Automatic generation of trusted test cases based on adaptive genetic algorithm

Danyang Wu*, Xuejun Yu
School of Beijing University of Technology, Beijing, China

*Corresponding author: danyangwu@bjut.edu.cn

Abstract. In recent years, the development and operating environment of software system has developed from the traditional closed and static environment to an open and dynamic Internet environment. The software system has become increasingly large and difficult to control, and the emergence of defects and loopholes is inevitable, resulting in the problem of software credibility [1]. How to improve the credibility of software has become the core hot issue in the field of software engineering [2]. In this paper, we will test the credibility of web applications based on the idea of "consistency of words and deeds". By analyzing the characteristics of web applications, define the trusted behavior statement rules of web applications, and combine the genetic algorithm to realize the automatic generation of trusted test cases. Because the basic genetic algorithm has the shortcoming of "premature convergence", in this paper, we will use adaptive parameters to implement genetic algorithm, and through the experimental verification, the adaptive parameter genetic algorithm can effectively improve the efficiency of searching the optimal solution.

Keywords: Software credibility, behavior declaration, genetic algorithm, Adaptive parameters, automatic generation of trusted test cases

1. Introduction
Software credibility refers to the characteristic that the actual operation results of the software always keep consistent with the expected results, and the software application can still work normally under external interference. The research object of this paper is the application software in the web environment. Based on the characteristics that the requests between the web application browser and the server communicate through the HTTP protocol, the credibility verification method of web application is established. The research process of this paper is as follows:

In first part, analyzes the credible requirements and the definition of the behavior statement based on the characteristics of the web application; In second part, designs the branch function instrumentation method and the fitness function calculation method; In third part, analyzes the shortcomings of the traditional genetic algorithm in the generation of test cases, and proposes an optimization plan for adaptive parameters; The fourth part is to verify the optimization efficiency of genetic algorithm through experiments and draw conclusions.
2. Trusted requirement analysis and trusted behavior declaration definition of web application

2.1. Trusted demand analysis
Based on the characteristics of web application, this paper mainly describes its trusted behavior from the following aspects:

(1) Trusted requirements of user access rights: in web applications, users can obtain the corresponding user identities through registration and login, and the system will give different user permissions to different identities. In order to ensure that the login user identity is not maliciously stolen, when a user logs in, the credibility of the logged-in user information is verified by limiting the number of times the user enters the wrong password. If the number of errors exceeds 5 times, it indicates that the identity of the logged-in user is not credible.

(2) Trusted requirements of sensitive resource access: every web application has some private resource information, which may be classified resources or resources related to user name and password, and can not be exposed to all visitors. Only users with specific permissions can access it, so we need to protect this kind of resource information, and trusted verification is required before users want to access it.

(3) Trusted requirements for the execution of sensitive operations: there are many sensitive operations in web applications, such as the button to delete some important files and important information, or operations to upload or download files. Therefore, it is necessary to verify the identity and access rights of the login user before performing sensitive operations to ensure the credibility of the operation process.

(4) Trusted requirements of interface access process: in web applications, the communication and data transfer between browser and server are based on HTTP protocol. Browser can access back-end database resources by accessing interface address and carrying corresponding parameters. Therefore, it is easy to inject malicious statements into interface information in the form of parameters from the input box or other means which affects the credibility of the application. This kind of malicious statement may exist in every statement or parameter, so it is necessary to restrict the credibility of sensitive statements to ensure the credibility of interface access process.

