Design Studies and Intelligence Engineering
L.C. Jain et al. (Eds.)

doi:10.3233/FAIA250355

A Study of Automated Software Testing and Optimization Models Based on Behavioral Data

Haoran LI[a,1]
[a]*Beijing Institute of Control Engineering,Beijing 100190,China*

Abstract. In order to solve the problem of low efficiency of traditional software testing algorithms, the research on automated software testing and optimization model based on behavioral data was proposed. The model proposed in this paper uses K-means to initialize EM and adaptively determine the number of clusters. In this process, the clustering results can be evaluated. At the same time, all parameters of the Gaussian mixture model are given. These parameters are used as parameters for a new round of iterative calculation of each cluster, and the final results tend to be the optimal solution. The experimental results show that the reduction rate of the algorithm proposed in this paper is higher than the two algorithms mentioned above, and its total reduction rate is 11.46% higher than that of the software test case set reduction algorithm based on the fuzzy K-Means clustering model, and 6.42% higher than that of the software test case set reduction algorithm based on the Gaussian mixture model. Conclusion: Compared with Gaussian mixture model algorithm and fuzzy K-Means clustering algorithm, the proposed algorithm has higher reduction rate and error detection rate. After reduction, although the scale of software test case set is simplified, the coverage rate is high.

Keywords.software testing,use case approximation,gaussian mixture modeling

1. Introduction

Automated software testing is the process of executing test cases using specialized testing tools or other automated means to verify that the functionality, performance, security and other attributes of the software meet expectations [1]. It aims to reduce manual involvement, improve testing efficiency, and ensure software quality [2]. Compared with manual testing, automated testing has higher execution speed, less error rate and better test coverage [3].

The basic principles and methods of software automation testing. Software automation testing utilizes tools and techniques to perform testing tasks to replace manual operations [4]. It includes aspects such as test case generation, execution and result analysis. Automation testing by writing scripts or using automated testing tools, can realize automated execution of large-scale test cases, improve testing efficiency and accuracy [5]. In software automation testing, common methods include regression testing, functional testing, performance testing, security testing and so on [6]. By building test

[1] Corresponding Author.Haoran LI,Beijing Institute of Control Engineering,E-mail:53157112@qq.com

scripts and test data, automated testing tools can simulate user operations and system responses, automate the execution of test cases and generate test reports [7]. This can reduce manual errors, improve test coverage, and obtain reliable test results in a shorter time [8].

In today's digital era, software has penetrated into every aspect of our lives, from cell phone applications to complex enterprise-level systems, the quality of software is directly related to the user experience and business success [9]. Therefore, ensuring the quality of software has become a core task in the process of software development. Automated software testing technology as an important means to ensure the quality of software, through the simulation of user behavior and system scenarios, can automatically detect defects and vulnerabilities in the software, so as to improve the reliability and stability of the software [10]. However, with the increase in software complexity and the diversification of testing needs, automated software testing is also facing many challenges.

2. Literature review

The larger the size of the software, the higher its logical complexity, and the higher the reliability requirements for such software. The hidden errors of software can cause its own operation failure [11]. That is to say, the hidden error of software is one of the most critical factors affecting the reliability of software. For small-scale software, most of the engineering practices use manual breakpoints to test in order to find out the error. However, the manual method to determine the location of the error is more complex and difficult, and is not applicable to large-scale software testing [12]. Therefore, in order to find and eliminate software errors more accurately and efficiently, experts and scholars have carried out extensive research, put forward a number of solutions, and developed tools for unit, static and dynamic testing of software [13]. These methods and test software, all require the test case set to cover as comprehensive as possible, only in this way can accurately locate errors and improve the effectiveness of testing [14]. However, the number of use cases included in the test case set should be kept as simple as possible in order to reduce the cost of testing.

