Automated Test Data Generation Using Fuzzy Logic-Genetic Algorithm Hybridization System for Class Testing Of Object Oriented Programming

Swapan Kumar Mondal, Hitesh Tahbildar

Abstract-In this paper we have explained automatic test data generation particularly for class testing of object oriented programming. During test data generation we have implemented the Genetic program - Fuzzy logic control auxiliary hybridization techniques. Some cases genetic algorithm has been used for optimized the desired results. As a future challenges we have made comments on the utilization of this new proposed technique. This proposed technique can be used for testing of industrial production oriented software. Production oriented software is use in Computer numerical control (C.N.C) machine.

Keywords- Binary tree, Fuzzy logic control (FLC), Genetic programming (GP), Genetic algorithm (GA), Mutation testing.

I. INTRODUCTION

Main objective of software industry is to customer satisfaction. Object oriented programming uses widely in every sector. Bug free software need to be produced with minimum testing cost and in minimum time. Automatic testing is necessary for adapting fast development of software industry as well as cut down cost and time. Testing including execution of a program by some sets of test data and compare the results with expected outcomes is shows many binding anomalies due to dynamic behavior of inheritance, method overriding, polymorphism, templates. Class is the basic building block of object oriented programming. Traditional testing like structural testing, functional specification-based testing and heuristics testing are used for it. Structural testing is important because it’s located the bugs in codes by control flow testing, path coverage testing, data flow testing. Functional testing meets the requirements and specification of software. Heuristics testing technique test the abstract classes. Automatic test data is generated in object oriented programming from traditionally code-based technique and model-based or design based technique. In code-based technique, test data is generated after analyzed the change impact between source code and instrumented code. In model-based technique test data is generated after analysis of two different versions of models. Model is drawn during system requirement analysis phases.

Hitesh et al [12] has explained automated test data generation technique for soft ware testing in their paper. Some researchers explained automated test data generation based on UML designing. But nowadays researcher has concentrated on search based [2] technique for automated test data generation of OOP. They have utilized evolutionary algorithm like genetic algorithm (GA), genetic programming (GP). Some researchers made comments on the using of GA with Fuzzy logic as well as neural networks [10] for various type of software testing.

In this paper automated test data is generated on the new model based approach. Advantage of this approach is that [3] test data can be available earlier during software requirement analysis phases. Dynamic behavior [1] of objects has been utilized throughout the model. Information is extracted from .UML file by java parser. Tree structure of objects is formed by the extracted information. The tree is then converted to optimized tree structures by Genetic Programming in association with Fuzzy logic control. The optimized tree is then converted to binary trees. Test data generation, Validity checking, Termination, all are done from binary trees by using depth first search algorithm [1,3].

II. OBJECTIVE

UML class diagram help the developer for developing software requirement analysis in software development life cycle. Tester can also predict not only the multiple inheritances, multilevel inheritances of classes but also conformance of black box testing early from it. Testing of pointers in java code through UML diagram gives extra advantages also. Java swing code is used in java parser. We can early predict the tree structures of nodes from parser. Using only genetic programming cross over tree for optimization of desired results is a cumbersome process. Optimization occurred in GP crossover trees in various levels and used large number of steps for it. This time consuming and cumbersome process has been eliminated in our proposed methodology through Fuzzy Logic control. Soft computing hybridization technique can be used [10] for different type of software testing. We have implement GA-FLC auxiliary hybridization technique [2, 6] throughout the tree structures in couple with Genetic programming. It is better than sequential hybridization technique [5] and individual use of Genetic programming. Hybridized techniques of these soft (GA-FLC) computing gives the healthy hybridization and tremendous potential [5] for optimal of the patterns easily. This technique having some disadvantages also e.g: writing of algorithm on this proposed technique is a very complicated task.

