Efficient Algorithm Selection for Detecting Suitable Test Case Prioritization

A.Pravin
Research Scholar,
Sathyabama University,
Chennai.

S.Srinivasan
Director – Affiliation,
Anna University of Technology,
Madurai.

ABSTRACT

Genetic algorithms have been successfully applied in the area of software testing. The demand for automation of test case generation in object oriented software testing is increasing. Genetic algorithms are well applied in procedural software testing but a little has been done in testing of object oriented software. This paper discusses genetic algorithms that can automatically select an efficient algorithm which is suitable for test cases selection. This algorithm takes a selected path as a target and executes sequences of operators iteratively for efficient algorithm selection to evolve. The evolved efficient algorithm selection can lead the program execution to achieve the target path. An automatic path-oriented test data generation is not only a crucial problem but also a hot issue in the research area of software testing today. We also propose genetic algorithm for the selection of the suitable algorithm, which perform much better than the existing methods and can provide very good solutions.

General Terms

Automatic selection process, Apriori algorithm, Pincer search algorithm, FP-Tree algorithm

Keywords

Conformance testing, prioritized test case generation, test case selection

1. INTRODUCTION

Software being utilized in various situations and software quality becomes more important than ever. Being main means of software quality assurance, software testing is very laborious and costly due to the act that it is accounts for approximately 50 percent of the elapsed time and more than 50 percent of the total cost in software development [4, 5]. Automatic test data selection is a key problem in software testing and its implementation can not only significantly improve the effectiveness and efficiency but also reduce the high cost of software testing[3, 4]. In particular, it is notable that various structural test case selection problem can be transformed into a selection of the best suitable algorithm. Moreover test case selection; strategy can detect almost 95 percent of errors in program under test [8]. Although efficient algorithm selection for detecting suitable test case prioritization is an undesirable problem [6], researchers still attempt to develop various methods and have made some progress. These means can be classified as dynamic selection of suitable algorithm. Dynamic methods include random selection of suitable algorithm. Dynamic methods include random selection of algorithms and extract suitable test case selection it's a kind of goal-oriented approach [15], and evolutionary approach [13, 14-16]. As values of input variables are determined when programs execute, dynamic test case selection mechanism can avoid those problems with that of the existing methods are confronted. In this article we propose to investigate the use of multi objective algorithms in order to combine a test case generation technique with a test case selection and prioritization method. The objective of the proposed work is to generate optimized algorithm to select a suitable test case in order to their importance with respect to test goals [3]. A multi objective algorithm can be applied to test case selection and prioritization problems. The need for multi objective algorithms to tackle the kind of problems will also be considering in this paper [4].

2. RELATED WORK

After development and release, software undergo regress maintenance phase of ten to fifteen years. Modifications in software may be due to change in customer's requirements or change in technology/platform. This leads to release of numerous versions/editions of the existing software. Also in case of the version/edition's test only modified and affected parts are to be tested to impart confidence in the modified software which is the process of testing. During development phase, time and budget of software permits for its thorough testing but same is not the case for regression testing. So, effective and intelligent prioritization of the actual software's test suite has to be done to make remarkable savings of time and budget. Several attempts have been made in finding techniques/algorithms for the test case prioritization. The time-constrained testing can be reduced to NP-complete problem and Test case selection and prioritization are well studied and understood testing techniques. Equally, test case generation is an active research area. Yet the combination of these techniques remains largely unexplored. Here we present a new model for the algorithm selection problem in protocol conformance testing, the goal of which is to select a suitable algorithm for a particular test case prioritization from a given set of algorithms. We also propose genetic algorithm for the selection of the algorithm, which perform much better than the existing methods and can provide very good solutions.

