OPTIMIZATION OF SOFTWARE TESTING USING HEURISTIC TECHNIQUE

Report submitted for the partial fulfilment of the requirements for the degree of Bachelor of Technology in Information Technology

Submitted by

TITAS SAHA
University Roll No.: 11700215102

MOHIT JAIN
University Roll No.: 11700214042

SOUMALLYA GHOSH
University Roll No.:11700215099

Under the Guidance of

DR. DIPANKAR MAJUMDAR
Associate Professor, Department of Information Technology

RCC Institute of Information Technology, Kolkata

RCC Institute of Information Technology
Canal South Road, Beliaghata, Kolkata – 7OOO15
[Affiliated to West Bengal University of Technology]

Department of Information Technology
RCC Institute of Information Technology, Kolkata
Canal South Road, Beliaghata, Kolkata – 7OOO15

Approval

This is to certify that the project report entitled "Graph Based Simulation & Optimization of Software Testing" prepared under my supervision by (**Titas Saha, Roll Number: 11700215102 and Mohit Jain, Roll Number: 11700214042 and Soumallya Ghosh, Roll Number: 11700215099**), be accepted in partial fulfilment for the degree of Bachelor of Technology in Information Technology.

It is to be understood that by this approval, the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn thereof, but approves the report only for the purpose for which it has been submitted.

....

Dr.Abhijit Das, HOD (I.T Dept)

Dr.Dipankar Majumdar ,Mentor
And professor (I.T Dept)

Acknowledgment

We would like to express our sincere gratitude to Mr. Dipankar Majumdar of the department of Information Technology, whose role as project guide was invaluable for the project. We are extremely thankful for the keen interest he / she took in advising us, for the books and reference materials provided for the moral support extended to us.

Last but not the least we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staff for the gracious hospitality they offered us.

Place:

Date:

.....

.....

.....

Index

1. Introduction

Software Engineering is extremely relevant for the important stage of systems development. Since its conception in the 1970s, important advances in software quality were achieved due to research models, standards and methodologies to support systems development. The application of such methods takes place in several aspects related to the development process, such as Project Planning, Requirements Analysis and Software Testing.

Unfortunately, in some cases, the conventional methods developed by the specialized scientific community are not able to solve certain problems that arise during the process of software development. These problems occur in inherently complex problems, such as those involving the selection of a solution in a prohibitively large set of possibilities. Problems of such kind require automated resolution methods, so the problem can be solved efficiently.

In the Software Testing phase, we find some problems that can be modelled using automated methods. For example, consider the activity of test cases prioritization. This activity consists in determining the best execution order for the test cases of a system. The quality of an order is defined by means of a coverage metric calculated mathematically that determined how soon a test set covers the entire system, rendering the rest of the tests unnecessary. Although this metric can be described using text, the resolution of the problem itself is too costly given the number of possible permutations among the test cases. Hence, the solution to the problem of test cases prioritization cannot be easily described by rules written in textual documents or steps and standards. Nevertheless, the mathematical modelling of the problem as an optimization problem is desirable given the existence of mathematical characteristics in the problem.

2. Problem Definition

Software Testing is the phenomenon of finding errors after executing the programs. Software testing can be defined by many processes designed sequentially and does not do anything unintended. The objective of software testing is to finalize the application software against the user requirements. It must have good test coverage to test the application software and perform as per the specifications. For generating list of coverage's, the test cases should be designed with maximum possibilities of finding various errors or bugs. The test cases should be very effective and is measured through the number of defects or errors reported. Generation of test cases or test data is a method to identify the data set which satisfies the criteria.

Optimization intends to maximize or minimize a mathematical function defined by coefficients and variables. The variables that define the function may be restricted, i.e., the variables must satisfy a set of equations defined according to each problem instance. Heuristics are a subset of the techniques that can be used to solve optimization problems.

The term heuristics was introduced by Glover and represents a class of generic search algorithms. Thus, heuristics are generic kind of heuristic, i.e., that can be used in different kinds of problems. These methods use ideas from different areas as an inspiration to make the process of finding the solution to optimization problems. As examples of a heuristic we can underscore Simulated Annealing which is based on a physical process in metallurgy. Other examples are Genetic Algorithms, which are based on concepts of evolution population.

The overall execution process of a heuristic is the search for a subset of solutions in the solution space guided by fitness functions. The fitness function is a mathematical function that assigns a value to each solution in the search space. As a comparison, consider a exhaustive search: to determine the best solution to a problem, all existing solutions are visited and in the end the best is returned. Another type of technique that can be used to solve optimization problems are the exact methods. In this approach the search is done from decisions based on mathematical theorems. The operation of heuristics works in a way that to determine the final solution, only some existing solutions are actually visited. The search is conducted under a process that is specific to each heuristic, but it is a way that attempts to intelligently find good solutions. However, there is no guarantee the solution returned by a heuristic is the best.

