A Meta-heuristic Test Case Prioritization Method Based on Hybrid Model

Wenjun Su[1st]
School of Automation
Chongqing University of Posts and Telecommunications
Chongqing, China.
swj19947@163.com

Zhihui Wang[3rd]
School of Automation
Chongqing University of Posts and Telecommunications
Chongqing, China.
384738799@qq.com

Zhao Li[2st]
School of Automation
Chongqing University of Posts and Telecommunications
Chongqing, China.
305158129@qq.com

Dengxin Yang[4rd]
School of Automation
Chongqing University of Posts and Telecommunications
Chongqing, China.
786849114@qq.com

***Abstract*—Software testing is an important and complex part of the software development life cycle. Along with version changes and defect repairs of the software system under test, regression testing is required to ensure that the modified parts have no impact on the unmodified parts. In a resource-constrained environment, it is necessary to select a more valuable test case from the test case library to execute first. However, the existing prioritization methods of test cases are still insufficient in terms of Average Percentage of Faults Detected (APFD) and time execution performance, and there is a problem of large search space. Aiming at the test case priority ranking problem, this paper proposes a meta-heuristic test case prioritization method based on a hybrid model to reduce test cost. This method first establishes a hybrid model by using the correlation between test cases and the importance of test data, and then uses an improved firefly algorithm based on the hybrid model to find an optimal test sequence. This article has carried out experiments on three benchmark test programs. The test suite is from Software-artifact Infrastructure Repository ([1]SIR). The experimental results show that the method proposed in this paper has better performance in terms of APFD and time execution compared with existing methods, such as Greedy, Particle Swarm Optimization (PSO) and Firefly Algorithm (FA).**

Keywords-component; Software testing; test case priority; hybrid model;firefly algorithm

I. INTRODUCTION

In a software development life cycle, the software testing phase is usually an essential and critical step. With the growing demand for highly reliable, scalable, and robust software, software testing accounts for approximately 50% of the workload in software engineering. Software test execution times are often long and come with significant test costs. Due to time and resource constraints, some studies have pointed out that test case prioritization (TCP) can enhance the effectiveness of software testing and improve the efficiency of the entire testing process.

TABLE I. TEST CASE FAILURE STATISTICS

Test Case	Fault revealed by test case									
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
TC1	√			√	√			√	√	√
TC2	√				√	√	√			
TC3					√	√	√	√	√	√
TC4	√	√	√	√	√				√	
TC5	√	√	√		√			√		

TCP sorts a set of test cases according to the preferred attributes [1], [2] to achieve early optimization. Prioritization is done to detect failures as early as possible. Obviously, as shown in Table 1, by sorting the test cases, we can see that TC3-TC4 is far superior to any others, because all faults can be detected earlier in the order of TC3-TC4. It is difficult to determine which tests will show errors in real problems. Therefore, the priority step of a test case depends on several methods, and it is expected that a selected method will expose errors as early as possible. There are many aspects to test case prioritization methods. Paper [6] mentioned 8 dimensions to describe, these TCP technologies are based on their commonalities in the selection process, input type and output type [12].

Where, the frequency of use and the probability of failure are particularly important. TCP can implement metrics by using strings of input data. These metrics can distinguish test cases based on character differences, and then prioritize according to a predefined fitness function. In recent years, existing work has prioritized test cases through information related to test cases. For example, test case input and historical version test case sequences, software testers can prioritize test cases before software development is successful. This reduces overall test time. TCP can be started immediately after the software is formed. Although many TCP technologies have been proposed, there is still room for improvement in terms of time execution

[1]https://sir.csc.ncsu.edu/content/sir.php

978-1-7281-5904-1/20/$31.00 ©2020 IEEE
DOI 10.1109/ICCEA50009.2020.00099

performance [4], [5]. Improvements are needed, especially in terms of APFD and execution time [7].

II. Related work

In TCP technology, string metric, [8] which is used by applications in tasks such as information retrieval, text classification, document clustering, topic detection, topic tracking, question generation, question answering, essay scoring, short answer scoring, machine translation, and text summary [9], is mainly used to calculate the difference between characters. And then the priority of test cases is determined according to the size of the difference. String metrics can be further classified based on their metric calculation strategy. When calculating the distance between test cases, you need to use the specificity and reliability of the string distance in the current priority target. In the existing work of [10], only the distance of one string is used, which may not solve the problem of redundancy, because the equally spaced weights of test cases are common.