Genetic Algorithms follow the concept of solution evolution by stochastically developing generations of solution populations using some given fitness function. They are particularly applicable to large, non-linear and possibly discrete in nature kind of problems. Evolutionary algorithms (EA) when applied for the selection of algorithm the procedural software can be used to specifically look for test scenarios that cover certain branches of a program. These kinds of algorithms are based on reproduction, evaluation and selection. The GA in general has mainly four stages, which are evaluation, selection, crossover and mutation. The evaluation procedure measures the fitness of each individual solution (also called chromosome) in the population and assigns it a relative value based on the defining optimization (or search) criteria here we are taking the chromosomes as different test cases. The selection procedure randomly selects individuals of the current population for development of the next generation. Various alternative methods exist but all follow the idea that the fittest have a greater chance of survival. Selection chooses the chromosomes to be recombined and mutated out of this initial population. Recombination reproduces the selected individuals and exchanges their information (pair-wise) in order to produce new individuals. This information exchange is called crossover. The crossover procedure takes two selected individuals and combines them about a crossover point thereby creating two new individuals. Mutation introduces a Small change to each newly created individual. The resulting individuals are then evaluated through the fitness function. It transfers the information encoded in the chromosome, the so called genotype, into an execution of getting a new solution for the problem that occurs in selection of the test case. The fitness function measures how well the chromosome satisfies the test criterion. The implementation of the fitness function follows earlier standards in evolutionary testing, described in other articles [11, 12]. This iterative process continues until one of the possible termination criteria is met: if a known optimal or acceptable solution level is attained; or if a maximum number of generations have been performed; or if a given number of generations without fitness improvement occur.

3. AUTOMATIC SELECTION PROCESS

Genetic Algorithms begins with a set of initial individuals as the first generation, which are sampled at random from the problem domain. The algorithms are developed to perform a series of operations that transform the present generation into a new, fitter generation [22]. Each individual in each generation is evaluated with a fitness function. Based on the evaluation, the evolution of the individuals may approach the optimal solution. Figure1 explains the overall proposed architecture and the most common operations of genetic algorithms are designed to produce efficient solution for the target problem [15].

The input of our system is a program module which is given as an input to preprocess. The input comprises the general information of particular domain provided by the user. For instance, if the domain provided by the user is Student academic details. The metadata provided by the user is used to confine the search through the entire database. In addition to metadata the user asked to provide desiderata, which is the information about the required output from the system as expected by the user.

The selection of the criteria is done by the user by listing all the attributes of the database given. The attributes selected are also influences the selection of the algorithm to be used for mining. In this criteria selection there are some parameters are also used as the input to the selection of the algorithms. The parameters are Runtime, Accuracy, Comprehensibility, Data input size, Memory utilization.

i) Randomly select two individuals as a couple from the parent generation.
ii) Randomly select a position of the genes, corresponding to this couple, as the crossover point. Thus, each gene is divided into two parts.
iii) Exchange the first parts of both genes corresponding to the couple.
iv) Add the two resulted individuals to the next generation.

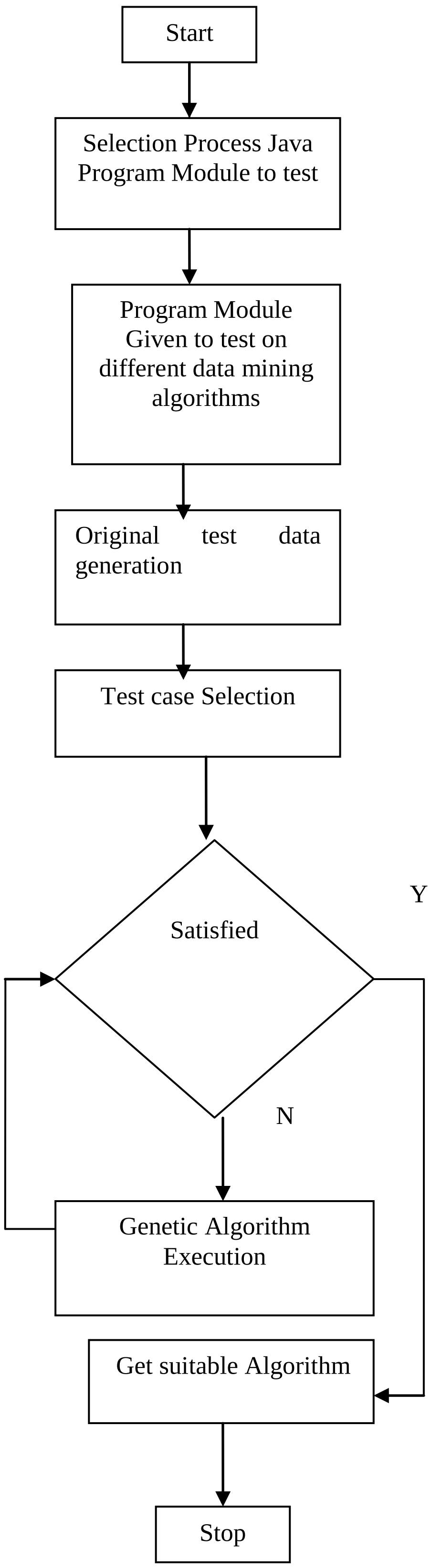
Start
Selection Process Java Program Module to test
Program Module Given to test on different data mining algorithms
Original test data generation
Test case Selection
Satisfied
Y
N
Genetic Algorithm Execution
Get suitable Algorithm
Stop

