Test Data Automatic Generation Based on Modified Condition/Decision Coverage Criteria

Fan Xin[1,a], Zheng Wei[1],Yang Feng-yu[1],Liang Qi-jun[1]
[1]Department of Software Engineering
School of Software, Nanchang HangKong University
Nanchang, 330063, China
[a]fanxinzh2@126.com

***Abstract*—Software testing is one of the most important means that guarantee software quality and reliability. Meanwhile, improving the automation level of testing is also very important to ensure software development quality and decrease software development cost. DO-178B provides the different criteria of structure coverage for different levels of software. This paper presents a test data automatic generation method based on genetic algorithm. This approach builds the decision tree from truth table to extract the minimum data set according to modified condition/decision coverage criteria, converts the problem of testing case data generation to another problem, which is searching the optimized solution from program code by applying the genetic algorithm. At last, the analysis of experiments proves the effectiveness of the approach.**

Keywords—modified condition/decision coverage; minimum data set; genetic algorithm; test data generation

I. INTRODUCTION

DO-178b[1] plays a critical role in airborne software development and testing. According to the safety assessment and hazard classification, DO-178b categorized the aviation airborne software into five levels, Level A-E. For the most critical (Level A) software, modified condition/decision coverage (MC/DC) criterion[2] is highly recommended to ensure adequate testing. However, it is troublesome for the software testing engineer to find the every condition in a decision that is shown to independently affect the outcome of the decision during manual testing. The paper proposes an algorithm of extracting minimum data set satisfying MC/DC criteria by analysis of the program source code, presents the approach to build the fitness function in genetic algorithm(GA) based on the minimum data set obtained in the previous step, introduces the process to generate initial population, setup crossover and mutation rule in detail.

The paper is further organized as follows. In Section 2 we describe the present study of test data generation together with a review of related work. In Section 3, we introduce the process of extracting minimum data set from decision tree according to MC/DC criteria. The proposed approach of test data generation based on genetic algorithm to satisfy MC/DC criteria is described in detail in Section 4. Section 5 describes the experiments carried out in this study and presents discussion on the result of the experiments. Section 6 concludes the paper with the studies of future work.

This paper is supported by National Defense Basic Scientific Research Program of China(Grant No.A0520132029), Natural Scientific Foundation of JiangXi Province of China (Grant No.20142BAB217017) and Foundation of Education Committee of JiangXi Province of China (Grant No. GJJ12457) .

II. RELATED WORK

Many Researcher has studied the automatic test data generation approach based on metaheuristic algorithm such as hill-climbing algorithm, genetic algorithm, tabu search algorithm[3], quantum particle swarm optimization[4] etc. Three approaches were introduced for ordering branches for selection as targets for coverage with a genetic algorithm as in [5]. In [6], the issue of test data generation with program-based criteria is addressed, and test data generation using metaheuristic techniques are introduced particularly satisfying branching coverage criterion. In [7], to deal with the problems existing in PSO multi-path covering optimization on software testing, Multi-path oriented PSO automatic test case generation algorithm was proposed. In [8] an experiment is conducted on the scalability of search-based software testing with the final conclusion that GA can outperform random testing as complexity, in terms of number of branches and statements, is increased. Meanwhile, many other studies has focused on test data generation satisfying MC/DC adequacy criteria. In [9], the author studies syntactic tree of combination of conditions and decision, and what is influence of decision of the left and right branches of the syntactic tree, and an algorithm is presented to rapidly generate minimum test case.

III. GENERATING DATA SET BASED ON MC/DC

Test adequacy criteria is often used to ensure the structural testing (i.e. white box testing). The general test criteria are Statement coverage(SC), Decision coverage(DC), Condition coverage(CC), and Condition/decision coverage(C/DC), but MC/DC criterion is most stronger among all the criteria mentioned above[10]. In this section. We introduce an searching algorithm to extract minimum data set satisfy MC/DC criteria from decision tree which is build to represent the logical calculations of all the condition in a decision.

A. Definition of MC/DC

The modified condition/decision coverage (MC/DC)[11] is a code coverage criterion that requires all of the below during testing.