Many researchers have proposed different methods for test case set reduction. A research team proposed a test case set reduction method based on vector similarity [15]. Another team proposed a method to solve the test case reduction problem by using the search tree, which can find the global optimal solution [16]. Other researchers introduced requirements analysis into test case set reduction and proposed a two-stage optimization method: first, requirements slicing, and then use case set optimization [17]. In addition, a research team proposed an online test case set reduction method, which embedded the test set reduction into the test generation process, and realized the test set reduction through the satisfaction relationship between the test sequence and the test goal [18]. In addition, by defining and combining the equivalent migration and equivalent state generated based on the ASM model, the research has reduced the invalid state and invalid migration path, and realized the reduction of the use case set space [19]. Finally, a test case reduction method for error location requirements is proposed, which not only reduces the complexity of error location, but also improves the accuracy of location [20].

These methods used to simplify the set of test cases also have some major problems as follows: 1) how to model the relationship between the set of test cases to improve the

accuracy of finding software errors; 2) how to achieve the simplification of parameter settings, or to achieve adaptive parameter settings. In order to solve these problems, this paper proposes a model study of automated software testing and optimization based on behavioral data. The algorithm introduces Gaussian mixture model to find the relationship between test cases, extract the test cases that meet the test requirements, and realize the simplicity of the set of use cases; meanwhile, through the adaptive strategy, the parameter settings are simplified.

3. Methodology

3.1 Behavioral Data Software Testing Classification Model

For web-architecture software test data, the feature classification includes the following steps.

1) For the input network software test data, the feature points are extracted, and the set of data feature points is F.

2) According to the feature expansion algorithm, the sparse decomposition algorithm is used to sparse the test data of the network software. The sparse data set obtained is D, and the relationship between the set F and the set D is F=nD.

3) Aggregate all sparse data to generate dataset W, then W=Pooling (F).

4) Aggregation of the data is performed using the square root aggregation method, and the formula for the aggregation calculation is.

$$W_i = \sqrt{\frac{1}{M}} \sum_{i=1}^{M} (F_i - 1) \tag{1}$$

$$W = (W_1 + W_2 + \cdots W_n) \tag{2}$$

Classification of web-architecture software test data based on the expression of sparse data, concept tree, weight calculation and expression calculation are used to express the characteristics of web-architecture software test data, which are classified and saved according to the data characteristics.

5) Save the network structure of the software test data for the number, easy to extract and query.

The classification model based on feature expansion of network structure software test data has the following advantages: firstly, the model maps the features of network structure software test data into vector space, which is conducive to feature expansion and performance; secondly, through accurate calculation, it can reduce the classification error and the calculation speed is faster, which improves the data relevance processing ability; finally, after the data is classified, it is labeled with the category and number in each database, which is conducive to later data retrieval and querying. data retrieval and query.

3.2 Feature Expansion Algorithm

Input: Net structure software test data, generate data files to be tested.

Output:Feature vector space after feature expansion of net-constructed software test data.

1) For any one feature item in the test data of the network structure software to be tested, calculate the minimum confidence and minimum support of the feature item, and the calculation formulas are shown in Eq. (3) and Eq. (4).

$$s = \frac{f+v}{\sqrt{\frac{1}{N}}} \tag{3}$$

$$c = \frac{f+v}{\sqrt{\frac{1}{N}}} \tag{4}$$

2) The feature items with the minimum execution degree and the minimum support degree are generated into the query feature co-occurrence set. The network software test data in the query feature co-occurrence set is defined as the rule feature item, and the rule degree of each item is calculated. The calculation method of the rule degree H is shown in Formula (5).

$$H = \frac{1}{N}\left(\frac{\sqrt{s}}{\sqrt{f}} + \sqrt{v}\right) \tag{5}$$

3) If the H value is greater than the set rule threshold, the rule feature item can be considered as a rule item; If there is a unique rule item in the query feature co-occurrence set, step 4) will be executed; If there are two or more rule items, step 5) will be executed;

4) Inclusion of unique rule terms in the feature space set;

5) Matching web-constructed software test data characteristics;

6) The data in the feature space set to generate the feature vector space, continue to calculate the next set of network structure software test data.

