ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20210311060442
Original scientific paper

Test Case Prioritization Based on Artificial Immune Algorithm

Hongwei XU, Pengcheng LI, Zhongxiao CONG, Fengzhi ZHANG, Yi PAN, Xu REN, Xingde WANG*, Ying XING

Abstract: Regression testing is an essential and critical part of smart terminal program development. The test case suite is usually preprocessed by test case prioritization technology to improve the efficiency of regression testing. To address the problems of traditional genetic algorithm in solving the test case prioritization problem, this paper proposed a test case prioritization algorithm for intelligent terminal based on artificial immune algorithm. Firstly, different sequences of test case sets were used as the encoding of antibodies to initialize the antibody population; secondly, the Hemming distance was introduced as the concentration index of antibodies to calculate the incentive degree; finally, the antibodies were immunized to find the optimal test case set sequence. The experimental results showed that the algorithm based on the artificial immune algorithm was more capable of global search and less likely to fall into local optimum than the genetic algorithm, which indicated that the artificial immune algorithm was more stable and could better solve the test case prioritization problem.

Keywords: artificial immunity algorithms; intelligent terminal; regression testing; test case prioritization

1 INTRODUCTION

In order to better meet the needs of users, the update and iteration rate of intelligent terminal software systems has been accelerating. The intelligent terminal software systems have formed iterative development driven by regression testing [1]. Regression testing of the system is required to ensure correctness between program branches after the smart terminal program defects have been fixed. During the development and maintenance of intelligent terminal software systems, the requirements of the program change frequently [2]. With the iteration of intelligent terminal software system versions, the size of the corresponding test case set becomes larger and larger, and the cost of regression testing keeps increasing. Through a certain strategy to prioritize the test case set, the test cases that can better cover system defects are placed in front of the test case set. The test cases with higher priority can be executed quickly during regression testing, so that errors in the software can be found in time and the cost of software development can be reduced.

Test Case Prioritization (TCP) refers to determine the order of test case execution according to a certain strategy for one or more test objectives to be optimized. The TCP technique can help testers find a better test case execution sequence. By executing sequenced test cases, testers are able to find software defects earlier or at a lower cost [3]. It has been shown that test case prioritization is NP-hard problem [5], and traditional deterministic algorithms have difficulty in obtaining optimal solutions when dealing with the TCP problem. For this reason, swarm intelligence algorithms have been created and many optimization algorithms have been proposed [4, 6, 7]. In contrast, both genetic algorithms and swarm intelligence optimization algorithms have shown better results in solving the TCP problem [8-10]. However, these algorithms are prone to problems such as falling into local optimum and premature convergence when solving the TCP problem [11].

The Artificial Immune Algorithm (AIA) is an intelligent algorithm that was born after the swarm intelligence algorithm. In common with swarm intelligence algorithms, it is an algorithm that has been transferred to computer science through observational studies of the life sciences [12]. The AIA is able to generate and maintain the diversity of populations with the characteristics of adaptivity, stochasticity, parallelism, global convergence, and population diversity [13], which overcomes the inevitable "premature" problem in the process of finding the optimal solution, and can obtain the global optimal solution of the problem to be optimized. The AIA aims to establish the corresponding engineering model by studying the information processing mechanism of the immune system in the body of an organism during the occurrence of biological immunity using mathematical knowledge to model the mechanism accordingly through the observation of this mechanism. Currently, the AIA has good performance in many fields such as wireless sensing [14], military applications [15], and control engineering [16]. Based on the AIA, this paper proposed an optimization algorithm applied to the TCP problem for intelligent terminals. The optimal test case sequence was found by this optimization algorithm, and the performance of that was analyzed by experimental results.

