ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20220328054159
Original scientific paper

A Study of Wolf Pack Algorithm for Test Suite Reduction

Zemin LI, Shaobo LEI, Fugui TAN, Yupeng LIU, Bin XIAO, Xin HUANG, Xu REN*

Abstract: Modern smart meter programs are iterating at an ever-increasing rate, placing higher demands on the software testing of smart meters. How to reduce the cost of software testing has become a focus of current research. The reduction of test overhead is the most intuitive way to reduce the cost of software testing. Test suite reduction is one of the necessary means to reduce test overhead. This paper proposes a smart meter test suite reduction technique based on Wolf Pack Algorithm. First, the algorithm uses the binary optimization set coverage problem to represent the test suite reduction of the smart meter program; then, the Wolf Pack Algorithm is improved by converting the positions of individual wolves into a 0/1 matrix; finally, the optimal test case subset is obtained by iteration. By simulating different smart meter programs and different size test suites, the experimental result shows that the Wolf Pack Algorithm achieves better results compared to similar algorithms in terms of the percentage of obtaining both the optimal solution and the optimal subset of test overhead.

Keywords: smart meters; software testing; test suite reduction; wolf pack algorithm

1 INTRODUCTION

With the premise of satisfying customers' needs, the iterative rate of smart meter programs is gradually accelerating. Software iterative development driven by software testing has become an important development method for smart meter software [1]. Software testing requires continuous testing of the test suite as the coherence between different functional modules may be changed after the program is modified. In the process of designing and developing applications for smart meters, the functional requirements change frequently due to actual circumstances and the test overhead of software testing is constantly increasing [2]. According to statistics, software testing costs account for 80% of software development costs and more than half of overall maintenance costs [3]. The efficiency of software testing directly affects the efficiency of smart meter software development.

Normally, for a coverage test of a given program under test, the original test suite will have a large amount of redundant data. For reducing the cost of software testing losses, Harrold et al [4] first proposed the concept of test suite reduction and based on this a heuristic method for incomplete selection testing of test suite. Chvatal [5] summarized the existing research foundation and innovatively applied the greedy algorithm (greedy) to solve the test suite reduction problem. Chen et al. [6] improved the greedy algorithm and based on this, Greedy Redundant Essential (GRE) was proposed to solve test suite reduction problem. Although the above methods can obtain the near-optimal solution to the test suite reduction problem, they cannot guarantee to obtain the optimal representative set. Moreover, the worst time complexity is relatively large, which is not suitable for the reduction of large test suites.

In order to better solve the test suite reduction problem, the researchers introduced the swarm intelligence algorithm to solve it. The swarm intelligence algorithm belongs to bioheuristic method with good parallelism and autonomous exploration, which provides new ideas and means to solve test suite reduction problem. It has become the focus of more and more researchers' attention. Li Hua et al. [7] applied the Genetic Algorithm (GA) to Ant Colony Optimization (ACO) to generate pheromones for ACO by using the global search capability of GA, so that a subset of test suite can be obtained quickly by iteration. Wenjing Liu et al. [8] accelerated the process by exploiting the diversity and maintenance of extreme values of the immune system. Chenchen Li et al. [9] optimized the Particle Swarm Optimization (PSO) and applied it to solve test suite reduction problem. The particle swarm relies on the update of position to achieve the iteration of the population. Weiwei et al. optimized the Firefly Algorithm (FA) [10] and Ant Lion Optimizer (ALO) [11] to reduce the test overhead efficiently by considering both the smart meter program branch coverage and the test overhead.

The Wolf Pack Algorithm (WPA) is a random search algorithm. Different individuals in the wolf pack cooperate with each other, and can quickly find the optimal solution with a large probability. Moreover, the wolf pack algorithm also has parallelism, which can start searching at multiple points at the same time, and the points do not affect each other, thereby improving the efficiency of the algorithm [12]. In this paper, we try to apply the latest WPA [13] to solve this problem.

Following are the main contributions of this paper. The test suite reduction problem is defined as an objective optimization problem, and the test overhead is usually chosen as the optimization objective. We introduce the WPA to solve the test suite reduction problem. The WPA is a stochastic probabilistic search algorithm that enables it to find the optimal solution quickly with a large probability. The WPA also has parallelism, which can search from multiple points at the same time without affecting each other between points, thus improving the efficiency of the algorithm.

