International Journal of Software Engineering and Its Applications
Vol. 10, No. 12 (2016), pp. 13-20
http://dx.doi.org/10.14257/ijseia.2016.10.12.02

Techniques for Automated Test Cases Generation: A Review

Pranali Mahadik[1], Debnath Bhattacharyya[2] and Hye-jin Kim[3*]

[1]*Computer Engineering Department, Bharati Vidyapeeth Deemed University College of Engineering, Pune, Maharashtra, India*
[2]*Computer Science and Engineering Department, Vignan's Institute of Information Technology, Vishakhapatnam-530049, AP, India*
[3]*Sungshin Women's University, 2, Bomun-ro 34da-gil, Seongbuk-gu, Seoul, Korea*
[1]*pranali72@gmail.com,* [2]*debnathb@gmail.com,* [3]*hyejinaa@daum.net*

Abstract

Generation of test cases is challenging part of software testing phase. Test cases must have ability to cover more test objective features i.e., achieve more test coverage. The different techniques for automatic generation of test cases find small set of test cases reduce human efforts as well as cost of software testing and produce most efficient and effective testing of software product. This survey gives overview of different techniques of automatic test cases generation.

Keywords*: software testing; test case generation; techniques of automatic test cases generation*

1. Introduction

Software testing is done for finding faults in program. In different phases of SDLC testing is very vital and expensive phase in both development and maintenance manner. Software testing phase generally estimate 30 to 50% of total development cost in this also challenging part is generating test cases. More number of test cases will be able to number of faults. For these reasons, the different techniques for automatic generation of test cases used to find out small set of test cases which are able to fulfill criteria and reduce software cost. A test case is a set of tests which takes input values and observed output and compare it with expected output. Main feature of test case is it has quality to cover most of the test objective and contribute to reduce the cost of software testing.

Table 4 presents a classification of automatic test case generation approaches along with advantages and disadvantages of each technique.

2. Automatic Test Data Generation Techniques

Automatic test data generation is done for two most important reasons.

Reduce cost of software testing: cost can increase while testing phase because of inappropriate test cases it also lead to loss of time.

Reduce human errors: manual test cases are totally depend upon the testers understanding as well as skill set of tester it may also lead to inclusion of bugs after testing to overcome this problem automatic test data generation is used.

* Corresponding Author

ISSN: 1738-9984 IJSEIA

2.1. Random Test Data Generation

Random test data generation techniques [1] choose inputs arbitrarily until useful inputs are found. This technique may fail because info about the test requirements isn't incorporated. The various disadvantages of this method are like it's appropriate just for simple and small programs, many sets of values might lead to a similar observable behavior and are so redundant and therefore the probability of selecting specific inputs that cause buggy behavior.

Random testing is, as previously represented, a fast and easy way to generate a large number of test cases. Random testing is mostly utilized in conjunction with different techniques. Mostly in order to get an initial test case from which other analyses may be used to generate consequent input values.

An example of a tool which uses random testing in conjunction with different techniques is DART [2] and JCUTE [3]. Here random testing is used to get an initial set of tests. These tests are then run and dynamic analysis of however the program behaves under random testing is then used to generate new inputs to systematically direct the program on various methods.

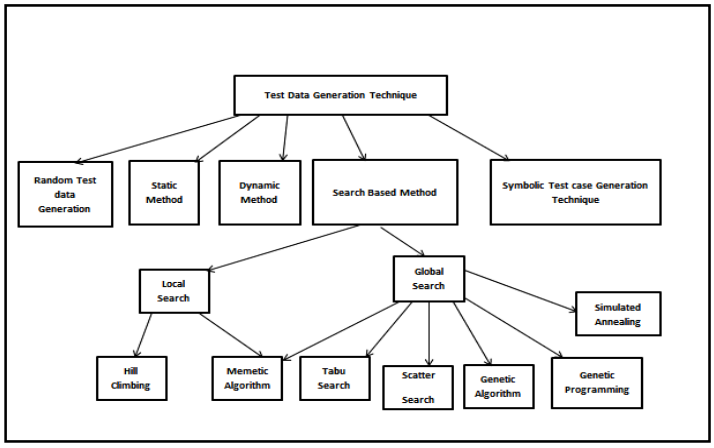

Figure 1. Test Data Generation Techniques

Table 1. Tools Related to Random Test Data Generation Technique [6]. Table Shows the Tools which are using Random Test Sata Generation Technique