- Each entry and exit point is invoked
- Each decision tries every possible outcome

- Each condition in a decision takes on every possible outcome
- Each condition in a decision is shown to independently affect the outcome of the decision

B. Truth Table

A truth table is a mathematical table used in logic to compute the functional values of logical expressions on each combination of values taken by their logical variables. Truth table is the basis to analyze the data set which can satisfy MC/DC criterion. For instance, the truth table of logic expression A and B and C is shown in Table 1. According to the rule of MC/DC criteria, each condition in decision should affect the whole outcome of the decision independently. The final data set satisfying the criteria includes the following four data groups: $< A, B, C >, < A, B, \overline{C} >, < A, \overline{B}, C >, < \overline{A}, B, C >$

TABLE I. TRUTH TABLE OF DECISION : A AND B AND C

	Value of A, B, C	Value of A and B and C	Included In MC/DC Data Set ?(Y/N)
1	(T, T , T)	T	Y
2	(T, T , F)	F	Y
3	(T, F , T)	F	Y
4	(F, T , T)	F	Y
5	(F, F , F)	F	N
6	(F, F , T)	F	N
7	(F, T , F)	F	N
8	(T, F , F)	F	N

C. Build Decision Tree Based on Decision Tree

According to the truth table described above, we find that when one condition affects the whole outcome of the decision, the logical values of other conditions are almost the same. If we put the situation which one condition is true into the left child of the tree, put the situation which one condition is false into the right child of the tree, a decision tree based on truth table can be built . The steps for building a decision tree are listed as below.

Step 1: Assuming that the decision is $c_1\ op_1\ c_2\ op_2\ c_3\ \ldots\ op_{n-1}\ c_n$, and the conditions are c_1, c_2, $c_3 \ldots$, c_n, logical operators are op_1, op_2, $\ldots\ op_{n-1}$, computes the whole decision expression and produce the truth table T.

Step 2: Retrieves the data groups in truth table T one by one.If data group $< c_1, c_2, c_3 \ldots, c_n >$ is the i*th* group, set R as the root, if c_1 is true, set $c_1$as the left child of R , else set c_1 as the right child of R, if c_2 is true, set c_2 as the left child of c_1 , else set c_2 as the right child of c_1 , repeats the same process from c_2 to c_n,

Step 3: Repeats the building steps from first group to the last data group in truth table T. and mark the leaf node in decision tree with the final outcome of data group $< c_1, c_2, c_3 \ldots, c_n >$, After all the steps mentioned above are done , the whole decision tree is built.

If the count of conditions in one decision is n, the whole truth table should have 2^n data groups. After converting the truth table into a decision tree, the depth of the tree should be a full binary tree with the depth $n+1$ and nodes $2^{n+1}-1$. Assuming the decision expression is A and B and C, The whole building steps are illustrated in Fig. 1.

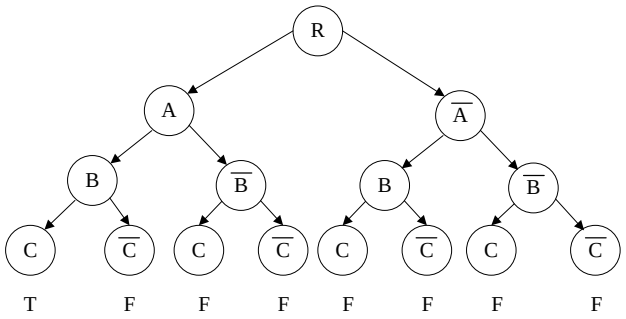

Fig. 1. Building a decision tree

D. Algorithm of Extracting data set from Decision Tree

After the decision tree has been built based on truth table, we design an algorithm to extract minimum data set from decision tree , which is derived from depth first searching algorithm. If the decision tree is represented by T, the depth of T is n, the extracting algorithm is described as below.

Algorithm 1 Extracting minimum data set of MC/DC

Input: the decision tree T built in section C

Ouput: the minimum data set S satisfying MC/DC

```
Begin
 1: S' = Depth_Traverse(T, n-1)
 2: for each last node node_{n-1} in sequence from S' do
 3:   if ( ( lchild(T, node_{n-1}) == true and rchild(T, node_{n-1})== flase )
 4:      Or (lchild(T, node_{n-1}) == false and rchild(T, node_{n-1})== true) ) then
 5:        node_n = lchild(T, node_{n-1})
 6:        node'_n = rchild(T, node_{n-1})
 7:        add sequence L_node1, L_node2,..., L_node n-1, L_node n to S
 8:        add sequence L_node1, L_node2,..., L_node n-1, L_node' n to S
 9:   end if
10: end for
11: for each sequence in S   do
12:   for node_i from n-1 to 1 step -1  do
13:      if ( finalvalue(L_node1, L_node2,...,L_nodei,..., L_node n-1, L_node n) and
14:         finalvalue(L_node1, L_node2,...,brother(T,L_nodei),..., L_node n-1, L_node n)) =
15:          false and sequence L_node1, L_node2,...,brother(T,L_nodei),..., L_node n-1, L_node
                n)
16:          not In S  then
17:          add sequence L_node1, L_node2,...,Brother(T,L_nodei),..., L_node n-1, L_node n) to
S
18:      end if
19:   end for
20: end for
end
```

In algorithm 1, procedure lchild and rchild can judge whether the logical value of childs of $node_{n-1}$ in decision tree T is true or false. Procedure brother can return the brother node

of $node_i$. Procedure finalvalue can return the whole outcome of the decision expression.