Our problem definition requires us to create a Control Flow Graph and apply Genetic Algorithm

3. LITERATURE SURVEY

For the given project requires Heuristic technique. For this reason we have studied Praveen Ranjan Srivastava's article on Automated "Software testing using meta-heuristic technique" which is based on optimization in software testing efficiency. We have studied this article for referencing GA(Genetic Algorithm), A GA starts with guesses and attempts to improve the guesses by evolution. A GA will typically have five parts: a representation of a guess called a chromosome, an initial pool of chromosomes, a fitness function, a selection function and a crossover operator and a mutation operator. A chromosome can be a binary string or a more elaborate data structure. The initial pool of chromosomes can be randomly produced or manually created. Genetic algorithms are often used for optimization problems in which the evolution of a population is a search for a satisfactory solution given a set of constraints. In this paper we have learnt how possible to apply Genetic Algorithm techniques and fitness function for finding the minimum shortest paths for improving software testing efficiency. The Genetic Algorithms also outperforms the exhaustive search and local search techniques.

Also we have studied Rudolf Ferenc's article on "Optimization of software test by the application of Heuristic Technique" and it is based on how implement heuristic technique and what is heuristic technique. We have studied this article for referencing Heuristic Optimization, within which we have studied different types of Heuristic technique which are Hill Climbing, Simulated Annealing and Genetic Algorithm. Besides we have studied Wenfei Fan's article on 'Distributed Graph Simulation' and it is based on undamental problems for distributed graph simulation. We have learnt how to simulate a graph model.We have also reffered to an article named "optimization in software testing Using Metaheuristics " by Fabrício Gomes de Freitas1 , Camila Loiola Brito Maia2 , Gustavo Augusto Lima de Campos3 . Jerffeson Teixeira de Souza for study of different techniques and approaches that can be applied in our project that is Optimization in software testing Using Heuristic Approach.

4. SRS (SOFTWARE REQUIREMENT SPECIFICATION)

1.) Functional requirements:-

a.) There must be a user to

1.) Give input for threshold for matrix in truncation

2.) To give choice weather the output of Floyd Warshall output is upto the mark or not.

2.) Software requirements

a.) Operating System

To run any programme in the system OS must be present .The OS used in our project is UBUNTU 16.04 LTS which is for 64bit processer.

b.) Editor or IDE

To write the programme we require a IDE or a editor. We have used Gedit editer, Gedit is the official text editor of the GNOME desktop environment. While aiming at simplicity and ease of use, Gedit is a powerful general purpose text editor.

3.) Hardware requirements

a.) A desktop or PC/laptop.

A desktop or Pc is required for working on the programme i.e. writing and editing of code. Here we have used laptop which DELL inspirion 3453 i5 7^{th} generation

b.) A pen drive or a CD

A pen drive or a CD is required for transferring of data, which may be

articles, software or any other data required for project

4.) Non Functional requirement

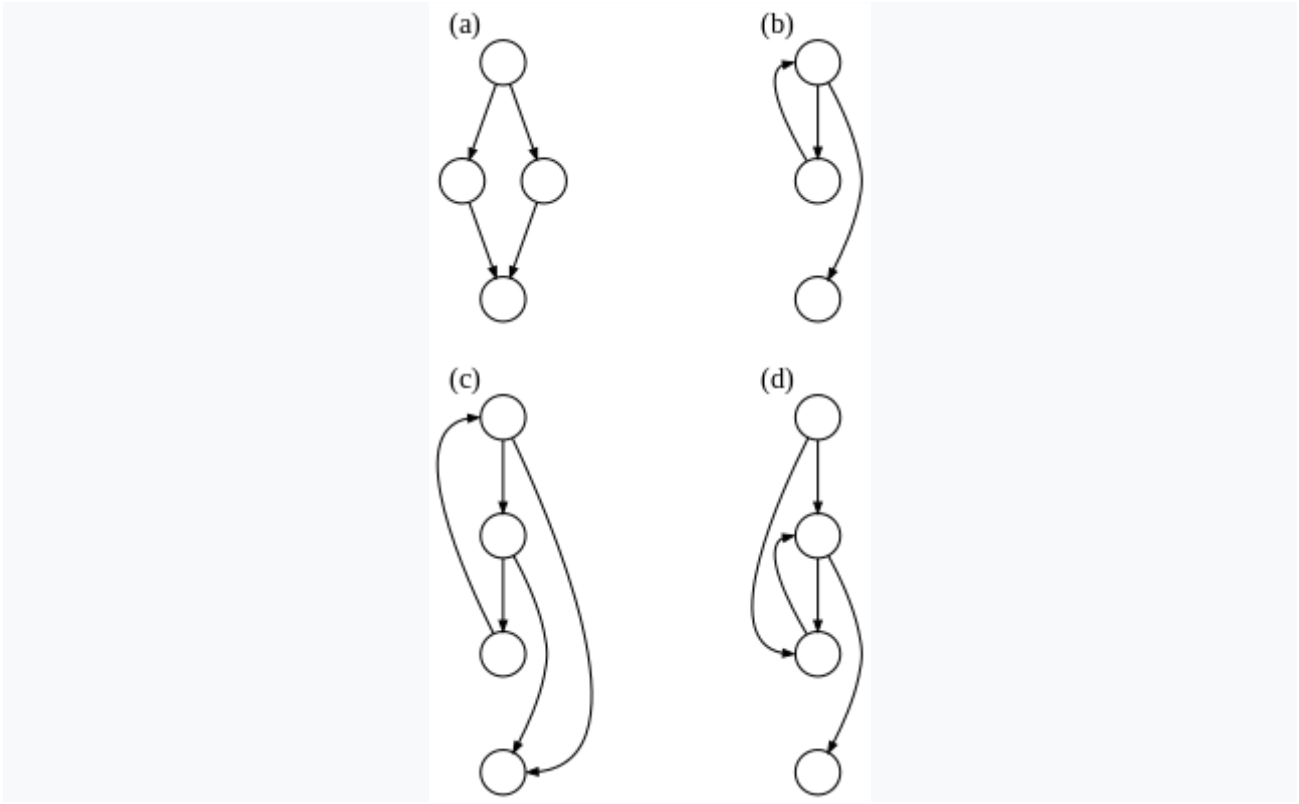
The programme must be able to generate or simulate a Cfg (control Flow Graph) without any error and traversal of the cfg using various random value must be smooth.and there must be application of Genetic algorithm

