A Parallel Early Binding Recursive Ant Colony Optimization (PEB-RAC) Approach for Generating Optimized Auto Test Cases from programming Inputs

Yogita Dubey  
M.Tech VI Semester  
Department of CS  
BUIT, BU, Bhopal, India

Divakar Singh  
Head of Department  
Department of CS  
BUIT, BU, Bhopal, India

Anju Singh  
Assistant Professor  
Department of CS and IT  
UTD, BU, Bhopal, India

ABSTRACT
Software testing is one of the most important parts of the software development. It also takes too much time to complete, because there are test cases are used for the testing of the software. So data mining techniques are used to improve the performance of the testing by reducing the size of the test cases. In this paper a Parallel Early Binding Recursive Ant Colony (PEB-RAC) System technique is presented with automated testing to provide an efficient way of the software testing. A result analysis is shown in the result section, this analysis shows that proposed technique provide better result as compare to the other technique.

Keywords
Optimization techniques, testing approach, auto case generation, SDLC

1. INTRODUCTION
Software engineering is the field in computer science in which programs are made and compile to design software. Provide reliable software for the user is a basic requirement in software engineering. Thus software testing is one of the key steps in software development life cycle. In software testing, testing to check correctness, completeness and reliability of the software is performed. There are many ways by which defects can be generated in the software like error in the programming is introduced by the programmer, or user requirement cannot be fetched correctly during the requirement analysis phase etc. These issues are generates bugs in the software and degrades the performance of the software.

This paper focuses on automated testing for the software. In this some tools, programs or applications are used to test the quality of the software. In olden days testing is conducted by manually which does not provides performance that is required. So testing by automatic tools can enhance the performance of the testing. In automated testing test cases are used to test the quality of the software. Test cases are the programs which are used to verify functionality and feature of the software. But sometimes redundancy is generated in the test cases that cause degradation in the performance of the automated testing.

Thus data mining techniques are used to provide reliable and small number of test cases for testing that can speed up testing process and also enhance the performance of the automated testing. Data mining is technique in which suitable no. of patterns are generated from the data. And technique can be used for application like classification, regression, clustering etc. thus it provides a suitable no of patterns of test cases which can be provide desired performance as require by the system.

For proposed work a PEB Recursive Ant Colony system is used to provide optimize test cases for the automated testing.

2. RELATED WORK
In this section a study over techniques which used for generating test cases for automated testing is presented. There are different techniques which are used for test cases generation for automated testing. Like in [1] a technique is presented in which UML diagram with genetic algorithm is used to generate test cases. In this technique genetic crossover’s technique is used to generate test cases from sequential diagram. And a mutation testing is implemented to test the effectiveness of the methodology. In [2] a test case reduction technique using clustering is presented. In clustering data objects can be group into cluster in which each object having similar properties with respect to other objects in the cluster. In a combination of software engineering and data mining technique is used to provide reduction in the test cases which improves the performance of the testing. WEKA a machine learning tool which is written in java is used to implement the whole process. In [3] automated testing technique with data mining for software testing is presented. In this technique firstly software requirements are converted into UML diagrams and then test cases from that diagram is generated. Then data mining techniques are used to reduce these test cases in that way an efficient way to reduce test cases for automated testing is presented. In [4] technique to categorize functional and nonfunctional requirements of the software is presented. In this a data mining technique over SRS state charts to mine test cases for automated testing is presented and in that way it generated a reliable and small number of test cases for automated testing.

In [24] an ant colony based optimization based test case generation is presented. In test case generation, test cases are generated that satisfies testing criteria. ACO optimization technique which uses the behavior of the ants to provides solution for the many optimization problems. In case of test case generation by ACO, first a transformation graph is generated and then an optimal path is formed to and a condition to stop generation and a process to frame these result to find an optimal path. By using these step test cases is generated from UML diagram. In [25] an intelligent technique based scheduling technique is presented. In that technique intelligent water drop (IWD) is used with white box testing to generate basis bath testing and ACO ant colony optimization is used for test case generation. That way, this technique provides a method which uses ACO ability to generate test cases and bath testing a technique of white box testing. That provides an intelligent water drop testing technique for software testing. In [25] an ant colony optimization (ACO) based scheduling technique is presented in that technique, ACO’s feature a set of software agents and artificial ants is
presented to provide solution for software testing in that case. In that artificial ants are used to provide solution for the problem and each and provide solution in constructive manner by collection component step by step to form a solution. Thus as mentioned in the related work, a data mining technique is can be provide a significant way of test case categorization. For propose work a data mining technique with an optimizing mechanism can be presented.

2.1 Existing technique (ACO Technique)

Ant colony optimization technique is applied while determining the test case generation. Where the natural ant behavior is used to opt test cases. Pseudo code describes the complete process using ACO technique.