The rest of this paper is organized as follows. Section 2 illustrates the problem model for the test suite reduction problem. In Section 3, we optimize the WPA algorithm. In Section 4, we perform the experimental analysis and empirical evaluation of the proposed method. Section 5 summarizes the conclusions of this paper and highlights the directions for future research.

2 TEST SUITE REDUCTION MODEL

The test suite reduction problem is a typical *NP* - *complete* problem [14]. The binary relationship $S(T,R)=\{(t,r)\in T\times R\}$ between the set of test suite T and test requirements R [15]. For example, the test requirements are $R=\{r_1, r_2, \ldots, r_m\}$ and the test suite $T=$

$\{t_1, t_2, \ldots, t_n\}$ satisfies all the test requirements R. The test suite reduction problem boils down to finding a minimal subset RS of T in which the test cases can cover all the requirements in R.

As shown in Tab. 1, the test suite T has 5 test cases $\{t_1, t_2, t_3, t_4, t_5\}$ and the test requirement set R has 6 test requirements $\{r_1, r_2, r_3, r_4, r_5, r_6\}$, where ▪ represents the test relationships S and the test overhead C has 5 test overheads $\{c_1, c_2, c_3, c_4, c_5\}$.

Table 1 Test suite example

Test suite T	Test requirements R						Execution cost C
	r_1	r_2	r_2	r_4	r_5	r_6	
t_1		▪	▪				c_1
t_2	▪			▪	▪	▪	c_2
t_3					▪	▪	c_3
t_4		▪					c_4
t_5				▪		▪	c_5

The above problem is equivalent to the following format: The test relationship S is converted into a 0/1 matrix. The ▪ is replaced by 1 to indicate that the test case can cover the test requirement, and 0 elsewhere to indicate that the test case cannot meet the test requirement. A vector is used to represent the test overhead C. It can be transformed from Tab. 1 into the following matrix.

$$\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{cccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}$$

Let the vector x denote the reduction result, $x_i = 1$ means the test case t_i is selected, $S_{ij} = 1$ means t_i can satisfy the test requirement r_j, and the problem objective is expressed as Eq. (1) and Eq. (2).

$$\min f(x) = \vec{x} \cdot \vec{c} \tag{1}$$

$$\text{s.t.} \prod_{j=1}^{n} \left(\sum_{i=1}^{m} (x_i \cdot s_{ij}) \right) > 0 \tag{2}$$

where (1) is the reduction objective and (2) is the constraint.

3 TEST SUITE REDUCTION BASED ON WPA

3.1 Description of WPA

The Wolf Pack Algorithm (WPA) simulates the predatory behaviour and prey distribution of wolves, and abstracts three intelligent behaviours: wandering, summoning, and besieging [12]. The WPA has the following rules:

1) Head wolf generation rule

After initializing the wolf population, the individual with the optimal objective function value among them will be marked as the head wolf. During each iteration, the marked head wolf does not perform the three intelligent behaviours of wandering, summoning, and besieging. If the marked head wolf is replaced by another individual with better objective function value during the iteration, the current head wolf is unmarked.

2) Wandering behaviour

Except for the head wolf, S_{num} artificial wolves were regarded as probe wolves, and S_{num} was randomly selected as an integer between $\left[\frac{n}{\alpha+1}, \frac{n}{\alpha} \right]$, where α was the probe wolf proportion factor and n was the number of wolves. Firstly, we need to calculate the objective function value Y_i of probe wolf i. If Y_{in} is larger than the head wolf's objective function value Y_{lead}, the head wolf needs to be updated, $Y_{\text{lead}} = Y_i$, and probe wolf i initiates the call instead of the head wolf. If $Y_{\text{lead}} > Y_i$, the probing wolf takes one step forward in each of the h directions (the wandering step is $Step_a$) and we record the function value after each step forward. The position of probe wolf i after advancing in the pth (p = 1, 2, …, h) direction is shown in Eq. (3).

$$x_i^p = x_i + \sin(2\pi \cdot \frac{p}{h}) \cdot Step_a \tag{3}$$

Since there are differences in the prey search mode of each wolf, the value of h is different. In practice, we can choose a random integer between [$h_{\min}$, $h_{\max}$], and the larger h is, the finer the wolf searches, but the search speed is also relatively slow.