Figure1. Explains the Proposed Architecture

These operations are iterated until the expected goal is achieved. Genetic algorithms guarantee high probability of improving the quality of the individuals over several generations [5].

4. IMPLEMENTED ALGORITHMS

The input of our system is given by the user. The input comprises the general information of particular domain provided by the user. The domain provided by the user can be of a testing application of a java based program module. The selection of the criteria is done by attributes of the different test case conditions. The attributes selected are also influences the selection of the algorithm to be used for mining.

4.1 Apriori algorithm

Uses a Level-wise search, where k-itemsets (An itemset that contains k items is a k-itemset) are used to explore (k+1)-itemsets, to mine frequent itemsets from transactional database for Boolean association rules. First, the set of frequent 1-itemsets is found. This set is denoted L1. L1 is used to find L2, the set of frequent 2-itemsets, which is used to fine L3, and so on, until no more frequent k-itemsets can be found. The apriori algorithm is an efficient algorithm for knowledge mining in form of association rules [2]. We have recognized its convenience for document categorization. The original apriori algorithm is applied to a transactional database of market baskets. In our case, instead of a market basket, we work with the testing and test case selection (represented by sets of significant terms).

4.2 Pincer search algorithm

Pincer Search Algorithm uses both, the top-down and bottom-up approaches to Association Rule mining. It is a slight modification to Original Apriori Algorithm. In this the main search direction is bottom-up (same as Apriori) except that it conducts simultaneously a restricted top-down search, which basically is used to maintain another data structure called Maximum Frequent Candidate Set. As output it produces the Maximum Frequent Set i.e. the set containing all maximal frequent itemsets, which therefore specifies immediately all frequent itemsets. The algorithm specializes in dealing with maximal frequent itemsets of large length.

4.3 FP-Tree Algorithm

It is one of the data mining algorithm here we are using it for test case selection. The features of the algorithms are:

- Size of FP-tree depends on how items are ordered.
- In the previous example, if ordering is done in increasing order, the resulting FP-tree will be different and for this example, it will be denser (wider).
- At the root node the branching factor will Increase from 2 to 5 as shown on next slide.
- Also, ordering by decreasing support count doesn't always lead to the smallest tree.
- This algorithmrates frequent itemsets from FP-tree by traversing in bottom-up fashion.
- This algorithm extracts frequent itemsets [6].

5. RESULTS & DISCUSSION

The algorithm is selected based on the parameters selected. Here input to the genetic algorithm is the bit corresponding to each parameter. The input will be a row which contains the value of the parameter. For some parameters the values may be kept as levels. The various steps involved are:

(i) The values of the parameters are passed from the criteria selection module. Parameterized input: [2 3 2 1 0]

(ii) These values are used to form a matrix, which is given as input to a fitness function.

(iii) A mapping between the input parameters (chromosome) and the output which are generated automatically by the number of iterations.

(iv) The output is in the form of decimal values which corresponds to all the three algorithms.

i.	A-priori	:	0.0003
ii.	Pincer search	:	0.0001
iii.	FP-tree	:	0.0002

(v) The higher value is assumed to be selected algorithm and that algorithm which is been suitable for that particular module.

(vi) The result of the experiment is shown in Figure 2 from which it is clear that GP approach outperforms of the selection of different algorithms.

