RESEARCHERID
THOMSON REUTERS

[Rjmann* *et al.*, 6(7): July, 2017]
IC™ Value: 3.00

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

APPLICATION OF CLONAL SELECTION ALGORITHM IN SOFTWARE TEST DATA GENERATION

Wasiur Rhmann[*1], Gufran Ansari[2]

[*] Department of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow India
Department of Information Technology, College of Computer, Qassim University, Al-Qassim, Saudi Arbia

DOI: 10.5281/zenodo.829221

ABSTRACT

This paper presents an artificial immune system (AIS) based optimized test data generation technique. Software test data generation is a critical activity in software testing. Redundant test data impose an extra cost in form of more testing and increase in efforts. So the generation of effective test data can save time and efforts spent on extra testing of the software. Optimized test data is generated for critical paths of the program. Paths criticality is defined based on the presence of loops and decision. The clonal selection algorithm is an immune system based machine learning algorithm and generates optimized test data by the maturation of the affinity. Generated results indicate the effective application of clonal selection technique in software test data generation.

KEYWORDS: Test data, Test paths, Control Flow Graph (CFG), Clonal Selection Algorithm.

INTRODUCTION

Software testing is a quality assurance activity in the software development and improves the reliability of the software. Software testing is primarily main verification and validation activity of software. Testing is a creative and labor intensive activity which consumes more than 50% of the total software development cost [1]. Testing involves the generation of test data and effectiveness of testing depends on the quality of test data. Test data generation activity is heavily affected by the redundant test data which is responsible for extra efforts. The intent of software testing is to design a set of test data which can reveal a large number of possible faults. Structural test techniques consider the program code under test and are based on the Control Flow Graph (CFG) of the code. CFG is a representation of the program under test by a directed graph where each node represent the statement of code and arrow connecting the nodes represent the flow of control in the code. How much testing is required for reliable code, is guided by the coverage criteria. Some of these criteria are statement coverage, branch coverage, and path coverage. Path coverage criteria are strongest among all criteria available in the literature. Recently automated test data generation has gained the attention of researchers. Artificial intelligence techniques are used in software engineering for the advancement of software engineering. In software testing, each test data is used to traverse a test path of control flow graph. Testing of all paths is not possible due to the presence of loops and presence of unfeasible paths poses the challenge for test data generation. So testing all paths is computationally infeasible and NP-complete problem. Different types of test data generation techniques are present in literature for generation of automated test data. The principles of Artificial Immune System (AIS) lead to the development of some well-known biological inspired computing algorithms like: clonal selection algorithm, negative selection algorithm, and Immune network algorithm. The clonal selection algorithm is fundamental to immunology. The clonal selection algorithm is applied in different fields like vehicle scheduling [2], construction site planning utilization [3] and in traveling salesman problem [4]. Different researchers used different soft computing technique for software test data generation. Jiang et al. [5] used reduced adaptive particle swarm optimization algorithm for automatically test data generation. They reduced the particle swarm equation without velocity. Algorithm's search capability in local and global search is balanced by adaptive adjustment scheme based on inertia weights. Yao et al. [6] proposed a multi-objective test data generation technique with the constraint of statement coverage. Authors solved the proposed model with the genetic

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

algorithm. Test suite generated from presented technique satisfied the better spatial distribution of test data and satisfied the statement coverage. Authors used a genetic algorithm for solving the proposed model of test data generation. Zhang and Gong [7] formulated test data generation problem as multiple objective problem and authors used a weighted genetic algorithm to solve the presented model. Yao and Gong [8] suggested a model for test data generation for multiple paths coverage. Authors solved proposed a model with the multi-population genetic algorithm using individual sharing. The performance of proposed approach is measured with experiment and theoretically. In the present work, Weighted CFG is used for test data generation weights are assigned to components based on the criticality of the component. Priority is given to loops, decision nodes. Proposed algorithm works on Control Flow Graph (CFG). CFG represents the flow of control within the program. Independent path of CFG introduces at least one new node of the statement in the path. Section 2 of this article explains the clonal selection algorithm, In section 3 proposed methodology is presented, in section 4 proposed algorithm is applied for a program and finally, section 5 concludes the presented work.

