International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 3, Issue 4, April 2014 ISSN 2319 - 4847

Application of Genetic Algorithm for generation of Artifacts and its usages as test data for unit testing

Ankita Dwivedi1, Navneet kumar Verma2 and Rahul Saxena3

[1] Research Scholoar ,Apaji Institute of Mathematics & Applied Computer technology, Banasthali Vidyapith, Jaipur, India

[2&3]School of Computer & Systems Sciences, Jaipur National University, Jaipur, India

ABSTRACT

This paper present a method for optimizing software testing efficiency by identifying most critical path based on the artifacts, in a program. For identifying this path, we develop a variable length Genetic Algorithm that optimize and select the critical path that assigned weight according to criticality of the path. As we know that exhaustive testing is not possible, only some part that is error prone can be selected for the testing. Therefore, we develop an approach that is more focusing on those parts that are most critical i.e. error prone so they can be tested first. Identify the most critical path; the testing efficiency should be increased.

Keyword- Genetic Algorithm, Artifacts, Software Testing

1. INRODUCTION

Software testing plays a crucial role in software quality assurance. It is a primary technique that used to gain consumers' confidence in the software industry. Software testing is a time and cost consuming task. 50% of the development costs are allocated to the software testing [1]. The objective of software testing is to design minimum number of test cases so that they identify as many faults as possible [1][2]. There are number of techniques used for the generation of test data and these can be categorized as structural and functional testing [2][4][7]. We present the result as using the genetic algorithm identify the most critical path i.e. error prone based on the artifacts during testing in a software construct [7, 8].

2. GENETIC ALGORITHM

Genetic Algorithm is computational method that solves the problems using biological evolution. Genetic Algorithms have been used to generate test sets automatically by searching the domain of the software for suitable values to satisfy a predefined testing criteria [10] . Genetic Algorithm are used to solved the optimization problems and are also known as search algorithm to search best possible solution. A genetic algorithm starts with guess [3][6][9]. A genetic algorithm consists of five parts:

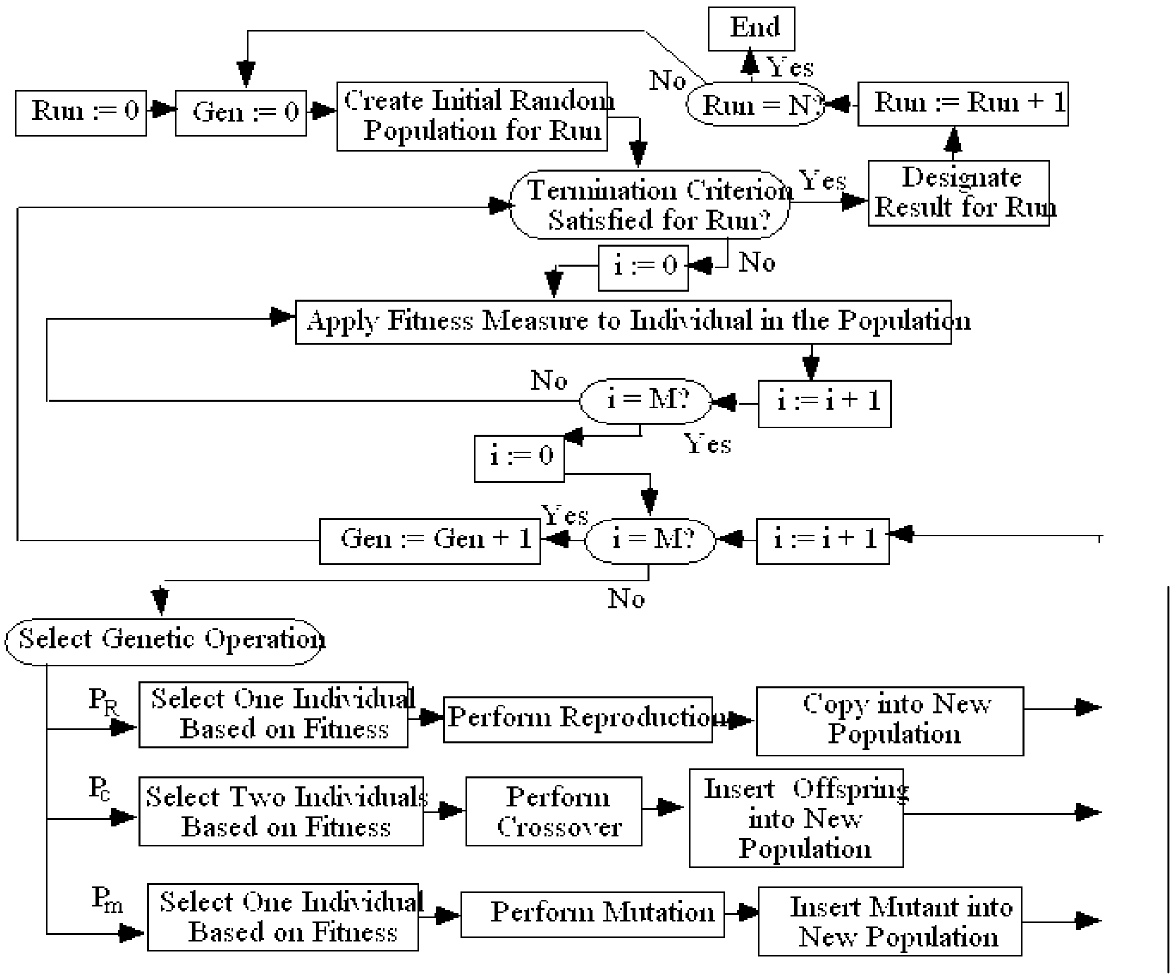
1. Chromosome(that shows the guess)
2. Initial pool of chromosomes
3. A fitness functions
4. A selection function
5. A crossover operator and mutation operator