After character string calculations between test cases are completed using string metrics, an effective test case prioritization algorithm is needed to arrange test cases based on the weight of character differences between test cases. Studies in [3], [10] show that the swarm intelligence (SI) algorithm has a significant effect on TCP ranking. Therefore, the paper [5],[11] and [12] used the firefly algorithm in the test case priority problem and achieved good results in SI. But this method also has the following problems: (1) there is a problem of large search space and too many iterations; (2) only the string metric is used, and the metric is too single; (3) it is easily affected by local optimal influences.

In view of the above problems, the main work of this article is as follows:

1) This paper proposes an improved firefly algorithm to establish a hybrid search space based on the importance of test cases and edit distances. Through the combination of local search and global search, the search results are more effectively moved to higher priority targets. To get the best possible test sequence;

2) Apply the improved firefly algorithm to three different benchmark test procedures for experimental verification;

3) Compare the experimental results obtained with existing work in APFD and execution time.

The rest of the paper is arranged as follows: The third part describes the construction process and evaluation criteria of the hybrid model based on test case importance and edit distance; the fourth part proposes the firefly algorithm based on the hybrid model and its application in TCP; The fifth part verifies the algorithm in this paper and compares it with the existing work. The sixth part summarizes this paper.

III. Model construction process and evaluation criteria

A. Hybrid Model Building Process

The algorithm proposed in this paper is a hybrid model based on the edit distance of test cases and the importance of data. The edit distance is measured by the correlation coefficient matrix between test cases, and the data importance is measured by the word frequency inverse text frequency (TFIDF) of the test data. The specific process is as follows:

The content in test cases is measured by string spacing, and the string spacing of different test cases is measured by edit distance. Use the formula shown below to measure:

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ min\begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1(a_i \neq b_j) \end{cases} & \text{others} \end{cases} \quad (1)$$

Where, $lev_{a,b}(i,j)$ represents the edit distance between two test cases a and b when the subscripts of the string are respectively i and j, a_i represents the character at position i in a, and b_j represents the character at position j in b. From the calculated distances between a and b calculated above, we can get the overlap distance between a and b as

$$\max(i,j) - lev_{a,b}(i,j) \quad (2)$$

which is denoted as P_i, Editing distance $Q_i = lev_{a,b}(i,j)$ is recorded as Q_i. Therefore, we can get the similarity coefficient $S_{i,j}$ between different test cases:

$$S_{i,j} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} {P_i}^2 + \sum_{i=1}^{d} {Q_i}^2 - \sum_{i=1}^{d} P_i Q_i} \quad (3)$$

According to the similarity coefficient $S_{i,j}$, the similarity between the i-th test case and the j-th test case can be obtained, so the similarity coefficient matrix M_s is obtained:

$$M_s = \begin{bmatrix} 0 & S_{1,2} & \cdots & S_{1,n} \\ S_{2,1} & 0 & \cdots & \vdots \\ \vdots & \vdots & \ddots & S_{n-1,n} \\ S_{n,1} & \ldots & S_{n,n-1} & 0 \end{bmatrix} \quad (4)$$

Calculate the importance of test data, including the following steps: Count the number of occurrences of each test data in all test cases $n_{k,j}$. For a specific test case, its importance expressed as:

$$tfidf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} * log \frac{|D|}{|\{d: d \ni t_{i,j}\}|} \quad (5)$$

In the above formula, $n_{i,j}$ is the number of occurrences of the term in the test case d_i, and the denominator is the sum of the occurrences of all words in the file d_i. $|D|$ represents the total number of test cases, $|\{d: d \ni t_{i,j}\}|$ represents the number of test cases containing test data $t_{i,j}$, and $t_{i,j}$ represents the i-th test data for the j-th test case.

Further, Further, initialize the brightness of the firefly agent (FA):

$$Brightness_{i,j} = W_i / \left(\frac{n_{i,j}}{\sum_k n_{k,j}} * log \frac{|D|}{|\{d: d \ni t_i\}|} \right) \quad (6)$$

Where, W_i represents the weight of test case i.