2 RELATED WORK

Wong et al [17] were the first to conduct research on test case prioritization related techniques. They approximated and prioritized the test cases set based on the modified information of the code and the historical executed information of the test cases. Rothermel et al [18, 19] proposed Total strategy and Additional strategy and experimentally verified the effectiveness of these two strategies in solving the TCP problem. Li et al [5] transformed the TCP problem into a knapsack problem and proved that the TCP problem is a combinatorial optimization NP-complete problem. They applied Hill Climbing and Genetic Algorithm (GA) to solve the TCP problem in their experiments. The experimental results showed that the GA performed better in solving the TCP problem. Zhang et al [20] solved the TCP problem based on GA and proposed two test case prioritization evaluation metrics. Zhang et al [21] proposed an algorithm based on the Particle Swarm Optimization (PSO) of Tent chaos, which improved the three main characteristics of Tent mapping and effectively avoided the premature and convergence of the standard PSO at the later stage. Zhang et al [19] proposed a hybrid optimization method of initial selection-ranking-again

selection that combined test case selection and priority ranking, using the results of change impact analysis of code based on function call paths.

3 AIA-BASED TEST CASE PRIORITIZATION

3.1 Test Case Priority Definition

Test case prioritization technique is one of the important tools to improve the efficiency of regression testing. Rothermel et al [18] defined the test case prioritization problem as follows:

Known: a test case set T, PT is the set consisting of all possible orderings of test cases in T, and f is a mapping from PT to the space of real numbers.

Purpose: To find $T' \in PT$ that makes $(\forall T'')(T'' \in PT)(T'' \neq T')\left[f(T') \geq f(T'')\right]$.

In this definition, PT is the definition domain of the optimization objective function f, and f is a quantitative description of the ranking objective for measuring the effectiveness of the ranking. The larger the value of f, the better the test case sequence is sorted. T' is the optimal test case sorting sequence and T'' is the test case sequence that is worse than T'.

3.2 Introduction to Artificial Immunity Algorithm

The Artificial Immune Algorithm (AIA) is a new type of intelligent optimization algorithm constructed artificially by imitating biological immune mechanism and combining with the evolutionary mechanism of genes. It has the characteristics of a general immune system and uses a population search strategy to obtain the optimal solution of the problem by iterative computation. Compared with other algorithms, the AIA uses its own characteristics of diversity generation and maintenance mechanism to ensure the diversity of the population and overcome the inevitable "premature" problem in the general optimization search process, and can find the global optimal solution. The AIA has the advantages of adaptivity, stochasticity, parallelism, global convergence, and population diversity.

3.3 Coding Strategy Design

In the test case prioritization problem, the input is the test case set T and the antibody individual is a test case sequence. In this paper, different sequences of the test case set are put into the AIA as the encoding of an antibody. Suppose the test case set is $T = \{t_1, t_2, \ldots, t_n\}$, the coding of one of the antibodies is shown in Fig. 1.

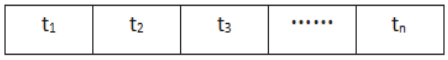


Figure 1 Schematic diagram of individual antibody coding

3.4 Affinity Function Design

Affinity characterizes the binding strength of immune cells to antigens and is similar to fitness in GA. The evaluation of affinity is related to specific problems. For different optimization problems, the affinity evaluation function should be defined according to the characteristics of the problem with the understanding of the problem substance. Usually, function optimization problems can be evaluated in terms of function values or simple treatments of function values (e.g., taking the reciprocal, opposite, etc.) as affinity, while for combinatorial optimization problems or more complex optimization problems in applications, problem-specific analysis is required [23].

In this paper, affinity is the evaluation of the similarity between the individual antibody and the optimal solution, which is a function of the match between antigen and antibody in the immune system. In this paper, the Average Percentage of Statement Coverage (APSC) corresponding to each test case sequence is used as the affinity index of the organism, and the closer the APSC is to 1, the higher the affinity of the organism and the closer the test case sequence is to the optimal solution.

3.5 Antibody Concentration Design

The antibody concentration characterizes the diversity of the antibody population, and a high antibody concentration means that there are a large number of very similar individuals in the population, and the search for the optimal solution will be concentrated in one region of the feasible solution interval, which is not conducive to global optimization. Therefore, the optimization algorithm should suppress the individuals with too high concentration to ensure the diversity of individuals [23].