Manuscript received November, 2013.
Swapan Kumar Mondal, Research Scholar, Department of CSE, University of Science and Technology, Meghalay, 9th Mile, Techno-city, Baridua, Ri-Bhoi, Meghalaya-793101, India
Hitesh Tahbildar, working as HOD, Computer Engineering Department, Assam Engineering Institute, Govt of Assam, Guwahati, India

III. SURVEY OF RELATED WORKS

M. Prasanna et al. [1] have proposed automatic test case generation for UML object diagrams using genetic algorithm in genetic crossover tree structures. As a case study, they have used Banking System. Test case is generated from binary tree. Depth first search algorithm is used for test data generation, validity checking, and termination. It is a very lengthy and time consuming process. Several steps are necessary for selection of fittest crossover tree until optimized value is reached. Optimization in different levels of genetic crossover tree using genetic algorithm is a cumbersome and more stepping process. A.V.K. Shanthi et al. [3] have proposed to use GA for optimization of crossover tree structures. GA implemented as data mining concept for optimization of test cases. They have also explained the advantages of model based automatic test data generation for object-oriented programming (OOP). As a model, they have used UML class diagram, optimized tree structure of objects, binary tree. Finally test data is generated and checked the validity by depth first search algorithm. Any single soft computing technique having less potential than hybridized soft computing. Their proposed testing technique can be fine tuned with my proposed hybridized technique.

A.V.K. Shanthi et al. [3] has proposed to use GA for optimization of crossover tree structures. GA implemented as data mining concept for optimization of test cases. They have also explained the advantages of model based automatic test data generation for object-oriented programming (OOP). As a model, they have used UML class diagram, optimized tree structure of objects, binary tree. Finally test data is generated and checked the validity by depth first search algorithm. Any single soft computing technique having less potential than hybridized soft computing. Their proposed testing technique can be fine tuned with my proposed hybridized technique.

IV. OUR PROPOSED METHOD

1. Source code is converted to the model of association, aggregation, generalization, among classes and inherited classes. These are structural modeling or static modeling of the system and does not change with time. This UML diagram drawn by Rational Rose software. The file is saved with the extension UML.
2. Java parser analysis the UML file and evaluates its expression. Java swing program is used. It collects the information from UML file and produces object tree.
3. Genetic programming in association with fuzzy logic control is used for fittest crossover tree formation. Fuzzy logic control generates imprecise data. Genetic algorithm converts that imprecise value to optimized value in some cases. In other cases distance based approach [4] and compound fuzzy prediction rules have been utilized for desired results.
4. The new tree is converted to binary tree. Optimized test data [1, 3] is generated from binary tree by depth-first search algorithm. Checking of validity, termination of test data generation is done by this algorithm.
5. Mutation testing is used for testing of efficiency of test data.

Graphical representation of our proposed method is given below.

CASE STUDY:

STEP-1: A simple object-oriented program about Ph.D evaluation process has been taken consideration here.

```c++
#include<iostream.h>
class phdscholarEvaluation {
protected:
    char universityName[20];
    char department[10];
    char guidename[20];
    double yearadmission;
public:
    void insertuniversity()
    {
        cout<<"university";
        cin>>universityname;
        cout<<"dept";
        cin>>department;
        cout<<"gdname";
        cin>>guidename;
        cout<<"yearadmission";
    }

```
cin>>yearadmission;
}

void displayuniversity()
{
cout<<"University name= "<<universityname<<"\n";
cout<<"Department name="<<department<<"\n";
cout<<"Guide name="<<guidename<<"\n";
cout<<"Year of admission="<<yearadmission<<"\n";
}
};
class student:public phdscholarevaluation//multilevel class
{
protected:
char name[20];
double roll;
double phonenumber;
public:
void inputstudent()
{
cout<<"student";
cin>>name;
cout<<"roll";
cin>>roll;
cout<<"phone";
cin>>phonenumber;
}
void disstudent()
{
cout<<"Name of student="<<name<<"\n";
cout<<"Roll number is="<<roll<<"\n";
cout<<"Phone number="<<phonenumber<<"\n";
}
};
class synopsis //multiple inheritance
{
protected:
char projecttitle[20];
double datesubmit;
double marks;
public:
void getsynopsis()
{
cout<<"title";
cin>>projecttitle;
cout<<"submit";
cin>>datesubmit;
{
cout<<"Date of submission="<<datesubmit<<"\n";
cout<<"Marks on thesis="<<marksontesis<<"\n";
}
};
class thesisevaluation
{
protected:
date submit;
double marksontesis;
public:
void getthesisevaluation()
{
cout<<"date=";
cin>>datesubmit;
cout<<"marks on thesis";
cin>>marksontesis;
}
void disthesisevaluation()
{

