Optimization of Basis Path Testing using Genetic Tabu Search Algorithm

Chetan Sharma  
PG Student  
Department of Computer Engineering  
U.I.E.T., Kurukshetra, India

Dr. Karambir  
Assistant Professor  
Department of Computer Engineering  
U.I.E.T., Kurukshetra, India

Abstract

The test cases are generated manually as well as automatically to test software product. Manual test case generation require more time and cost than automatic test case generation. Automated generated test cases decrease the time and cost of testing process. Nowadays software companies are turning towards automated test case generation. After the generation of test cases, there is need to optimize the generated test cases. For their optimization we will apply artificial intelligence techniques. The genetic algorithm is based on the evolutionary ideas of natural selection and genetics. Genetic Algorithm (GA), a heuristic search algorithm is used for optimization. Here, we will integrate tabu search algorithm with genetic algorithm (GTSA) for optimization the result using an analysis of data.

Keywords: Genetic Algorithm, Automated Testing, Test Cases, Testing Tools

I. INTRODUCTION

Software is a collection of programs, which are designed to perform particular operation. A program is a sequence of instructions, which are written to solve a particular problem. Nowadays, software is also used into industrial production, commercial activities, scientific experiments. The quality of software has got more and more attention. To assure software quality, software testing plays a significant role which can't be replaced by other ways. Software testing is an expensive and labor-intensive work, which often accounted for about 50% of the total workload in the software development [1]. The main goal of software testing is to prove that the product meets all the pre-established requirements to ensure better functionality. It is a costly process which increases the total development cost of a software product. Software testing can be divided into static testing and dynamic testing. Dynamic testing takes a great portion in software testing. Dynamic testing usually has high efficiency compared to static testing. Various static testing techniques are informal review, technical review, walkthrough, inspection, static code review. Various dynamic testing techniques are unit testing, integration testing and system testing. Non-functional testing like performance testing, security testing falls under dynamic testing. The automation process of test data generation is an important step in reducing the cost and time of software development. Applying the artificial intelligence technology to the process of automatic generation of software test data can greatly improve the efficiency of automatic generation of test data. In this case, the test data generation problem is transformed to the test data search optimization problem. The general aim of testing is to confirm the quality of software systems systematically by exercising the software in carefully controlled circumstances [2]. The difference between static and dynamic testing are:

<table>
<thead>
<tr>
<th>Static Testing</th>
<th>Dynamic Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing Occur Without Executing The Program.</td>
<td>Testing Occur By Executing The Program.</td>
</tr>
<tr>
<td>It Can Occur Without Compilation.</td>
<td>It Is Performed After Compilation.</td>
</tr>
<tr>
<td>Require Large Number Of Meeting.</td>
<td>Require Less Number Of Meetings.</td>
</tr>
</tbody>
</table>

As a result, improvement in automation of software testing tools becomes an important way for improving the software test efficiency and ensuring software quality. Dynamic testing includes white box testing and black box testing. In black box testing internal structure/ design/ implementation of the item being tested is not known to the tester. While in white box testing internal structure/ design/ implementation of the item being tested is known to the tester. White box testing needs to cover all possible paths of a program as much as possible.

Optimization is the process of making something better. In engineering, various optimization algorithms have been developed and well used in all respects for a long time. An engineer or a scientist asks about a new idea and optimization improves on that idea. Optimization consists in trying variations on an initial concept and using the information gained to improve on the idea [3].

For optimization we can apply various algorithms such as: Genetic algorithm, simulated annealing, ant colony optimization, particle swarm optimization and tabu search. To produce results more accurate we can also combine the algorithms mentioned above. Here we take genetic algorithm. Researchers introduce some algorithms which have better effect on search, such as simulated Annealing algorithm and Ant Colony Optimization and tabu search algorithm. As a result, they produce some new
algorithms such as Mixed annealing genetic algorithm, Hybrid genetic ant colony algorithm and Genetic tabu search algorithm.