IV. APPLYING GENETIC ALGORITHM

Genetic algorithm(GA) is a model of simulating biological genetics and natural selection to search the optimized solution. In a genetic algorithm, a population of candidate solutions (called individuals) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution process often starts from a population of randomly generated individuals. The objective function is created to calculate the fitness of every individual in the population. The more fit individuals are eventually selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

In this section, we introduce the approach to apply GA to generate test data satisfy MC/DC criteria. The main difficulty is how to convert the problem of test data generation into an optimization searching problem.

A. Fitness Function

When minimum data set have been created in the above sections , many conditions are contained in one data group such a>b or a<b etc. If the objective is to generate the value of a and b which satisfy the relation a>b, the branch distance are defined to show the distance between the value a and b . When the acutal value of a and b satisfies the less than relation, the branch distance is 0. Korel defined the computation of branch distance for different conditions(shown in Table 2). These are basics to create the fitness function.

TABLE II. BRANCH DISTANCE COMPUTATION

NO.	*Decision to make*	*Branch distance*
1	a<=b	a-b
2	a>=b	b-a
3	a= =b	abs(a-b)
4	a!=b	abs(a-b)
5	a&&b	a+b
6	a||b	min(a,b)

When the value of a and b satisfies the logic relation , the fitness function is 0. When the value of a and b don't satisfy the relation , fitness function takes the branch distance as the fitness value. In the paper, the fitness function is described as Equation (1).

$$fit(x) = \frac{1}{1+f(x_1)} + \frac{1}{1+f(x_2)} + \frac{1}{1+f(x_3)} + ... \quad (1)$$

$f(x_i)$ is the fitness function for each condition.

B. Encoding Strategy

When apply GA to solve optimization problem, mapping the actual value into individual's genome is a critical process. In this paper , we use binary encoding method to encode the individual. If the range for some parameter is $[x_{min}, x_{max}]$, a binary code with the length L is used to represent the parameter. so we can obtain 2^L different codes. If δ is used to represent the precision, δ can be calculated by Equation (2).

$$\frac{X_{\max} - X_{\min}}{2^L - 1} \quad (2)$$

If individual x is encoded as code : $a_L a_{L-1} a_{L-2} ... a_2 a_1$, we can decode and compute the value of x by Equation (3).

$$X_{\min} + (\sum_{i=1}^{L} a_i \cdot 2^{i-1}) \cdot \frac{X_{\max} - X_{\min}}{2^L - 1} \quad (3)$$

C. Population Replacement Strategy and Other Parameter

1) Selection Strategy: In the following selection process, we use roulette wheel selection. If the fitness value of some individual is f(xi), so the chosen probability is computed as Equation (4).

$$P_s = \frac{f(x_i)}{\sum f(x_i)} \quad (4)$$

2) Crossover Strategy: set the crossover probablity P_C as 0.8.

3) Mutation Strategy: set the mutation probablity P_m as 0.1

4) Population Size: 20,30,40,…100

5) Maximum number of generation: 1000

D. Design GA Algorithm

The detail genetic algorithm is described in Algorithm 2 shown as below.

Algorithm 2 Genetic Algorithm

Input: the minimum MC/DC data set

Ouput: test data set.

Begin

1: randomly generate an initial population
2: **While** (terminal condition is not true)
3: Calculate the fitness *f(x)* for every individual *x* in the population.
4: Initialize the intermediate population to empty for selection.
5: **Repeat begin**
6: Select a pair of parent indivuduals from current population
7: With probablity P_C, crossover the parents to form two children
8: Mutate the two children with probability P_m, and place the new
9: individuals in intermediate population.
10: **end until** (n individuals have been added to intermediate population)
11: Replace individuals in current population with new individuals
12: from intermediate population.
13: **end while**

end

V. EXPERIMENT AND ANALYSIS

This section setups several experiments to validate the effectiveness of the approach mentioned above. We selected 4 cases to generate the minimum data set satisfying MC/DC criteria using algorithm 1 and then generate test data using algorithm 2. the four selected cases are list as below:

case1: A&&B case2: A&&(B||C)

case3: (A||B)&&(C||D) case4: (A&&B)||(C&&D&&E)

TABLE III. TIME ELAPSED IN DIFFERENT APPROACH

	Case	***Elapsed Time in* [9]**	***Elapsed Time in the paper***
1	Case1	0.002 601	0.002 116
2	Case2	0.003 464	0.003 128
3	Case3	0.005 408	0.005 198
4	Case4	0.006 727	0.006 567

In this table, we compare the time used by different approach to get the minimum data set satisfying the MC/DC criteria. It is more faster to obtain the data set in this paper than the method applied in [9].