- A chromosome can be a bit string and in genetic algorithm population typically take the form of bit strings.
- The set of chromosomes can be randomly produced as initial pool of data or may be manually created.
- The fitness function defined as the probability that the organism will live to reproduce. It means how much chromosomes can be fitter for the reproduction.
- The selection function selects the fitter chromosome that will participate in the evolution stage of genetic algorithm.
- The locus can be randomly chosen using crossover operator and exchange genes from two chromosomes in order to create two new chromosomes (called offspring).
- To create one new chromosome, the he mutation operator randomly flips some of the bits in a chromosome.

A Simple Genetic Algorithm (SGA) is as follows[6]:
The pseudo code for genetic algorithm is:

```
 Initialize (population)
 Evaluate (population)
While (stopping condition not satisfied) do
{
    Selection (population)
    Crossover (population)
    Mutate (population)
    Evaluate (population)
}
```

Figure 1: Flow chart for Genetic Algorithm


End
Yes
No
Run := 0
Gen := 0
Create Initial Random Population for Run
Run = N?
Run := Run + 1
Termination Criterion Satisfied for Run?
Yes
Designate Result for Run
i := 0
No
Apply Fitness Measure to Individual in the Population
No
i = M?
i := i + 1
Yes
i := 0
Yes
Gen := Gen + 1
i = M?
i := i + 1
No
Select Genetic Operation
P_R
Select One Individual Based on Fitness
Perform Reproduction
Copy into New Population
P_c
Select Two Individuals Based on Fitness
Perform Crossover
Insert Offspring into New Population
P_m
Select One Individual Based on Fitness
Perform Mutation
Insert Mutant into New Population

3. PROPOSED APPROACH

In this paper our proposed approach is to generate the test data using genetic algorithm based on artifacts and define the fitness function. We identify the critical path , so for the path testing we use control flow graph (CFG)[1]. The flow graph depicts logical control flow using some notations. A flow chart is used to depict program control structure[1, 2]. CFG[1][2] is a directed graph in which nodes are either statements or fragment of statements and edges represent flow of control.

3.1 PROCEDURE

According to the genetic algorithm we assign some weights to the control flow graph. These weights are randomly generated between 0 and 1.

3.2 SELECTION

The selection of chromosomes to become parents for the reproduction is done accordingly to a probability based on the fitness values. First we calculate the fitness value of the chromosomes using fitness function that is used in algorithm.