5. Planning

The project simulates white box testing .genereating a model control flow graph we are required to apply genetic algorithm to make it optimized .the CFG(control Flow Graph) is generated using a randomly generated number array.the array is truncated by a random value given by the user .After that it is put into floyd Warshall such that we have a precursor to control flow graph . The precursor model is then used to create a dedicated node and a end node such that the graph looks like the true form of CFG.After a CFG is created we are initialising a population before applying GA(genetic Algorithm) ,For this we have used random number as seeds and from that seeds tracked down paths traversed in CFG the two seeds which covered longest path is taken as parents which are then stored as binary strings .From these two parents children are generated using crossover function which is nothing but making new binary strings by reshuffling and exchanging parent Binary strings
.The newly generated childrens strings are passed into mutation function where there binary strings are mutated or changed at certain bit position.Now when the population is created genetic algorithm is applied which is based on Darwin's theory of survival of fittest ,for that purpose a fitness function is then taken which selects the parents for next generation ,this step is followed by crossover, mutation and again fitness function the cycle continues until the desired result is achived.

6.Design

1.Control Flow Graph :-



An example of Control Flow Graph

A **control flow graph** (**CFG**) in computer science is a representation, using graph notation, of all paths that might be traversed through a program during its execution

In a control flow graph each node in the graph represents a basic block, i.e. a straight-line piece of code without any jumps or jump targets; jump targets start a block, and jumps end a block. Directed edges are used to represent jumps in the control flow. There are, in most presentations, two specially designated blocks: the *entry block*, through which control enters into the flow graph, and the *exit block*, through which all control flow leaves

a.)algorithm for Genteration of matrix:

```
generate()
{
 int i,j,n;
 srand(Null);        //initializes the starting value (seed )for generation of random
for(i=0:dimrow-1)                                  number
  for(j=0:dimcol-1)
    {
     n=rand()%100;
     m[i][j]=n;
     if(i==j)
       {
        m[i][j]=0;         //initializes  the diagonal elements=0
       }

     }
```

```
}
```

b.)algorithm for truncation and normalization of matrix

```
truncate()
{
float thresh;
int i,j;
scanf(thresh);    //taken user input for truncating the matrix
for(i=0:dimrow-1)
  for(j=0:dimcol-1)
    {
     if(m[i][j]<=thresh)
      {
       am[i][j]=0;   // every value less than threshold or equal to it is re-initialized
                     //    as 0

      }
     else
      {
      am[i][j]=m[i][j]; //greater than threshold are left in the adjacency matrix

 }
    }


print(am);


  normalize();
  print(am);
}

normalize()

{
 int i,j,k,sum;
for(i=0:dimrow-1)
{
  sum=0;
  for(k=0:dimcol-1)
  {
  sum=sum+am[i][k]; // sums up each element in the row for division
  }
   for(j=0:dimcol-1)
   {
   am[i][j]=am[i][j]/sum;  //each element in matrix becomes a fractional value
   }
}
 for(i=0:dimrow-1)
 {
   for(j=0:dimcol-1)
```

```
   {

    if((i!=j)&&(am[i][j]=0))  //since  the diagonal elements are already 0
    {                         //so excluding them those elements which are
                              //truncated
     am[i][j]=999.99          //and normalized are reiniialized as 999.99 which is
                              //a
    }                         //representation of  infinity
   }
  }
}
```

c.)after truncation of matrix has been done application of it in floyd warshall algorithm is done on matrix am[][]

```
int floyd()
{
 int i,j,k,n;
 for(k=0:dimrow-1)
  {
   for(i=0:dimrow-1)
    {
     for(j=0:dimcol-1)
     {
      if(am[i][k]+am[k][j]<am[i][j])
       {
         am[i][j]=am[i][k]+am[k][j];
       }
     }
    }
  }

 print(am);
 printf("would you like to truncate again if result of floyd warshall is not uptomark if yes
give 1 or else give 0 ");
 scanf("%d",&n);                    //asks for user choice
 return(n);
}
```

d.) Algorithm for creating a dedicated node(ghost node or a node from where we can go to anywhere in the matrix) and end node .