CLONAL SELECTION ALGORITHM

Immune system inspired algorithms are recently gained the attention of researchers and are applied in various optimization problems. These types of algorithms are inspired by the biological immune system and based on the Darwinian theory of survival of fittest. The immune system protects the host body from infection by outside invaders. The adaptive immune system protects the body from outsider antigens. Antigen triggers the production of antibodies by the immune system to fight with outsider invaders. Artificial immune system algorithms are inspired by the functionality of the human immune system. The clonal selection algorithm belongs to the artificial immune system. The clonal selection algorithm is a machine learning approach. The clonal selection algorithm depends on acquired immunity of clonal selection which helps to understand how B and T lymphocytes improve their response to antigens. Antibodies with higher affinity are selected to proliferate and selected antibodies are strengthened with hypermutation process. Hypermutation process brings diversity in the clones. Hypermutation process can be performed in a number of a way but affinity inversion is the main operator used for introducing the diversity in the population [9]. Then receptor editing is used to eliminate antibodies with worse affinities and new antibodies are introduced. Clonal selection algorithm resembles the Genetic algorithm without recombination. Clonal selection algorithm proposed in [10] used mutation for effective local search. In the clonal selection, cells which are capable of recognizing the antigens will proliferate [11].

The clonal selection algorithm is described below in brief:

Step 1. Randomly generate j antibodies;

Step 2. Repeat the process till the predefined goal is achieved

a. Affinity of each antibody is determined with objective function;

b. Select n antibodies and clone the antibodies proportionally to their affinities;

c. Hypermutation is performed on the cloned antibodies. Mutation rate is inversely proportional to the affinity of antibodies [12];

d. Select j highest affinity antibodies from set of clones and antibodies;

e. Replace the d lowest affinity antibodies by new antibodies generated randomly;

Step 3. End repeat.

Fig. 1 represents the clonal selection algorithm used in software test data generation

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

Start Initial random population generation.

1. **While** (Specified level of Affinity of test data achieved)

{

2. **Evaluate** the affinity of each generated antibody

3. **Select** half of the antibodies with affinity higher than other half;

4. **Hypermutate** each clone;

5. **Generate** new test data randomly to maintain the initial test data number;

} **end while**

} **end**

Figure 1. Clonal Selection Algorithm for Software Test Data Generation

PROPOSED METHODOLOGY

Based on the concepts of clonal selection in software testing the methodology presented here deals with effective test data generation. Figure 2 represents the activity diagram for test data generation.

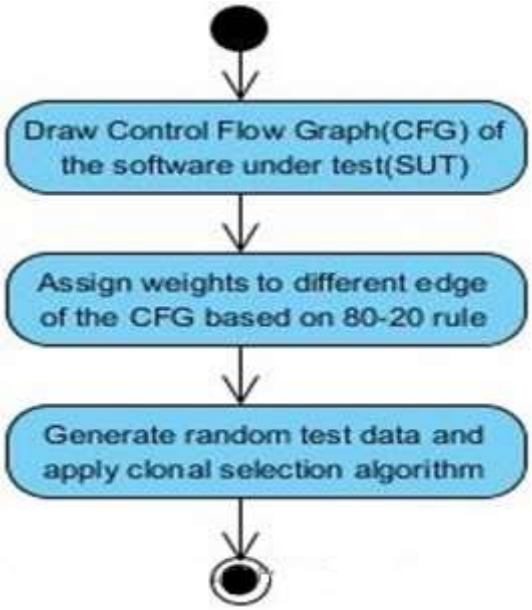

Figure 2. Activity Diagram for Test Data Generation

The steps of proposed test data generation are briefly described as follows:

Step1. The program under test is converted into Control Flow Graph (CFG), and different weights are assigned to edges of the control flow graph. 80-20 rule is followed in the assignment of weights to different edges of the control flow graph.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

The initial weight of an edge is assigned as 10 which is divided on the occurrence of branches in the ratio of 80:20. For denser control flow graph more initial weights to the edges may be assigned. Initial weights of incoming edges are divided in such a way that 80% weight is given to the loops and branches while only 20% of sequential statements. The weight of incoming edges is equal to the weights of outgoing edges. Figure 3 represents the activity diagram of the application of clonal selection algorithm in test data generation.

