suite is reduced by 50% which optimizes the overall regression testing process. The proposed algorithm reduces the test suite data by approximately 50%. Although the percentage reduction is found little bit low in comparison to existing algorithm but the overall algorithm is found efficient as it provides an authentic way of selecting test cases using fuzzy rule base.

Key words: Software testing • Regression testing • Test cases • Honey bee mating • Fuzzy logic • Inference system • Optimization.

INTRODUCTION

Maintenance is the common term concerned with the deterioration of the system because of its usage and age. However, this is not true for the software, because software do not deteriorate with the passage of time and usage. Software maintenance is mainly concerned with the modifications and changes done to the software by the changing user needs and adaptation to the changed environment with the passage of time. Software maintenance activity needs to validate that the modifications has not led to degradation of software quality. This requires software regression testing (RT), which is performed on the modified software to instill confidence that changes are correct and have not adversely affected unchanged portion of the software. The dominant strategy for performing the RT is to rerun the test cases. Test cases are set of conditions under which a tester determines whether software is working correctly or not. RT is expensive, it accounts for almost half of the total software maintenance cost [1]. Running all the test cases requires a large amount of time and effort. Report shows that it takes 1000 machine hours to execute approximately 30,000 test cases and 100 of man hours for monitoring the process of RT [2]. This sets a requirement of optimizing the regression test cases.

Optimization tends to provide the best result with least investment. The classical methods and techniques of solving the optimization problems come up with exponential computational complexities as one of their
biggest drawback. This calls up for a new method to built up, shedding away the shortcomings of the traditional methods. Developing, working creatures and natural science specially biology has shown themselves as one of the most peculiar source of inspiration for designing and inventing new system and algorithms [3]. The present scenario has marked the shift from the traditional to a new species of optimization techniques. Recently, evolutionary and meta-heuristic algorithms have been used as search and optimization tools. Computational intelligence (CI) is the best known research field, which is the study and design of intelligent agents. Intelligent agent is a system that perceives its environment and then takes action to maximize the chance of its success. CI includes nature-inspired computational methodologies and approaches to address the complex problems of the real world applications [4]. It can be categorized as shown in Fig. 1.

Over the last couple of decades the behavior of social insects, like ants and bees are being moed for the purpose of searching and optimization problems. It has been the emerging area of swarm intelligence (SI). The intelligence of these swarms lies in the networks of interactions among the agents and between agents and environment. SI is becoming an important research area for computer scientists, bioinformaticians, operational researchers and many other disciplines. This is all because the problems that natural intelligent swarms can solve have important counter parts in several engineering areas in the real world. The requirements of SI like self organization and division of labor features are strongly and clearly seen in the honey bees. Honey bee comb build-up and management is a classic example of teamwork, experience, co-ordination and synchronization. The way they build and maintain their comb is remarkable. This gives rise to interest of using honey bees for regression test case optimization. In this paper we attempt to develop an algorithm for regression test case optimization using honey bee mating optimization (HBMO) and fuzzy logic.

This paper is organized in several sections. Section 2 describes the brief introduction of the related concepts. Section 3 presents the background studies. Section 4 gives the proposed algorithm. Section 5 provides the implementation of the algorithm. Section 6 gives the results and Section 7 concludes the overall paper.

**Brief Introduction of Related Concepts**

**Regression Testing and Test Case Optimization:** RT is the process of retesting the modified part of the software and ensuring that no new regression errors has been introduced into previously unmodified part of the software [5]. It makes the maintenance work easy and less scary. There are several RT techniques namely retest all, regression test selection (RTS), regression test prioritization (RTP) and hybrid approach [6] [7]. Retest all consumes excessive time and resources as compared to other techniques. RTS is an alternate approach but sometimes it omits some fault revealing test cases. So we use the hybrid approach by combining both RTS and RTP.

**Honey Bee Mating Optimization (HBMO) Algorithm:** It is a meta-heuristic, which belongs to the class of nature-inspired population-based algorithm. A natural system has become important source of ideas and models for development of various artificial systems. HBMO uses a similarity among the way in which the bees in nature look for mating and the way in which optimization algorithms search for an optimum in combinatorial optimization problems. The major idea behind the HBMO is to find the best queen bee. Here, the best queen bee is the bee with maximum fault coverage in minimum time, which is decided based on the selection priority calculated by the designed fuzzy inference system (FIS). In order to discover superior and superior solutions, honey bees cooperate with each other and exchange information. Via collective knowledge and giving out
information among themselves, the honey bee focus on more-promising areas and discard the solutions from less promising areas.

