International Journal of Computational Intelligence Research
ISSN 0973-1873 Volume 13, Number 6 (2017), pp. 1521-1531
© Research India Publications
http://www.ripublication.com

A Literature Survey on Automation of test data generation for Branch coverage testing using genetic Algorithm

Shivani Sharma
Uttarakhand Technical University, Dehradun, Uttarakhand, India.

Dr. Umesh Chandra
Glocal University, Saharanpur, Uttar Pradesh, India

Dr. Parag Jain
Roorkee Institute of Technology, Roorkee, Uttarakhand, India

Abstract

Test data generation plays an important role in the field of software testing. Earlier test data used to generate manually which was very time consuming and complex. Then random test data generation approach was used. However, it was not much successful and not able to generate appropriate test cases. It seems unlikely that a random test data generation can also do well for some programs but requires extra amount of efforts. Results from Random test generation are generally poor for realistic programs. In general, small programs are unable to show the weaknesses of random test data generation. Much work has been done in automating the testing with the help of heuristic approaches.In this paper, we have presented a survey on use of different evolutionary techniques especially Genetic Algorithm for optimizing the testing results with the help of automation of test data generation for white box testing such as statement coverage, branch coverage and path coverage testing.We found that though lot of work has been done in path testing. There is still a need of optimization in test data generation for branch coverage testing.

Keywords: Software Testing, Genetic Algorithm, Test Data generation, Test Case, Branch coverage, Path Testing.

1. INTRODUCTION

Software testing includes some of the main issues like how good a test input tests a program or lines of codes. The main objective of testing is to find maximum faults or failures possible with the help of the test cases, and obviously, those tests which can find many faults or failures will be better than those which can get a few. In general it's almost impossible to guess or predict that how many faults or failures would be uncovered by a particular set of tests. It is because of the fault is regularly defined that is why we need some standards for test adequacy to decide when a program has been tested enough. And that is why we have to establish a test adequacy criterion (Christoph C.,2001)

Once we defined the test adequacy criterion, then the focus is how we can create test cases which are well enough with the test adequacy criterion and we can think how much difficult it will be by hand that is why we need to consider automated test data generation. Test adequacy criterion needs the program to do some specific tasks to reach to a statement. Generally test data generation needs some heuristics and it does not always guarantee to get succeed to get appropriate test cases [1]. Sometimes it seems very unlikely that a random approach can also do well for real test generation problems which requires extra amount of efforts that makes it costly.

If we really want to satisfy many test requirements, then we have to consider many function minimization but sometimes functions which are being minimized are quite similar that's why we are able to solve some other problem while we are solving one particular problem. In other words,it's not necessary that whatever tests we are generating are only for one problem, we could reuse results of one problem to solve another one (Christoph C et al.,2001).

The paper has been divided into 4 sections. Section 2 defines the basic of Genetic Algorithm. In section 3 we have presented the detailed survey on automation of test data generation using genetic algorithm and other evolutionary techniques for various types of testing

2. INRODUCTION TO GENETIC ALGORITHM

1. Genetic Algorithm:

Genetic algorithms follow Darwinian's principal, are very important to get the optimum solution. Whenever search space is too large we could get benefited from genetic algorithms, but the main question is why only genetic algorithms because in genetic algorithms we generate number of generations and then pick out the optimum or best result. Genetic algorithms guarantee for many systems to get good results, if we should apply genetic algorithms on big or complex systems. Initially population is generated randomly then we will have to apply genetic operators to get the optimum solution.

2. Individuals:

An individual is a set of solution for the given problem which is being solved. An individual is represented by a string; we generally represent individuals by binary strings. We could also define or use some other data structures.

3. Initial population:

As we know, an individual is generally represented by a bit of string. Initial population is generally generated randomly and we have to decide the length of the bit string according to the problem which is being solved.

4. Algorithm:

Genetic Operators:

4.1. Selection:

Selection is the process of extracting individuals from the initial population after applying objective function on each and every individual

4.2. Crossover

The next step is to create offspring's form the two selected parents with the help of crossover operator. There is one simple e.g. of crossover is given below.