cout<<"Date of submission="<<datesubmit<<"\n";
cout<<"Marks on thesis="<<marksontesis<<"\n";
}

class result : public student,public synopsis,public researchmethodology,public thesisevaluation
{
protected:
double total;
double average;
cout<<"marks";
cin>>marks;
}

void dissynopsis()
{
cout<<"Title of the project="<<projecttitle<<"\n";
cout<<"Submit date="<<datesubmit<<"\n";
cout<<"Marks obtained="<<marksontesis<<"\n";
}
}
};
class researchmethodology
{
protected:
double paper1;
double paper2;
double paper3;
public:
void getmethodology(double x,double y,double z)
{
paper1=x;
paper2=y;
paper3=z;
}
void dismethodology()
{
cout<<"Marks on paper1="<<paper1<<"\n";
cout<<"Marks on paper2="<<paper2<<"\n";
cout<<"Marks on paper3="<<paper3<<"\n";
}
};
class thesisevaluation
{
protected:
date submit;
double marksontesis;
public:
void getthesisevaluation()
{
cout<<"date=";
cin>>datesubmit;
cout<<"marks on thesis";
cin>>marksontesis;
}
void disthesisevaluation()
{

cout<<"Date of submission="<<datesubmit<<"\n";
cout<<"Marks on thesis="<<marksontesis<<"\n";
}

public:
void displayresult();
};
void result::displayresult()
{
total=marks+paper1+paper2+paper3+marksontesis;
cout<<"total marks="<<total<<"\n";
average=(marks+paper1+paper2+paper3+marksontesis)/5;
cout<<"average marks="<<average<<"\n";
displayuniversity();
disstudent();
dissynthesis();
dismethodology();
disthesisevaluation();
}
int main()
{
result r;
r.insertuniversity();
r.inputstudent();
r.getsynopsis();
r.getmethodology(20,10,20);
r.getthesisevaluation();
r.displayresult();
return 0;
}

STEP-2: UML Class diagram of aggregation, directive association are drawn by Star Rational Rose software.

DRAWING-2:

STEP-3: Java parser program parse the UML file and yields the objects form it. java swing program is used.

STEP-4: Left sub trees of each class always represent input membership function with attributes or terminals. Right sub trees of each class always represent output membership function with terminals. Multilevel class inheritance is denoted by vertical nodes.

TABLE-1: EXPLANATION OF NODES USED IN THE TREE
<table>
<thead>
<tr>
<th>SERIAL NO.</th>
<th>USED CLASS, METHODS, AND TERMINALS IN TREE MODEL</th>
<th>NODES</th>
<th>SERIAL NO.</th>
<th>USED CLASS, METHODS, AND TERMINALS IN TREE MODEL</th>
<th>NODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>University name</td>
<td>CN 2</td>
<td>Department</td>
<td>DPT</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>College name</td>
<td>CN 4</td>
<td>Year admission</td>
<td>VA</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Semester evaluation</td>
<td>F203</td>
<td>Insert university</td>
<td>BI</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Student</td>
<td>DSU 8</td>
<td>Student</td>
<td>SSD</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Important</td>
<td>BPS 10</td>
<td>Student</td>
<td>BRS</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Name</td>
<td>NM 12</td>
<td>Roll</td>
<td>RL</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Project title</td>
<td>PTL 16</td>
<td>Synopsis</td>
<td>VSN</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Marka</td>
<td>MR 15</td>
<td>Gamasopita</td>
<td>GSN</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Electrophysiology</td>
<td>DBSNY 20</td>
<td>Research methodology</td>
<td>BDS</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Paper1</td>
<td>PI 22</td>
<td>Paper2</td>
<td>P2</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Paper2</td>
<td>PI 24</td>
<td>Markothesis</td>
<td>GAI</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Dentotheseology</td>
<td>DBSN 24</td>
<td>Dentotheseology</td>
<td>AGM</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Dentotheseology</td>
<td>DBSN 26</td>
<td>Thotheseology</td>
<td>TRHE</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Dentotheseology</td>
<td>DBS 28</td>
<td>Markothesis</td>
<td>MBST</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Dentotheseology</td>
<td>DBS 30</td>
<td>Dentotheseology</td>
<td>MBST</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Dentotheseology</td>
<td>DBS 32</td>
<td>Total</td>
<td>DTE</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Dentotheseology</td>
<td>AVG 34</td>
<td>Total</td>
<td>DTE</td>
<td></td>
</tr>
</tbody>
</table>

New crossover technique as per Janet et al [8] is implemented in genetic programming below for construction of tree structure of objects.