ACO Algorithm:

Input: Data Input Program, TC Test condition i-n;
Output: Test case generation table -GenTCi-n;
Begin:

{ Load the complete program code by line by line
For each line proceed:
Apply preprocessing technique and parse the conditional statement clusters.
Apply construct Ant Solutions ();
Extracting Possible test conditions matched with available
Apply Local Search ();
Generate the Test cases based on Ant Solutions and refine for publish.
Apply Pheromones ();
} END;
Upon processing a dataset the following optimized data is received for the result process.

Optimized Data using Existing algorithm

In order to calculate the precision, recall, accuracy and detection rate we have various parameter values from the dataset progress:

Which are TP, FP, TN, and FN, which is getting in the form of confusion matrix, using which the result parameters calculated using the following formulae

Accuracy = (TP+TN)/(TP+TN+FP+FN)

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

The above formulae terms usage for result computation. Upon processing this data further the observed values are:

TP= 38, FP = 63, TN = 19, FN = 16

Accuracy = (38+19/ (38+63+19+16))*100 = 41.30

Precision = (38/(38+63))*100 =37.62

Recall = (38/(38+19))*100 = 70.37

3. PROPOSED WORK

As discussed in the related work section data mining techniques can provide assistance for automated testing which can enhance the performance of the testing programs.

Dataset: In our approach here further a input sample program is taken [27] example. In [2] a test case reduction technique using clustering is presented. In clustering data objects can be group into cluster in which each object having similar properties with respect to other objects in the cluster. In a combination of software engineering and data mining technique is used to provide reduction in the test cases which improves the performance of the testing. WEKA is a machine learning tool which is written in java is used to implement the whole process

Test case dataset [28] table is further described here which is used in our experimental setup and solution:

Table-1 Large Dataset for experimental Setup

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Input program purpose</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bockyroberts</td>
<td>Multiple Tutorial Programs</td>
<td><a href="https://github.com/buc">https://github.com/buc</a> kyroberts/</td>
</tr>
<tr>
<td>Career monk</td>
<td>Data structure algorithm programs</td>
<td><a href="https://github.com/car">https://github.com/car</a> eermonk/</td>
</tr>
</tbody>
</table>

Thus this is the programming large dataset which further consider for the experimental setup and analysis as described in table-1 and considered below program.

3.1 Software Test Cases Table

There are various possible test cases associate with the any input dataset program. Where the accuracy and efficiency of the program can be tested, that what cause it may fail while dealing with the program execution. Here is some sample test case which often consider while doing programming with any of the Object oriented programming language. In usual a test case given with the function or its behavior, further more test case ID, description, remarks etc. can be embed with the sample test case.

Table-2 Sample Test Case Table

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Expected Test Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify the mobile number 10 digit numeric value.</td>
<td>If(Length(mobile)==10) True; If (Mobile!='numeric') False;</td>
</tr>
<tr>
<td>If ((a==a)&amp;&amp;(b==b))</td>
<td>If ( a&gt;0,b&gt;0,c&gt;0) &amp; not belongs to Alphabet category.</td>
</tr>
<tr>
<td>If((a+b)&lt;c) &amp;((c+a)&gt;b) &amp;((b +c)&gt;a))</td>
<td>If(variable != alphabets) OR If(Variable all solution in same format)</td>
</tr>
<tr>
<td>Email format verification</td>
<td>If(email.contains(&quot;@&quot;)), If(email.contains(&quot;@domain name&quot;))</td>
</tr>
</tbody>
</table>
As per described in above table the sample test cases are always available with the any of the small programming.
Here a dataset input taken triangle classification problem a C language program and further examine the auto test cases generation on processing the program as dataset.

```c
#include<stdio.h>
#include<conio.h>
void main ()
{
    int a, b, c, valid;

    printf("Enter the value of three sides: ");
    scanf("%d %d %d", &a, &b, &c);

    valid=0;

    if ((a>=0)&&(a<=100)&&(b>=0)&&(b<=100)&&(c>=0)
        &&(c<=100))
    {
        if (((a+b)>c)&&(c+a)>b)&&(b+c)>a))
        valid=1;
    }

    if (valid==1)
    {
        if ((a==b)&&(b==c))
            printf("Equilateral triangle.");
        else if ((a==b)||(b==c)||(c==a))
            printf("Isosceles triangle.");
        else
            printf("Scalene triangle.");
    }
    else{
        printf("Invalid input (out of range ");
    }
}
```

In this paper a PEB Recursive Ant Colony system based optimization technique is presented which provide optimize test cases for the automated testing.