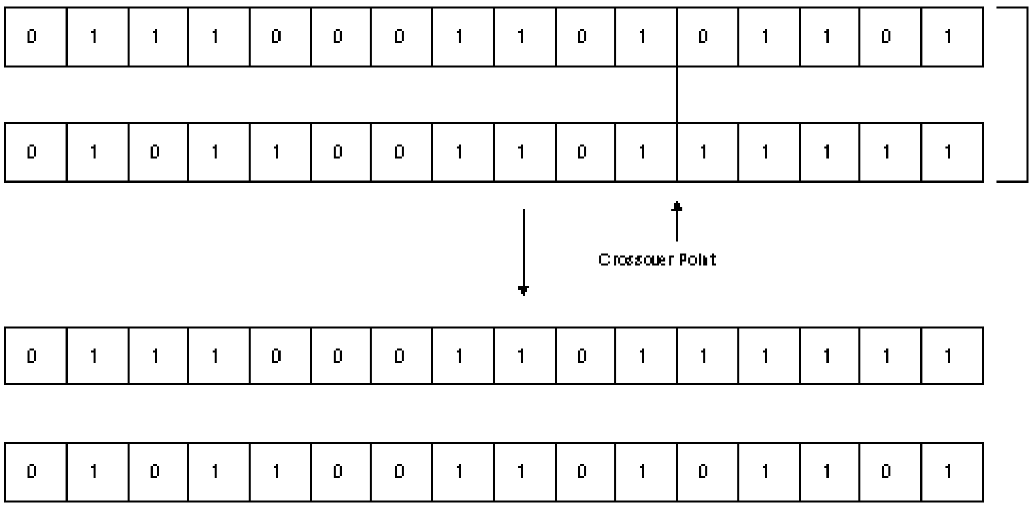

Fig.1. Cross Over Point

Figure 2 Mutation

In mutation we just simply interchange bits from 0 to 1 or 1 to 0. It is clearly shown in Figure 2

3. LITERATURE REVIEW

Xanthakis et al. in (Xanthakis et al.,1992) is the first one for generating test data by applying genetic algorithm. For this implementation test- data was generated for structures which are not covered in the random search with the help of genetic algorithm. User chooses a path where all the branch statements are extracted from the source code.Genetic Algorithm was used for finding the data to be input for all the branch predicates in one go,where the fitness function is calculated by adding the distance values of branch.

Pei et al. (Pei et al.,1994) in 1994 proposed a test data generator for single-path coverage by applying the technique of genetic algorithm. In his survey he found that maximum of the test- data generators used symbolic evaluation which were implemented during that time. They came to a conclusion that dynamic testing was not effective while static was not practical. They found that both the gradient-descent-based static and dynamic testing were developed.

Roper et al. (Roper et al.,1995) developed a test data generator in 1995 based on genetic algorithm with the aim of traversing all the possible branches of a source code. A Program is given to the generator which is instrumented by it automatically and provides the feedback on the achieved branch coverage.

Alander, J. et al (Alander, J. et al,1996) in the year 1996 derived a test method which falls under the category of dynamic stress testing done automatically by surveying all the possibilities for software testing using Genetic algorithm. Its plan was to design the test cases for finding problematic situations.

Similar test data generator that was based on genetic algorithm was developed to achieve branch coverage by Jones et al. (Jones et al.,1996) in 1996.He used a sequence of binary strings for representing the individual that was converted to a decimal number before the program gets executed. With the help of Control flow graph one or more iterations for each loop was represented, due to which acyclic CFG is formed. With the execution of each branch automatic traversal of CFG takes place to the other branch using breadth first search.

Michael et al. (Michael et al.,1997) in 1997 developed GADGET (Genetic Algorithm Data Generation Tool)which generates test data, with the help of which a program can be instrumented automatically without the limitation of programming language. The only limitation with it is in this scalar inputs can only be accepted. Its adequacy criteria is condition–decision coverage. Both the simple and differential GA is used in

GADGET. GADGET is the first test data generator which is tested fora large real world problem, b737 (part of autopilot system i.e real-world control software). He concluded that the test data generator which generates uses random approach does not perform well for the large problems.

Tracey N. et al. (Tracey N. et al,1998) in 1998 built a test-case data generator which was based on the optimization technique. In this the number of test criteria can be included for both functional as well as non-functional properties. Optimization is applied to testing specification failures and exception conditions. In this output of different case studies are shown to prove the effectiveness and efficiency of this optimization technique for generating test case data.

Pargas et al. (Pargas et al.,1999) in 1999 improvised the outcome of work performed by Jones et al.'s.In earlier work the evaluation of fitness function used branch information, while here the control dependency graph was used to evaluate the fitness. According to which they said this approach gives better evaluation of fitness as compared to Jones and Michael approaches discussed above. He also claimed that this technique with minor changes can also provide path coverage.

Lin and Yeh (Lin and Yeh,2000) in the year 2000 again improvised the work performed by Jones. Instead of branch coverage, path coverage criterion is used. Ordinary (weighted) hamming distance was also extended so as to handle different sequence of paths have same branch nodes. Here the logic is simple hamming distance is not suitable when same branches can have two different paths but in different order. SIMILARITY is the name given for fitness function as it evaluates the items which are similar with respect to their sequencing within the two paths which are different. The greater SIMILARITY will result in better fitness.