3) Summoning behaviour

The head wolf calls M_{num} wolves to rapidly approach the location of the head wolf, where $M_{\text{num}} = n - S_{\text{num}} - 1$. The wolf quickly approaches the location of the head wolf with a relatively large running step $Step_b$. The position of wolf i at the (k + 1)-th iteration is shown in Eq. (4).

$$x_i(k+1) = x_i(k) + Step_b \cdot (g_k - x_i(k)) / |g_k - x_i(k)| \tag{4}$$

where, g_k is the position of the k-th generation of head wolf. Eq. (4) consists of 2 parts, the former is the current position of the artificial wolf, and the latter indicates the tendency of the artificial wolf to gradually gather towards the head wolf position.

On the running, if the function value of fierce wolf i is $Y_{\text{lead}} < Y_i$, then let $Y_{\text{lead}} = Y_i$. At the same time, the fierce wolf i is transformed into the head wolf and initiates the summoning behaviour. If $Y_{\text{lead}} > Y_i$, fierce wolf I continues to run until the distance d_{is} less than d_{near} between it and the head wolf s when it turns to besieging behaviour. Let the variables to be searched for superiority take values in the range [l_b, u_b], then the decision distance d_{near} can be estimated by Eq. (5).

$$d_{\text{near}} = \frac{1}{D \cdot w} \sum_{d=1}^{D} (u_b - l_b) \tag{5}$$

where, ω is the distance determination factor. The increase of ω will accelerate the convergence of the algorithm.

4) Besieging behaviour

After running, the fierce wolves are closer to the prey. The fierce wolves have to unite with the probe wolves to besiege and capture the prey. The position of the head wolf is considered as the position of the prey. For the k-th

generation of wolves, we set the location of the prey as G_k, then the besieging behaviour of the wolves is represented by Eq. (6).

$$x_i(k+1) = x_i(k) + \lambda \cdot Step_c * | g_k - x_i(k) | \quad (6)$$

where, λ is a random number uniformly distributed between [−1, 1], and $Step_c$ is the attack step length of artificial wolf i when performing besieging behaviour.

Let the value range of the variable to be searched for be $[l_b, u_b]$, then the relationship between the step lengths of the three intelligent behaviors involved in the wandering step $Step_a$, the running step $Step_b$, and the attack step $Step_c$ exists as shown in Eq. (7).

$$Step_a = \frac{Step_b}{2} = 2 \cdot Step_c = | u_b - l_b | / s \quad (7)$$

where, s is the step factor, which indicates the fineness of the artificial wolf's search for the optimal solution in the solution space.

3.2 Binary Optimization Problem Application

The test suite reduction problem is a binary optimization problem. The WPA is improved by a cellular automaton classification model that converts the positions of artificial wolves into a 0/1 matrix, and the j-th dimensional position of the i-th artificial wolf is represented as x_{ij}. The $sign$ function is used to classify the cell values as shown in Eq. (8).

$$s_{ij} = \frac{1}{1 + e^{-x_{ij}}} \quad (8)$$

The cellular automata model classification is expressed as Eq. (9).

$$x_{ij}(t+1) = \begin{cases} 0 & \text{sign}(x_{ij}(t)) < 0.5 \\ 1 & \text{sign}(x_{ij}(t)) \geq 0.5 \end{cases} \quad (9)$$

where t represents the moment.

3.3 Binary Optimization Problem Application Algorithm Design

The proposed WPA-based test suite reduction technique for smart meters is shown below.

In the below algorithm, lines 1-5 represent the inputs and output, lines 6-13 mean the basic parameters of WPA, line 14 introduces the initialization algorithm parameters, line 15 is used to initialize the artificial wolf population, line 16 represents the calculation of the fitness corresponding to the wolf individuals, lines 17-26 illustrate the population iteration to update the wolf population until the maximum number of iteration is reached, and the optimal subset of test suite after the reduction is returned at line 27.