Set of function = \{f1, f2, f4, f5\}. Set of terminals= \{1 or 2\} First and Last are randomly choose function, second two integers are random choice of terminals, next integers are random choice of inputs for the function from the set 1 or 2 or 3.

CREATING INITIAL POPULATION BY RANDOM CHOICE OF INPUT FROM TERMINALS 1 or 2, AND FROM nodes 3,4,5

<table>
<thead>
<tr>
<th>f1, f2, f5</th>
<th>f2, f4, f5</th>
<th>f1, f2, f5</th>
<th>f5, 3, 5</th>
<th>f2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RANDOM CHOICE OF OUTPUT FROM TERMINALS 1or 2 AND ALL NODES 1, 2, 3, 4, 5 TO THE FUNCTION f1, f2, f3, f4, f5.

output

TREE REPRESENTATION OF OUTPUTS ARE:

DRAWING-3:

It is seen that Genetic Programming (GP) generates the tree structure of objects of inherited classes. Large number of steps e.g. cross over tree stage, mutation stage are required for converging it. For converging the GP tree structure we have been implemented GA-FLC auxiliary hybrid system in our proposed method. In auxiliary hybrid system [5] the input value makes a loop within GA and FLC until desired output comes out. Here we have been used fuzzy constraints (f1, f2, f3, f4, f5) and various fuzzy linguistic variables [5]. The violating approximate value is optimized by GA and extended compound fuzzy logic prediction [4].

STEP-5B:

Nodes used in the classes :

f1 f2 f3 f4 f5

Assume value of nodes as per their sequences of function. This value is implemented in FLC.

f1 f2 f3 f4 f5

1 2 3 4 5

Linguistic variables used PS- Positive small, NS- Negative small, PM- Positive medium, NM- Negative medium, NL-Negative large, ZE-Zero, PL- Positive Large

Rule 1: If input value of f1 is NL and input value of f5 is ZE then possible come out NS.

Rule 2: If input value of f1 is ZE and input value of f5 is NM then possible come out NL.

Rule 3: If input value of f1 is ZE and input value of f5 is PS then possible come out ZE.

Rule 4: If input value of f1 is NM and input value of f5 is PM then possible come out PS.

Rule 5: If input value of f1 is PS and input value of f5 is NS then possible come out PS.

Rule 6: If input value of f1 is NS and input value of f5 is ZE then possible come out NM.

Rule 7: If input value of f1 is NL and input value of f5 is NM then possible come out PM.
DRAWING-4:

RULE STRENGTH CALCULATION:
Rule 1: \( \min (0.8, 0) = 0 \)
Rule 2: \( \min (0, 0.11) = 0 \)
Rule 3: \( \min (0, 0.44) = 0 \)
Rule 4: \( \min (0.11, 0.55) = 0.11 (PS) \)
Rule 5: \( \min (0.44, 0) = 0 \)
Rule 6: \( \min (0, 0) = 0 \)
Rule 7: \( \min (0.8, 0.11) = 0.11 (PM) \)

DRAWING-5:

FOR FUZZY SET PS:
Centroid point on x axis is = 4.5
Area between a1 and b1 = \( \frac{1}{2}h \times (a1+b1) \) where \( h = 0.11 \)
\[ = 0.5 \times 0.11 \times (1.8 + 1.2) = 0.165 \]

FOR FUZZY SET PM:
Centroid point on x axis = 5.4
Area between a2 and b2 = \( \frac{1}{2}h \times (a2+b2) \) where \( h = 0.11 \)
\[ = 0.5 \times 0.11 \times (1.8 + 1.2) = 0.165 \]
C.G = \( 0.11 \times 4.5 + 0.11 \times 5.4 \times 0.165 + 0.165 = 4.95 \)

Approximated value of FLC is = 4.95 ≈ 5. It is confirmed that node 1(f1) and 5(f5) both are not connected with node 5. Both are connected with less than any node of 5.