### 3.2 Parallel Early Binding Recursive Ant Colony Optimization Algorithm

1. Test case generation presenting different dataset (Triangle classification, Bocky Roberts, Career monk) etc. Apply Ant colony optimization algorithm to generate test cases.
2. Partition test cases in to sub-classes.
3. Perform thread creation for each subclass and process for execution in parallel.
4. Perform thread based ant colony optimization for each sub class, apply early binding execution stops.
5. Collect results from each sub-class.
6. Merge results of the sub-classes.
7. Form final optimal result to generate test cases.

#### 3.2.1 Proposed PEB-RACS: In our approach here further a input sample Program

**Input:** Sample input program, GenTCi.n.
**Output:** Test case generation./optimal test cases

**Steps:** Begin – Apply PEB-RAC

1. Construct Ant Solutions ()
2. Local Search on Ant Solution;
3. Perform Pheromones;
4. Generate subset partition program ();
5. Thread Creation Partition ();
6. Apply Early Binding ();
7. For each partition using
   1. New Thread ();
   2. Obtain Cluster CLi-n;
   3. Further ACO on each thread Ti CLi-n;
   4. Apply early Binding ();
8. Returns merge final result; //optimal test cases.

#### 3.3 ACO Technique

The existing approach which is Ant colony optimization where data is processed at using the approach which is stated in related work, in order to proceed with the technique following steps were involved:

**3.3.1 Input (Dataset) Program:**
As per discussed the above triangle classification program given by the author in paper, here is the Triangle.C considered.

```c
#include<stdio.h>
#include<conio.h>
void main ()
{
    int a, b, c, valid;

    printf("Enter the value of three sides: ");
    scanf("%d %d %d", &a, &b, &c);

    valid=0;

    if ((a>=0)&&(a<=100)&&(b>=0)&&(b<=100)&&(c>=0)
        &&(c<=100))
    {
        if (((a+b)>c)&&(c+a)>b)&&(b+c)>a))
        valid=1;
    }

    if (valid==1)
    {
        if ((a==b)&&(b==c))
            printf("Equilateral triangle.");
        else if ((a==b)||(b==c)||(c==a))
            printf("Isosceles triangle.");
        else
            printf("Scalene triangle.");
    }
    else{
        printf("Invalid input (out of range ");
    }
```
valid=1;
}
}
if (valid==1)
{
if ((a==b)&&(b==c))
printf("Equilateral triangle.");
else if ((a==b)||(b==c)||(c==a))
printf("Isosceles triangle.");
else
printf("Scalene triangle.");
}
else{
printf("Invalid input (out of range ").;
}

3.3.2 Preprocessing Procedure
Upon taking the program as input the complete structure is analyzed and parse using the scanner and further entity such as separator, operators, keywords and such other components are removed and finally the remaining part is pass to the next phase.

3.3.3 Apply ACO Technique
After preprocessing of triangle program coming to the step where ACO is applied with the algorithm and following optimal result is obtain where the optimized statements retrieved.

3.3.4 Result Obtain (Optimal)
Upon applying the ACO technique, following optimal data is retrieved which shows the different mathematical and other available statement is considered.

3.4 PEB-RAC Technique
PEB-RACS system based on the fact that performance of the ant colony system for the small size problem is better but in case of large scale data set it degrades in the performance. In PEB-RACS partition scheme similar to the merge sort is presented. In that technique all the nodes of the data set are divided in to two small sub sets and perform the operation of optimization and in that way separate operation for these subsets is performed and desired performance have been reached.

3.4.1 Input (Dataset) program:
In order to process the proposed algorithm further the same Triangle.C program is considered which is given below.

```c
#include<stdio.h>
#include<conio.h>
Void main ()
{
int a, b, c, valid;
printf("Enter the value of three sides: ");
scanf("%d %d %d", &a, &b, &c);
Valid=0;
If ((a>=0)&&(a<=100)&&(b>=0)&&(b<=100)&&(c>=0)&&(c<=100))
{
if (((a+b)>c)&&(c+a)>b)&&(b+c)>a)
{
valid=1;
}
if (valid==1)
{
if ((a==b)&&(b==c))
printf("Equilateral triangle.");
else if ((a==b)||(b==c)||(c==a))
printf("Isosceles triangle.");
else
printf("Scalene triangle.");
}
else{
printf("Invalid input (out of range ");;
}
```
Thus a method, which combines automated testing with the PEB-RACS system, is presented which improve the performance of the existing technique and provide better performance as compare to the existing technique.

In order to calculate the precision, recall, accuracy and detection rate we have various parameter values from the dataset progress:

Which are TP, FP, TN, and FN, which is getting in the form of confusion matrix, using which the result parameters calculated using the following formulae

\[
\text{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)}
\]

\[
\text{Precision} = \frac{TP}{(TP+FP)}
\]

\[
\text{Recall} = \frac{TP}{(TP+FN)}
\]

The above formulae terms usage for result computation.