In 2000 Bueno and Jino (Bueno and Jino,2000) proposed an approach for path coverage testing that uses dynamic information for control and data flow. This approach also deals in identifying the program paths which are infeasible bytesting the progress of the search for the required test data. To cover the feasible path this technique work for the continuous improvement of fitness function of the population.

Wegener et al. (Wegener et al.,2000) in 2002 build Genetic Algorithm based test data generator for real world embedded software whose focus was white box testing, specially statement and branch coverage. The two building blocks he used for fitness function are local distance of normalized predicate and approximation level. Overall Fitness function is evaluated by adding the local distance value and the approximation level value, where the local distance value is evaluated for the individuals and the approximation level the number of matched branching nodes between the traversed branches and a target branch by an individual this is called as "partial aim". In this work normalization scheme for the local distance value is not described. This technique works only for one partial aim at a time. The test specially works on those partial aims which are hard to reach. Completion of all the branch coverage or reaching the number of generations whichever is first satisfied forms the stopping criteria.

Unfortunately this paper does not describe can we cover multiple targets in one attempt or not. In their report they claimed full coverage of few programs and not for all the programs. They are trying to find out the reason (which can be the number of generations or the infeasible branches/statements) why they are unable to cover all the possible branches in all the programs.

In 2003 Daz E., et al. (Daz E., et al.,2003) describes a technique for optimizing the testing that includes Tabu Search with the Korel's chaining approach. With the help of this approach automatic test-data generation was done for branch coverage testing.

Berndt D., et al. (Berndt D., et al.,2003) in 2003 includes a simple case study is used to illustrate the approach that design test cases for software testing using genetic algorithm. A new fitness function was evolved which is based on the organisms fossil record which results in different search behaviors defined on the concept of severity, novelty and proximity.

In 2003 Hermadi et al (Hermadi et al,2003) used the approach of genetic algorithm for test data generation for path testing. With the help of this approach for a set of target paths set of test data was generated. Better path coverage was achieved with this approach and the performance was improvised in terms of 1) Search space exploitation, 2) Exploration, 3) Allows fast convergence.

Tonella P., et al. (Tonella P., et al,2004) in 2004 developed a test case generator using genetic algorithm focusing on unit testing in a generic scenario.In this approach Chromosomes represent as test cases that include everything about the objects need to be created, the methods to be called and the values need to be used as input. This algorithm does the mutation with the objective of maximizing a given coverage measure. Paper describes how this algorithm is implemented and applied to the classes from java standard library.

D. J. Berndt et al. (D. J. Berndt et al.,2005) in 2005 defined a new strategy which is combination of technique for automated test suite generation with large volume testing or long sequence testing. Test cases are repeated many times in long sequence testing, which simulates extended intervals. These techniques for testing have proved to be useful for removing errors that results from system resource consumption and component co-ordination problem. The approach can be more effective and scalable in the field of testing.

James Miller et al. (James Miller et al.,2005) in 2005 proposed aanalysis techniques which was different from the existing for program dependence using genetic algorithm for the generation of test data. Set of problems used this approach which showed efficiency and effectiveness based on defined criterion.

Abdelhamid Bouchachia (Abdelhamid Bouchachia,2007) in 2007 a new algorithm was defined which was named as (IGA) Immune Genetic Algorithm. Complete elaboration of this hybrid algorithmis given but it was not tested in terms of test data generation using some standard programs. Its comparison is done with other evolutionary algorithms.

Two fitness function was proposed by Yong Chen et al.'s (Yong Chen et al.,2009) in 2009 that were based on normalized extended hamming distance(SIMILARITY) and the other one on branch distance (BDBFF),both were applied on GA based test data generation focusing on path oriented. A triangle classification program was selected as an example for the comparing the performance of both the fitness functions.

Yang et. al., (Yang et. al,2009) in 2009 proposed a technique for generation of test data with the help of genetic algorithm for single specific path. Genetic algorithm was used for searching the best solution by selecting the fitness value as the closeness of execution path and target path with overlapping of sub path to calculate the fitness of each individual of the population. Few experiments were conducted to see the effectiveness of the proposed fitness function and the performance of the function was measured with respect to the consumed time and convergence ability. Results proved that the proposed function out performs the other two fitness function for single specific path.