Fitness function value of the string is known as string's fitness[5, 6]. Fitness function can be used as that the weights that are used in the path, calculates the fitness of Chromosomes.
Let we take n chromosomes and in the control flow graph we have i edges so for calculating the fitness function we use notation as

$$F = \sum_{i=1}^{n} w_i$$

Where w_i=weight assigned to ith edge on the path.
After the calculation of fitness function we have to select the best fitted chromosomes to find more critical path based on artifacts.
For Selection there is a selection probability also used that is Pj for each path j that calculated as

$$Pj = Fj / \Sigma. Fj$$

Where, j=1 to n
n= initial population size
Again we also have to calculate the comulative probability Ck for each path k,to compare it with the randomly generated weights so that which can be selected for the crossover

$$Ck = \sum_{j=1}^{k} pj$$

3.3 CROSSOVER
A locus is randomly chosen using the crossover operator and exchange the subsequences before and after that locus, between the two chromosomes in order to create two new offsprings.
Crossover can be done according to the crossover probability Pc which is an adjustable parameter.
In the single/one point crossover technique, crossover done at the midpoint of input bit string.In this half of bit of one parent are swapped with corresponding half of the bit of other parent.
C1=11001110 (1100:1110)
C2=11000111 (1100:0111) (: shows locus)
C(1,2)=1100011 C(2,1)=11001110
For each parent selected, generate a random real number r in the range (0, 1); if r<Pc then select the parent for crossover. Once the crossover is done the selected data are then formatted randomly.
3.4 MUTATION
Mutation means the genes of chromosomes are modified. Every bit of chromosomes in the population has an equal chance to mutate (changes from 0 to 1 and 1 to 0). Mutation is also done according to the mutation probability Pm, which is also an adjustable parameter. For doing mutation generate a random number r in the range (0,1) and if r<Pm then mutate the bit.

4. EXAMPLE

```
1.  Power (int a, int b){            1
2.  p <- 1;          2
3.  z <- 1.0;          3
4.  p = b;          4
5.  if (b>0)          5
        a. while(p != 0){     6
        b. z = z * a;        7
        c. p = p - 1;        8
        d. }
6.  else                     9
        a. while (p != 0){       10
        b. z = z * a;        11
        c. p = p + 1;           12
        d. }
7.  z = 1 / z;          13
8.  return z;          14
9.  }
```

International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 3, Issue 4, April 2014 ISSN 2319 - 4847

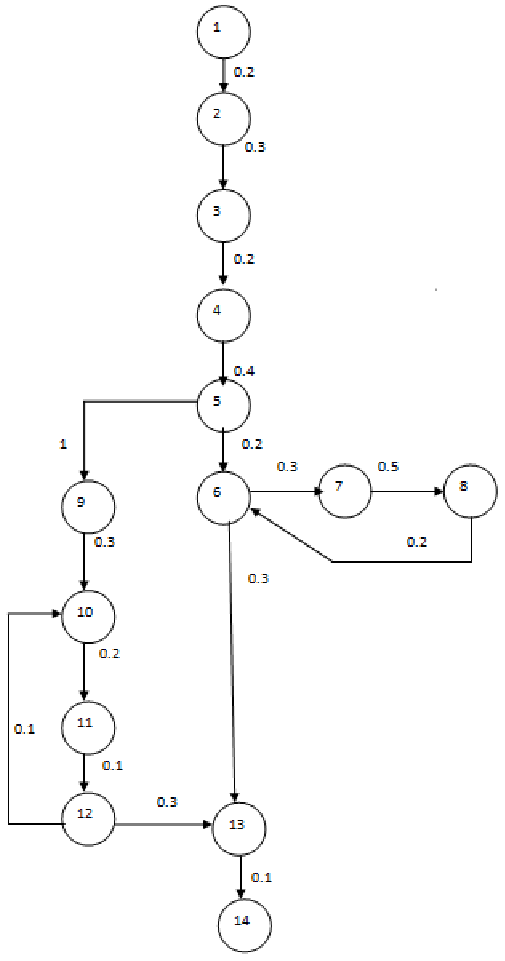


Figure 2: The Control Flow Graph for the above algorithm

Line1 to 4 are simple code but node 5 become the predicate node because it is a “if” statement. There are two possible outcomes of the “if” statement. Node 6 is “while” statement so it also has two outgoing edges in the control flow graph.

4.2 ASSIGN THE WEIGHTS

In control flow graph the initial credit for each node can be given randomly.

4.3 SOLUTION USING GENETIC ALGORITHM

One table used for solving the problem. The entries of the table are as followes:

- X: Test data set
- F(x): Fitness value calculated for the each test data by adding the weights of the path.
- P_i: Probability for corresponding data

  $P_i=F(x_i)/(\Sigma F(x_i))$
- C_i: Cumulative Probability
- Ns: The generated test data number that has cumulative probability and is just greater than the random number.
- Ran: Random number generated for test data set.
- Mating Pool: contains the number of times the test data appears in the Ns column.