2.2. Definition of trusted behavior declaration
This paper defines the trusted behavior declaration in the form of XML. XML format can show the relationship between data more clearly, and it also has advantages in data expansion. According to the different types of trusted requirements analyzed above, the trusted behavior types in behavior declaration can be divided into four categories: UserAccessPermission, SensitiveInfoAccess, SensitiveOperationExecution, InterfaceAccessProcess. According to different types of trusted requirements, there are the following basic rule subitems when defining behavior declaration:

```
<BehaviorRulesList>  // List of behavior rules
  <BehaviorType>  // Trusted behavior type
    <BehaviorName>***</BehaviorName>  // behavior name
    <BehaviorId>***</BehaviorId>  // Unique identification id of behavior rules
    <OperationPath>***</OperationPath>  // behavior path
    <OperationParams>  // Technical Parameters
      <paramName>***</paramName>
      ......
    </OperationParams>
    <ExpectedResult>***</ExpectedResult> //expected results
    <securityLevel>Safety/Suspicious/Dangerous</securityLevel>//Credibility level
  </BehaviorType>
</BehaviorRulesList>
```
Uploading and downloading files are sensitive operations. Examples of behavior declaration are as follows:

```xml
<SensitiveOperationExecution>
  <BehaviorName>upload_file</BehaviorName>
  <BehaviorId>001</BehaviorId>
  <OperationParams>
    <Authority>admin</Authority>
    <Size>1024</Size>
    <Type>jpg/png</Type>
  </OperationParams>
  <securityLevel>Safety</securityLevel>
</SensitiveOperationExecution>

<SensitiveOperationExecution>
  <BehaviorName>upload_file</BehaviorName>
  <BehaviorId>002</BehaviorId>
  <OperationParams>
    <Authority>Admin</Authority>
    <Size>2048</Size>
    <Type>jpg/png</Type>
  </OperationParams>
  <securityLevel>Dangerous</securityLevel>
</SensitiveOperationExecution>
```

3. Behavior declaration analysis and fitness function design

The trusted behavior statement is defined by using the xml file. In this paper, we use dom4j to parse the behavior declaration. Parsing the uploaded trusted behavior declaration file mainly includes two parts:

3.1. Behavior declaration parsing and equivalence class partition

(1) Extract behavior parameters: extract the behavior name, technical parameters, security level, expected results and other important information defined in the behavior declaration rules to prepare for the initialization of genetic algorithm.

(2) Partition equivalence class: in order to ensure that the generated trusted test cases can cover all situations, the concept of equivalence class needs to be referenced. In the behavior declaration, the behavior parameters of each path node corresponds to valid equivalence class and invalid equivalence class. According to the behavior parameters and security level defined in the behavior declaration, equivalence class is divided. When the safety level is security, the value of the behavior parameter corresponds to the valid equivalence class. When the safety level is suspicious or dangerous, where the corresponding behavior parameter is different from the corresponding parameter value when the security level is safe, it is regarded as an invalid equivalence class. At the same time, after the equivalence class is divided and before the genetic algorithm is coded, the user can also input some valid or invalid equivalence classes that are not defined in the behavior statement. According to the behavior declaration analysis result defined in the previous chapter and user input, the analysis result is shown in table 1. And the equivalence class is divided, and the results is shown in table 2.

| Permission | BehaviorName | Trigger Parameters | Trigger Value | Security Level |
|------------|--------------|--------------------|---------------|----------------|
| Admin      | Upload file  | type               | jpg/png       | Safety         |
| Admin      | Upload file  | type               | exe           | Dangerous      |
| Admin      | Upload file  | size               | \leq1024      | Safety         |
| Admin      | Upload file  | size               | >1024          | Dangerous      |
Table 2. Equivalence class division

| Permission | Behavior Name | Parameters | Effective equivalence class | Invalid equivalence class |
|------------|---------------|------------|----------------------------|--------------------------|
| Admin      | Upload file   | type       | jpg/png                    | exe                      |
| Admin      | Upload file   | size       | ≤1024                      | >1024                    |

3.2. Program instrumentation and fitness function design

In order to generate test cases automatically, it is necessary to insert the source program, so as to understand the running state of the source program under test and obtain the key information of the program under test in real time [3]. This paper uses Korel [4] branch function insertion method to insert the program under test. The branch function needs to insert the branch function $f_i = (x_1, x_2, \ldots, x_p)$ ($p$ is the number of parameters) before the judgment statement of program execution path. The rules for writing branch functions are shown in table 3.