Step 2. Initial test data is generated randomly. The affinity of each test data is calculated and affinity of test data is the total weight of the weight of edges of the paths which will be traversed by test data. Half of the test data are selected which have higher affinity values compared to another half. Selected test data is cloned using the formula given below:

Affinity (d_i) =Sum of weights of edges occurred in path of traversal of data d_i (1)

CP=Affinity (di)/ ΣAffinity(di) (2)

Number of clone s=CP*n (3)

Where CP is clonal probability and n is a number of initial data items. Affinity(di) is the affinity of data $d_{i.,}$

After cloning, mutation of cloned data is performed by performing mutation. Mutation is performed based on mutation probability which is the inverse of the affinity value of that data.

Mutation Probability (MP) =1/Affinity (4)

For performing mutation, a random number is generated between (0, 1) for each bit of the test data and if generated the random number is greater than the mutation probability then that bit of data is flipped.

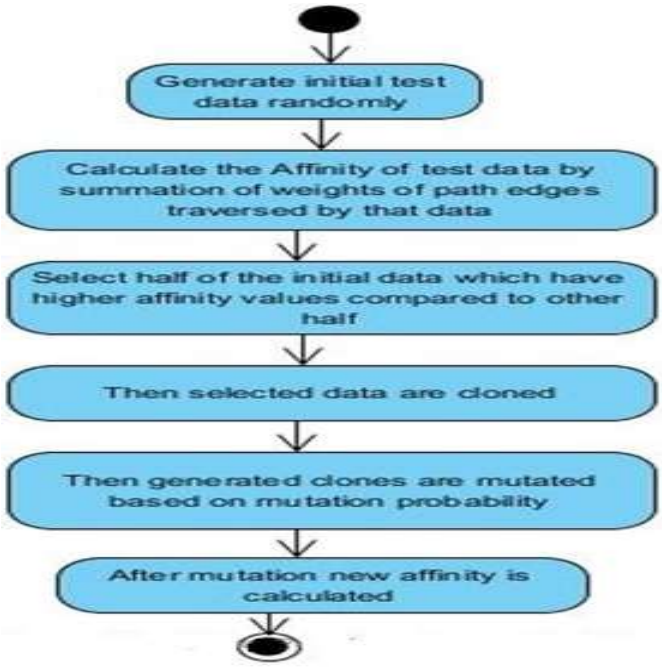

Figure 3. Activity Diagram for Clonal Selection Application in Test Data Generation

In each iteration, new test data is randomly generated to maintain the initial number of test data in population.

EXPLANATION OF PROPOSED ALGORITHM WITH EXAMPLE

For the application of the presented clonal selection algorithm authors have considered a well-known example of Greatest common divisor [13] given in Figure 4. There are 13 statements from 0 to 12 and there is a node corresponding to each statement of the program. Node 1 is predicate node so there are 2 branches coming out from that node. In

```
0 gcd(int m, int n)
{   int r;
1. if(n>m){
2.r=m;
3. m=n;
4.n=r;
}
5. r=m%n
6. while(r!=0)
{
7.m=n;
8.n=r;
9.r=m%n;
10. }
11. return n;
12. }
```