In this paper, the individual antibodies are coded for the test case set sequences, so the Hemming distance between antibodies is used as the determination of the antibody-antibody affinity, and the formula for calculating the Hemming distance is shown in Eq. (1):

$$aff\left(ab_i, ab_j\right) = \sum_{k=0}^{L-1} \partial_k \tag{1}$$

where, in this paper, the similarity of encoding between two antibodies is used as a calculation of the Hemming distance. ∂_k is defined as shown in Eq. (2):

$$\partial_k = \begin{cases} 1, ab_{i,k} = ab_{j,k} \\ 0, ab_{i,k} \neq ab_{j,k} \end{cases} \tag{2}$$

In antibody populations, the higher the coding similarity of two antibodies, the greater the Hemming distance and the higher the antibody concentration.

3.6 Antibody Excitation Degree Design

Antibody excitation is the combined ability of antibody populations to respond to antigen and be activated by other antibodies, and is mainly influenced by affinity and concentration, which are proportional to affinity and inversely proportional to concentration [20]. In this paper, it was obtained by a simple mathematical operation using the results of antibody affinity and antibody concentration evaluation, as shown in Eq. (3):

$$act\left(ab_i\right) = a \times aff\left(ab_i\right) - b \times den\left(ab_i\right) \tag{3}$$

where, $act(ab_i)$ is the excitation degree of antibody ab_i, a and b are constants.

3.7 AIA-based Test Case Prioritization Optimization Algorithm Flow

The optimization algorithm proposed in this paper based on the AIA is shown in Algorithm 1.

Algorithm 1 AIA-based test case prioritization algorithm

```
begin
1       enter the set of test cases T
2 Initialize algorithm parameters(pop、iter_time、a、b、mp)
3 Initialize the antibody population
4 get_affinity()  //fetch the antibody affinity and get the maximum aff_max of affinity and its corresponding antibody individual best
6 m = 1
7 while(m ≤ iter_time):
8get_density()  //get the inter-antibody concentration
9       get_inspire()  //get inter-antibody stimulus
10 get_selection()  //select high quality antibodies for activation based on the affinity and concentration calculation of the antibodies in the population
11      get_clone()  //clone the activated antibodies to get several copies
12      get_density()  //update inter-antibody concentration
13      get_inspire()  //update inter-antibody stimulus inspire_values
14      get_inspire_values of populations for clonal inhibitionins_selection
15      forі = 1 :pop :
16      ifinspire_values[i] <ins_selection:
17              generate a random antibody individual and calculate its corresponding antibody affinity
18              ifantibody_aff>affinity_values[i] :
19                  antibodies[i] = antibody
20                  affinity_values[i] = antibody_aff
21              end if
22          end if
23          if affinity_values[i] >aff_max :    //update the optimal solution and its corresponding individual antibodies
24              aff_max = affinity_values[i]
25              best = antibodies[i]
26          end if
27 end fori
28      m++
29 end while
30 post-processing and visualization of results
end
```

In the above algorithm, line 1 represents the input, lines 2 and 3 represent the initialization of the algorithm parameters and the antibody population, line 4 represents the affinity evaluation of each feasible solution in the population to obtain the maximum *aff_max* of affinity in the current population and its corresponding antibody individual best, lines 5 - 28 represent the iterative optimization search of the antibody population, where the immune processing methods are immune selection, cloning, mutation and clonal suppression, and line 29 represents the post-processing and visualization of the results.

The immune processing part of this paper makes corresponding changes for the TCP problem, as shown in Algorithm 2.

In the above algorithm, lines 1 - 3 represent the input and output of the algorithm, lines 4 - 9 represent the immune selection for the current antibody population, and lines 10 - 26 represent the immune clone selection for the antibody population.

4 EXPERIMENTAL VERIFICATION

4.1 Algorithm Flow Selection of Optimization Targets and Experimental Environment

4.1.1 Selection of Optimization Targets

For the test case prioritization optimization problem, the Average Percentage of Statement Coverage (APSC) of the test case sequence is selected as the optimization objective for the experiments in this paper [24].

APSC represents the coverage of a sequence of test cases over lines of statements in the software code and is defined as in Eq. (4):

$$APSC = 1 - \frac{TS_1 + TS_2 + ... + TS_i}{NM} + \frac{1}{2N} \tag{4}$$

where, N denotes the number of test cases, M denotes the number of statements of the program, and TS_i denotes the position in the execution sequence of the test case in which statement i is first detected.