Upon processing this step the further values computed are –

\[
\begin{align*}
\text{TP} &= 41, \\
\text{FP} &= 58, \\
\text{TN} &= 26, \\
\text{FN} &= 12
\end{align*}
\]

\[
\begin{align*}
\text{Accuracy} &= \frac{(41+26)}{(41+58+26+12)}\times100 = 60.90\% \\
\text{Precision} &= \frac{41}{(41+58)}\times100=41.41 \% \\
\text{Recall} &= \frac{41}{(41+12)}\times100= 77.35 \%
\end{align*}
\]

3.5 Different Dataset outcomes

As per our work performance and applying same procedure with different dataset programs, the following result is obtained using Bocky Roberts and career monk program, the confusion matrix parameter value is obtained.

**Bocky Roberts:** the obtained value is performed using the ACO technique

\[
\begin{align*}
\text{TP} &= 27, \\
\text{FP} &= 97, \\
\text{TN} &= 17, \\
\text{FN} &= 2
\end{align*}
\]

\[
\begin{align*}
\text{Accuracy} &= 21.77, \\
\text{Recall} &= 93.10, \\
\text{Precision} &= 30.76
\end{align*}
\]

**PEB-RAC Technique:** the following result with Bocky Robert is obtained

\[
\begin{align*}
\text{TP} &= 43, \\
\text{FP} &= 28, \\
\text{TN} &= 7, \\
\text{FN} &= 2
\end{align*}
\]

\[
\begin{align*}
\text{Accuracy} &= 60.56, \\
\text{Recall} &= 95.55, \\
\text{Precision} &= 62.5
\end{align*}
\]

**Career Monk:** the obtained value is performed using the ACO technique

\[
\begin{align*}
\text{TP} &= 29, \\
\text{FP} &= 51, \\
\text{TN} &= 19, \\
\text{FN} &= 6
\end{align*}
\]

\[
\begin{align*}
\text{Accuracy} &= 45.71, \\
\text{Recall} &= 82.25, \\
\text{Precision} &= 36.25
\end{align*}
\]

**PEB-RAC Technique:** the following result with Career monk is obtained

\[
\begin{align*}
\text{TP} &= 45, \\
\text{FP} &= 37, \\
\text{TN} &= 7, \\
\text{FN} &= 6
\end{align*}
\]

4 RESULT ANALYSIS

All the experiments were performed using an i5-2410M CPU @ 2.30 GHz processor and 4 GB of RAM running windows 8. The discussed feature selection algorithms were implemented using language Java.

### Table 3 Comparison Analysis between Existing and Proposed Algorithm

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Algorithm</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangle classification</td>
<td>ACO</td>
<td>41.30</td>
<td>37.62</td>
<td>70.37</td>
</tr>
<tr>
<td></td>
<td>PEB-RAC</td>
<td>60.9</td>
<td>41.41</td>
<td>77.35</td>
</tr>
<tr>
<td>Bocky Roberts</td>
<td>ACO</td>
<td>21.77</td>
<td>30.76</td>
<td>93.1</td>
</tr>
<tr>
<td></td>
<td>PEB-RAC</td>
<td>60.56</td>
<td>62.5</td>
<td>95.55</td>
</tr>
<tr>
<td>Career Monk</td>
<td>ACO</td>
<td>45.71</td>
<td>36.25</td>
<td>82.25</td>
</tr>
<tr>
<td></td>
<td>PEB-RAC</td>
<td>54.73</td>
<td>54.87</td>
<td>88.23</td>
</tr>
</tbody>
</table>
Figure 3: Precision Bar Graph comparison between existing and proposed algorithm using Triangle program.

Figure 3: Recall Bar Graph comparison between existing and proposed algorithm using Triangle program.

Figure 3: Accuracy Bar Graph comparison between existing and proposed algorithm using Triangle program.

In this graph we have discussed and compared the obtained result analysis obtain from the experimental setup and result part by which we observe the obtained parameter value prove that the proposed system algorithm suite best as compared with existing algorithm.

5 CONCLUSION
Software testing is one of the key phases in software development process. In this paper various issues related to the automated software testing like redundancy in the test cases is presented. To reduces these issues in automated testing data mining techniques are used which enhance the performance of the whole automated testing technique. A PEB-RACS based optimizing technique with automated techniques is proposed. A result analysis for that method is shown, which shows that proposed technique provide better result as compare to the existing technique.

6 REFERENCES
[1] Paramjit Kaur, Rupinder Kaur “approaches to generating test cases automatically to test the software” IJEAT, Vol. 2 Issue 3, and February 2013.


[11] P. Samuel R. Mall A.K. Bothra “Automatic test case generation using unified Modeling language (UML) state diagrams” Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur 721302, West Bengal, India E-mail: philips@cusat.ac.in


[18] Karl Nils Gundersen “An Application of Association Rule Mining To Unit Test Selection” Fargo North Dakota June 2013