SR NO	TOOL	INSTITUTION	LAST MODIFICATION ON	TYPE	INPUT CODE TYPE	REQUIRED INPUT	OUTPUT	TESTING TECHNIQUE	DOMAIN	LICENSE
1.	C++ Test	Parasoft	2010	Commercial	C++	Source and binary code	Unit tests	Static analysis	Desktop	Commercial 14 days validity
2.	AgitarOne	Agitar Technologies	2015	Commercial	Java	Source code	Junit tests	Observation derived testing	Desktop	Commercial 30 days validity
3.	CodeProAnalytix	Google Inc	2010	Commercial	Java	Source code	Junit test cases	White box Testing technique	Desktop	Apache license 2.0
4.	Randoop	MIT CSAIL	2015	Open source software	JAVA	Source code	Unit test suite	Feedback directed random testing	Desktop	MIT license
5.	Jwalk	University of *Sheffield*	2013	Academic research	JAVA	Source code	Test report	lazy systematic testing	Desktop	JWalk License

6.	GRT	Lei ma, cheng, hiroyuki, Johannes, rudolf	2015	Academic research	JAVA	Source or byte code	Junit Test cases	Static and runtime analysis	desktop	Free
8.	Jcrasher	Christoph and yannis	2007	Academic research	JAVA	Source code	Test File	Random or robust testing	desktop	Apache Licence

2.2. Symbolic Test Case Generation Technique

James C. King was proposed Symbolic execution as a tool for test-data generation in 1976 [12].Symbolic test data generation techniques [7, 8] create algebraic expression for several constraints in program by assigning symbolic values to the variable. This technique also used for many purposes like bug detection, verification of program, program debugging, maintenance and localization of fault[5].Symbolic state maintains by this technique which maps symbolic path constraint (pc) and symbolic expression.

Symbolic execution technique is a key element for precise automatic code-driven test generation. The main aim of verification is to prove absence of program errors.Symbolic execution for automatic test generation has two types of groups.

- Static test generation
- Dynamic generation

Table 2. Tools Related to Symbolic Execution Technique [8]. Table Shows the Tools which are using Symbolic Execution Technique

SR NO	TOOL	INSTITUTION	LAST MODIFICATION ON	TYPE	INPUT CODE TYPE	REQUIRED INPUT	OUTPUT	TESTING TECHNIQUE	DOMAIN	LICENSE
1.	JCUTE	University of *Illinois* By kaushik sen	2006	Academic research	Symbolic execution	JAVA	Source code	Unit tests	Search based concolic testing	Desktop
2.	CATG	University of *Illinois* By kaushik sen	2015	Open source software	Symbolic execution	JAVA	Source code	Test cases	Random testing	desktop
3.	Symbolic PathFinder	Microsoft	2015	Open source software	Symbolic Execution	JAVA	Software	Test cases	Model checking	Web application
4.	PET	Elvira and miguel	2011	Academic Research	Symbolic Execution	JAVA	Byte code	Test cases	White box testing	desktop

2.3. Static Test Generation

Static test data generation is done without executing program it analyze the program statically by using symbolic execution technique. it compute input by considering different constraints. Computational cost of this technique is high.

Disadvantage of this approach is the statement cannot be reasoned about symbolically.

For Example [14] of this drawback is,

```
int foo(int x, int y)
{
If (x==hash(y)) return -1;//error
```

```
Return 0; //ok
}
```

Here assume that for function hash the constraint solver can't symbolically reason because it is much complex or simple because it is not available. it means that the constraint solver not able to generate 2 values for input x and y gives guarantee to satisfy or violate the constraint x==hash(y). Here test generation is not only sufficient but also it is also necessary to generate specific values for x and y which satisfy or violate this constraints.

2.4. Dynamic Test-data Generation

Dynamic test-data generation [*e.g.,* 10, 11, 9] technique determines which test cases satisfy the requirement during the execution of program. After that test input modify incrementally until one of them satisfy the requirement. Search based techniques are most of the time used by dynamic techniques. Symbolic execution performing dynamically collects symbolic constraint on input which is gathered from the predicates in branch statement along the execution of program.

2.5. Search-based Automatic Test Case Generation

To solve software engineering problem Search-based Software Engineering (SBSE) is the application of optimization technique (OT). Process to find best possible solution among all available is known as optimization. Search-based approach is very useful for un-decidable problem [13] as well as for programs which have large search space. Input domain of the system under test is used as search space from this search space test data fulfils objective of test data generation. For purpose of test cases generation there are different search algorithms to use like.