Algorithm 2 Immunoprocessing algorithm

```
begin
    Input: test case set T
    Output: Immunoselection array imm_selection
    Output: the antibody population antibodies and their corresponding affinity_values after clonal mutation
    defget_selection():   //immune selection
        get the minimum antibody incentive for the antibody population
        get the max_inspire of the antibody population
        limit = min_inspire + (max_inspire–min_inspire) * 0.3
        get a boolean array of the antibody excitations greater than the immune definition line
        return imm_selection
    defget_clone():    //perform clone mutation operation
            get the number of individual antibodies to be immunoclonal mutated based on the antibody affinity and
            imm_selection array
        forі = 1 :pop :
            ifimm_selection[i]:
                num = int(m_nums[i])   //the number of clonal variants corresponding to the i-th antibody
                get the array clone_abi formed by cloning the num number of antibodyі
                clone_abi = mutation(clone_abi)    //perform the antibody mutation operation
                forj :num :
                    get the affinity clone_abi_aff corresponding to the jth clone variant individual
                    if clone_abi_aff>affinity_values[i]
                        antibodies[i] = clone_abi[j]
                        affinity_values[i] = clone_abi_aff
                    end if
end for
            end if
end for
        return antibodies, affinity_values
end
```

4.1.2 Experimental Environment

The experimental environment of the article is a 64-bit operating system with x64-based processor i5-8250U, system version Windows 10, 1.6 GHz main frequency, 8.00G DDR4-2400 running memory, and python 3.9.0 programming language. The optimal sequence of test cases is obtained by iterative execution of the AIA algorithm. The number of antibody populations is set to 100, the constants of antibody excitation are 0.66 and 0.34, and the probability of clonal variation is 0.6.

4.2 Test Data Set

The programs under test in this experiment are the codes of five modules in the intelligent terminal system, and test cases of different sizes are prepared for different programs under test, which can achieve full coverage of program statements. The specific information is shown in Tab. 1.

Table 1 Program information under test

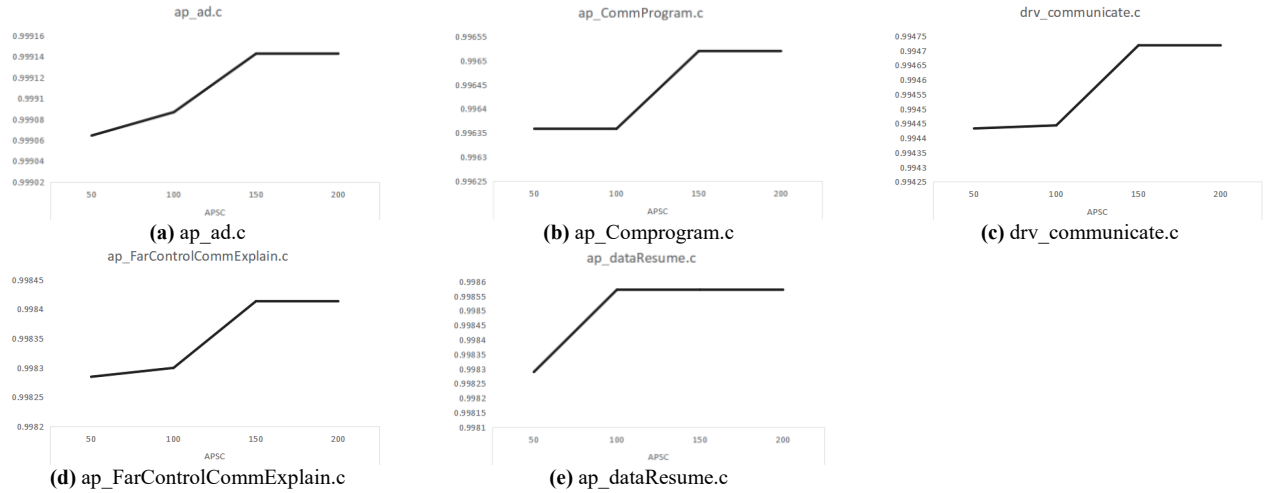
Program Under Test	Number of Statements	Test Case Set Size	Description
ap_ad.c	167	1050	AD detection module
ap_CommProgram.c	355	1014	Program record flag clearing, and corresponding event recording
drv_communicate.c	104	866	Uart port parameter configuration mode initialization
ap_FarControlCommExplain.c	130	1432	Transfer all remote commands to global variables
ap_dataResume.c	114	2035	Electrical parameter inspection measurement chip data calibration