3.3 Classification of test data for web-constructed software

There are more interference factors in the feature vectors of the net structure software test data after feature expansion, due to the similarity between the net structure software test data, which affects the feature expression ability of the net structure software test data, and the data analysis can improve the expression ability of the data, which is more conducive to the realization of data classification and query.

First of all, the conceptual tree carries out the conceptual description of its data through the attribute analysis of the data, and defines the attribute weight formula of the web-constructed software test data as shown in equation (6).

$$J(n) = \sum_{i=0}^{I} [\log_i (Deep + 1)] \tag{6}$$

Due to the similarity of the data, there is not much difference between the similar data, in order to accurately categorize the data, after calculating the weights of the test data of the net structure software, the expression ability is calculated, as shown in equation (7).

$$G(n) = \sin J(n)\sqrt{\frac{1}{M}} \tag{7}$$

According to the expression ability of the net structure software test data, the eigenvalues are fully expressed and reasonably categorized according to the differences in the expressed eigenvalues.

3.4 Gaussian mixture modeling

The probability distribution of software test case set data is usually complex, so Gaussian mixture model is introduced to simulate approximation and reduction of software test case set to simplify the problem. The Gaussian mixture model (GMM) is M The weighted sum of individual Gaussian models (clusters of clusters) of the data, the M Gaussian estimation of quasi probability density distribution. Each Gaussian probability density function corresponds to a class. EM (expectation maximization) algorithm is used in training.

Let each sample correspond to a class, that is, a Gaussian probability density function, and the entire sample set corresponds to M Gaussian probability density functions. However, the specific Gaussian probability density function corresponding to each sample xi is uncertain, and the weight of each Gaussian probability density function in GMM is also uncertain. Formula (8) shows GMM:

$$p(x_i) = \sum_{j=1}^{M} \varphi_j f_j\left(x_j/\mu_j, \sum_j\right) \tag{8}$$

Each Gaussian probability density function,i.e., single Gaussian model, has three parameters。 When modeling the Gaussian mixture model, it is necessary to determine 3N parameters.

All parameters of GMM need to be estimated through sample set $X\,\Phi$, the probability of sample xj is shown in equation (9):

$$p(X \mid \Phi)\prod_{i=1}^{N}\sum_{j=1}^{M} \varphi_j N_j\left(x_i \mid \mu_j, C_j\right) \tag{9}$$

3.5 Experimental analysis

3.5.1 Experimental data sets

This paper uses Siemens software test case set which is widely used and easy to compare. Developed by experts from Siemens Corporate Research, it contains seven types of program codes: Print_tokens1, Print_tokens2, Schedule1, Schedule2, Replace, tacs, and tot_info. Each type of program code has one error free version and some error containing versions. The number of lines, initial test cases, errors in the program, and test cases of each type of program code in the software test case set are shown in Table 1. For each type of program code, the software test case set uses five types of programming languages for code writing to meet the code requirements of common programming languages.

Table 1. Data set for software test case approximation

data sets	the number of initial use cases	Number of errors	Number of rows	number of test cases
Print_tokens1	30	20	565	4130
Print_tokens2	30	18	510	4115
Schedule1	32	24	412	2650
Schedule2	32	32	307	2710
Replace	30	22	563	5542
tacs	32	24	173	1608
tot_info	32	28	406	1502

3.5.2 Evaluation criteria

In the experiments, three indexes, namely, the approximation rate, the error detection rate and the loss rate of error detection, are used to evaluate the experimental results in order to verify the effectiveness, correctness and stability of the algorithm.

