Research on Automatic Generation of Abnormal Unit Tests Based on Genetic Algorithm and Log Analysis

YuHao Liu[1,a]

[1]*School of Software Engineering, A National Pilot Software College, Beijing University of Technology Beijing, China*
a. liuyuhao1203@163.com

Keywords: Code robustness, log analysis, abnormal unit testing, genetic algorithm, test data generation.

Abstract: In order to improve the robustness of commercial software code, an automatic generation method of abnormal unit tests based on genetic algorithm and log analysis is proposed. Intelligently analyze the past program logs, select parameters with static parameter values and continuous parameter values, and construct a parameter database as the initial value and mutation value of the genetic algorithm. And use a reasonable fitness function and use genetic algorithms to generate test cases. Experimental results show that this method is superior in terms of code function coverage and recall of abnormal problems.

1. Introduction

This topic is based on the actual project of the enterprise, and the technical background of the enterprise commercial software is to solve the online problem of the commercial product, namely the robustness problem of the system code, with high efficiency and high quality. With the continuous and rapid iteration of commercial product functions and the continuous increase of business functions, most developers develop code to meet business functions as a premise, resulting in poor code robustness of online systems, which makes business functions contrary to actual goals. According to statistics, in the second quarter of 2019, 18 of the 37 problems that appeared on a Baidu commercial product line online were abnormal capture and processing of abnormal data, which had a significant impact on online revenue. Code robustness needs to be addressed.

Software testing is the key method and necessary link to ensure the reliability and stability of software. The design and selection of test cases will directly affect the quality of software testing[1]. Exception testing is one of the testing methods to improve the robustness of system code. Anomaly testing can be broadly divided into four categories, stress-based, functional testing, unit testing, and static code scanning exception testing methods. Through comparison, it is found that in addition to labor costs, the recall, location, and closed-loop capabilities are the best. Other methods have a certain lack of exception coverage, which cannot effectively intercept abnormal code problems, so that online stability problems still occur frequently. Therefore, this article uses the method of abnormal unit testing to solve the current problem. In order to intelligently select test cases with high coverage, improve test quality, reduce test time and labor consumption, experts and scholars have conducted a lot of research. Reference[2] uses genetic algorithms in object-oriented class test case generation, and has significantly improved the branch coverage of the methods in the class and the number of test case generation. Reference[3] improved genetic algorithms based on increasing branch coverage, making the improved method superior in coverage and evolutionary algebra. This paper proposes a genetic algorithm combined with log analysis. The main idea is to use the parameter information in the log to build a parameter database. Statistics are performed on fixed-value, continuous-value, and discrete-value data to provide a reference for the initial generation and mutation operation of the genetic algorithm. It can quickly generate high-quality test cases, and experiments prove the effectiveness and efficiency of the algorithm.

2. An Introduction to the Automatic Generation of Abnormal Unit Test

2.1. Intelligent Log Analysis Technology

There are often a lot of branch judgments in the code of business function modules to meet the requirements of covering different business scenarios, such as condition judgment if (a == 1 && b <c), but in the automatically generated test cases, (a == 1) The condition is often not easily met, so it cannot cover the branch. Similar situations include fixed string values. Since the online system of commercial products prints a special business log every day according to the real user request, the log will include the parameter names, parameter values, and other indicators. Based on this, this paper proposes an improved genetic algorithm idea, intelligently analyzes the log content, obtains target parameter information, and constructs a parameter database to provide a reference for the initial value and variation value of the genetic algorithm.

2.1.1. Dynamic Estimation Parameter Values

Offline analysis of the historical data of each target parameter in the business log, found that in historical tasks, the value types of numerical parameters can be roughly divided into fixed values, discrete values, continuous values, and stable values. Parameters of fixed value types do not need to be processed, and are directly assigned as the final predicted values. The parameter value of the parameter statistics of the discrete value type has the highest occurrence frequency as the final predicted value. The continuous value type parameter gives the parameter threshold value according to the maximum and minimum values of the historical parameter, and then randomly selects one as the final predicted value. The largest type of parameter is the parameter of the stable value type. The stable value parameters are in historical tasks, and most of them show a stable trend. However, when factors such as functional iteration or flow have a large influence on the parameters, the curve An inflection point will appear, and the parameter value after the inflection point will stabilize again, as shown in figure 1 below.