4.3 Experimental Design and Analysis of Results

4.3.1 Experiment 1: Exploring The Effect of the Number of Population Iterations on the Test Case Prioritization Algorithm

The main purpose of this experiment is to investigate the effect of different iteration numbers on the AIA-based test case prioritization algorithm under a certain number of antibody populations. The number of iterations was set as 50, 100, 150 and 200, and then the test case sets of the five smart terminal programs under test were prioritized. The optimal test case sequences and the corresponding APSC values were recorded under different iterations. The APSC values obtained under different iterations were shown in Fig. 2.

In Fig. 2, we have counted the APSC values corresponding to the number of 50, 100, 150 and 200 iterations, and the closer the APSC is to 1, the better the sorted test case sequence obtained. As can be seen from Fig. 2, with a certain number of antibody populations, the APSC of the intelligent terminal programs under test gradually tend to be optimal as the number of iterations of the AIA increases. After a certain number of iterations, the APSC values tend to be stable and do not change significantly.



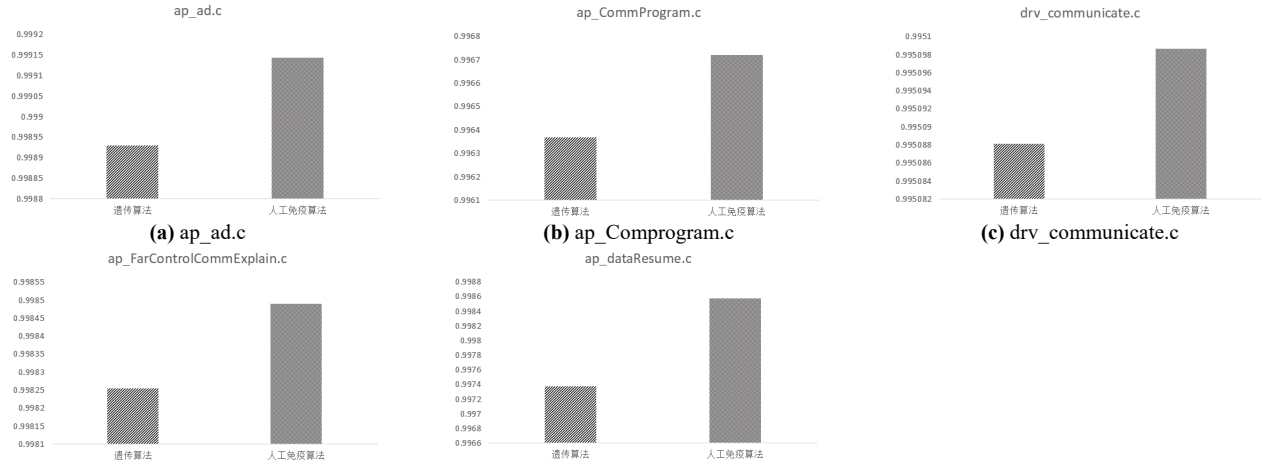
(a) ap_ad.c **(b)** ap_Comprogram.c **(c)** drv_communicate.c

(d) ap_FarControlCommExplain.c **(e)** ap_dataResume.c

Figure 2 APSC value distribution of the tested program under different iteration times

4.3.1 Experiment 2: Comparison of Artificial Immune Algorithm and Genetic Algorithm

This experiment is designed to compare the AIA with the GA in solving the TCP problem. We set the population size to 50 and the number of iterations to 200. The experimental procedure replicates that of the GA-based algorithm in the literature [17]. The experimental process will be performed on the test program using GA and AIA for test case prioritization, respectively, and the APSC values obtained from both are shown in Fig. 3.