Example: **Initial Population: (a, b)**

(2, 3) (2, 1) (3, 4) (7, 5) Tables are shown for the calculation. First of all Fitness value can be calculated for the all initial populations. Then in the 3 and 4 column probability Pi and cumulative probability Ci calculated respectively. Then Random number generated to simulate the genetic algorithm process.

Table 1: Iteration1 Calculating fitness value and other entries for initial input

S. N.	X	F(x)	P_i	C_i	Ran	Ns	Mating Pool
1.	(2, 3)	4.8	0.233	0.233	1.0	4	1
2.	(2, 1)	2.7	0.131	0.364	0.729	3	0
3.	(3, 4)	5.5	0.262	0.626	0.297	1	2
4.	(7, 5)	7.7	0.373	0.999	0.668	3	1

Table 2: Crossover and Mutation for above Ns entries

S. N.	Ns	Mating Pool	Crossover	Mutation
1.	4	(7, 5) 01110101	(7, 5) 01110101	(7, 7) 01110111
2.	3	(3, 4) 00110100	(3,4) 00110100	(3,5) 00110101
3.	1	(2,3) 00100011	(2,4) 00100100	(2,4) 00100100
4.	3	(3,4) 00110100	(3,3) 00110011	(3,7) 00110111

Table 3: Iteration2 calculate new fitness value and entries for last mutated chromosomes

S. N.	X	F(x)	Pi	C_i	Ran	Ns	Mating Pool
1.	(7, 7)	8.7	0.292	0.292	0.629	3	0
2.	(3, 5)	6.5	0.218	0.51	0.65	3	1
3.	(2, 4)	5.8	0.195	0.705	1.000	4	2
4.	(3, 7)	8.7	0.292	0.997	0.499	2	1

Table 4: Crossover and Mutation for above Ns entries

S. N.	Ns	Mating Pool	Crossover	Mutation
1.	3	(2,4) 00100100	(2,4) 00100100	(6,5) 01100101
2.	3	(2,4) 00100100	(2,5) 00100101	(3,5) 00110101
3.	4	(3,7) 00110111	(3,7) 00110111	(3,7) 00110111
4.	2	(3,5) 00110101	(3,4) 00110100	(3,4) 00110100

Table 5: Result for critical path finding

S. N.	X	F(x)	P_i	C_i	Ran	Ns	Mating Pool
1.	(6, 5)	6.5	0.236	0.236	0.600	3	2
2.	(3, 5)	6.5	0.236	0.472	0.347	1	0
3.	(3, 7)	8.7	0.316	0.788	0.216	1	2
4.	(3, 4)	5.8	0.210	0.998	0.748	3	0

5. CONCLUTION

The Genetic Algorithm is a probabilistic search algorithm that iteratively transforms a set (population) of mathematical objects (fixed length binary string), each with an associated fitness value, that maps onto a new population of offspring objects using crossover and mutation techniques. As the genetic algorithms are used for generation of solution to optimization problems and hence in this paper genetic algorithm are used for finding the most critical paths based on artifacts during testing in a software construct and can improve the software testing efficiency.

REFERENCES

[1] Aditya P mathur,"Foundation of Software Testing", 1st edition Pearson Education 2008.
[2] S. Sidiroglou and A.D. Keromytis, "Countering Network Worms through Automatic Patch Generation," IEEE Security and Privacy, vol. 3, no. 6, pp. 41-49, Nov./Dec. 2005.
[3] E. Fast, C. Le Goues, S. Forrest, and W. Weimer, "Designing Better Fitness Functions for Automated Program Repair," Proc. Genetic and Evolutionary Computing Conf., 2010.
[4] Rajat Kumar Bal,Software Testing.
[5] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, 1992.
[6] Goldberg, D.E, "Genetic Algorithms: in Search, Optimization & Machine Learning," Addison Wesley, MA.1989.
[7] Mark Last, Shay Eyal1, and Abraham Kandel, "Effective Black-Box Testing with Genetic Algorithms," IBM conference
[8] Dr. Velur Rajappa, "Efficient Software Test Case Generation Using Genetic Algorithm Based Graph Theory," ICETET '08, pp.298-303, 2008.
[9] S. Forrest, "Genetic Algorithms: Principles of Natural Selection Applied to Computation," Science, vol. 261, pp. 872-878, Aug. 1993.
[10] B.F. Jones, H.-H. Sthamer and D.E. Eyres. Automatic structural testing using genetic algorithms. Software Engineering Journal, pages 299-306, September, 1996.