Table 3. Branch function definition rules

| Branch predicate | Branch function |
|------------------|-----------------|
| $V_1 < V_2 \text{ OR } V_1 \leq V_2$ | $V_1 - V_2$ |
| $V_1 > V_2 \text{ OR } V_1 \geq V_2$ | $V_2 - V_1$ |
| $V_1 = V_2 \text{ OR } V_1 \neq V_2$ | $|V_1 - V_2|$ |
| $V_1 \text{ AND } V_2$ | $\text{Max}(V_1, V_2)$ |
| $V_1 \text{ OR } V_2$ | $\text{Min}(V_1, V_2)$ |

When the value of branching function $f_i$ is 0, the branching predicate is true. In this paper, the key to judging the quality of test cases in this paper is the coverage of the related paths of their coverage behavior declaration. According to the definition rules of the branch function, when the values of the branch function are all 0, it indicates that the path is fully covered, which is the best test case we want to find.

The design of fitness function will affect the speed of finding the optimal solution and the quality of solution. By designing a reasonable fitness function, the population can be guided to the optimal direction of evolution.

We need to insert the branch function before each branch statement and put the fitness function of the target path at the end of the program. Assuming that there are $n$ branches on the target path and the number of parameters of each branch is $p$, the calculation expression of fitness function is as follows:

$$F(x_1, x_2, \ldots, x_p) = \frac{1}{\sum_{i=1}^{n} f_i(x_1, x_2, \ldots, x_p)}$$

(1)

The specific implementation of program insertion and fitness function design is as follows:

```c
double test(int x, int y, int z) {
    int k=0; j=0;
    F1 = max(x<25, 10-y); // Branching function F1
    if ((x<25) && (y>10)) {
        k=x*y-1;
        j=sqrt(k);
    }
    F2 = min(10-y, abs(5-y)); // Branching function F2
    if(y>10|(x<5)) |
        j=x*y=10;
    }
    j=y|z;
    if(k<10) F1=0;
    if(k<20) F2=0;
    F=1/F1+1/F2;
    return F;  // Fitness function F
}
```

Figure 1. Example 1.
3.3. Parameter coding
In this paper, we encode the result of behavior declaration parsing in binary mode [5]. The steps are as follows:

1. Get the behavior parameter list of a behavior and the number of parameters n;
2. Obtain the value list of a behavior parameter and the number of value types m;
3. Calculate the smallest multiple of 2 that is greater than the number of parameter values m, and assign the multiple value to N, then the value of N is the code length of the parameter;
4. Code the parameter list value from 0 in decimal, and allocate the unmatched codes from the beginning until all 2N codes are assigned to the corresponding parameter values;
5. Convert the decimal code of the parameter to binary, and each binary parameter code value corresponds to the gene value on the individual chromosome.
6. Finally, the coding results of multiple parameters are arranged and combined in order to generate chromosomes;

\[
\text{Chromosome coding} = \sum_{i=1}^{n} \text{Gene coding}
\]  

(2)

4. Design of genetic algorithm based on adaptive parameters
Genetic algorithm (GA) is a method to search for the optimal solution by simulating the natural evolution process [6]. This paper will be based on the trusted behavior statement combines the genetic algorithm to automatically generate trusted test cases, and its basic structure is as follows:

![Diagram of Genetic Algorithm Process](image-url)

**Figure 2.** Genetic algorithm process.
4.1. Shortcomings of traditional genetic algorithm
Although genetic algorithm is often used to find the optimal solution, as a search algorithm based on natural selection, genetic algorithm has some shortcomings, such as premature convergence and low convergence speed in the later stages, and the traditional genetic algorithm uses fixed crossover rate and mutation rate to realize evolution, which will cause the problem of higher and higher degree of convergence and lower population diversity in the later stage of evolution. It is not conducive to global optimization.