(1) Approximate rate

Simplification rate is calculated as shown in equation (10), which is the ratio of the difference between the number of test cases in the existing software test case set and the number of test cases in the simplified software test case set, compared with the number of test cases in the existing software test case set.

$$\mathrm{RR} = \frac{\mathrm{USN}-\mathrm{USN}'}{\mathrm{USN}} \times 100\% \tag{10}$$

(2) False detection rate

The error detection rate is calculated as shown in Equation (11), which is the ratio of the number of program code errors detected from the test cases to the number of actual program code errors in the test cases after learning the initial use cases in an approximate way.

$$\mathrm{EDR} = \frac{\mathrm{EN}'}{\mathrm{EN}} \times 100\% \tag{11}$$

(3) False detection loss rate

The error detection loss rate is calculated as shown in Equation (12), which is the ratio obtained from the difference between the number of program code errors detected in the test cases and the number of actual program code errors in the test cases after learning the initial cases in a simplified manner, compared with the number of actual program code errors in the test cases.

$$ELR = \frac{|EN|-|EN'|}{|EN|} \times 100\% \tag{12}$$

From Eq. (10), Eq. (11) and Eq. (12), we can compare the reduction rate, error detection rate and error detection loss rate of Gaussian mixture model, fuzzy K-Means clustering model and the algorithm proposed in this paper.

4. Results and discussion

In order to verify the performance of the software test case set reduction algorithm proposed in this paper based on the adaptive Gaussian mixture model, the algorithm proposed in this paper, the software test case set reduction algorithm based on the Gaussian mixture model and the software test case set reduction algorithm based on the fuzzy K-Means clustering model are compared and analyzed.

The number of test cases in the set of software test cases obtained after simplification using the three algorithms is shown in Table 2.

Table 2. Initial number of use cases after simplification

data sets	the number of initial use cases	The algorithm in this paper, the	Gaussian mixing	Fuzzy K-Means
Print_tokens1	30	17	23	19
Print_tokens2	30	13	17	14
Schedule1	32	14	20	17
Schedule2	32	12	15	16
Replace	30	15	14	16
tacs	32	18	21	18
tot info	32	16	20	19
Total	218	105	130	119

The 3 algorithms approximation rates are shown in Table 3.

Table 3. The approximation rates of the three algorithms

data sets	Simplification rate/%		
	The algorithm in this paper, the	Gaussian mixing	Fuzzy K-Means
Print_tokens1	43.33	23.33	36.67
Print_tokens2	56.67	43.33	53.33
Schedule1	56.25	37.50	46.88
Schedule2	62.5	53.13	50.00
Replace	50.00	53.33	46.66
tacs	43.75	34.38	43.75
tot_info	50.00	37.50	40.63
Total	51.83	40.37	45.41

Through analysis, the reduction rate of software test case set reduction algorithm based on fuzzy K-Means clustering model is better than that based on Gaussian mixture model. The reduction rate of the algorithm proposed in this paper is higher than the two algorithms mentioned above. Its total reduction rate is 11.46% higher than that of the software test case set reduction algorithm based on the fuzzy K-Means clustering model, and 6.42% higher than that of the software test case set reduction algorithm based on the Gaussian mixture model. The improvement of simplicity can reduce the complexity of software testing to a certain extent and improve the efficiency of software testing.

5. Conclusion

This paper proposes a research on automated software testing and optimization model based on behavioral data. The algorithm does not need to fix the number of clusters during initialization and iteration. It can adaptively determine the number of clusters according to the characteristics of different software test case set data, improve

the adaptability, accuracy and generalization of clustering, and achieve optimal reduction of the use case set. Experimental results show that the reduction rate and error detection rate of the proposed algorithm are higher than those of the Gaussian mixture model algorithm and the fuzzy K-Means clustering algorithm. After reduction, although the scale of software test case set is simplified, the coverage rate is high.

References

[1] Yayan, U. . (2023). Rosmutation: mutation based automated testing for ros compatible robotic software. advances in electrical and computer engineering, 23(3), 47-56.

[2] Chandra, S. S. V. , Sankar, S. S. , & Anand, H. S. . (2022). Smell detection agent optimization approach to path generation in automated software testing. Journal of Electronic Testing, 38(6), 623-636.