i.)

```
void create_dedicated_node()
{
  int i,j,k;                          //basicly the previous matrix n x n is being
copied into a
  float sum;                          // a 2d matrix of (n+1) x (n+1) named m2
  m2[0][0]=0;                        //the variable  sum is used for summing up the
colum of
```

```
 for(i=1;i<=dimrow;i++)
 {
   sum=sumcol(i);                    //sumcol() is nothing  but a function which is
required for
   print(i,sum);                     //summing up of column
   m2[0][i]=(sum/(dimcol+1));
 }
 For i=1 to dimrow
 {
  For j=0 to dimcol
  {
   if(j==0)
   {
     m2[i][j]==0;                          //  since we can go to each and every node
from
   }                                        // ghost node or dedicated node but
reverse process is
   if((j!=0))                               //not true
   {
     m2[i][j]=am[i-1][j-1];
   }
  }
 }
```

ii.)

```
void creatend()

{
inti,j;
float sum; //the sum variable is used for summing up the values in the particular
//row
For i=0 to dimrow

{

  sum=sumrow(i);                  //sumrow() returns sum of particular  row

  print(,i,sum);

  For j=0 to dimcol

   {

   m3[i][j]=m2[i][j];             // the m2 matrix  which was created during creation
of

   }                              //dedicated node

    m3[i][dimrow+1]=((sum)/(dimrow+2));

  }
```

```
   For j=0 to dimcol

   {

    m3[dimrow+1][j]=0.0;                    //the last row of matrix is filled with zeros
because

   }                                                    // so as to create a exit condition for
control flow graph

}
```

3.)Creation of Population for Genetic Algorithm:-

a.)Algorithm for creation of seed:-

```
  void seed()

{

  int i;

  srand(rand())                              //seeding of random number genretor

  for i=0 to 6

  {

   a[i]=random number between 0 to 100;

   print(a[i]);

  }

 }
```

b.) Algorithm for creation of paths and there distance travelled :-

```
void using_seeds()

{

 int k;

 float distance;

 int nxtnode;

 int r;
```

```
for k=0 to 6
 {
   srand(a[k]);                    // seeding of random number generator
   nxtnode=0;
   distance=0.00;
   while(nxtnode!=dimcol+1)       //generation of path form  random numbers
    {
      r=(random number  from 0 to dimrow+1);
      distance=distance+m[nxtnode][r];
      if(r==0)
        {
          nxtnode=dimcol+1;
        }


 nxtnode=r;
}
   path[k]=distance; // the path variable stores value of total distance for different random
                              //number
 }
}
```

c.)Algorithm for initializing population

```
void initiate()

{

 int i,j,k,top,second,temp;

 float  b1,b2;

 b1=path[0];

 top=a[0];

 for j=1 to 6

   {

     if(b1<=path[j])              //those numbers are chosed as parents who have greater distance

      {                          // covered

        b1=path[j];

        top=a[j];                 //top selects the number as parent which has covered the largest distance

        temp=j;

      }

}



 b2=path[0];

 second=a[0];

 for(j=1;j<7;j++)

   {

     if((b2<=path[j])&&(j!=temp))

      {

        b2=path[j];

        second=a[j];

      }
```

```
   }

 tobinary(top,parent1);      //the to binary function converts number in binary string format

 tobinary(second,parent2);

}
```

4.)Genetic Algorithm:-

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. You can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear. The genetic algorithm can address problems of mixed integer programming, where some components are restricted to be integer-valued.

The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:

- *Selection rules* select the individuals, called *parents*, that contribute to the population at the next generation.
- *Crossover rules* combine two parents to form children for the next generation.
- *Mutation rules* apply random changes to individual parents to form children.

a.)crossover rule

Crossover:-

```
void crossover()

{

  int i,j,k,n;

  n=random number from 0 to 7;

  for i=0 to n-1

  {

    child1[i]=parent1[i];        // assignment of certain parts of parent's binary in the children
child2[i]=parent2[i];      // so as to create a population for 0' th  generation
```

```
 }
 For j=n to 8
 {
   child1[j]=parent2[j];  // assignment of childen
   child2[j]=parent1[j];
 }
 For i=0 to 8
 {
   print("%d",child1[i]);        //prints the first child
 }
 For j=0 to 8
{
   printf("%d",child2[j]);  //the second children is printed
 }

}
```

b.) Mutation rule:-

Algorithm for mutation of children in a particular generation

```
void mutation()
{
 int i,j,k,n;
 srand(rand());
 n=random number from 0 to 7;
 if(child1[n]==1)    //the binary  bit is changed in child1
 {
```

```
child1[n]=0;
 }

 else

  {

    child1[n]=1;

  }

  if(child2[n]==1) //the binary bit is mutated in child2

  {

    child2[n]=0;

  }

 else

  {

    child2[n]=1;

  }

}
```

c.) Selection Rule:-

algorithm for Fitness function

```
void fitness()

{

 int  i,j,k,top,second,nxtnode ;

 int gen[4];   //basicly gen array stores the parent and children at index 0,1 and at
index 2,

 float dis[4];   //dis gets distance traveled by respective population

 gen[0]=todecimal(parent1);//toodecimal converts binary to decimal using power
 gen[1]=todecimal(parent2);//function
```