(a) ap_ad.c **(b)** ap_Comprogram.c **(c)** drv_communicate.c

(d) ap_FarControlCommExplain.c **(e)** ap_data Resume.c

Figure 3 Comparison of APSC values of the tested program under different algorithms

In Fig. 3, the diagonal stripes represent the APSC values obtained from the GA iteration and the small squares represent the APSC values obtained from the AIA iteration. As can be seen from Fig. 3, for a certain scale of test program, when the APSC value of test case prioritization reaches stability, the APSC value obtained by the AIA-based test case prioritization algorithm is significantly better than that of the GA-based test case prioritization algorithm. In solving the test case prioritization problem, the artificial immune algorithm used in this paper can better preserve the optimal solutions obtained during the iterative process and globally keep searching for possible optimal solutions, effectively avoiding the situation of falling into local optimum.

5 CONCLUSION

In this paper, we proposed a test case prioritization algorithm based on AIA for smart terminal systems. Firstly, different sequences of the test case set were used as the encoding of antibodies to initialize the antibody population; secondly, the Hemming distance was introduced as the concentration index of antibodies to calculate the excitation degree of the antibody population; then, the antibody population was immunized and the optimal sequence of the test case set is iteratively found; finally, the obtained experimental data were analyzed and processed. The experimental results showed that the AIA combined the idea of antibody immune variation into TCP problem, and the proposed optimization algorithm has a strong global search capability and good algorithm stability, which can obtain better APSC values for the intelligent terminal test case set compared with GA.

We will try to use more optimization algorithms to solve the problems related to the prioritization of smart terminal test cases in the future. In addition, in the future, we will try to analyze and improve the research of artificial immune algorithms on the multi-objective test case prioritization problem.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. U1736110 and 61702044).

6 REFERENCES

[1] Ding, G., Zheng, Y., & Zhang, L. (2009). Study of test suite minimization based on ant colony algorithm. Computer Engineering, 35(6), 213-215+218.

[2] Zhang, L., Hu, S., Mei, N., Li, R., & Xiao, X. (2020). Overview of research on reliability of smart meter. *Electrical Measurement & Instrumentation*, *57*(16), 134-140.

[3] Elbaum, S., Malishevsky, A., & Rotherel, G. (2002). Test case prioritization: a family of empirical studies. *IEEE Transactions on Software Engineering*, *28*(2), 159-182. https://doi.org/10.1109/32.988497

[4] Mengxia, L., Ruiquan, L., & Yong, D. (2016). The Particle Swarm Optimization Algorithm with Adaptive Chaos Perturbation. *International Journal of Computers Communications & Control*, *11*(6), 804-818. https://doi.org/10.15837/ijccc.2016.6.2525

[5] Li, Z., Harman, M., & Hierons, R. (2007). Search Algorithms for Regression Test Case Prioritization. *IEEE Transactions on Software Engineering*, *33*(4), 225-237. https://doi.org/10.1109/TSE.2007.38

[6] Ji, W., Wang, J., & Zhang, J. (2014). An improved real hybrid genetic algorithm. *Tehnical gazette*, *21*(5), 979-986.

[7] Liu, H. & Ki, K. Y. (2020). Wireless sensor network based improved immune gene algorithm in airport floating personnel positioning. *Computer Communications*, 160. https://doi.org/10.1016/j.comcom.2020.04.036

[8] Khatibsyarbini, M., Isa, M., Jawawi, D., & Rooster, T. (2018). Test case prioritization approaches in regression testing: A systematic literature review. *Information & Software Technology*. https://doi.org/10.1016/j.infsof.2017.08.014

[9] Gao, D., Guo, X., & Zhao, L. (2015). Test case prioritization for regression testing based on ant colony optimization. *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS 2015)*. https://doi.org/10.1109/ICSESS.2015.7339054

[10] Mishra, D., Mishra, R., Das, K., & Arup, A. (2017). A systematic review of software testing using evolutionary techniques. *Proceedings of Sixth International Conference on Soft Computing for Problem Solving*. https://doi.org/10.1007/978-981-10-3322-3_16

[11] Sun, J., Chen, J., & Wang, G. (2018). Multi-Objective Test Case Prioritization based on Epistatic Particle Swarm Optimization. *International Journal of Performability Engineering*, 14, 2441-2448. https://doi.org/10.23940/ijpe.18.10.p20.24412448

[12] Castro, D. & Zuben, V. (2002). Learning and optimization using the clonal selection principle. *IEEE Transaction on Evolutionary Computation*, *6*(3), 239-251. https://doi.org/10.1109/TEVC.2002.1011539

[13] Jiao, L. C. & Du, H. F. (2003). Development and prospect of the artificial immune system. *Acta Electronic Sinica*, *31*(10), 1540-1548.