In this there is a single egg-laying long-lived queen, several thousands of drones responsible for mating with queen and 10 to 60 thousands of workers responsible for local search and newly born broods care [8]. Queen is specialized in egg-laying, she mate with different drones during the mating flight far from the nest [9]. Queen mates multiple times, but the drones only once. This makes the bee mating the most spectacular mating among insects. After mating, the queen bee returns to hive and starts breeding. The worker bees are chosen proportional to their fitness in order to improve the generated broods. The best brood is selected and the queen is replaced with the selected brood. Remaining broods are killed and a new mating flight begins until the stopping condition is reached or all mating flights are completed.

HBMO and Fuzzy Logic: In most of the models it is assumed that the problem data are deterministic quantities. But, here in this paper the choice of modeling HBMO with fuzzy logic started from the assumption that the perceptions of bees are “fuzzy”. In HBMO the selection of the queen bee and the best brood is uncertain and not defined clearly. Hence, in order to improve this uncertainty we combine HBMO with fuzzy logic [10]. In this, the bees use approximate reasoning and fuzzy rules for their action and communication. During the selection of queen bee and the best brood the bees perceives the selection priority of other bees as ‘very low’, ‘low’, ‘medium’, ‘high’ and ‘very high’. Based on the values of number of faults covered and execution time the selection priority of each bee is calculated with the designed FIS.

Fuzzy Inference System (FIS): It is a system that uses the theory of fuzzy set in order to map the inputs to outputs. There are two FIS namely mamdani and sugeno [11]. In this paper we have used mamdani style of inferencing. Here we have designed two inference systems namely FIS1 and FIS2. FIS1 gives the selection priority of the test cases [12]. FIS2 gives the priority of the subset of test cases in the final list. FIS1 take faults covered and remaining execution time (difference of given time constraint and execution time), as input. Whereas, FIS2 takes one more parameter, number of test cases as its input. FIS1 categorizes faults covered and remaining execution time as very low, low, medium, high and very high. Both the system have different sets of rules, based on which they provide the outcomes. FIS1 provides the selection priority as very low, low, medium, high and very high, whereas FIS2 provides the priority of the final solution as low, medium, high and very high. The detailed description is there in the implementation section.

Background Study: Over the last one or two decades, the emerging area for research of SI is, modeling the behavior of social insects for searching and optimization problems. Colonies of various social insects are characterized by swarm behavior. Studies are primarily based on the collective decision, pheromone laying, reproduction, mating, marriage, foraging behavior of the swarms. Most closely studied social insects are bees. Mating behavior of the bees is considered as one of the swarm based approach for optimization.

Researches using honey bees started in late 90’s. Honey bee behavior was used in the year 1996 by Y. Yonezawas for giving an ecological algorithm for optimal ordering [13]. In the very next year in 1997, T. Sato and M. Hagiwara had given a bee system for finding solution by a concentrated search [14]. Later, in 2001 H.A. Abbas used marriage in honey bee optimization approach and proposed MBO algorithm [15]. Researches using honey bees algorithms continued for different purposes like, fault-tolerant routing algorithms [16], dynamic transport planning [17], cluster analysis [18], numerical function optimization [19], for training feed forward neural network [20], scheduling problems in cosedesign [21], optimization of fuzzy logic controller [22], resource allocation [23, 24], optimization of neural network for identification of wood defects [25]. HBMO algorithm was first used by A. Afshar et.al in 2007 [26]. This was applied to a single reservoir optimization problem. Later, the same algorithm was applied to three benchmark mathematic problem by Bozor Hadded et.al [27]. Recently, honey bee algorithms have found their utility in many fields of software engineering. In the year 2011, Kilic et.al had proposed an approach using honey bees for automated software refactoring problem [28]. Soimart et.al had applied bee algorithm for the development of an automated MLD programming tool [29]. Li and Ma had proposed honey bees based solution method for logic reasoning [30].

Research on regression testing spans a wide variety of topics. Recently, the optimization of size-constrained regression testing was taken up by Siavash Mirarab et.al in 2012 [31]. A safe regression testing based on the program dependence graph was given by Jianchum Xing et.al in the same year [32]. Honey bee algorithms have
also been used for the purpose of optimizing the software test cases and the overall testing process. Mala et al. had applied bee algorithm for the purpose of software test suite optimization in 2010 [33]. In the very next year Adi Srikanth et.al had introduced a software test case optimization method using honey bees [34].