[3] Xue, F. , Junsheng, W. U. , Zhang, T. , Wang, W. , & Cheng, J. . (2022). Visual judgment approach of isomorphic gui for automated mobile app testing. Journal of Northwestern Polytechnical University, 40(4), 804-811.

[4] Ruiz, B. . (2023). The role of automation in modern software testing. Journal of Computer Engineering & Information Technology, 12(4), 1-1.

[5] Wang, Y. , Zhao, Y. , Wang, X. , Tang, W. , Zhang, J. , & Wang, S. , et al. (2024). Robotic process automation efficiency for mobile app testing: an empirical investigation. International Journal of Software Engineering and Knowledge Engineering, 34(07), 1025-1046.

[6] Smith-Miles, K. , & Munoz, M. A. . (2023). Instance space analysis for algorithm testing: methodology and software tools. ACM computing surveys, 55(12), 1-31.

[7] Arasteh, B. , Imanzadeh, P. , Arasteh, K. , Gharehchopogh, F. S. , & Zarei, B. . (2022). A source-code aware method for software mutation testing using artificial bee colony algorithm. Journal of Electronic Testing, 38(3), 289-302.

[8]Zhi-boLI, Qing-baoLI, Ming-jingLAN, & Jian-fanSUN. (2022). Mart algorithm based on mirror selection order optimization. Acta Electronica Sinica, 50(02), 314-325.

[9] Zhuang, M. , Chen, Z. , Wang, H. , Tang, H. , He, J. , & Qin, B. , et al. (2022). Anatomysketch: an extensible open-source software platform for medical image analysis algorithm development. Journal of Digital Imaging, 35(6), 1623-1633.

[10] Khudhur, A. A. , Mohamed, I. , & Zainudin, S. . (2024). Enhancing automated test case generation through knn algorithm in software testing. Journal of Electrical Systems, 20(3s), 927-936.

[11] Sezgin, A. , & Boyac, A. . (2023). Aid4i:an intrusion detection framework for industrial internet of things using automated machine learning, 76(8), 2121-2143.

[12] Xue, F. , Junsheng, W. U. , Zhang, T. , Wang, W. , & Cheng, J. . (2022). Visual judgment approach of isomorphic gui for automated mobile app testing. Journal of Northwestern Polytechnical University, 40(4), 804-811.

[13] Miyaki, R. , Yoshida, N. , Fujiwara, K. , Tsuzuki, N. , Yamamoto, R. , & Takada, H. . (2022). Fuzz4b: a support tool for fuzzing with afl. Computer Software, 39(2), 124-142.

[14] Yayan, U. . (2023). Rosmutation: mutation based automated testing for ros compatible robotic software. advances in electrical and computer engineering, 23(3), 47-56.

[15] Araghi, M. A. , Rafe, V. , & Khendek, F. . (2024). Using data mining techniques to generate test cases from graph transformation systems specifications. Automated Software Engineering, 31(1), 1-43.

[16] Hacks, S. , Persson, L. , & Nicklas Hersén. (2023). Measuring and achieving test coverage of attack simulations extended version. Software and systems modeling, 22(1), 31-46.

[17] Giamattei, L. , Guerriero, A. , & Russo, P. S. . (2024). Automated functional and robustness testing of microservice architectures. The Journal of Systems and Software, 207(Jan.), 1-14.

[18] Krishna, V. V. , & Gopinath, G. . (2022). Agile test automation for web application using testng framework with random integration algorithm in machine learning to predict accuracy and response time on automated test results. journal of theoretical and applied information technology, 100(16), 4909-4917.

[19] Dai, X. , Ning, B. , & Gu, C. L. S. . (2022). Automated test case generation based on competitive swarm optimizer with schema and node branch archive. Technical Gazette, 29(3), 915-925.

[20] Hacks, S. , Persson, L. , & Nicklas Hersén. (2022). Measuring and achieving test coverage of attack simulations extended version. Software and Systems Modeling, 22(1), 31-46.