2.5.1. Automatic Test Case Generation using Hill Climbing: Hill climbing (HC) is mathematical optimization technique comes under local search algorithm family. HC is an iterative algorithm it starts with arbitrary solution to problem then in next attempts it find better solution by incrementally modifying single element of solution. If modification generates better solution than previous one then it made new solution then this process repeated until further improvements can found.

It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.

2.5.2. Automatic Test Case Generation using Genetic Algorithm: Genetic algorithms [4] represent a class of adaptive search techniques and procedures supported the method of natural genetics and Darwin's principle of the survival of the fittest. Genetic algorithm searching mechanism starts with a collection of solution referred to as a population. One solution within the population is named a chromosome. The search yield for variety of generations, for every generation the fitter solutions (based on the fitness function) are selected to create a new population. During the cycle, there are 3 main operators namely reproduction, crossover and mutation. The cycle can repeat for variety of generations till certain termination criteria are met.

In a method of generating test cases for structural testing based on genetic algorithms to cover multiple target paths in one run is presented. First, drawback the matter of generating test cases is developed as a multi-objective optimization problem within which the number of objectives decreases at the side of generation of test cases. Then, test cases are generated using genetic algorithms incorporating with domain knowledge.

In a Constraint-based [19] Genetic algorithm technique is employed to get optimized test cases from UML Activity diagram and Collaboration diagrams. [11] Is regarding test cases generated for software security supported GA. the method selects the foremost favorite test cases so as to get the vulnerabilities within the software. This methodology will reduce the test time and improve the test efficiency to a certain extent.

2.5.3. Automatic test case Generation exploitation Genetic Programming: Genetic programming (GP) is an evolutionary algorithm-based methodology inspired by biological evolution to find computer programs that perform a user-defined task. It's a specialization of genetic algorithms (GA) wherever every individual may be a computer program. It's a machine learning technique used to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task.

a search-based approach[2] to automatically generating test cases for object oriented software. It depends on a tree-based illustration of technique call sequences. Strongly-typed genetic programming is utilized to generate method call trees that respect the call dependences among the methods. A new kind of distance-based fitness function applied that accounts for runtime exceptions. The approach proved successful and produced test cases leading to full branch coverage for four test objects completely automatically.

2.5.4. Automatic test case Generation using Memetic Algorithm: The Memetic algorithms (MAs) are metaheuristics that use both global and local search (*e.g.*, a GA with a HC). Here the thought is to mimic the process of the evolution of these memes. From an optimization point of view, a MA is described as a population based mostly met heuristic during which, whenever an offspring is generated, a local search is applied to it till it reaches a local optimum.

The MA used in [4] for testing of object-oriented containers is fairly straightforward. it's built on GA, and therefore the solely difference is that at every generation on every individual a Hill climbing is applied till a local optimum is reached. the cost of applying those local searches is high, hence the population size and the total range of generations is lower than in the GA.

2.5.5. Automatic test case Generation using Tabu search: Tabu Search (TS) is a metaheuristic search technique based on the premise that, in order to qualify as intelligent, problem solving should incorporate adaptive memory and responsive exploration. Thus, the algorithm of Tabu Search is based on that of the next k neighbors, whereas maintaining a Tabu list (memory) of visited neighbors that tabu. The Tabu Search rule features a range of parameters that have to be chosen on the idea of the problem to be solved: the objective operate (fitness function) that has to measure the cost of a solution, an appropriate Candidate list strategy (to try to select good neighbor candidates that goes beyond a local optimum without exploiting the examination of parts within the neighborhood) and, it's additionally necessary to outline short - term memory and memory and their several methods. Short -term memory stores the recent moves of the search as Tabu. memory, on the other hand, stores the frequency with that a move occurs so as to penalize often visited moves that diversify the search. TSGen [3] may be a Tabu search metaheuristic rule for the automated generation of structural package tests. The goal that TSGen needs to achieve is that of generating the tests that obtain the most branch coverage for the program below test.TSGen [22] Created an economical testing technique that combines Tabu Search with Korel'schaining approach. The technique automatically generates test data so as to get branch coverage in software testing.

2.5.6. Automatic test case generation using Simulated Annealing: Simulated annealing is one type of global search technique it is similar in principle to Hill Climbing. To improve the result it recombines result after sampling the whole area.

For example if result value y1 is chosen in simulated annealing and the result which has minimum cost. Comparative and desirability of particular solution is defined by cost function. Cost function is referred to minimizing objective of function and maximizing the objective of function is referred as Fitness Function.