In test field, researchers have already used Mixed annealing genetic algorithm and Hybrid genetic ant colony algorithm to test the result of experimental data.

Genetic tabu search algorithm (GTSA) is created by introducing the genetic algorithm into tabu search algorithm, which takes the both advantages and disadvantages of the two algorithms to improve the result produced and the efficiency.

II. BASIS PATH TESTING

Basis path testing is the oldest structural testing technique. The technique is based on the control structure of the program. Basis path testing is a type of white box testing technique that is used to test the code based on control flow. Basis path testing, a white box testing technique used for designing test cases intended to examine all possible paths of execution at least once. The method uses a control flowchart and a control flow graph to convert the code into a model and then derive independent test paths from it. Basis path testing is a white-box testing technique first proposed by Tom McCabe. Basis path testing was proposed by Thomas McCabe in the eighties of last century. Based on cyclomatic complexity measure to CFG and specific algorithm, independent basis paths can be created and test cases can be design according to these paths.

The method devised by McCabe to carry out basis path testing has four steps. These are:
1) Compute the program graph.
2) Calculate the cyclomatic complexity.
3) Select a basis set of paths.
4) Generate test cases for each of these paths.

The cyclomatic complexity of a strongly connected graph is provided by the formula \( V(G) = e - n + p \). where \( e \) = number of edges, \( n \) = number of nodes and \( p \) = number of connected areas. If we apply this formula to the graph given in Figure 1, the number of linearly independent paths is: \( 11-7+1=5 \)

The five linearly independent paths of our graph are as follows:
- Path 1: A, B, C, G.
- Path 2: A, B, C, B, C, G.
- Path 3: A, B, E, F, G.
- Path 4: A, D, E, F, G.
- Path 5: A, D, F, G.

III. RELATED WORK

Sheng Zhang et al. [1] generated test data automatically based on GA-PSO. Researchers have proposed a hybrid algorithm (GA-PSO) which combines Genetic Algorithm and Particle Swarm Optimization (PSO). The new algorithm is proved effective by a representative test of the "triangle type of discrimination". The experiment shows that the new algorithm has higher performance. Kewen Li et al. [2] introduced particle Swarm Optimization into Genetic algorithm to generate test data automatically. To prove the capability of the GPSMA method, the problems was investigated through the Triangle Classify Program taken as an example. The comparison with ant colony optimization and traditional genetic algorithm shows that the GPSMA is a good alternative for test data generation problems. Pengfei Guo et al. [3] presented three different kinds of the novel enhanced genetic algorithm procedures including the hybrid genetic algorithm, interval genetic algorithm and hybrid interval genetic algorithm respectively.
The interval genetic algorithm and hybrid interval genetic algorithm can avoid calculating system slope in traditional interval analysis and determines the optimum interval range of the parameters under allowable corresponding objective error boundary. Zhang Zhonglin et al. [4] described an improved method of acquiring basis path for software testing. Here, researches tried to avoid selecting infeasible paths from the control flow graph. In order to avoid selecting infeasible basis paths, researchers analyzed the source code before generating the CFG. Imran Hermadi [5] exposed some of challenges posed by path testing, and to analyze what control parameters most affect GA’s performance with respect to path testing. Each and every step in path testing is analyzed based on its complexity. Some challenges posed by path testing are: (1) justifying its use in the first place, (2) adequate target paths generation, (3) guiding heuristic, (4) computational complexity, and recognizing infeasible paths. The disadvantage of path testing is it requires more work than other coverage criteria (e.g., for target path generation, and path traversal to evaluate executed input data). T. K. Wijayasiriwardhanan et al. [6] developed a new tool to generate test cases for basis path testing. Manually generation of test cases was very difficult task. This makes the automated test case generation a need. Algorithm determines a basis set in a flow graph of a program unit. The algorithm starts determining the elements of a basis set from the source node and descends recursively through the flow graph of the program unit. Gentiana Ioana Latiu et al. [7] described a comparison between three important Evolutionary Algorithms which were used for automatic test data generation. Here researchers presented a technique that forces the execution of a desired path of the program called target path. Two new approaches, which are based on Particle Swarm Optimization and Simulated Annealing algorithms, are compared with Genetic Algorithms for generating test data. The metric used in this method calculates the difference between the selected path to be traversed, and the actual path traversed by the input values. Shiveta Parnami et al. [8] described a survey on generation of test data and test cases using Artificial intelligence techniques. The AI areas including Neural Networks, Expert Systems, Automatic Speech Recognition, Genetic Algorithms, Intelligent Agents, Natural Language Processing, Robotics, Logic Programming, and Fuzzy Logic. Test data can be generated by static and dynamic approach. Static approach include domain reduction and symbolic execution, while dynamic includes random test cases, goal oriented approach local search approach, chaining approach and evolutionary approach. Yerimese Suresh and Santanu Ku Rath [9] demonstrated about genetic algorithm approach for test data generation using basis path testing. Test data generation classified into three types: Path wise test data generators, Direct specification generators, random test data generator. When we use these techniques practically these require complex algebraic computation. All paths selected in a module needs to be executed and thus generating a large set of test data for these paths was a difficult task. GA helps to achieve this goal by optimizing test data required to cover the path in a control flow graph.