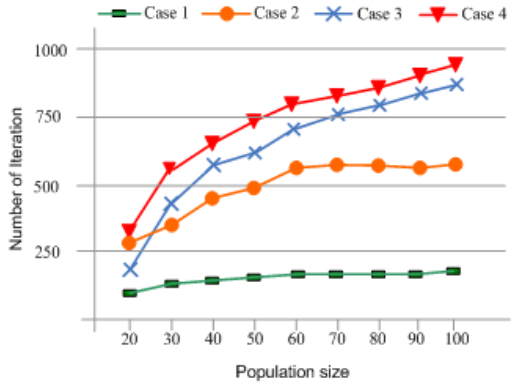

Fig. 2. Plots of population size vs number of iteration

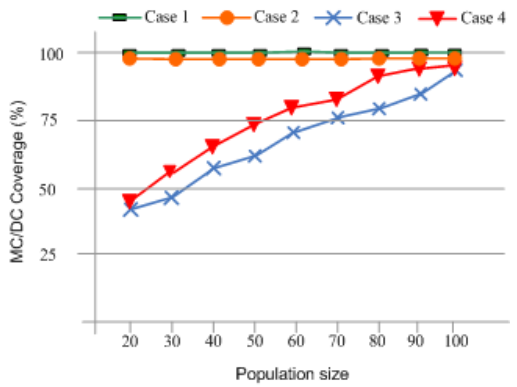

Fig. 3. Plots of population size vs number of iteration

From Fig.2 and Fig 3, we find that the complexity of the decision will influence the generating efficiency by genetic algorithm. But for different cases, the final MC/DC coverage can reach to 100%,which proves the effectiveness of the approach proposed in this paper.

VI. CONCLUSION

This paper presents the detail process to get the minimum data set satisfy MC/DC criteria based on decision tree. The genetic algorithm is also described in the paper to solve the problem of generating test data. After analyzing the result produced by the experiments. The whole approach is proven to be effective. Future work is mainly about improving the extracting algorithm and genetic algorithm to deal with more complicated decision, solving the issue of avoiding falling into local optimal solution while applying genetic algorithm..

REFERENCES

[1] DO-178B/ED-12B. Software considerations in airborne systems and equipment certification. RTCA/EUROCAE, 1992.

[2] NASA. A Practical Tutorial on Modified Condition/Decision Coverage[Z]. Washington D. C., USA: NASA, 2001: 7-9, 73-76.

[3] D áz, E., Tuya, J., Blanco, R.. Automated software testing using a metaheuristic technique based on tabu search. In: Proceedings of the 18th IEEE International Conference on Automated Software Engineering, Montreal, Canada (ASE'03), pp:310–313

[4] Agarwal, K., Srivastava, G. Towards software test data generation using discrete quantum particle swarm optimization. In: Proceedings of the 3rd India Software Engineering Conference (ISEC'10), ACM, New York, NY, USA.

[5] Ankur Pachauri, Gursaran Srivastava. Automated test data generation for branch testing using genetic algorithm: An improved approach using branch ordering, memory and elitism,The Journal of Systems and Software , 2013, 86,pp: 1191–1208

[6] Chen, Y., Zhong, Y., Shi, T., Liu, J., Comparison of Two Fitness Functions for GA-Based Path-Oriented Test Data Generation[J]. In: In Proceedings of the 2009 Fifth International Conference on Natural Computation, vol. 4, Washington, DC, USA (ICNC 09), pp: 177 - 181.

[7] Nie Peng, Geng Ji, Qin Zhi-guang, Multi-path oriented pariticle swarm optimization automatic test case generation algorithm[J]. Computer Integrated Manufacturing Systems. 2012,18(1): 216-223

[8] Mehrmand, A.. A Factorial Experiment on Scalability of Search-based Software Testing. Master's Thesis, Thesis Number: MSE-2009:20, Blekinge Institute of Technology, Sweden.

[9] DUAN Fei-lei, WU Xiao, ZHANG Fan, DONG Yun-wei , Rapidly Generating Algorithm for Minimum Test Case Set on MC/DC, Computer Engineering (China), 2009,17(35):40-43

[10] Chilenski J J. An Investigation of Three Forms of the Modified Condition Decision Coverage (MCDC) Criterion[J]. Computer Programming and Software, 2001, 18(4): 214-219

[11] Chilenski J J. Miller S P. Applicability of Modified Condition/Decision Coverage to Software Testing[J]. Software Engineering Journal, 1994, 9(5): 193-200.