Researches for the purpose of regression testing using fuzzy logic are very scant. It started in the year 2005 when Xu, Gao et.al had firstly shown the application of fuzzy for regression testing [35]. Recently in 2012 and 2013 Ali M. Alakeel and H.B. Gupta proposed a technique for the regression testing using fuzzy logic respectively [36] [37].

Proposed Algorithm: In this paper we have combined the fuzzy rule base with HBMO algorithm for the purpose of regression test case optimization. In order to optimize the regression test cases the proposed technique selects the subset of test cases from the available test suite that covers all the faults in minimum execution time.

Input: < NT, NF, TC, S, Available, r = 0, c = 0 >

Where,
NT = Number_of_Test_case
NF = Number_of_faults
S = Test_suite = {T, FC_T, NF_T, ET_T, \ldots, RET_T}; 1 <= i <= NT
TC = Time_Constraint
T_i = i^{th} Test_case; 1 <= i <= NT
FC_T_i = Faults_covered by i^{th} Test_case
NF_T_i = Number_of_faults_covered by i^{th} Test_case
ET_T_i = Execution_time of i^{th} Test_case
RET_T_i = Remaining Execution_time of i^{th} Test_case.
Available = <T_1, T_2, T_3, \ldots, T_n>

Other Terms Used:
Queen = Queen_bee
Drone = Drone_bee
Brood = Brood_bee
FIS = Fuzzy_Inference_System
P_T_i = Priority of i^{th} Test_case
List = Subset of Test_cases with all faults covered within TC
List_Final = List in order of decreasing priority.
n = |Available| = Number_of_Test_case in Available
NF_Queen = Number_of_faults_covered by Queen_bee
NF_Drone = Number_of_faults_covered by Drone_bee
NF_Brood = Number_of_faults_covered by Brood_bee
FC_Queen = Faults_Covered by Queen_bee
FC_Drone = Faults_Covered by Drone_bee
FC_Brood = Faults_Covered by Brood_bee
ET_Queen = Execution_time of Queen_bee
ET_Drone = Execution_time of Drone_bee
ET_Brood = Execution_time of Brood_bee
RET_Brood = Remaining Execution_time of Brood_bee

Begin Main:
Begin For:
For( i=0; i<NT; i++)
RET_T_i = TC - ET_T_i
P_T_i = FIS1 ( NF_T_i, RET_T_i )
End for:
Each Test Case Is Assigned to Each Bee.
Begin For:
For( i=0; i<NT; i++ )
Queen = Best bee based on the priority.
Drones = Available - Queen
Begin for:
For( j=0; j<n; j++ )
FC_Brood[j] = FC_Queen || FC_Drone[j]
ETF_Brood[j] = ET_Queen + ET_Drone[j]
RET_Brood[j] = TC - ET_Brood[j]
Begin if:
if( NF_Brood[j] == NF & and ET_Brood[j] <= TC )
then List[r][c] = Brood[j]
c = c + 1
P_Brood[j] = FIS2(NF_Brood[j], RET_Brood[j])
List[r][c] = P_Brood[j]
r = r + 1, c = 0
End if:
P_Brood[j] = FIS1 ( NF_Brood[j], RET_Brood[j] )
End for:
6) Select Best Brood based on Priority
7) Begin if:
if (P_Selected_brood[j] > P_Queen)
then Queen = Selected_brood[j];
Drones = Drones - Queen;
Repeat from Step 5;
End if:
8) Begin else:
Else goto previous stage and Repeat from Step 4 with next
Best Queen;
End else:
End for:
9) List_Final = List in order of priority.
End main:

Implementation: The proposed algorithm has been
implemented on several examples. One of the examples is
explained here in detail. In this section we explain the
working of the proposed algorithm using the following
example.

Consider a test suite having 8 test cases in it,
covering 10 faults, with the execution time for each test
case.
Let, the time constraint be 22.
So, NT = 8; NF = 10; TC = 22.

Here, we are using binary representation for
representing the faults covered by the particular test case.
1 at an index means that the test case detects the fault at
that index and 0 for vice-versa. Like, if test case T1 covers
faults F1, F4, F7 and F9. Then we will represent it as
1001001010 from left to right.