Algorithm WPA-based test suite reduction technique for smart meters

```
   begin
1      Input: Test suite T
2      Input: Test Requirements R
3      Input: Testing Relationships S
4      Input: Test Overhead C
5      Output: Subset RS
6      iter_max: the maximum number of population iteration
7      cur_iter: the number of current iteration of the population
8      s: the head wolf
9      pro_wolf: the probe wolf population
10     fie_wolf: the fierce wolf population
11     Step_a: the wandering step length of probe wolf
12     Step_b: the running step length of fierce wolf
13     Step_c: the attack step length of fierce wolf
14     initialize the algorithm parameters
15     initialize the artificial wolf population
16     calculate the adaptation corresponding to the individual wolf pack
17     for cur_iter = 1: iter_max:
18        initialize the head wolf s
19        initializing the probe wolf population pro_wolf
20        initializing the fierce wolf population fie_wolf
21        the probe wolves search prey randomly with wandering step Step_a
22        the fierce wolves listen to the head wolf's call and run towards the head wolf's position with a larger running step Step_b
23        the fierce wolves near the head wolf attack the head wolf with the attack step Step_c
24        cull poorly adapted wolves and regenerate new packs
25        update the wolf populations and calculate the corresponding adaptations
26     end for
27     return RS
   end
```

4 EXPERIMENTAL VERIFICATION

4.1 Experimental Environment

The CPU used in the experimental environment of this paper is Intel Pentium G2030 3.00 GHz, 4G DDR3 memory, running under Ubuntu 16.10 64-bit operating system, and the programming language is Python 3.9.0 and shell language.

During each execution, the number of population iterations is 100. In order to make the experimental results more general, the algorithm is executed 20 times successively, and the obtained results are analyzed and processed.

4.2 Testing Data Set

The five functional modules of the smart meter system species are selected as the system under test in this experiment. The scale of the test suite varies for different functional modules, and all of them can cover the whole test points. The specific information is shown in Tab. 2.

Table 2 Program information under test

Program Under Test	LOC	Number of branches	Test Suite Size	Description
ap _ ad	167	105	1 680	detect AD analog-to-digital conversion
ap _ CommProgram	355	78	1 326	program record flag clearing, and corresponding event recording
ap _ recorder	292	96	1 632	responsible for reading load records
GuoWangPrepay	114	118	2 006	intelligent detection of plug-in users
Ap _ ClockBatLowV	232	37	1 221	detect if the clock battery is under-voltage

4.3 Experimental Problem

The experiments in this paper are designed to analyze the following problems:

1) What about the ability of WPA to reduce the test overhead?

The most obvious indication of improving testing efficiency is to reduce the test overhead. In this paper, the Execution Cost of the Representative Set (*ECRS*) is used to denote the effect of reduction [16], and the formula is shown in Eq. (10).

$$ECRS = \sum_{i \in RS} c_i \tag{10}$$

where c_i represents the overhead of the i-th test case, and the smaller the value of *ECRS*, the better the reduction effect. The test overhead of this experiment uses the number of branches of the program under test that can be covered by each test case.

2) What is the difference in the effectiveness of the WPA algorithm in test suites of different sizes?

3) How does the WPA perform in different systems under test?

4.4 Experimental Investigation

Each program under test was performed separately as follows: 100, 200, 500, 1000, and the full set were randomly selected to form a new test suite in the total test suite, and each group was randomly selected 20 times, while the coverage rate was guaranteed to be 100%. In this paper, we reproduced the FA [10], the greedy and the GRE for comparison experiments. In the experimental process, this paper compares the four algorithms in two aspects: one aspect counts the number of times the four algorithms WPA, FA, GRE and greedy algorithms perform optimally after the same number of reduction experiments, and another aspect counts the *ECRS* values after each set of experiments. The experimental results are shown in Tab. 3, and all subsequent analyses are based on Tab. 3.