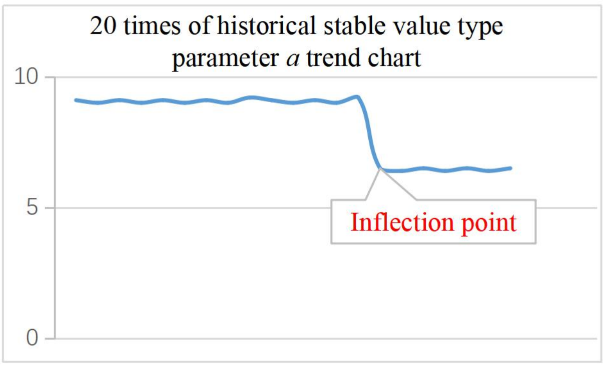


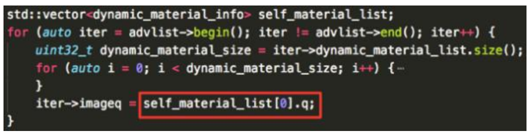
Figure 1: 20 times of historical stable value type parameter a trend.

Chart The change trend of parameter a can be seen in the figure, where a function iteration occurs at the inflection point, which causes the parameter value to decrease from 9 to 6.5. For such parameters, this paper uses the exponential smoothing method to predict the parameter values. The exponential smoothing method is a special weighted moving average method. Its characteristics are: First, the exponential smoothing method further strengthens the effect of recent observations on the predicted value during the observation period, and the weights given to observations at different times are different, thereby increasing the weight of recent observations, Forecasts can quickly reflect actual market changes. The weights are reduced in equal series. The first term of this series is the smoothing constant a, and the common ratio is (1-a). Second, the exponential smoothing method has scalability for the weights given to the observations. You can take different values of a to change the rate of change of the weights. If a is a small value, the weight changes more quickly, and the recent change trend of the observed value is more quickly reflected in the exponential moving average. Therefore, using the exponential smoothing method, you can choose different values of a to adjust the uniformity of the time series observations (that is the smoothness of the trend change). According to the different smoothing times, the exponential smoothing method is divided into:one exponential smoothing method, two exponential smoothing methods and three exponential smoothing methods. This article adopts the exponential smoothing method and its calculation formula is as follows:

$$St = a * yt + (1 - a)St - 1 \tag{1}$$

Among them, St is a smoothed value of time t, St-1 is a smoothed value of time t-1; a is a smoothing constant, which ranges from [0, 1]; yt is the actual value of time t. According to the exponential smoothing resume model, the parameter value of the stable value type is predicted as the final parameter value. In this way, parameters of fixed value, discrete value, continuous value, and stable value type can all obtain predicted values based on historical logs.

In addition, because traditional genetic algorithms cannot solve the automatic generation of non-numerical data, literature[4] proposed a method for automatically generating non-numerical software test data based on genetic algorithms. Operator to automatically select path fitness functions or character fitness functions to guide non-numeric software test data generation for a given path. In this article, because the parameter naming of commercial products is very standardized, the type and meaning of the parameter can be basically determined by the parameter name. For parameters with non-numeric values, such as strings, you can analyze the parameter value through logs. Use the parameter values analyzed by the log. The default initial value (predicted value) of this article that cannot be obtained through log analysis is: "Test 123 @", which contains Chinese and English numbers and special symbols. For another example, the parameter is an array. Most developers will ignore the out-of-bounds problem of arrays, as shown in figure 2:



```
std::vector<dynamic_material_info> self_material_list;
for (auto iter = advlist->begin(); iter != advlist->end(); iter++) {
    uint32_t dynamic_material_size = iter->dynamic_material_list.size();
    for (auto i = 0; i < dynamic_material_size; i++) {⋯
    }
    iter->imageq = self_material_list[0].q;
}
```