Let, set S = {T_i = < FC_T_i, NF_T_i, ET_T_i, RET_T_i >}i

Test Cases | Faults Covered | Number of Faults | Execution Time | Remaining Execution Time |
-----------|----------------|-----------------|----------------|-------------------------|
T1         | 1 0 1 0 0 1 0 0 1 0 | 4 | 7 | 15 |
T2         | 0 1 0 0 0 0 0 1 0 0 | 2 | 3 | 19 |
T3         | 0 1 0 0 0 1 0 0 0 0 | 3 | 5 | 17 |
T4         | 0 0 0 0 1 0 1 0 1 1 | 4 | 5 | 17 |
T5         | 1 0 0 0 0 1 0 0 0 0 | 3 | 3 | 19 |
T6         | 0 0 0 1 1 0 0 0 0 0 | 3 | 6 | 16 |
T7         | 1 0 0 0 0 0 1 1 0 0 | 3 | 3 | 19 |
T8         | 0 1 0 0 0 0 0 0 0 0 | 2 | 2 | 20 |

Available = < T1, T2, T3, T4, T5, T6, T7, T8 >

The selection priority of each test case is calculated
by using the fuzzy inference system (FIS 1). The
membership values for faults covered and remaining
execution time corresponding to each test case is
calculated. As,
For, test case T1
\[ \mu_{\text{faults covered}}(T1) = \frac{\text{NF}_T}{10} = \frac{4}{10} = 0.4 \]
\[ \mu_{\text{remaining execution time}}(T1) = \frac{\text{RET}_T}{22} = \frac{15}{22} = 0.682 \]

Based on these membership values the value of
selection priority for each test case is calculated as:

\[ P_{T1} = FIS1(\mu_{\text{FC}_T1}, \mu_{\text{RET}_T1}) = FIS1(0.4, 0.682) = 0.6545 \]

Similarly, the membership values and the selection
priority for all the test cases are calculated.

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>( \mu_{\text{FC}_T1} )</th>
<th>( \mu_{\text{RET}_T1} )</th>
<th>( P_T1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0.4</td>
<td>0.682</td>
<td>0.6545</td>
</tr>
<tr>
<td>T2</td>
<td>0.2</td>
<td>0.864</td>
<td>0.4255</td>
</tr>
<tr>
<td>T3</td>
<td>0.3</td>
<td>0.773</td>
<td>0.3582</td>
</tr>
<tr>
<td>T4</td>
<td>0.4</td>
<td>0.773</td>
<td>0.7179</td>
</tr>
<tr>
<td>T5</td>
<td>0.3</td>
<td>0.864</td>
<td>0.5033</td>
</tr>
<tr>
<td>T6</td>
<td>0.3</td>
<td>0.727</td>
<td>0.2832</td>
</tr>
<tr>
<td>T7</td>
<td>0.3</td>
<td>0.864</td>
<td>0.5033</td>
</tr>
<tr>
<td>T8</td>
<td>0.2</td>
<td>0.909</td>
<td>0.4255</td>
</tr>
</tbody>
</table>

Queen = T4 FC_Queen NF_Queen ET_Queen RET_Queen P_Queen
0 0 0 1 0 1 0 1 1 0 4 5 17 0.7179

Drone = Available - Queen = < T1, T2, T3, T5, T6, T7, T8 >

Drones = T1 T2 T3 T5 T6 T7 T8

Mating flight begin with queen (T4). The queen will mate
with all the drones one by one and breed respectively in
the hive. The process of mating is as follows:

FC_Queen = 0 0 0 1 0 1 0 1 1 0
NF_Queen = 4
ET_Queen = 5
FC_Drone [1] = FC_T1 = 1 0 1 0 0 1 0 0 1 0
NF_Drone [1] = NF_T1 = 4
ET_Drone [1] = ET_T1 = 7

After mating, the brood generated is brood [1].

FC_Brood [1] = FC_Queen || FC_Drone [1] = 0 0 0 1 0 1 0 1 1 0
NF_Brood [1] = Number of 1’s in FC_Brood [1] = 6

P_Brood [1] = FIS1(\( \mu_{\text{NS_Brood}[1]} \), \( \mu_{\text{RET_Brood}[1]} \)) = FIS1(0.6, 0.4545) = 0.5023

Similarly all the broods are generated as shown in table.