Shaukat Ali Khan et al. [10] developed a new tool for data flow testing. Here we can apply evolutionary approach. Data flow testing is a type of white box testing technique that uses both flow of data and flow of control through the program for testing purpose. Evolutionary testing selectively generates test data by applying optimization based search techniques. The developed tool generate random initial population of test paths and then based on the selected data flow testing criteria new path were generated by applying Genetic algorithm. Mei Jia [11] improved the results of genetic algorithm for test data generation using Queen-bee evolutionary genetic algorithm (QBEA). In this algorithm, sequences of operators iteratively executes for test cases to evolve to target paths. The best chromosome called queen among the current population is crossover with drones selected according to a certain crossover probability, which enhanced the exploitation of searching global optimum. Genetic algorithm had been used to find automatically a program’s longest or shortest execution time. Minjie Yi [12] combined the genetic algorithm and ant colony algorithm. The genetic algorithm and Ant colony algorithm were used to generate test data, and the both can improve the efficiency of test data generation. Here we will try to improve the results of Genetic algorithm and ant colony algorithm by using Genetic algorithm with tabu search algorithm.

IV. PROBLEM FORMULATED

In practical use, Genetic Algorithm will bring a problem called premature convergence. The premature convergence of a genetic algorithm arises when the genes of some high rated individuals quickly attain to dominate the population, constraining it to converge to a local optimum. In this case, the genetic operators cannot produce any more descendants better that the parents, the algorithm ability to continue the search for better solutions are therefore substantially reduced. In addition, Genetic Algorithm has low search efficiency in later period of revolution. In order to overcome such shortcoming of Genetic Algorithm, researchers introduce some algorithms which have better effect on search, such as Annealing algorithm, Ant Colony Optimization and tabu search algorithm. Here we apply Hybrid genetic tabu algorithm to test the results of experimental data. Hybrid genetic tabu algorithm has a good effect of restraining local convergence and improving the search efficiency.

V. CONCLUSION

Software testing is the checking of code for building confidence of the programmer that shows, the software does what it is intended to do, which in turn improves the consistency of the software. Automation of software testing process helps in achieving it with reduced cost and time. After having gone through the different literatures published on this topic of testing optimization, we propose here a technique Genetic Tabu Search Algorithm (GTSA) algorithm has a good effect of restraining local convergence and improving the search efficiency. GTSA may not be the answer to the approach of software testing, but do
provide an effective strategy. The classical triangle discrimination problem is chose as a test program for analyze the efficiency of GTSA.

The further extension of this work can try to improve the presented algorithm or they can try to use any other algorithm to optimize the results.

REFERENCES