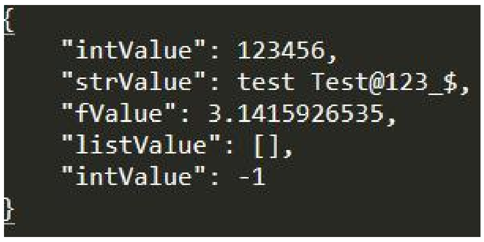
Figure 2: Array out of bounds case.

The normal business logic of the parameter self_material_list is not empty, so the developer ignores the problem of array overflow and directly performs the value operation. The actual situation is that self_material_list is empty due to online data delay, which causes problems. Therefore, this article is based on the actual situation in the industry. In order to improve the robustness of the code, more code exceptions are detected. The parameters are the initial values of the array type (predicted values)is empty.

2.1.2. Building a Parameter Database

Based on the standardization of the parameter naming of industrial products, and the parameter naming of the same role in different programs is the same, a parameter database is constructed, and the database is a MySQL database. MySQL database is fast, reliable and adaptable. It is easy to use and open source. Therefore, it is a database with high usage of lightweight software in the industry. The amount of data stored in this article is in the thousands. The MySQL database can be fully qualified. . The database design is: (1). Increment id (primary key); (2). Parameter name, varchar type, 20 characters, cannot be empty;(3). Parameter value, varchar type, 255 characters, can be empty; (4), Parameter type, varchar type, 20 characters, can't be empty; (5). Whether the parameter is available, int, 1 is available, 2 is not available, the default value is 1. Each parameter and parameter value is a piece of data to ensure the real-time nature of the data. The data is updated twice a week, and the log data 20 times before the update is selected as the basis for data selection and prediction. Update the existing parameter values and insert operations for the newly added parameter values. For the convenience of management, this article adds the parameter availability flag field, and rewrites this field to 2 state for temporarily unneeded parameter values.

In use, this article selects the parameter name field as the key, and based on the dynamic parameter prediction value (parameter value field) obtained from log analysis as the value, constructs a key-value pair in the form of key-value, and stores it in the form of a JSON string to construct an offline parameter prediction Value list. As shown in figure 3 below.



```
{
    "intValue": 123456,
    "strValue": test Test@123_$,
    "fValue": 3.1415926535,
    "listValue": [],
    "intValue": -1
}
```

Figure 3: Offline parameter vocabulary.

2.2. Introduction to Genetic Algorithm

The genetic algorithm was created by Professor Holland of the University of Michigan in the United States. It is derived from the theory of biological evolution and simulates the evolutionary laws of "natural selection" and "superiority." , Has a good global search ability[5]. The genetic algorithm is mainly a "production-iteration" process, and the main links are population selection, chromosome crossover, and chromosome mutation. The entire workflow of the genetic algorithm is shown in figure 4 below.

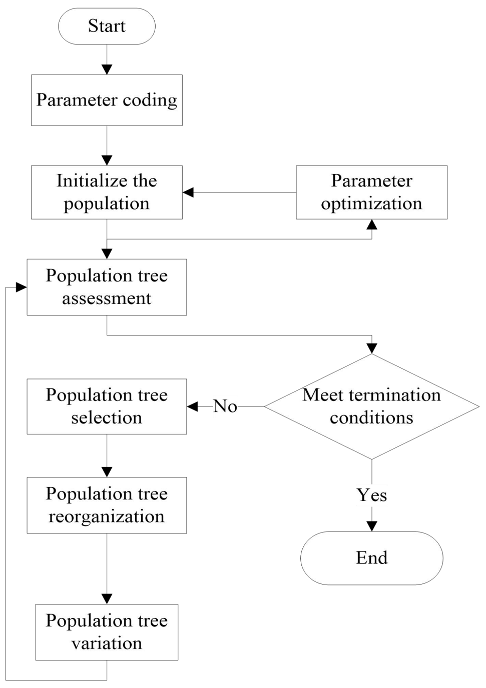

Start
Parameter coding
Initialize the population
Parameter optimization
Population tree assessment
Meet termination conditions
No
Population tree selection
Yes
End
Population tree reorganization
Population tree variation

Figure 4: Genetic Algorithm Workflow.