In 2010 Hermadi et al (Hermadi et al,2010) illustrates the difficulties and challenges faced in path testing and has shown the parameters which effects the performance of GA. The analysis of path testing is done on the basis of automation and complexity. Some experiments are performed in this paper. The parameters used by the author are :1)Population size,2) Number of generations,3)Allele range,4)Mutation rate.

Nirpal and Kale, (Nirpal and Kale,2010)in 2010 defined the strategy of automatic test data generation using genetic algorithm for path coverage, which they compared with the Yong Chen approach (Chen Yong, 2009**)** of test data generation for path testing. They proved that this approach performed better as compared to Yong Chen approach, as Genetic algorithm reduces the time required for long testing process by generating more suitable test cases for path testing.

Rauf et. al, (Rauf et. al,2010) in 2010.The author presented a technique for coverage analysis and Graphical user interface testing by using the event driven nature of GUI. Event-flow graph technique is used in the field of automation GUI testing. In the same way as the control-flow graph, GUI also represents all the possibly executed paths in a program. Similarly event-flow model, shows all the promising progressions of the events which can be executed on the Graphical User Interface. Genetic algorithm searches for the best possible test parameter combinations according to predefined test criterion, like coverage function that measures according to the given test criterion as to how much of the automatically generated optimization parameters gets satisfied.

In 2011, Jungsup Oh et. al. (Jungsup Oh et. al.2011).A messy –GA was used by the author in Simulink/state flow models for transition coverage. A tool was introduced to implement this approach of messy GA which was examined on three benchmarks. When this messy-GA was examined and compared with the random search and Simulink/State flow which is a commercial tool formodel testing, the messy-GA achieved better coverage statistically.

In 2012, BinithaS.,et. al. (BinithaS.,et. al.,2012)illustrate that in computer science there are certain complex problems which are difficult to solve such as NP-problems. Here nature comes into play for solving such problems with the help of nature inspired algorithms i.e. Meta heuristics which gives the optimal solution for these problems.

Many such nature inspired algorithms are discussed by the author in this paper eg. Genetic Programming, Genetic Algorithms, Different Evolution Strategies and Paddy Field Algorithm. Other techniques which are also discussed are swam intelligence algorithms, Artificial Immune System Algorithms, Fish Swam Algorithm, Shuffled Frog Leaping Algorithm etc.

In 2014 Hooda et al (Hooda et al,2014)presents a survey of study on different techniques of test case generation. The paper comprises of the review which mainly focused on:

1) Different methods for generating test case.

2) Minimization of test cases

3) Techniques for selection

4) Prioritization technique

5) Evaluation technique

Software testing (Myers, G. J., 2004) can only discover the presence of errors in a program but can never guarantees the absence of errors

4. CONCLUSION

After the detail analysis of various evolutionary techniques especially genetic algorithm used for automated test data generation for different types of testing such as statement coverage, branch coverage and path testing. Lot of research has been done in path testing. With so much of research in automation of the test data generation, no one could still achieve the best performance result in branch coverage testing which can be accepted by industry. So, we can say that we still have a scope of improvement in branch coverage testing for achieving the best performance result which will give path for our future work in this direction.

REFERENCES

[1] Christoph C. Michael, Member, IEEE, Gary McGraw, and Michael A. Schatz, 2001. "Generating Software Test Data by Evolution," IEEE Transcations on so FTW Engineering.December 2001, Vol. 27, No. 1 2,

[2] Xanthakis S, Ellis C, Skourlas C, Le Gall A, Kastiskas S, Karapoulios K. 1992 "Application of genetic algorithms to software testing". In 5th International Conference on Software Engineering and its Applications, pages 625-636, Toulouse, France.

[3] Pei, M., Goodman, E.D., Gao, Z., and Zhong, K., 1994 "Automated Software Test Data Generation Using Genetic Algorithm", Technical Report GARAGe of Michigan State University, June 1994.

[4] Roper, M., Maclean, I., Brooks, A., Miller, J., and Wood, M., .1995. "Genetic Algorithms and the Automatic Generation of Test Data", http://citeseer.ist.psu.edu/135258.html.

[5] Alander, J.T., Mantere, T., and Turunen, P., "Genetic Algorithm Based Software Testing", http://citeseer.ist.psu.edu/40769.html. 1997.

[6] Jones, B., Sthamer, H., and Eyres, D., "Automatic Structural Testing Using Genetic Algorithms", 1996. Software Engineering Journal 11(5), September 1996, pp. 299-306.

[7] Michael, C.C., McGraw, G.E., Schatz, M.A., and Walton, C.C., 1997 ."Genetic Algorithms for Dynamic Test Data Generation", Technical report, Reliable Software Technologies, Sterling, VA. May 23, 1997.