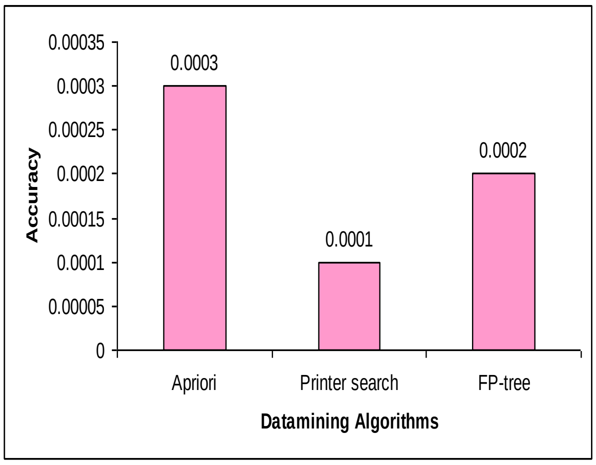

Figure 2. Different datamining algorithm Vs Accuracy of the output

6. CONCLUSION

In this paper, the genetic algorithm is used to automatically select the suitable algorithm for a test case prioritization. The greatest merit of genetic algorithm in program testing is its simplicity. Each iteration of the genetic algorithms generates a generation of individuals. In practice, the computation time cannot be infinite, so that the iterations in the algorithm are been limited by fixing a randomized threshold. The quality of test case selection produces by genetic algorithms is higher than the quality of test cases produced by random way because the algorithms that can direct the generation of test cases to the desirable range fast.

7. REFERENCES

[1] Alspaughy,S.,Walcotty,K.R.,Belanichz,M.,Kapfhammerz ,G.M.,and Soffa,M.L.," Efficient Time-Aware Prioritization with Knapsack Solvers", Proceedings of the ASE 2007 Workshop on Empirical Assessment of Software Engineering Languages and Technologies. Atlanta, Georgia, pp. 1-6.

[2] Askarunisa, A., Shanmugapriya, L., Ramaraj. N., "Cost and Coverage Metrics for Measuring the Effectiveness of Test Case Prioritization Techniques", INFOCOMP Journal of Computer Science, pp. 1-10.

[3] Shin Yoo and Mark Harman,"Using hybrid algorithm for Pareto effcient multi objective test suite minimization", Journal of Systems Software, 83(4):689–701, April 2010.

[4] Mark Harman," Making the case for morto: Multi objective regression test optimization", In The 1st International Workshop on Regression Testing (Regression 2011), Berlin, Germany, 2011.

[5] Antonia, B., "Software Testing Research: Achievements, Challenges, Dreams", in 2007 Future of Software Engineering: IEEE Computer Society, 2007.

[6] Christian Borgelt, "An Implementation of the FP-growth Algorithm".

[7] Kant Singh,V., Shah,V., Kumar Jain,Y., Shukla,A., Thoke,A.S., Kumar Singh,V., Dule,C., and Parganiha ,V.,"Proposing an Efficient Method for Frequent Pattern Mining".

[8] Wegener,J., and Grochtmann,M., "Verifying timing constraints by means of evolutionary testing", Real-Time Systems, Vol.3, No.15, pp. 275-298, 1998.

[9] Wappler,S., and Lammermann,F., "Using evolutionary algorithms for the unit testing of object-oriented software", Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, Washington DC, USA, June 25-29, ACM, New York, pp. 1053-1060, 2005.

[10] Tonella,P., "Evolutionary Testing of Classes", Proceedings of the 2004 ACM SIGSOFT Intl. Symposium on Software Testing and Analysis, Boston, July 11-14, pp. 119-128, 2004.

[11] B Jones et al. "Automatic Structural Testing Using Genetic Algorithms", Software Engineering Journal, Vol.11, No.5, 1996.

[12] McMinn,P., "Search-Based Software Test Data Generation: A Survey", Software Testing, Verification and Reliability, Vol.14, No.2, pp. 105—156, 2004.

[13] Koza,D., Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.

[14] Seesing,A., and Gross,H.G., "A Genetic Programming Approach to Automated Test Generation for Object-Oriented Software", International Transactions on Systems Science and Applications, Vol.1, No.2, pp.127-134, 2006.

[15] Peter M. Kruse and Magdalena Luniak" Automated test case generation using classification trees. ASQ Software Quality Professional, 13:4–12, December 2010.

[16] Arturo Hern/andez Aguirre, Salvador Botello Rionda, Carlos A. Coello Coello, Giovanni Liz/arraga Liz/arraga, and Efr/en Mezura Montes," Handling Constraints using Multiobjective Optimization Concepts", International Journal for Numerical Methods in Engineering, 59(15):1989–2017, April 2004.