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>NF_Brood</th>
<th>ET_Brood</th>
<th>RET_Brood</th>
<th>P_Brood</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1, T2</td>
<td>6</td>
<td>12</td>
<td>10</td>
<td>0.5023</td>
</tr>
<tr>
<td>T4, T1</td>
<td>5</td>
<td>8</td>
<td>14</td>
<td>0.5751</td>
</tr>
<tr>
<td>T1, T3</td>
<td>7</td>
<td>10</td>
<td>12</td>
<td>0.7168</td>
</tr>
<tr>
<td>T4, T5</td>
<td>6</td>
<td>8</td>
<td>14</td>
<td>0.5751</td>
</tr>
<tr>
<td>T5, T6</td>
<td>5</td>
<td>11</td>
<td>11</td>
<td>0.5033</td>
</tr>
<tr>
<td>T6, T7</td>
<td>6</td>
<td>8</td>
<td>14</td>
<td>0.5751</td>
</tr>
<tr>
<td>T7, T8</td>
<td>6</td>
<td>7</td>
<td>15</td>
<td>0.6575</td>
</tr>
</tbody>
</table>

658
As, the selection priority of none of the brood is better than the queen. So all the broods are killed and the process continues with the selection of next queen from the previous stage.

Available = < T1, T2, T3, T4, T5, T6, T7, T8 >

Next best Queen is:

<table>
<thead>
<tr>
<th>Queen</th>
<th>FC_Queen</th>
<th>NF_Queen</th>
<th>ET_Queen</th>
<th>RET_Queen</th>
<th>P_Queen</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0 1 0 0 0 1 0 0 1 0</td>
<td>4</td>
<td>7</td>
<td>15</td>
<td>0.6545</td>
</tr>
</tbody>
</table>

Drones = < T2, T3, T4, T5, T6, T7, T8 >

As, the selection priority of brood [2] is better than the queen. So the queen is replaced and all the remaining broods are killed. The process continues with the new queen.

<table>
<thead>
<tr>
<th>Queen</th>
<th>FC_Queen</th>
<th>NF_Queen</th>
<th>ET_Queen</th>
<th>RET_Queen</th>
<th>P_Queen</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1, T6, T3</td>
<td>1 1 1 1 1 1 1 1 1 1 1 0</td>
<td>9</td>
<td>18</td>
<td>4</td>
<td>0.8760</td>
</tr>
</tbody>
</table>

Drones = < T2, T4, T5, T7, T8 >

Again the mating flight begins with queen (T1, T6, T3). The queen mates with all the drones one by one and breed respectively in the hive to generate broods.

<table>
<thead>
<tr>
<th>Brood</th>
<th>Test Cases</th>
<th>FC_Brood</th>
<th>NF_Brood</th>
<th>ET_Brood</th>
<th>RET_Brood</th>
<th>P_Brood</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = 1</td>
<td>T1, T6, T3, T2</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0</td>
<td>9</td>
<td>18</td>
<td>4</td>
<td>0.8760</td>
</tr>
<tr>
<td>2</td>
<td>T1, T6, T3, T4</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0</td>
<td>9</td>
<td>18</td>
<td>4</td>
<td>0.8760</td>
</tr>
<tr>
<td>3</td>
<td>T1, T6, T3, T7</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0</td>
<td>9</td>
<td>18</td>
<td>4</td>
<td>0.8760</td>
</tr>
<tr>
<td>4</td>
<td>T1, T6, T3, T8</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0</td>
<td>9</td>
<td>18</td>
<td>4</td>
<td>0.8760</td>
</tr>
</tbody>
</table>


Hence, P_Brood[3] = FIS2(µNF_Brood[j], µ RET_Brood[j]) = FIS2(1.0, 0.04545) = 0.21

P_Brood[5] = FIS2(µNF_Brood[j], µ RET_Brood[j]) = FIS2(1.0, 0.09090) = 0.468


Similarly all the subset of test cases satisfying the condition is found and put into the List.

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>Priority (P_Brood)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1, T6, T3, T5</td>
<td>0.21</td>
</tr>
<tr>
<td>T1, T6, T3, T8</td>
<td>0.468</td>
</tr>
<tr>
<td>T1, T6, T8, T7</td>
<td>0.684</td>
</tr>
<tr>
<td>T1, T6, T2, T5, T7</td>
<td>0.206</td>
</tr>
<tr>
<td>T1, T3, T4, T5</td>
<td>0.468</td>
</tr>
<tr>
<td>T1, T3, T4, T8</td>
<td>0.637</td>
</tr>
<tr>
<td>T1, T5, T6, T7, T8</td>
<td>0.206</td>
</tr>
</tbody>
</table>

As, the selection priority of brood [5] is better than the queen. So the queen is replaced and all the remaining broods are killed. The process continues with the new queen.