According to the flowchart, the parameters are encoded based on the data obtained from the intelligent log analysis, and then the population is initialized to determine the maximum number of iterations, population size and other parameters. This paper uses an ordered tree method to represent the parameter sequence. The initialized population is each tree individual[2]. After evaluating the population tree to see if the search conditions are met, the termination conditions are: 1. The predetermined maximum number of iterations is reached and the end algorithm. 2. Find the optimal solution within a predetermined number of iterations and end the algorithm. If the search is satisfied, the search is terminated to obtain the optimal solution. Otherwise, the population selection, recombination, and mutation operations are performed, and the above operations are repeated until the search conditions are met to exit the search.

2.2.1. Parameter Coding Design

Coding In order to make the algorithm have better search ability, maintain the diversity of the population, and then convert the parameters of the problem space into the chromosomes of the genetic space, the commonly used encoding methods are: binary encoding, decimal encoding, floating-point encoding, etc. This article needs to be based on The parameter vocabulary is used for parameter optimization, so this paper proposes a special encoding scheme as follows:

<Chromosome> = <parameter name> @ <parameter value sequence>

<Parameter value sequence> = {int | float | double | boolean | string |}

Explanation of parameter encoding:in the course of the algorithm, the code is parsed, and the parameter optimization work and parameter value mutation operation are performed according to the parameter name in the chromosome.

2.2.2. Introduction to Genetic Operators

a) Selection operator: It is to eliminate inferior individuals from the parents' individuals according to certain rules, and retain and inherit the high quality to the next generation. In the selection of individuals, the rules of survival of the fittest and survival of the fittest in the natural reproduction process are observed and simulated.[6] Among them, classic methods include roulette, proportional selection, and random sampling. This article uses the roulette method. First, the formula is used to calculate the probability of the individual being selected. The calculation formula is as follows:

$$\frac{f_i}{\sum_{j=1}^{n} f_i} \qquad (2)$$

P_i represents the probability that the individual is selected, f_i is the fitness value of individual i, and the population size is n.

b) Crossover operator: The two chromosomes in the genetic space are cut, and then the other two chromosomes are spliced according to the defined parameter coding method under the premise of ensuring that the variable types do not conflict.

c) Mutation operator: Randomly select positions on the chromosome, parse the parameter name according to the parameter encoding generation rule, and search in the parameter database. If the parameter has a specific value in the database and is different from the current parameter value, a mutation operation is performed to re-set the parameter Assignment; if the parameter has a specific value in the database but is the same as the current parameter value, no mutation operation is performed; if the parameter is in the specified range in the parameter database, a similar type value is randomly generated within the specified range to Replace the original parameter value; if the parameter is not in the parameter database, you need to randomly generate the same type of value as the parameter to be mutated to replace the original value to complete the mutation operation. Under special conditions, you need to generate within its specified range . As shown in figure 5, the value of parameter A is changed from 3 of type (int) to 6 of type (int).

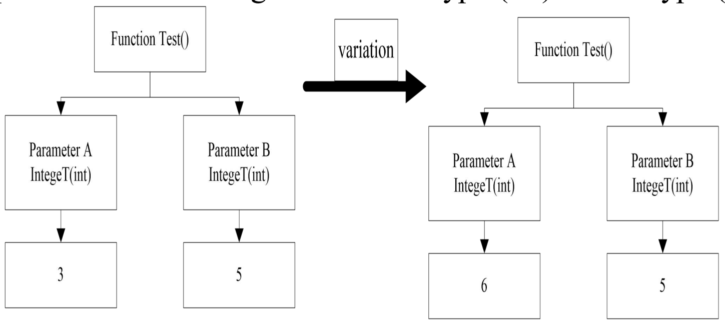


Figure 5: Parameter variation.