According to Distance based approach of Andrea et al [4], Precondition is 3, and post condition is 4.95. Violating real value or distance is: 4.95-3.0=1.95 nearly 2. It is necessary to minimization this distance. He has been used fuzzy prediction rule for minimization of violating results. For minimization of violating real values I have applied GA in this cases. Binary form of 2 in random chosen values are

a) 0 1
b) 0 0
c) 1 1
d) 1 0

Here optimized value comes out from the above population is 1 1. Its decimal value is 3, So, nodes 1(f1) and 5(f5) must be connected with the object of result node is 3.

Now, for searching the desired objects for f4 and f5 we applied the concept of Andrea et al [4],
RULE1: if (value of f4 is NL) and (value of f5 is ZE) then possible outcome is PL.
RULE2: if (value of f4 is ZE) and (value of f5 is NL) then possible outcome is PL.
RULE3: if (value of f4 is NM) and (value of f5 is ZE) then possible outcome is PM.
RULE4: if (value of f4 is NS) and (value of f5 is PS) then possible outcome is PS.
RULE5: if (value of f4 is PS) and (value of f5 is NS) then possible outcome is NS.
RULE6: if (value of f4 is PL) and (value of f5 is ZE) then possible outcome is NL.
RULE7: if (value of f4 is ZE) and (value of f5 is PS) then possible outcome is PS.
RULE8: if (value of f4 is ZE) and (value of f5 is PM) then possible outcome is PM.
PS: positive small NL=negative large ZE= zero PL= positive large NM= negative medium PM= positive medium NS= negative small,
Automated Test Data Generation Using Fuzzy Logic-Genetic Algorithm Hybridization System for Class Testing Of Object Oriented Programming

**DRAWING-6:**

**RULE STRENGTH CALCULATION:**

RULE1= min(0, 0.55) = 0
RULE2= min(0.55, 0) = 0
RULE3= min(0, 0.55) = 0
RULE4= min(0, 0.44) = 0
RULE5= min(0.44, 0) = 0
RULE6= min(0, 0.55) = 0
RULE7= min(0.55, 0.44) = 0.44 (PS)
RULE8= min(0.44, 0.55) = 0.44 (PM)

**DRAWING-7:**

FOR PS= ½* 0.44* (1.8+1.3)= 0.682
FOR PM= ½*0.44*(1.8+1.3)= 0.682
CG= (4.5*0.682+5.4*0.682)/(0.682+0.682)= 4.9494 ≈ 5

Output of FLC is 4.9494. Therefore, it is confirm that function 4(f4) and 5(f5) never connected with node 4 and 5. Connection must be with any other nodes but less than 4 or 5. In the tree remaining nodes are directly connected with node 1 and indirectly with node 3. Violating extended value or distance based approach is implemented. Distance based approach is necessary to optimized (minimized) by FLC. We have been implemented the concept of extended membership function as per Andrea et al [4] are given below.

\[
Mf (WFF) = [0,1] \\
Mf1(m<n) = \frac{1}{1+e^{-(n-m-1/2)}} \\
Mf2(m\leq n \text{ or } n<m) = \frac{1}{1+e^{-(n-m+1/2)}}
\]

**FUZZY LOGIC CONTROL PARAMETERS:**

<table>
<thead>
<tr>
<th>Function used in FLC</th>
<th>Evaluated values come out from FLC (Center of Gravity)</th>
<th>Destination nodes Or Precondition</th>
<th>Violating Value or post condition= C.G - N</th>
<th>Membership function Mf1</th>
<th>Membership function Mf2</th>
<th>(\mu_1(Mf1)) =1- Mf1, (\mu_2(Mf2)) =1- Mf2</th>
</tr>
</thead>
<tbody>
<tr>
<td>f4, f5</td>
<td>4.9494</td>
<td>1</td>
<td>4.9494 – 1 = 3.9494</td>
<td>0.7880</td>
<td>1-0.7880 = 0.212</td>
<td>(\mu_3(\mu_1 \lor \mu_2)) = min (\mu_1, \mu_2)</td>
</tr>
<tr>
<td></td>
<td>4.9494</td>
<td>3</td>
<td>4.9494-3 = 1.9494</td>
<td>0.400</td>
<td>1-0.400 = 0.6</td>
<td>(\mu_3=\min(.212, 0.6)) = 0.212</td>
</tr>
</tbody>
</table>