<table>
<thead>
<tr>
<th>Queen</th>
<th>FC_Queen</th>
<th>NF_Queen</th>
<th>ET_Queen</th>
<th>RET_Queen</th>
<th>P_Queen</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1, T6, T8, T7</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0</td>
<td>9</td>
<td>18</td>
<td>4</td>
<td>0.8760</td>
</tr>
</tbody>
</table>

Drones = < T2, T4, T5, T7, T8 >

Again the mating flight begins with queen (T1, T6, T3). The queen mates with all the drones one by one and breed respectively in the hive to generate broods.

<table>
<thead>
<tr>
<th>Brood</th>
<th>Test Cases</th>
<th>FC_Brood</th>
<th>NF_Brood</th>
<th>ET_Brood</th>
<th>RET_Brood</th>
<th>P_Brood</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = 1</td>
<td>T1, T6, T3, T2</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0</td>
<td>9</td>
<td>18</td>
<td>4</td>
<td>0.8760</td>
</tr>
<tr>
<td>2</td>
<td>T1, T6, T3, T4</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0</td>
<td>9</td>
<td>18</td>
<td>4</td>
<td>0.8760</td>
</tr>
<tr>
<td>3</td>
<td>T1, T6, T3, T7</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0</td>
<td>9</td>
<td>18</td>
<td>4</td>
<td>0.8760</td>
</tr>
<tr>
<td>4</td>
<td>T1, T6, T3, T8</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0</td>
<td>9</td>
<td>18</td>
<td>4</td>
<td>0.8760</td>
</tr>
</tbody>
</table>


Hence, P_Brood[3] = FIS2(µNF_Brood[j], µ RET_Brood[j]) = FIS2(1.0, 0.04545) = 0.21

P_Brood[5] = FIS2(µNF_Brood[j], µ RET_Brood[j]) = FIS2(1.0, 0.09090) = 0.468


Finally, after the completion of the process a Final List is formed, which contains the subset of test cases from the List in order of their priority.

<table>
<thead>
<tr>
<th>Final List</th>
<th>Test Cases</th>
<th>Priority (P_Brood)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1, T6, T8, T7</td>
<td>0.684</td>
<td></td>
</tr>
<tr>
<td>T1, T3, T4, T8</td>
<td>0.637</td>
<td></td>
</tr>
<tr>
<td>T1, T6, T3, T8</td>
<td>0.468</td>
<td></td>
</tr>
<tr>
<td>T1, T3, T4, T5</td>
<td>0.468</td>
<td></td>
</tr>
<tr>
<td>T1, T6, T3, T5</td>
<td>0.21</td>
<td></td>
</tr>
<tr>
<td>T1, T6, T2, T5, T7</td>
<td>0.206</td>
<td></td>
</tr>
<tr>
<td>T1, T5, T6, T7, T8</td>
<td>0.206</td>
<td></td>
</tr>
</tbody>
</table>
RESULTS

It is seen that the set of test cases in the final list has covered all the faults within the given time constraint. So, these set of test cases like \{T1, T6, T8, T7\}, \{T1, T3, T4, T8\} and many more in the final list are the required reduced set of test cases produced by our proposed algorithm. It can be observed that the reduction in the test suite using the proposed algorithm is approximately 50%. In order to produce more reduced set of test cases the above process need to be iterated.

CONCLUSION

This paper proposed an algorithm for regression test case optimization, which reduces the large test suites using HBMO algorithm with fuzzy rule base. This algorithm has been evaluated for different set of examples. One of the examples has been shown in this paper. The proposed algorithm reduces the test data by approximately 50%. The result was compared to different existing algorithms like ant colony optimization and bee colony optimization with genetic algorithm. It was found that, the percentage reduction of test cases was little bit low, even though the overall algorithm was found efficient as it provides an authentic way of selecting test cases using fuzzy logic. Fuzzy logic removes the uncertainty associated with the selection of test cases in existing algorithms.

Issues of future work include the consideration of more parameters for test case selection and, applying the proposed algorithm on many more large and complex examples, in order to prove its effectiveness and usability.

REFERENCES

10. Teodorović, Dušan, Tatjana Davidović and Milica Šelmić, "Bee Colony Optimization Overview."