3. Experimental Analysis of Abnormal Single Test Generation Method

3.1. Unit Test Code Generation

Paper uses Gtest as the testing framework. Gtest is an open source testing framework for the C ++ language launched by Google. In order to improve the speed of code generation, this paper builds various types of code generation operators, as shown in figure 6 below. According to the different parameter types (String, Pointer, Array, STL family, Struct, Class, etc.), based on the structure of the tested code and test cases Data, bottom-up, quickly generate Gtest framework executable unit- test code.

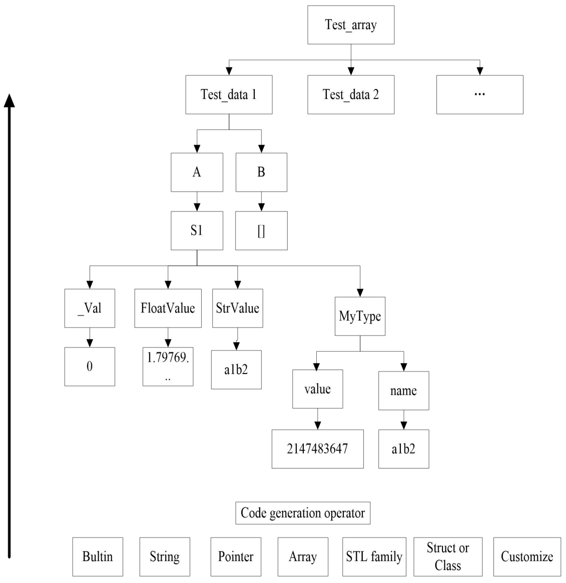
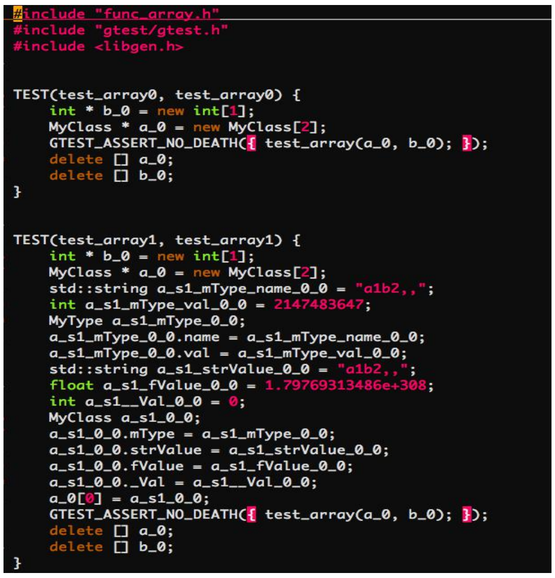


Figure 6: Unit test code generation process.

The automatically generated unit test case code is shown in Figure 7 below:



```
#include "func_array.h"
#include "gtest/gtest.h"
#include <libgen.h>


TEST(test_array0, test_array0) {
    int * b_0 = new int[1];
    MyClass * a_0 = new MyClass[2];
    GTEST_ASSERT_NO_DEATH({ test_array(a_0, b_0); });
    delete [] a_0;
    delete [] b_0;
}


TEST(test_array1, test_array1) {
    int * b_0 = new int[1];
    MyClass * a_0 = new MyClass[2];
    std::string a_s1_mType_name_0_0 = "a1b2,,";
    int a_s1_mType_val_0_0 = 2147483647;
    MyType a_s1_mType_0_0;
    a_s1_mType_0_0.name = a_s1_mType_name_0_0;
    a_s1_mType_0_0.val = a_s1_mType_val_0_0;
    std::string a_s1_strValue_0_0 = "a1b2,,";
    float a_s1_fValue_0_0 = 1.79769313486e+308;
    int a_s1__Val_0_0 = 0;
    MyClass a_s1_0_0;
    a_s1_0_0.mType = a_s1_mType_0_0;
    a_s1_0_0.strValue = a_s1_strValue_0_0;
    a_s1_0_0.fValue = a_s1_fValue_0_0;
    a_s1_0_0._Val = a_s1__Val_0_0;
    a_0[0] = a_s1_0_0;
    GTEST_ASSERT_NO_DEATH({ test_array(a_0, b_0); });
    delete [] a_0;
    delete [] b_0;
}
```