4.2. Improvement strategy of genetic algorithm
Through the understanding and application of the basic genetic algorithm, we can know that the key factor that determines the performance of the genetic algorithm lies in its parameter configuration method. Usually, when configuring the parameters, the final parameters are determined based on empirical values or several groups of comparative experiments, and once these values are set, they do not change. The idea of adaptive genetic algorithm is to adjust these parameters adaptively based on the state of the algorithm, so that the best parameter values can be used at any specific moment in the execution process, so as to improve the performance of the algorithm. The adaptive parameters in the genetic algorithm mainly refer to the mutation rate and the crossover rate. This paper calculates and updates these parameter values through the average population fitness and the current best fitness of the population.

4.2.1. Population initialization. Population initialization requires us to determine the size of the initial population and generate the corresponding number of initial individuals. In this paper, through analysis of behavior statement parsing results; Determines the length of each gene according to the number of values of each parameter, and according to the type of parameter determine the length of the chromosome. After knowing the length of the chromosome, randomly initialize some binary strings as the initial population.

4.2.2. Selection operation. The selection operation is the operation of selecting individuals to enter the next-generation population based on the calculation results of individual fitness. The higher the fitness, the greater the probability of being selected, and vice versa. This paper uses roulette [7] to realize the selection operation. Assuming that the size of the population is M and the fitness value of an individual is $F_i$, the probability of it being selected is:

$$P = \frac{F_i}{\sum_{i=1}^{M} F_i}$$  (3)

In addition to using "roulette" for selection, this paper also combines the idea of "elitism" when implementing the selection operation, preserving the best individuals in each generation, and directly assigning values to the next generation population without genetic operation. The size of the population determines the number of elite individuals. When the population size is 100, set the number of elite individuals to 2~5.

4.2.3. Crossover and mutation operation of adaptive parameters
In order to avoid the "premature convergence" defect of genetic algorithm, this paper uses adaptive parameters when performing crossover and mutation operations, using the average fitness value and the best fitness value of the population to calculate and update the crossover rate and mutation rate. Among them, $P_c$ is the crossover rate and $P_m$ is the mutation rate. The calculation formula is as follows:

$$P_c = \begin{cases} \frac{f_{\text{max}} - f_i}{f_{\text{max}} - f_{\text{avg}}} \times c, & f_i > f_{\text{avg}} \\ c, & f_i \leq f_{\text{avg}} \end{cases}$$  (4)
\[
P_m = \begin{cases} \frac{f_{\text{max}} - f_i}{f_{\text{max}} - f_{\text{avg}}} \times m, & f_i > f_{\text{avg}} \\ m, & f_i \leq f_{\text{avg}} \end{cases}
\]

\(^{\text{1}}\) \(f_i\) represents the individual fitness value, \(f_{\text{max}}\) represents the optimal fitness value of the population, \(f_{\text{avg}}\) represents the average fitness value of the population, \(c\) is the initial population crossover rate, and \(m\) is the initial population mutation rate.

\(^{\text{2}}\) When \(f_i > f_{\text{avg}}\), that is, the individual fitness value is greater than the average fitness value, the crossover and mutation rate can be reduced by multiplying the previous parameter value to retain good chromosome genes; on the contrary, when \(f_i \leq f_{\text{avg}}\), the initial crossover and mutation rate will continue to be used to increase the probability of crossover and mutation, so as to ensure the diversity of the population and avoid premature convergence of the population.