B. Evaluation Criteria

In this paper, the average fault detection rate (APFD) is used to quantify the fault detection rate. APFD [1],[2] is widely used in TCP experiments as a measure to quantify the rate at which test cases are detected by optimal sequencing. In this study, the APFD value is between 0 and 100, and the larger the value, the

better the failure display rate. The calculation formula of APFD value is as follows:

$$APFD = 1 - \frac{TF_1+TF_2+TF_3+\cdots+TF_n}{n\times m} + \frac{1}{2n} \quad (7)$$

Where, n represents the number of test cases in test case set T, m represents the number of defects of the software under test, and TF_i represents the location of the first test case detected defect I in T. The larger the value of APFD, the faster the defect detection rate.

IV. META-HEURISTIC TEST CASE PRIORITIZATION METHOD BASED ON HYBRID MODEL

According to the description in the previous section, preconditions and preset parameter values are provided for the algorithm proposed in this paper. This section mainly describes the specific process of the algorithm. Where, part A introduces the basic concept of the original firefly algorithm and the principle of the algorithm in the TCP environment, and part B describes the specific process of the hybrid model based firefly algorithm.

A. Overview of Firefly Algorithm

Fireflies attract other fireflies nearby by the light they emit. Therefore, by associating the emitted light with the objective function to be optimized, it can be formulated, so that a new optimization algorithm can be formulated. Figure 1 shows a schematic of the Firefly algorithm. The following three assumptions exist in the Firefly algorithm:

1) All fireflies are unisexual.Each firefly attracts or is attracted to another firefly.All fireflies are attracted to any firefly, no difference.

2) The attraction of a firefly is proportional to its brightness.When fireflies are attracted to other species, their brightness becomes a priority for attraction.

3) If no more attractive fireflies can be found nearby, they move randomly.If two or more fireflies have the same brightness, the firefly will move toward one at random.

Figure 1 shows the execution flow of the firefly algorithm to be applied in TCP, taking into account the previously described assumptions. First, the algorithm first defines the objective function. Then calculate the distance matrix between firefly agents and the adjacency relationship of their brightness, and encode them to determine the degree of attraction of each firefly. Subsequent movements of the firefly will be based on its brightness value. Once all fireflies have been visited, the movement stops. All flight paths are recorded. Finally, choosing the shortest flight path is the best test sequence.

B. Improved Firefly Algorithm Based on Hybrid Model

This method is based on the original firefly algorithm, using the degree of importance to search the next candidate node that fireflies may reach in the global, and join the candidate set. Then edit the distance to select the optimal next move destination in the candidate set. The specific process is as follows: By associating brightness with the objective function to be optimized, it can be formulated so that the fitness function of the new optimization algorithm can be formulated:

$$f(x_{i,j}) = \max_k\left\{\frac{W_{i-1}}{Random(\cdot)} * Brightness_{i,j} \middle| i-1 \in Order, i \in unOrder\right\} \quad (8)$$

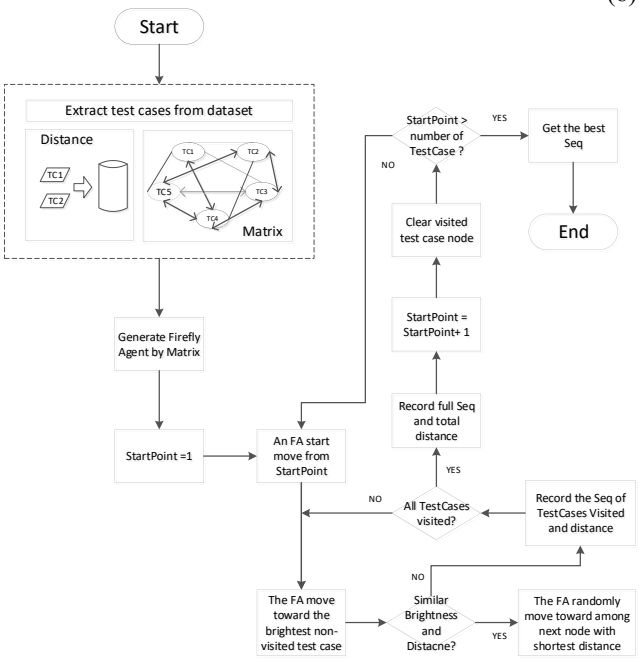

Figure 1. Execution flow of firefly algorithm.