Table 3 Frequency of optimal reduced test suites and corresponding *ECRS* values

Program under Test	Algorithm	Test Suite Size									
		100		200		500		1000		full	
		Frequency	ECRS	Frequency	ECRS	Frequency	ECRS	Frequency	ECRS	Frequency	ECRS
ap _ ad	WPA	5	175.6	**14**	**155.95**	**18**	**145.02**	**20**	**138.45**	**19**	**136.3**
	FA	7	174.8	2	163.15	2	148.45	3	140.55	3	138.45
	GRE	**8**	**173.8**	5	161.3	1	150.8	2	141	0	140
	greedy	3	183.8	2	167.45	1	152.5	0	151	0	141
ap _ CommProgram	WPA	2	117.1	**16**	**104.3**	**18**	**98.65**	**18**	**93.3**	**19**	**92.75**
	FA	7	115.55	1	108.7	2	101.15	5	95.75	5	94.1
	GRE	**9**	**114.3**	5	107.5	0	102.65	2	95	0	95
	greedy	2	122.65	0	114.1	1	105.55	0	103	1	94
ap _ recorder	WPA	6	155.15	**16**	**138.55**	**16**	**128.25**	**18**	**124.15**	**17**	**119.35**
	FA	**8**	**154.95**	1	144.8	3	132.95	4	125.75	3	123
	GRE	5	158.65	4	144	2	135.85	0	128	0	130
	greedy	3	164.4	1	148.1	1	137.15	0	127	0	136
GuoWangPrepay	WPA	**13**	**202.4**	**17**	**183.45**	**17**	**171**	**20**	**162.05**	**18**	**157.6**
	FA	6	204	0	191.55	1	175.1	0	166.7	2	161.35
	GRE	3	209.05	3	189.4	2	178.15	1	171.4	0	168
	greedy	0	216.35	1	194.8	1	179.55	1	174.3	0	167
Ap _ ClockBatLowV	WPA	**14**	**43.8**	**19**	**41.65**	**17**	**39.7**	**18**	**38.75**	**19**	**38.6**
	FA	8	44.7	1	43.5	9	41	12	39.2	12	39.1
	GRE	4	45.8	1	44.7	2	43.45	0	40	0	40
	greedy	0	49.9	1	44.8	1	44.4	0	44	0	44

Note: *ECRS* value is the average value after 20 times of algorithm execution

4.5 Experiment Analysis

1) Problem One

The WPA proposed in this paper was compared with FA, GRE and greedy algorithm in this paper. On the one hand, the *ECRS* values of the test suites were counted by the four algorithms after reduction respectively, as shown in Fig. 1a, and the mean values of *ECRS* were obtained as 120.075, 122.732, 124.312 and 128.272, respectively, and the WPA algorithm's result for the test suite after reduction is 2.16% less than FA. On the other hand, the statistics from the frequency of optimal reductions, as shown in Fig. 1b,

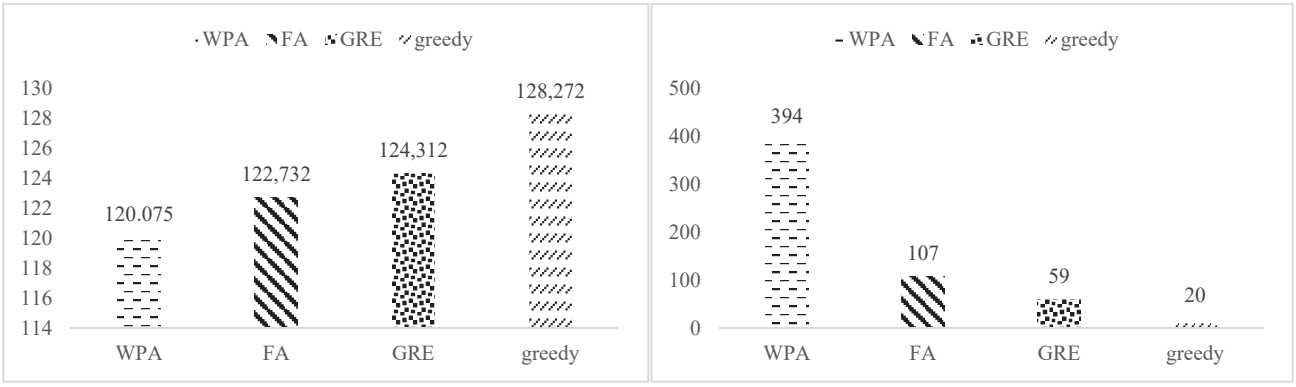
respectively, the frequency of the optimum that four algorithms reached after reducing the same test suite were 394, 107, 59, and 20. From both aspects, it can be seen that the WPA proposed in this paper is an efficient method for test suite reduction.