[14] Pang, M., Feng, Z., & Bai, W. (2020). DV-Hop node location algorithm optimized by improved artificial immune algorithms. *Journal of Terahertz Science and Electronic Information Technology*, *18*(06), 1133-1140.

[15] Gao, Y., Chen, S., Yu, M., Hai, J., & Fang, R. (2019). Multi-aircraft cooperative air combat target allocation method based on improved artificial immune algorithm. *Journal of Northwestern Polytechnical University*, *37*(02), 354-360. https://doi.org/10.1051/jnwpu/20193720354

[16] Luo, L., Zhu, L., Cao, Y., Wang, Q., & Xiao, J. (2020). Automatic access control model of power information system based on artificial intelligence technology. *Automation & Instrumentation*, *254*(12), 42-45.

[17] Wong, W., Horgan, J., London, S., et al. (1997). A study of effective regression testing in practice. *PROCEEDINGS The Eighth International Symposium On Software Reliability Engineering*. https://doi.org/10.1109/ISSRE.1997.630875

[18] Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (1999). Test Case Prioritization: An Empirical Study. *IEEE International Conference on Software Maintenance*. IEEE. https://doi.org/10.1109/ICSM.1999.792604

[19] Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (2000). Prioritizing test cases for regression testing. *IEEE Transactions on Software Engineering*, *27*(5), 102-112. https://doi.org/10.1109/32.962562

[20] Zhang, W., Wei, B., & Du, H. (2015). Test Case Prioritization Method Based on Genetic Algorithm. *Journal of Chinese Computer Systems*, *36*(09), 1998-2002.

[21] Zhang, N., Teng, S., Wu, B., & Bao, X. (2019). Test Case Prioritization Based on Tent Chaos. *Computer Measurement & Control*, *27*(06), 9-12.

[22] Zhang, L., Mou, Y., Zhang, Z., & Cui, Z. (2020). Test Case Hybrid Optimization Method Based on Function Call Path. *Science Technology and Engineering*, *20*(09), 3640-3647.

[23] Sun, N. (2006). Artificial Immune Optimization Algorithm and Applications. *Harbin Institute of Technology*.

[24] Li, Z., Harman, M., & Hierons, R. M. (2007). Search algorithms for regression test case prioritization. *IEEE Transactions on Software Engineering*, *33*(4), 225-237.

Contact information:

Hongwei XU, MA. Eng
Measurement center of Guizhou Power Grid Company,
32 Jiefang Road, Nanming District, Guiyang City, Guizhou Province, China
E-mail: 275853249@qq.com

Pengcheng LI, MA. Eng
Measurement center of Guizhou Power Grid Company,
7 Guanshui Road, Nanming District, Guiyang City, Guizhou Province, China
E-mail: 365248318@qq.com

Zhongxiao CONG, MA. Eng
Measurement center of Guizhou Power Grid Company,
120 Guanshui Road, Nanming District, Guiyang City, Guizhou Province, China
E-mail: 644397824@qq.com

Fengzhi ZHANG, B.S.
Holley Technology Co.Ltd,
181 Wuchang Street, Yuhang District, Hangzhou City, Zhejiang Province, China
E-mail:nb273145890@qq.com

Yi PAN, B.S.
Holley Technology Co.Ltd,
181 Wuchang Street, Yuhang District, Hangzhou City, Zhejiang Province, China
E-mail: 1046486119@qq.com

Xu REN, B.S.
Holley Technology Co.Ltd,
181 Wuchang Street, Yuhang District, Hangzhou City, Zhejiang Province, China
E-mail: 282097986@qq.com

Xingde WANG, postgraguate
(Corresponding author)
Beijing University of Posts and Telecommunications,
10 Xitucheng Road, Haidian District, Beijing, China
E-mail: wxd_sanders@163.com

Ying XING, Associate Professor, PhD
Beijing University of Posts and Telecommunications,
10 Xitucheng Road, Haidian District, Beijing, China
E-mail: xingying@bupt.edu.cn