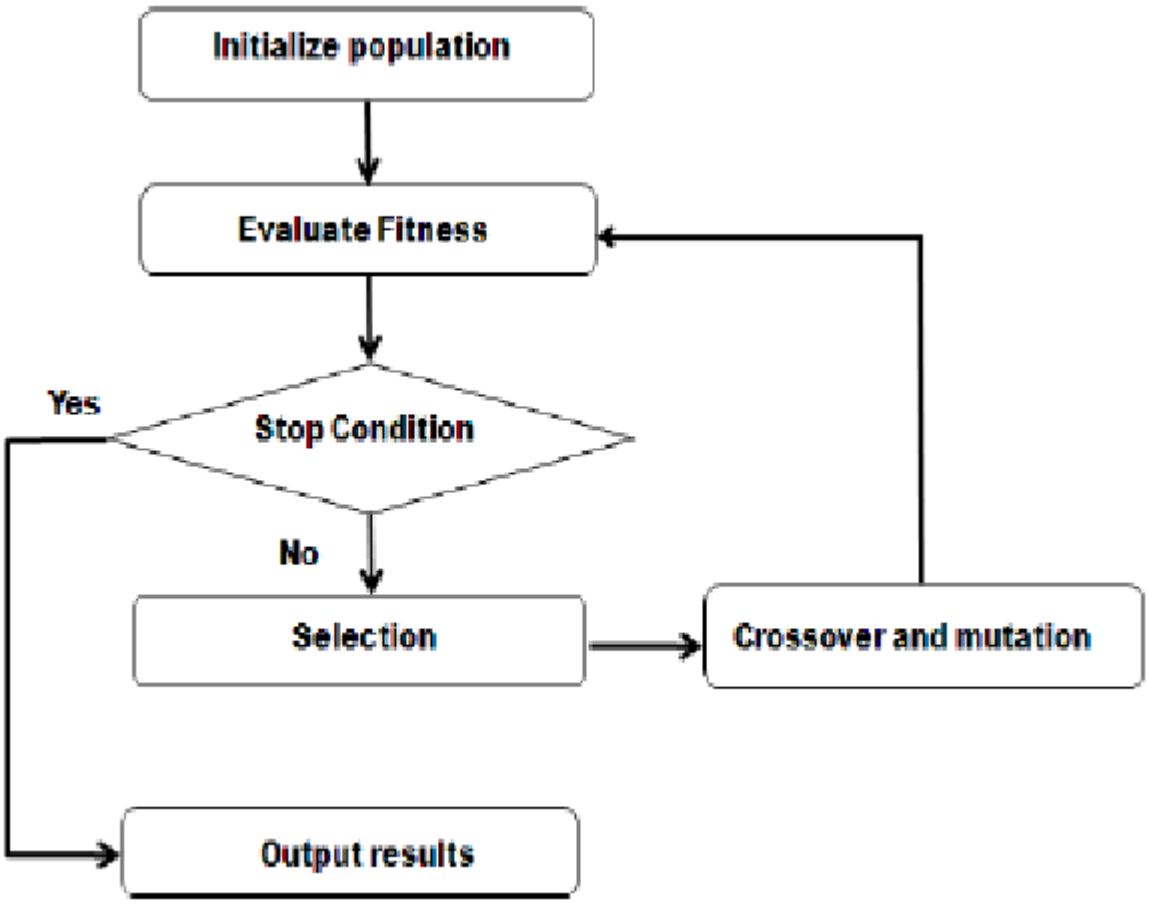
```
gen[2]=todecimal(child1);

gen[3]=todecimal(child2);

dis[0]=0.00;

nxtnode=0;

printf("\nthe first node for %d is 0",gen[0]);    //the distance traversed for parent1

srand(gen[0]);   // is    calculated

while(nxtnode!=dimrow+1)

{

  k=random number from 0 to dimrow +1;

  if(k==0)

  {

   k=dimrow+1;

  }

  dis[0]=dis[0]+m3[nxtnode][k];

  nxtnode=k;

}//parent1 ends

printf("\nthe total distance comes to be %f\n",dis[0]);

dis[1]=0.00;                                        //parent2 distance is again
nxtnode=0;                                          //calculated

 printf("\nthe first node for %d is n0",gen[1]);

 srand(gen[1]);


while(nxtnode!=dimrow+1)

{
```

```
k=(rand())%(dimrow+2);
  if(k==0)
  {
   k=dimrow+1;
  }
  dis[1]=dis[1]+m3[nxtnode][k];
  nxtnode=k;
  printf("\nThe nextnode for %d is %d",gen[1],nxtnode);
}
printf("\nthe tota distance comes to be %f\n",dis[1]);
dis[2]=0.00;
nxtnode=0;
printf("\nthe first node for %d is n0",gen[2]); //child1 distance is again calculated
srand(gen[2]);
while(nxtnode!=dimrow+1)
{
  k=(rand())%(dimrow+2);
  if(k==0)
  {
   k=dimrow+1;
  }

dis[2]=dis[2]+m3[nxtnode][k];
  nxtnode=k;
  printf("\nThe nextnode for %d is %d",gen[2],nxtnode);
```

```
 }

 printf("\nthe total distance comes to be %f\n",dis[2]);

 dis[3]=0.00;

 nxtnode=0;

 printf("\nthe first node for %d is n0",gen[3]); //child22 distance is again
calculated

 srand(gen[3]);

 while(nxtnode!=dimrow+1)

 {

   k=(rand())%(dimrow+2);

   if(k==0)

   {

    k=dimrow+1;

   }

   dis[3]=dis[3]+m3[nxtnode][k];

   nxtnode=k;

   printf("\nThe nextnode for %d is %d",gen[3],nxtnode);

 }

 printf("\nthe total distance comes to be %f\n",dis[3]);

 top=0; // from here own we are selecting nodes for next generation based on
distance


 for(i=0;i<4;i++)
{

  if(dis[top]<dis[i])

  {
```

```
    top=i;

  }

 }

 for(i=0;i<4;i++)

 {

  if((dis[second]<dis[i])&&(i!=top))

  {

    second=i;

  }

 }

 tobinary(gen[top],parent1);//to binary converts integer and store them as binary value in array and also it can be used to initialize

 tobinary(gen[second],parent2);//parents for next generation

}
```