Figure 4. Greatest Common Divisor

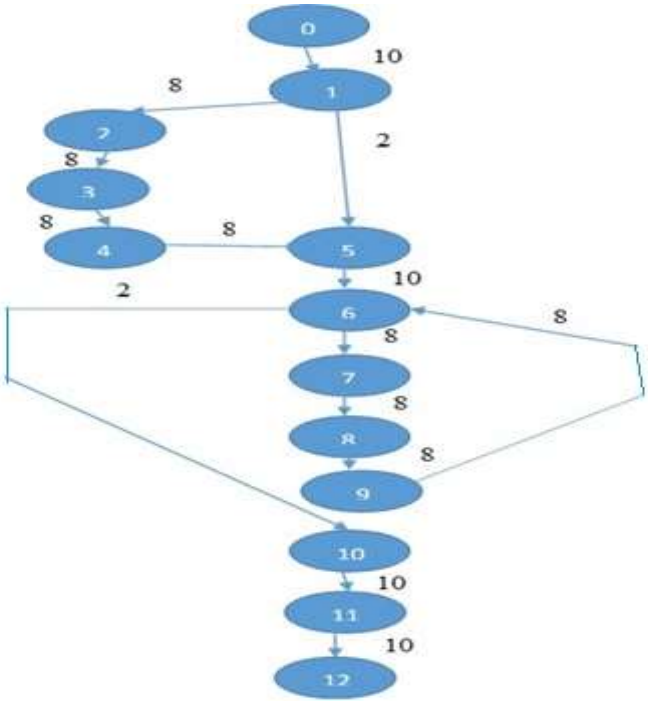

Figure 5. Weighted Control Flow Graph (WCFG)

Figure 5 control flow graph is created for the program taken as case study and weights are assigned to edges of the control flow graph. Initial weight to the edge is given as 10 which is divided at the edge in 80 to 20 ratio.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

Table 1 shows the initial test data and clones after mutation. Test data are generated in a random manner. Their affinity is calculated as follows:

(12, 8) will travel the following path:

0-1-5-6-7-8-9-6-10-11-12

The weight of edges: 10+2+10+8+8+8+8+2+10+10=76, similarly the other values of affinities are calculated and recorded in Table 1.

Table 1 and Table 2 shows the first iteration and Table 3 and table 4 shows the second iteration and Table 5 presents the final affinity values of the population. (2, 3) and (15, 4) are selected as they have higher affinity than other half data items. These selected data items are the cloned and the number of clone in determined by equation 2, then mutation probability is calculated by equation 3. Mutation is performed by generating a random number (ran) between (0, 1) and it is compared with the mutation probability if (ran<MP) then corresponding bit is flipped from 0 to 1 or 1 to 0.

Table 1. Initial Test Data and Test Data for Cloning

X	Affinity	Selected items	Number of clones=CP*n	Clones	Mutation Probability (MP)=1/Affinity	Item for mutation
(12, 8)	76		106*4/334=1.26(1)	(2,3)	0.0094	(2,3)
(2, 3)	106	(2, 3)				
(6, 2)	44		108*4/334=1.29(1)	(15,4)	0.0092	(15,4)
(15, 4)	108	(15, 4)	106*4/334=1.26(1)	(2,3)	0.0094	(2,3)

Table 2. Item After Mutation

Item for mutation	Mutation Probability	Item After mutation
(2, 3) (10,11)	0.0094	(3, 3)
(15, 4) (1111, 0100)	0.0092	(14, 4)

In the second iteration of algorithm, authors generated two new random data items (4, 5) and (15, 7) to maintain numbers of data items and similarly other steps are performed as in the first step.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

Table 4. Second Generation of Population

X	Affinity	Selected items	Number of clones=CP*n	Clones	Mutation Probability(MP)=1/Affinity	Item for mutation
(3, 3)	44					
(14, 4)	76	(14, 4)	76*4/334=0.91(1)	(14,4)	0.0131	(14, 4)
(4,5)	64	(4,5)	64*4/334=0.76(1)	(4,5)	0.0156	(4,5)
(15, 7)	34					

Table 4. Second Generation Population after Mutation

Item for Mutation	MP	Item for Mutation
(14, 4) (1110,100)	0.0131	(14, 3)
(4, 5) (100, 101)	0.0156	(6, 4)

Table 5. Affinity Values of Population Generated in Second Generation

X	Affinity
(15, 8)	108
(14, 4)	76
(14,8)	104
(4,5)	64

After the second generation, we get the test data (15, 8), (14, 4), (14, 8) and (4, 5).