Figure 7: Executable unit test code.

3.2. Analysis of Results

Experiments are performed on the genetic algorithm based on log analysis proposed in this paper. The initial basic parameter settings are: population size 100, crossover probability 0.6, mutation probability 0.1, and maximum number of iterations 400. When the coverage cannot reach a satisfactory value, modify the parameter settings appropriately until you are satisfied.

The experimental data was obtained on an Intel (R) PC running at 2.4GHz and a 4GB memory operating system, Windows 7. The tested code is the real online code of a commercial product. The code structure characteristics are shown in Table 1 below. The test case is judged by code function coverage. The initial test data is based on the data obtained by intelligent log analysis. The experimental results are shown in Table 2:

Table 1: Selected test code.

Test code	Lines of code	Number of constructors
User feedback function code	10456	267

Table 2: Analysis of Results.

Test code	Function coverage	Abnormal recall rate	Number of executions	execution time(s)	Number of cases
User feedback function code	90%	70%	2375	60	21

As can be seen from Table 2, the function coverage of the automatically generated abnormal single test for the tested code reached 90%, and the test cases also include branches that can only be executed under specific parameters. At 70%, most code robustness issues can be exposed. As can be seen from the results given in the table above, the actual results have achieved the expected results. The method of automatic generation of abnormal single test proposed in this paper is an effective method.

4. Concluding Remarks

This paper proposes a method for automatically generating abnormal single test based on log analysis and genetic algorithm. The intelligent analysis of log data to obtain parameter information, combined with genetic algorithm to generate test case data. Experiments show that the method proposed in this paper can make the automatically generated abnormal unit test code reach a function coverage of 90%, the number of use cases is significantly reduced, and the code generation and execution time is within the expected range. High-coverage test cases can more accurately discover the robustness of the code at an earlier stage, avoiding risky code from leaking online and causing loss of commercial products.

This method can meet the requirements for automatic generation of abnormal unit tests of general commercial products. However, due to the complexity of the functional code scenario, the function coverage rate of the code and the recall rate of exception issues are urgent indicators to be improved. How to adapt to more test scenarios, compatible with more code languages, more complex code structures, and more efficient and accurate automatic generation of single test code will be one of the important technologies for future research.

References

[1] Yao Yao. A new method of software test case generation based on genetic algorithm [J]. Computer and Digital Engineering, 2009, 37 (01): 18-21.

[2] Zhou Ruijie, Jiang Guohua. Research on Generating Object-Oriented Test Cases Based on Genetic Algorithms [C]. National Youth Communication Conference, 2010: 151-155.I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Jiang Yuanpeng, Dong Qingjie. Generating method of test data based on genetic algorithm and branch coverage [J]. Computer Engineering and Design, 2016, 37 (1): 112-117.

[4] Research on Automatic Generation Method of Non-numeric Software Test Data Based on Genetic Algorithm [D]. Li Cuicui. Beijing University of Chemical Technology, 2006.
[5] HOLLAND J H.Genrtic algorithms and the optimal allocation of trials[J].SIAMJ Comput,1973,2(2):89-104.
[6] Gao Xuedi, Zhou Lijuan, Zhang Shudong and Liu Haoming. Research on automatic generation of test data based on improved genetic algorithm [J]. Computer Science, 2017, 44 (03): 209-214.