[8] Tracey, N.J., Clark, J., Mander, K., and McDermid, J.,1998, "An Automated Framework for Structural Test-Data Generation", In Proceedings 13th IEEE Conference in Automated Software Engineering, Hawaii, October 1998.

[9] Pargas, R.P., Harrold, M.J., and Peck, R.R., 1999."Test-Data Generation Using Genetic Algorithms", Journal of Software Testing, Verification and Reliability..

[10] Lin, J.C. and Yeh, P.L.,2000. "Using Genetic Algorithms for Test Case Generation in Path Testing", In Proceedings of the 9th Asian Test. Symposium (ATS'00). Taipei, Taiwan, December 4-6, 2000

[11] Bueno, P.M.S. and Jino, M., 2000. "Identification of Potentially Infeasible

Program Paths by Monitoring the Search for Test Data", In Proceedings of the Fifteenth IEEE International Conference on Automated Software Engineering (ASE '00), pg 209218, Grenoble, France, 11-15 September 2000.

[12] Wegener, J., Buhr, K., and Pohlheim, H., 2002."Automatic Test Data Generation for Structural Testing of Embedded Software Systems by Evolutionary Testing", In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO2002, New York, USA, 9-13th July 2002.

[13] Daz E, Tuya J., and Blanco R., 2003. "Automated Software Testing Using a Metaheuristic Technique Based on Tabu Search", In 18th IEEE International Conference on Automated Software Engineering, pp. 310-313,.

[14] Berndt, D.J., Fisher, J., Johnson, L., Pinglikar, J., and Watkins, A., "Breeding Software Test Cases with Genetic Algorithms", In Proceedings of the Thirty-Sixth Hawai`i International Conference on System Sciences (HICSS-36), Hawaii, January 2003.

[15] Hermadi, Irman, and Ahmed, M .,2003 "Genetic algorithm based test data generator." Evolutionary Computation, 2003. CEC'03. The 2003 Congress on. Vol. 1. IEEE.

[16] Tonella P., "Evolutionary Testing of Classes", In Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing andanalysis (ISSTA 04), ACM Press, New York, NY (2004) 119-128.

[17] D. J. Berndt, A. Watkins, "High Volume Software Testing using Genetic Algorithms", hicss, pp.318b, Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 9, 2005.

[18] James Miller, Marek Reformat and Howard Zhang, "Automatic test data generation using genetic algorithm and program dependence graphs", Science Direct, Information and Software Technology ,2006Vol.48 , No. 586–605.

[19] AbdelhamidBouchachia, "An Immune Genetic Algorithm for Software Test Data Generation", IEEE, Seventh International Conference on Hybrid Intelligent Systems 2007.

[20] Chen Yong, Zhong Yong, Tingting Shi1, Liu Jingyong,,"Comparison of Two Fitness Functions for GA-based Path-Oriented Test Data Generation" ,2009 Fifth International Conference on Natural Computation, IEEE 2009

[21] Yang Cao, Chunhua Hu and LumingLi,"An Approach to Generate Software Test Data for a Specific Path Automatically with Genetic Algorithm", International Conference on Reliability, Maintainability and Safety,2009 pp.888-892.

[22] Hermadi, Irman, Chris Lokan, and RuhulSarker. "Genetic algorithm based

path testing: challenges and key parameters." Software Engineering (WCSE), 2010 Second World Congress on. Vol. 2. IEEE

[23] Nirpal, Premal B and Kale K.V. “Comparison of Software Test Data for Automatic Path Coverage Using Genetic Algorithm”, Internal Journal of Computer Science and Engineering Technology, 2010, ISSN:2229-3345 Vol. 1, Issue 1.

[24] Rauf A., Anwar S., Jaffer M.A.andShahidA.A. “Automated GUI Test Coverage Analysis Using GA”, 7th International Conference on Information Technology New Generations,2010 pp. 1057-1062.

[25] Jungsup Oh, Harman, M., Yoo S.,2011. “Transition Testing for Simulink/Stateflow Model Using Messy Genetic Algorithms”, ACM, GECCO'11, July 12-16, 2011, Dublin Ireland.

[26] Binitha S, S Siva Sathya.,2012, “A Survey of Bio inspired Optimization Algorithms, international journal of soft computing and engineering”(IJSCE) ISSN: 2231-2307,May 2012, Vol. 2, No. 2.

[27] Hooda, Itti, and Rajender Chhillar. "A Review: Study of Test Case Generation Techniques." International Journal of Computer Applications 2014, Vol .107, No .16.

[28] Myers, G. J., 2004 “The Art of Software Testing”, New Jersey: Wiley.