CONCLUSION AND FUTURE WORK

The presented work showed the application of immune system inspired clonal selection algorithm in software test data generation. The possibility of a huge number of test data which may be generated from the infinite number of possible test paths may lead to a very large amount of work to test the software. Critical paths of control flow graph are identified earlier by assigning weights to the edges. Identification of critical paths earlier

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

with clonal selection algorithm may find bugs earlier in the software testing. Proposed approach can be easily automated. In current work, authors have considered a small example in future research industrial benchmark may be considered.

REFERENCES

[1] Boris Beizer, "Software Testing Techniques", Second Edition, Dreamtech Press, (2011).
[2] Xinguo Shuia, Xingquan Zuoa, Cheng Chena, Alice E. Smith, "A clonal selection algorithm for urban bus vehicle scheduling", Applied Softcomputing, vol. 36, (2015), pp. 36-44.
[3] Xi Wang, Abhijeet S. Deshpande, Gabriel B. Dadi, and Baris Salma, "Application of Clonal Selection Algorithm in Construction Site Utilization Planning Optimization", Procedia Engineering, vol. 145, (2016), pp. 267-273.
[4] Zhu Y, Gao S, Dai H, Li F, Tang Z, Improved clonal algorithm and its application to traveling salesman problem. Int J ComputSci Network Secur, vol. 7, no. 8, (2007), pp. 109–113.
[5] Shujuan Jiang, Jiao, jiao Shin, Yanmei Zhang, Han Han "Automatic test data generation based on reduced adaptive particle swarm optimization algorithm", Neurocomputing, vol. 158, (2015), pp. 109-116.
[6] Xiangjuan Yao, Dunwei Gong, Gongjie Zhang, "Constrained multi-objective test data generation based on set evolution", IET Software, vol. 9, no. 4, (2015), pp. 103-108.
[7] Yan Zhang and Dunwei Gong, "Generating test data for both coverage and faults detection using genetic algorithms: multi-path case", Front. Comput. Sci., vol. 8, no. 5, (2014), pp. 726-740.
[8] Xiangjuan Yao and Dunwei Gong, "Genetic Algorithm-Based Test Data Generation for Multiple Paths via Individual Sharing", Computational Intelligence and Neuroscience, 2014, Article ID 591294, pp. 1-12.
[9] T. Jansen and C. Zarges, "Analyzing different variants of immune inspired somatic contiguous hypermutations", Theoretical Computer Science, vol. 412, no. 6, (2011), pp. 517–533.
[10] 10. L. N. De Castro, F. J. Von Zuben, "Learning and Optimization using Clonal selection algorithm", vol. 6, no. 3, (2002), pp. 239-251.
[11] 11. F. M. Burnet, "The clonal selection theory of acquired immunity", Cambridge University Press, (1959).
[12] Z. X. Ong, J. C. Tay and C. K. Kwoh, Applying the Clonal Selection Principle to Find Flexible Job-Shop Schedules, Artificial Immune Systems, Lecture Notes in Computer Science, vol. 3627, (2005), pp. 442-455
[13] Praveen Ranjan Srivastav and Tai-hoon Kim, "Application of Genetic Algorithm in Software Testing", International Journal of Software Engineering and its Applications, vol. 3, no. 4, (2009), pp. 87-96.

CITE AN ARTICLE

Rhmann, Wasiur , and Gufran Ansari. "APPLICATION OF CLONAL SELECTION ALGORITHM IN SOFTWARE TEST DATA GENERATION." *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY* 6.7 (2017): 452-59. Web. 15 July 2017.