5. Experimental verification

5.1. Experiment

In order to verify the timeliness of the improved genetic algorithm in the automatic generation of test cases, program instrumentation and fitness function design are carried out through the two programs of Example 1 and Example 2, and the genetic algorithm is used to generate test cases covering the entire path. The parameters are set as follows: Example 1 (population size: 50, chromosome length: 21, initial crossover probability: 0.9, initial mutation probability: 0.1, number of elite individuals: 2, value range of each parameter in the function: \([0, 127]\)). Example 2 (population size: 60, chromosome length: 21, initial crossover train: 0.9, initial mutation probability: 0.1, number of elite individuals: 2, value range of each parameter in the function: \([1, 128]\)); both the maximum evolution algebra is not set. Each group carries out 100 experiments, and the average running results are shown in Table 1, and the program code of Example 2 is as follows:

```
double TriangleType(double x, double y, double z){
    double f1=10, f2=10, f3=10, f4=10, F;
    F1=\(z-x+y\); //Branch function F1
    if(x+y>z){
        f2=x-(y+z); // Branch function F2
        if(y+z>x){
            f3=y-(x+z); // Branch function F3
            if(x+z>y){
                F4=min(min(abs(y+y+z-x-\(x\)),abs(x+\(x\)+z-y-\(y\))),
                \(abs(x+y+y-z)\)); // Branch function F4
                if((x=x+y+z2)\(y+y=x=x+z2)\(z=x=x+y))\{result= Right triangle \};
            }
            f1=0;
        }
        if(f3<0) f2=0;
    }
    f1=1/(f1+f2+f3+f4); //Fitness function F
    return F;
}
```

Figure 3. Example 2

The value of fitness function \(f\) is passed to the genetic algorithm package. When the value of branch function is 0, it shows that the use case satisfies all branch conditions and can cover the whole path branch. That is, when \(F\) is infinite, it satisfies the termination condition and ends the algorithm.
Table 4. Comparison of search results

| Experimental indicators                  | Example 2 | Example 1 |
|------------------------------------------|-----------|-----------|
|                                          | Average operation algebra | Average running time/s | Average operation algebra | Average running time/s |
| Adaptive parameter genetic algorithm     | 268       | 12        | 7          | 2          |
| Traditional genetic algorithm            | 897       | 15        | 21         | 6          |

Through comparative experiments, we find that the genetic algorithm using adaptive parameters can improve the evolution speed of the population and shorten the evolution time, and the more branches in the program, the more obvious the efficiency of the improved algorithm.

5.2. Conclusion
By analyzing the characteristics of web applications, this paper classifies the trusted requirements in web applications, describes the definition method of trusted behavior declaration, analyzes the trusted behavior declaration and divides the equivalence classes, and encodes the parameters of the analysis results for the initialization of genetic algorithm. Later, based on the basic genetic algorithm, this paper uses the parameter adaptive method to modify it. Finally, two groups of examples are used for comparative experiments to verify that the time to find the optimal solution and the operation efficiency of the genetic algorithm based on parameter adaptive are significantly improved.

References
[1] Liu Ke, Shan Zhiguang, Wang Ji, et al. Overview of major research projects for basic research on trusted software. National Science Foundation of China, 2008(3): 145-151.
[2] Allberto G. Automatic generation of cost-effective test oracles. ICSE2014, 36th International Conference on Software Engineering, Hyderabad, India, 2014.
[3] Xiaoyuan Xie’Baowen Xu’Liang Shi. A Dynamic Optimazation Strategr for Evolutionary Testing, 12th Asia-Pacific Software Engineering Conference (APSE’05), 2005.
[4] Korel B. Automated Software Test Data Generation [J]. Software Engineering IEEE Transactions on, 1990, 16(8): 870-879.
[5] Liu Miaochen. Research and application of REST-style software credibility testing based on behavioral statement [D]. Beijing University of Technology, 2017.
[6] Li Minqiang, Kou Jisong, Lin Dan, etc. The basic principles and application of genetic algorithm. Beijing: Science Press, 2002.
[7] Xia Jiamei, Zeng Jianchao. A random particle swarm algorithm based on genetic algorithm of roulette selection. Computer Engineering and Science, 2007, 26(9): 51–53.
[8] Fu Xuhui, Kang Ling. Research on the precocity of genetic algorithm. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2003, 31(1): 53-55.