A general flow diagram for genetic algorithm



7. Implementation and Flow chart

Step by step implementation of project and flow chart

1.) We have taken 20*20 matrix we have generated this matrix using random number generation.

2.) After that we have truncated the matrix using user input.

3.) Then we have normalized the matrix

4.) And then it is put in a Floyd Warshall algorithm the output of which is Precursor to control flow graph.

5.) Dedicated node or starting node and end node or exit node are created which is true form of control Flow Graph.

6.) After this populations for application of Genetic Algorithm is Created first parents are created by random number seeds based on distance travelled by seeds.

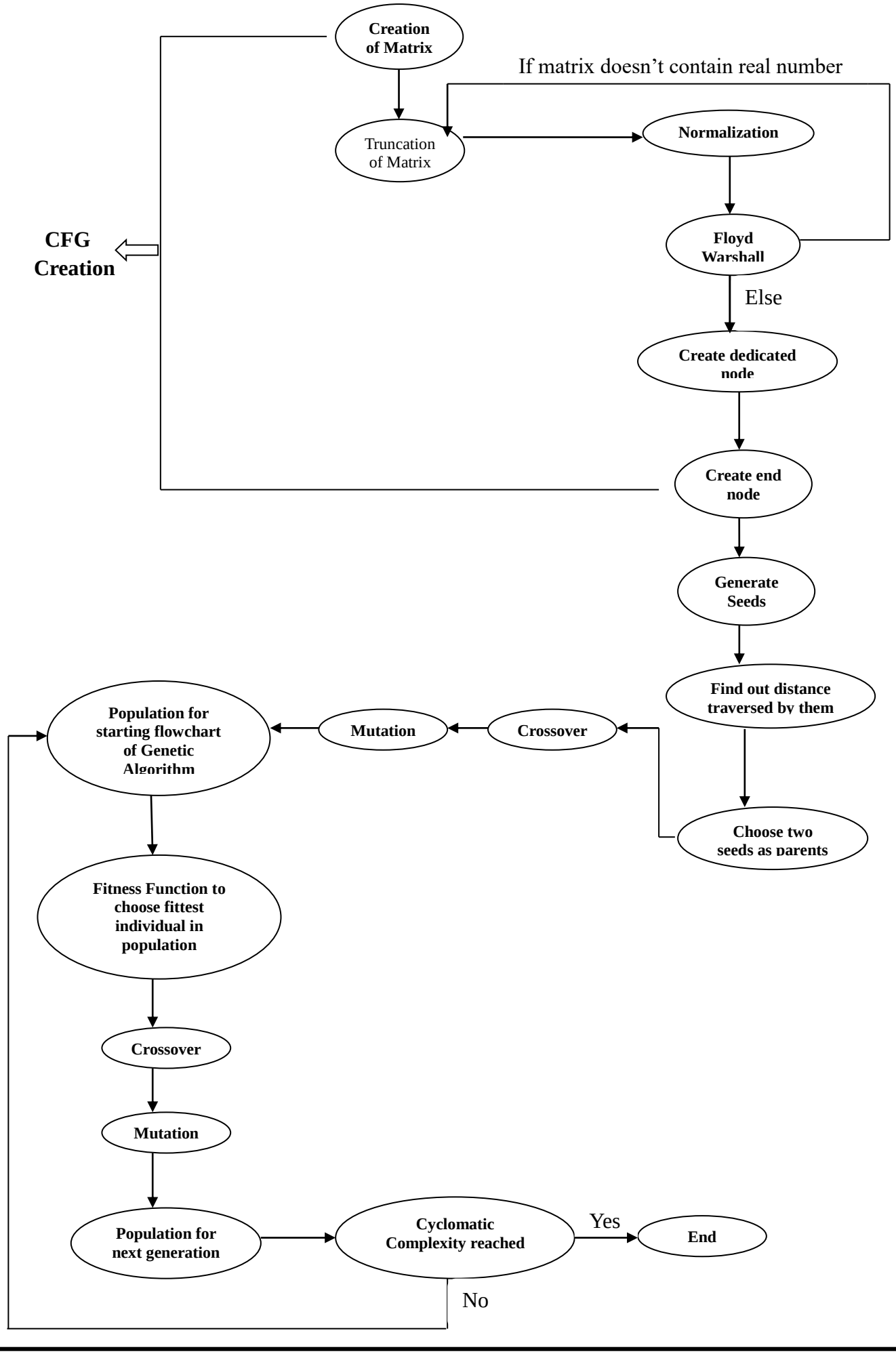
7.) From parents children genome are created by crossover and they are mutated for variation