It is seen from the above results that the nodes 4(f4) and 5(f5) having extended membership function is 0.212. This value is coming from desired node 1. Therefore, node 4(f4) and 5(f5) is connected with desired node 1. By the using of extended predicates we can proved that remaining nodes 1(f1) and 2(f2) are also connected with node 5(f5).

In this way we can jumped the huge number of steps of Genetic programming crossover tree. Therefore, new tree structure is formed according to the deduced results of FLC and GP hybridized system. New tree is then converted to binary tree. Finally optimized valid test data are generated by Depth first search tree algorithm.

**STEP-5C:**
Genetic Programming And Fuzzy Logic Compound Predicates Is Used For Optimized Following Tree.

**FUNCTION**
- f13
- f14
- f15
- f16
- f17
- f18

**TERMINALS**
- 12
- 13
- 14

**NEXT INTEGERS ARE**
- RANDOM CHOICE OF INPUT FOR FUNCTION
  - 10
  - 11

**REPRESENTATION OF TREE STRUCTURE OF OUTPUT**: DRAWING-8

Now FLC is used to determine the connected node of objects for
(A) f13, f12, and f10
(B) f14, f17, f18 in place of convergence technique of GP crossover tree.

- **R1-IF** (f12 is NL) and (f10 is ZE) and (f13 is PL) THEN output instance object is NM
- **R2-IF** (f12 is ZE) and (f10 is NL) and (f13 is PS) THEN output instance object is PM
- **R3-IF** (f12 is PL) and (f10 is NM) and (f13 is ZE) THEN output instance object is NS
- **R4-IF** (f12 is PS) and (f10 is NS) and (f13 is ZE) THEN output instance object is ZE
- **R5-IF** (f12 is ZE) and (f10 is PS) and (f13 is NS) THEN output instance object is PS
- **R6-IF** (f12 is NM) and (f10 is PM) and (f13 is NS) THEN output instance object is ZE
- **R7-IF** (f12 is PL) and (f10 is ZE) and (f13 is PS) THEN output instance object is PS
- **R8-IF** (f12 is PS) and (f10 is NS) and (f13 is ZE) THEN output instance object is PM

**ASSUME VALUE OF**
- f10
- f12
- f13
- f14
- f17
- f18

**DRAWING-9:**
Automated Test Data Generation Using Fuzzy Logic-Genetic Algorithm Hybridization System for Class Testing Of Object Oriented Programming

**RULE STRENGTH:**

R1: \( \min (0, 0.666, 0) = 0 \)

R2: \( \min (0.666, 0, 0.33) = 0 \)

R3: \( \min (0, 0, 0.666) = 0 \)

R4: \( \min (0.33, 0.33, 0) = 0 \)

R5: \( \min (0, 0.33, 0.66) = 0 \)

R6: \( \min (0, 0, 0.33) = 0 \)

R7: \( \min (0.33, 0.666, 0.33) = 0.33 \)

R8: \( \min (0.33, 0.33, 0.66) = 0.33 \)

**DRAWING:** 10

For fuzzy set PS

X axis centroid point = 14, rule strength applied for calculation of \( CG = 0.33 \)

Fuzzy set for PM

X axis centroid point = 17, rule strength applied for calculation of \( CG = 0.33 \)

Shaded area = \( \frac{1}{2}h(a1+b1) = \frac{1}{2} \times 0.33 \times (6+5.8) = 1.9 \)

Therefore, weighted average =

\[
CG = \frac{1.9503 \times 14 + 1.9503 \times 17}{1.9503 + 1.9503} = 15.5
\]