2) Problem Two

This paper performed reduction comparison experiments by randomly selecting test suites of different sizes for the five programs under test. The experimental results showed that the percentage of times the WPA got the optimal effect was 35.40%, 74.54%, 74.78%, 75.81% and 77.96% for the size of 100, 200, 500, 1000 and the full set, respectively. The experimental results fully demonstrated the effectiveness and superiority of the algorithm in the test suite reduction. The results were shown in Tab. 4.

3) Problem Three

In this paper, we tested a total of 5 different programs under test, and the following results can be obtained by counting the test results of the 5 programs under test, as shown in Tab. 5. Among the five tested programs, the optimal frequency ratios of WPA algorithm are respectively 66.08%, 64.60%, 67.59%, 80.18%, and 63.04% respectively. It can be seen that the WPA algorithm proposed in this paper has good performance in different programs under test, reflecting its feasibility in solving the test suite reduction problem.



(a) Comparison of ECRS values (b) Comparison of the best frequency of reduction

Figure 1 Comparison of the reduction effect under different algorithms

Table 4 Optimal number of times at different sizes

Algorithm	Test Suite Size				
	100	200	500	1000	full
WPA	**40**	**82**	**86**	**94**	**92**
FA	36	5	17	24	25
GRE	29	18	7	5	0
greedy	8	5	5	1	1

Table 5 Optimal number of times under different programs under test

Algorithm	Program under Test				
	ap _ ad	*ap _ CommProgram*	*ap _ recorder*	*GuoWangPrepay*	*Ap _ ClockBatLowV*
WPA	76	73	73	85	87
FA	17	20	19	9	42
GRE	16	16	11	9	7
greedy	6	4	5	3	2

5 CONCLUSION

The test suite reduction of smart meters is modeled as a set coverage problem, which is a classical *NP-Complete* class optimization problem where the execution time increases dramatically with the size of the problem. Due to the limitation of testing cost, it is not allowed to execute all test cases in sequence. This paper proposed the WPA to reduce the test overhead of software testing by setting up a cellular modeler to make it applicable to binary optimization problems and to reduce the test overhead by reduction. The results of the experiments with five programs under test in smart meter system showed that the WPA algorithm can reduce the scale of the test suite with high efficiency and high quality while guaranteeing the coverage of test requirements, which is better than FA, GRE and greedy algorithms. Meanwhile, it is also proved to have good robustness in experiments with different sizes and programs under test.

However, the WPA proposed in this paper still has some shortcomings and needs further research and development, mainly in the following two aspects: (1) the effectiveness of the mechanism in this paper is proved by the information of the test case coverage of the test requirements collected accurately from more smart meter systems; (2) it needs to be compared with other intelligent optimization algorithms in the next step.

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. https://doi.org/10.3969/j.issn.1000-3428.2009.06.075

[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. https://doi.org/10.19753/j.issn1001-1390.2020.16.023