8.) Now the population is iterated in genetic Algorithm cycle where every generation of population has to pass through process of selection ,crossover, mutation
until desired result is reached.

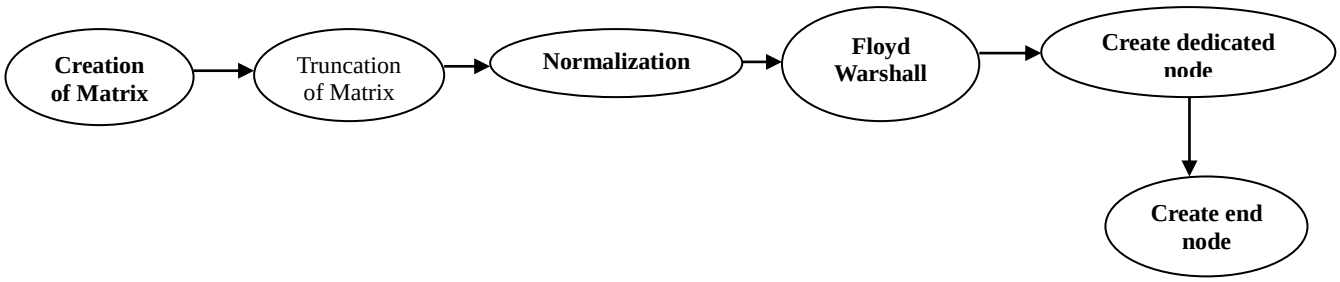
9.) We have calculated the cyclomatic complexity of CFG. And the generations of genetic algorithm is bounded by this cyclomatic complexity.

10.) As soon as generation number is more than cyclomatic complexity Genetic Algorithm ends.

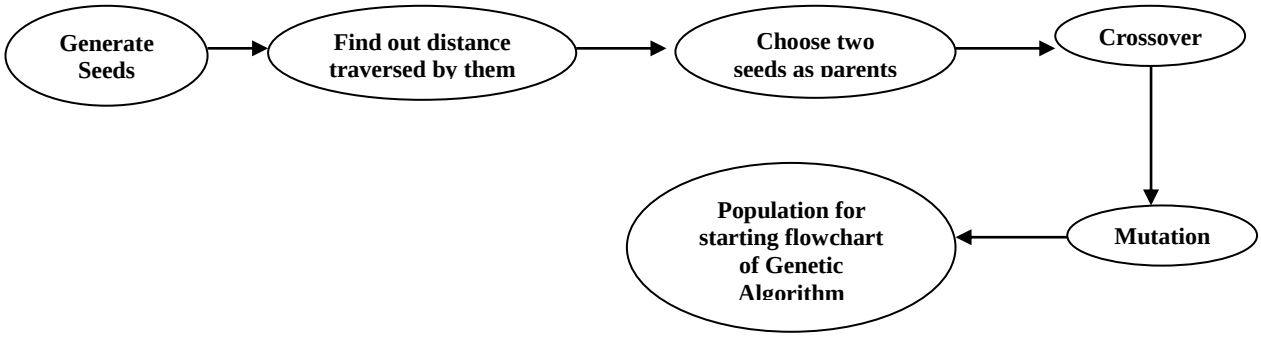
Flow Chart:


Creation of Matrix
If matrix doesn't contain real number
Truncation of Matrix
Normalization
Floyd Warshall
CFG Creation
Else
Create dedicated node
Create end node
Generate Seeds
Find out distance traversed by them
Choose two seeds as parents
Crossover
Mutation
Population for starting flowchart of Genetic Algorithm
Fitness Function to choose fittest individual in population
Crossover
Mutation
Population for next generation
Cyclomatic Complexity reached
Yes
End
No

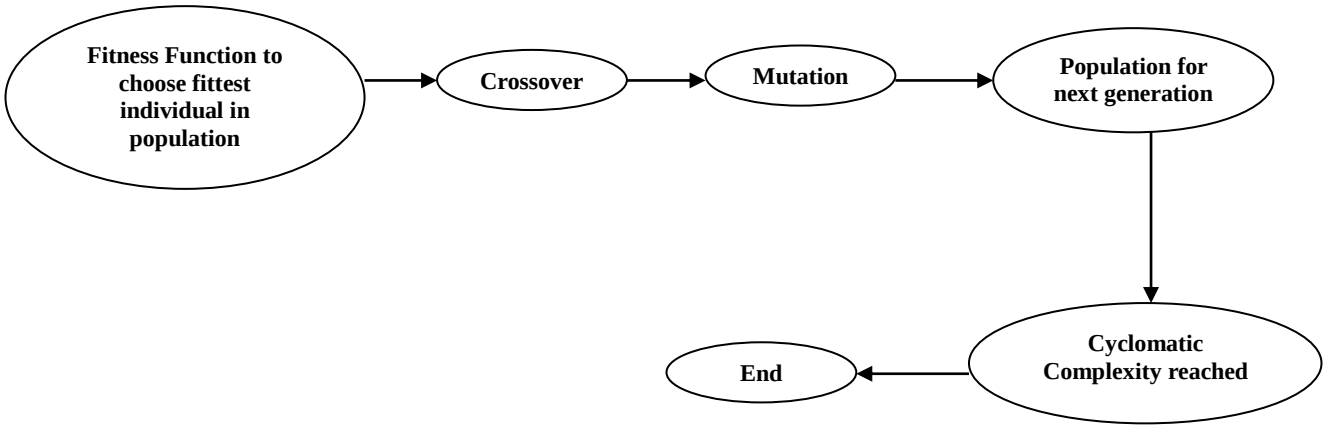
For CFG creation:



For initializing population for Genetic Algorithm:



For Genetic Algorithm:



8. Future Scope

The papers and formulations presented in this report serve as motivation for the perception of contributions in the formulations. Another aspect of the survey is the possibility, presented from the modelling of new problems in the area of Software Testing. As future work, we indicate the modelling and solving of problems in various areas of Software Engineering, particularly in the area of Software Testing. Specifically, the resolution of problems related to the activity of regression testing using heuristics that has not been used yet.
The future approach to this work could enhance the test case or test data generation for large programs automatically. The different parameters could be added which gives more optimized test cases and also increases the efficiency of heuristic techniques. Another perspective area could be the randomly generated test data by using various paths according to the control flow graph (CFG).Test Cases can be generated by using various kinds of hybrid heuristic algorithms like BCFA, FABA, BABC etc. since the test data generated by using Heuristic algorithm is compared with test data generated by PSOA and BCA and it was found that BCA produces optimal result in very less time and with more accuracy.

9. Bibliography

1) T. Dyba, “An empirical investigation of the key fa ctors for success in software process improvement”, IEEE Transactions on Software Engineering, IEEE, May 2005, pp. 410-424.

2) F. G. Freitas, C. L. B. Maia, D. P. Countinho, G. A. L. Campos, J. T. Souza, “Aplicação de Metaheurísticas em Problemas d a Engenharia de Software: Revisão de Literatura”, II Congresso Tecn ológico Infobrasil (Infobrasil 2009), 2009.

3) A. Auer and J. Korhonen. State testing of embed-ded software. In EuroStar-95London (UK), 1995.

4.) www.slideshare.net/KrishnasaiGudavalli/software-requirements-specification 17173967

5.) https://en.wikipedia.org/wiki/Program_optimization

6.) http://shodhganga.inflibnet.ac.in/bitstream/10603/36963/18/18_chapter%208.pdf

10. Conclusion

The application of heuristics to solve Software Engineering problems is part of a relatively new field called Search Based Software Engineering.

In this context, the phase of optimization of Software Testing showed an emphasis over other phases of systems development. Thus, the Search-based Software Testing subfield was created. The results in this area indicate the potential that this emerging field of research presents. In this sense, this way of viewing the problems of Software Engineering, likewise Software Testing, allows the resolution of problems that were unable to resolve satisfactorily before.

The works described in this report describe this project based on the problems in optimization of software testing using heuristic technique, mainly in test data generation, test cases selection and prioritization.

Heuristic technique is very tool for optimization of test cases or test data. It has been diversified the problems in a very effective manner for generating the test data automatically. This project report how the random test cases are generated and finding the optimal solution to maximize the problem. This project will inspire researchers to work on the various evolutionary algorithms by applying in computer science engineering area to generate the effective automated test cases.