**FUZZY LOGIC CONTROL PARAMETERS:**

<table>
<thead>
<tr>
<th>SERIAL NO.</th>
<th>CG VALUE OF FLC.</th>
<th>N</th>
<th>VIOLATION VALUE = C.G - N</th>
<th>M</th>
<th>MF1</th>
<th>MF2</th>
<th>MF3 DISJUNCTIO N OF PREDICATE = ( \min(MF1, MF2) )</th>
<th>FUNCTION USED IN FLC</th>
<th>CONNECTED NODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15.5</td>
<td>13</td>
<td>15.5-13=2.5</td>
<td>2.5</td>
<td>0.035</td>
<td>-</td>
<td>-</td>
<td>t0,f10,f12,f13</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>15.5</td>
<td>12</td>
<td>15.5-13=2.5</td>
<td>2.5</td>
<td>-</td>
<td>0.039</td>
<td>0.049</td>
<td>f10,f12,f13</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>15.5</td>
<td>10</td>
<td>15.5-13=2.5</td>
<td>2.5</td>
<td>-</td>
<td>0.030</td>
<td>0.035</td>
<td>f10,f12,f13</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>17.998</td>
<td>18</td>
<td>17.998-12=5.998</td>
<td>5.998</td>
<td>0.030</td>
<td>-</td>
<td>-</td>
<td>f17,f18</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>17.998</td>
<td>17</td>
<td>17.998-12=5.998</td>
<td>5.998</td>
<td>-</td>
<td>0.033</td>
<td>0.030</td>
<td>f17,f18</td>
<td>12</td>
</tr>
</tbody>
</table>

Now tree is converted into fittest tree. In this way we can jump the large number of steps for converging the GP fittest tree structure.

Now tree is converted to binary tree. Depth first search algorithm is used to generate valid test data with one time testing path of binary tree. Termination of test data generation is also done by this searching algorithm.

Binary tree structure is given below:
TABLE-2: SEQUENCE OF TESTDATA GENERATION

<table>
<thead>
<tr>
<th>SRL. NO</th>
<th>TESTDATAGENERATION</th>
<th>RESU LT</th>
<th>SRL. NO</th>
<th>TESTDATAGENERATION</th>
<th>RESU LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UN,DPT,GN,YAD</td>
<td>V</td>
<td>2</td>
<td>ISU,UN,DPT,GN,YAD</td>
<td>V</td>
</tr>
<tr>
<td>3</td>
<td>DSU,UN,DPT,GN,YAD</td>
<td>V</td>
<td>4</td>
<td>PHDSE,ISU,UN,DPT,GN,YAD,DSU,UN,DPT,GN,YAD,RES,ISU,UN,DPT,GN,YAD</td>
<td>V</td>
</tr>
<tr>
<td>5</td>
<td>RES,ISU,UN,DPT,GN,YAD,DSU,UN,DPT,GN,YAD</td>
<td>V</td>
<td>6</td>
<td>NM,RL,PHN,     V</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>IPS,NM,RN,PHN</td>
<td>V</td>
<td>8</td>
<td>DIS,NM,RN,PHN     V</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>SD,IPS,NM,RN,PHN,DIS,NM,RN,PHN</td>
<td>V</td>
<td>10</td>
<td>RES,IPS,NM,RN,PHN,DIS,NM,RN,PHN,</td>
<td>V</td>
</tr>
<tr>
<td>11</td>
<td>PTL,DTS, MK</td>
<td>V</td>
<td>12</td>
<td>GSYN,PTL,DTS, MK</td>
<td>V</td>
</tr>
<tr>
<td>13</td>
<td>SYN, GSYN, PTL, DTS, MK</td>
<td>V</td>
<td>14</td>
<td>SYN, DISSYN, PTL, DTS, MK</td>
<td>V</td>
</tr>
<tr>
<td>15</td>
<td>P1,P2,P3</td>
<td>V</td>
<td>16</td>
<td>GM,P1,P2,P3       V</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>RMD, GM,P1,P2,P3</td>
<td>V</td>
<td>18</td>
<td>DISSYN, P1,P2,P3  V</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>RMD, DISSYN, P1,P2,P3</td>
<td>V</td>
<td>20</td>
<td>DSU, MRKT,      V</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>GETE, DSU, MRKT</td>
<td>V</td>
<td>22</td>
<td>THSE, GETE, DSU, MRKT</td>
<td>V</td>
</tr>
<tr>
<td>23</td>
<td>DISTE, DSU, MRKT</td>
<td>V</td>
<td>24</td>
<td>THSE, GETE, DSU, MRKT, DISTE, DSU, MRKT</td>
<td>V</td>
</tr>
<tr>
<td>25</td>
<td>TOT, AVG</td>
<td>V</td>
<td>26</td>
<td>DISR, TOT, AVG   V</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>DISR, DSU, DISSYN, DIM, DISTE</td>
<td>V</td>
<td>28</td>
<td>RE, ISU, DSU, IPS, DIS, GSYN, DISSYN, GM, DM, GETE, DISTE</td>
<td>V</td>
</tr>
<tr>
<td>29</td>
<td>RE, DISR, DSU, DISSYN, DIM, DISTE</td>
<td>V</td>
<td>30</td>
<td>P1,P2,UN,DPT,GN,YAD</td>
<td>I</td>
</tr>
<tr>
<td>31</td>
<td>DIS, NM, RN, PHN, PTL, DTS</td>
<td>I</td>
<td>32</td>
<td>ISU, UN, DPT, GN, YAD, TOT, AVG, P1,P2,P3</td>
<td>I</td>
</tr>
<tr>
<td>33</td>
<td>DSU, MRKT, IPS, NM, RN, PHN</td>
<td>I</td>
<td>34</td>
<td>GSYN, PTL, DTS, MK, DISR, TOT, AVG, THSE, GETE</td>
<td>I</td>
</tr>
</tbody>
</table>