[3] Harrold, M. J. (2009). Reduce, reuse, recycle, recover: Techniques for improved regression testing. *IEEE International Conference on Software Maintenance*. https://doi.org/10.1109/ICSM.2009.5306347
[4] Harrold, M. J., Gupta, R., & Soffa, M. L. (1993). A methodology for controlling the size of a test suite. *ACM Transactions on Software Engineering and Methodology*, *2*(3), 207-285. https://doi.org/10.1109/ICSM.1990.131378
[5] Chvatal, V. (1979). A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, *4*(3), 233-235. https://doi.org/10.1287/moor.4.3.233
[6] Chen, T. Y. & Lau, M. F. (1998). A new heuristic for test suite reduction. *Information and Software Technology*, 1998, *40*(5-6), 347-354. https://doi.org/10.1016/S0950-5849(98)00050-0
[7] Hua, L., Wang, C., Gu, Q., & Cheng, H. (2012). Test-suite Reduction Based on Genetic Algorithm and Ant Colony Algorithm. *Chinese Journal of Engineering Mathematics*, *29*(04), 486-492. https://doi.org/10.3969/j.issn.1005-3085.2012.04.003
[8] Liu, W., Xing, Y., Han, H., & Gong, Y. (2017). Efficient regression test suite reduction mechanism based on artificial immune algorithm. *Journal of Southeast University (Natural Science Edition)*, *47*(S1), 170-175. https://doi.org/10.3969/j.issn.1001-0505.2017.S1.032
[9] Li, C. (2019). *Test case Generation Method Based on Optimized Reduction Particle Swarm*. Zhejiang Sci-Tech University.
[10] Wei, W., Su, J., Ye, L., Li, F., & Wang, X. (2021). Test Case Suite Reduction Based on an Intelligent Optimization Algorithm. *Journal of CAEIT*, *16*(02), 111-118+126. https://doi.org/10.3969/j.issn.1673-5692.2021.02.002
[11] Wei, W., Cheng, W., Ye, L., Xia, S., Wang, Y., Xing, Y., & Wang, X. (2022). A Study of Test Suite Reduction Based on Ant Lion Optimizer. *Tehnicki Vjesnik - Technical Gazette*, *29*(1), 246-251. https://doi.org/10.17559/TV-20210807090754
[12] Wu, H., Zhang, F., & Wu, L. (2013). New swarm intelligence algorithm--wolf pack algorithm. *Systems Engineering and Electronics*, *35*(11), 2430-2438. https://doi.org/10.3969/j.issn.1001-506X.2013.11.33
[13] Xin, J., Hao, G., Li, Z., & Tianxin, Y. (1995). Improved Wolf Pack Algorithm for Solving a Class of Multi-Objective Route Optimization Problem. *Journal of Physics: Conference Series*, *1995*(1). https://doi.org/10.1088/1742-6596/1995/1/012038
[14] Chen, T. Y. & Lau, M. F. (1998). A Simulation Study on Some Heuristics for Test Suite Reduction. *Information and Software Technology*, *40*(13), 777-787. https://doi.org/10.1016/S0950-5849(98)00094-9
[15] Xiaofang, Z., Lin, C., Baowen, X. et al. (2008). Survey of test suite r eduction problem. *Journal of Frontiers of Computer Science and Technology*, *2*(3), 235- 247. https://doi.org/10.3778/j.issn.1673-9418.2008.03.002
[16] Wang, S., Ali, S., & Gotlieb, A. (2015). Cost-effective test suite minimization in product lines using search techniques. *The Journal of Systems & Software*, *103*, 370-391. https://doi.org/10.1016/j.jss.2014.08.024

Contact information:

Zemin LI, Engineer
Inner Mongolia Power (Group) Co., Ltd.,
Jinxiu Fuyuan A District, Qianda Men Road, Saihan District, Hohhot,
Inner Mongolia, China
E-mail: lizemin@impc.com.cn

Shaobo LEI, MA. Engineer
Inner Mongolia Power (Group) Co., Ltd.,
Jinxiu Fuyuan A District, Qianda Men Road, Saihan District, Hohhot,
Inner Mongolia, China
E-mail: leishaobo@impc.com.cn

Fugui TAN, Engineer
Inner Mongolia Power (Group) Co., Ltd.,
Jinxiu Fuyuan A District, Qianda Men Road, Saihan District, Hohhot,
Inner Mongolia, China
E-mail: tanfugui@impc.com.cn

Yupeng LIU, MA. Engineer
Inner Mongolia Electric Power (Group) Co.,
9 Qianda Men Road, Saihan District, Hohhot, Inner Mongolia, China
E-mail: liuyupeng@impc.com.cn

Bin XIAO, Engineer
Inner Mongolia Power (Group) Co., Ltd.,
Jinxiu Fuyuan A District, Qianda Men Road, Saihan District, Hohhot,
Inner Mongolia, China
E-mail: xiaobin@impc.com.cn

Xin HUANG, MA. Engineer
Inner Mongolia Electric Power (Group) Co.,
9 Qianda Men Road, Saihan District, Hohhot, Inner Mongolia, China
E-mail: huangxin@impc.com.cn

Xu REN, Engineer
(Corresponding author)
Holley Technology Co., Ltd.,
181 Wuchang Avenue, Wuchang Street, Yuhang District, Hangzhou,
Zhejiang Province, China
E-mail: xu.ren@holley.cn