Table 3. Tools Related to Search based Test Data Generation Technique [6]. Table Shows the Tools which are using Different Search based Techniques to Generate Test Data Automatically

SR NO	TOOL	INSTITUTION	LAST MODIFICATION ON	TYPE	INPUT CODE TYPE	REQUIRED INPUT	OUTPUT	TESTING TECHNIQUE	DOMAIN	LICENSE
1.	EvoSuite	Research project by dr.gordon Fraser and dr.andrea arcuri	2015	Academic research and open source software	Search Based	JAVA	Source code	Junit tests	Search based approach	desktop
2.	JTExpert	Abedelilah sakti,gilles, yann-gael	2015	Academic research	Search Based	JAVA	Source code	Whole test suite	Search Based	Desktop

Table 4. Automatic Test Data Generation Technique

SR.NO.	TECHNIQUE	MAIN IDEA	ADVANTAGE	DISADVANTAGE
1	Random-Based	Generation of test cases done randomly	1. Easy to implement 2. Very useful for small program It introduces randomness in software testing process Which reflects irregularity of system environment 4. Easily combine with other technique. 5. Most of the time used as a benchmark because it is easy to implement.	1. does not suitable for large programs. 2. Does not use available information to generate test cases. 3.may not have acceptable test Coverage.
2	Search-Based	Problem of test case generation is considered as optimization problem and looking to find out best test set for class under test	1. Efficiency of approach reflected form its result	1.produce large search space

3	Symbolic-Execution	Execute program with symbol rather than value	1. Also use for bug finding.	1. it has scalability issues due to the large number of paths that need to be analysed And the complexity of the constraints that are generated.

3. Conclusion

This paper represent different techniques for automatic test case generation and brief look at each one along with tools related with technique are mentioned. there are several techniques available but depending upon the need of user which technique is applicable to find best solution this survey helps whether to use technique with combination or alone this analysis is given.

References

[1] Edvardson J, "A Survey on Automatic Test data generation", In proceedings of the second conference on computer science and engineering., vol. 2, no. 1, **(1999)**, pp. 343- 351.
[2] P. Godefroid, N. Klarlund and K. Sen, "Dart: directed automated random testing", SIGPLAN Not., vol. 40, no. 6, **(2005)**, pp. 213–223.
[3] K. Sen, D. Marinov and G. Agha, "Cute: a concolic unit testing engine for c", ACM, **(2005)** Sep 5, pp. 263–272.
[4] T. Feng and KasturiBidarkar, "A Survey of Software Testing Methodology", vol. 25, no. 3, **(2008)**, pp. 216- 226.
[5] J. M. Voas, J. Morell and Miller, "Predicting where faults can hide from testing", IEEE, vol. 8, **(1991)**, pp. 41-48.
[6] Pranali Prakash Mahadik, Prof. Dr. D. M. Thakore, "Survey on Automatic Test Data Generation Tools and Techniques for Object Oriented Code", vol. 4, Issue 1, **(2016)** January, pp. 358-364.
[7] W. E. Howden, "Symbolic testing and the DISSECT symbolic evaluation system", IEEE Transactions on Software Engineering ,vol. 3, no. 4, **(1977)**, pp. 266-278.
[8] J. Clarke, M. Harman and B. Jones, "The Application of Metaheuristic Search techniques to Problems in Software Engineering", IEEE Computer Society Press, vol. 42, no. 1, **(2005)** September 5, pp. 247-254.
[9] N. Gupta, A. P. Mathur and M. L. Soffa, "Generating Test Data for Branch Coverage", In Proceedings of the 15th IEEE International Conference on Automated Software Engineering, **(2000)** September, pp. 219–227.
[10] B. Korel, "A Dynamic Approach of Test Data Generation", In IEEE Conference on Software Maintenance, San Diego, **(1990)** November, pp. 311–317.
[11] A. J. Offutt, Z. Jin and J. Pan, "The Dynamic Domain Reduction Procedure for Test Data Generation", Software Practice and Experience, **(1997)**, pp. 167–193.
[12] J. C. King, "Symbolic execution and program testing", ACM, vol. 19, no. 7, **(1976)**, pp. 385–394.
[13] L. A. Clarke and D. J. Richardson, "Applications of symbolic evaluation. Journal of Systems and Software, **(1985)**, pp. 15–35.
[14] P. Godefroid, "Compositional Dynamic Test Generation", ACM, **(2007)** January, pp. 47–54.