V=VALID TESTDATA,  I= INVALID TESTDATA.

STEP-6: MUTATION TESTING:
Mutation testing is invoked for efficiency testing and analysis of test data. Faults injected to the program. So, the program would generate a set of mutant. Faulty program is called mutant. Run the mutant program against all the test cases one by one and fault is revealed from mutant. That fault is compare with original program with same inputs. The behavioral changes between mutant and original program is removed by test data. This is called killed mutant. In this way the complex fault which are generated from simple fault can be removed by this testing. It is widely used for unit testing of software. Though, mutation testing takes long time and making hard to predict for loop program. TABLE-5:

<table>
<thead>
<tr>
<th>MUTATION TESTING FOR CLASS PROGRAM OF CASE STUDY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRL. NO</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
IX. LIMITATION AND FUTURE CHALLENGES

One time testing tool. Tool to be used for testing particular software.

1. Many researchers has been focused on the automatic test data generation using soft computing technique like genetic algorithm, memetic algorithm. But still many research questions are in need of answer.

Two points we want to focus of our personal interest that need to investigate in future challenges.

At present time modern industry in India has been used computer aided design-computer aided manufacturing (CAD-CAM) System for increasing quality production. One of the software program that is computerized numerical control (C.N.C) is necessary for controlling production. The CNC program is written by Mechanical engineer based on CAD-CAM system. Automatic testing of that software (CNC) can be done by FLC-GA hybridization technique based on CAD-CAM system. Test data here traversed the path of tools and jobs movement through X, Y, Z axis. These automated generated test data save the productions from rejection. So, our proposed technique can increase the profit of modern industry.

2. Any where any place automatic software testing tool generation.


X. CONCLUSION

Automated test data generation minimizes manpower, cost, and time. FLC-GA auxiliary hybridization technique has removed the drawback of genetic crossover tree programming during automated test data generation. Jumping of Several steps of Genetic crossover tree operators has been done by this new proposed technique. Without class testing, others objects and method testing of object oriented programming is value less. Model based approach is suggested here. UML is a very popular and widely industrial accepted software is used for designing class diagram. Test cases to be available early during development of software lifecycle from models. So, we can made from it the effective test design as well as we can check the software coding . Tree structures of objects are made from collecting information of java parser. Tree structure having balancing property.Sorting and searching of nodes in tree structures is made easy in computer memory. Our approach is to use Genetic programming tree coupled with fuzzy logic control for optimized test data generation. Valid test data and termination is done by binary tree, applying with depth first search algorithm.

REFERENCES