Where, W_{i-1} represents the weight of the test case i-1, Order represents the sequence number of the test case whose priority is determined, unOrder represents the sequence number of the test case whose priority has not been determined, Random (·) = {N-2 < [[N_i-i]-0.1] <N}, where N is the total number of test cases. The fitness value $f(x_{i,j})$ of $x_{i,j}$ is affected by the point to be reached in the next step, so k nodes with the highest fitness value are found from the test cases that have not yet joined in $Set_{candidate}$.

Step1: Update the $Brightness_{i,j}$and the objective function $f(x_{i,j})$ of the firefly agent (FA), including the following steps:

1) Set the fitness function / objective function $f(x_{i,j})$, where $x_{i,j}$ represents the j-th agent of the it-h test case.

2) Use formula (6) to initialize the value of $x_{i,j}$ as the entrance of the firefly agent.

3) X_i represents the node reached by the firefly agent. The update formula of the node distance is as follows:

$$X_i^{t+1} = X_i^t + \beta e^{-\gamma S_{i,j}^2}\left(X_i^t - X_j^t\right) + \alpha\varepsilon^t \quad (9)$$

Where, β and α are constants, β is the absorption rate of light, usually taken as 1, $\alpha \in [0, 1]$. Here ε is a random factor obeying a uniform distribution, γ is the attraction coefficient, and $S_{i,j}$ is the node that the similarity coefficients of X_i and X_j are given by the matrix M_s.

Step2: The selection strategy of the next arriving node, including the following steps:

1) Calculate the distance between X_i^t and all nodes in the $Set_{candidate}$ according to formula (9).

2) Select the node with the smallest distance X_{i+1}^t as the i + 1th sequence.

3) Record the path of FA movement and update the node distance X_{i+1}^t with the hybrid model.

4) Repeat until unOrder is empty.

Step3: Output the optimal test sequence, including the following steps:

1) Change the starting position of the test case and restore the original initial conditions.

2) Repeat steps 1 through 3.

3) Find the optimal test sequence in all flight paths.

The pseudo-code of algorithm execution is given as follows:

TABLE II. PSEUDO-CODE OF ALGORITHM

Algorithm	***Improved Firefly Algorithm Based on Hybrid Model***
Initialize	1) Set the fitness function / objective function $f(x_{i,j})$, where $x_{i,j}$ represents the jth agent of the i-th test case. 2) Use formula (6) to initialize the value of $x_{i,j}$ as the entrance of the firefly agent. 3) Set parameters β, α, γ, ε^t, *MaxGeneration*
Input	Each test case node information X_i, X_i contains the firefly agent information $x_{i,j}$.
Output	optimal test sequence T_{best}
1:	*while (T< MaxGeneration)*
2:	*for i = 1:n*
3:	*for j = 1:n*
4:	*if* $(f(X_i) < f(x_{i,j}))$
5:	X_j *Join* $Set_{candidate}$
6:	*end if*
7:	*end for j*
8:	*for k = 1:* $Set_{candidate}$
9:	*calculate the* $distance_{i,k}$
10:	*end for k*
11:	*choose the best candidate, update the distance*
12:	*sort and record the current best sequence* T_{cur}
13:	*end for i*
14:	*output optimum sequence* T_{best}
15:	*end*

V. SIMULATION AND PERFORMANCE ANALYSIS

A. Experimental Environment and Process

The data involved in this experiment are from three benchmark program test sets of Software-artifact Infrastructure Repository (SIR), namely flex, gzip and grep. Where, flex contains 21 error versions and 567 test cases, gzip contains 55 error versions and 217 test cases, and grep contains 17 error versions and 809 test cases.

This experiment is mainly divided into two phases. Phase one: Priority order of test cases, which is mainly used to determine the priority of test cases, and provide test sequences for subsequent verification phases. Phase two: Testing and verification. The performance and effect of the algorithm are measured through the test sequence generated in the previous phase. The algorithm simulation platform of this paper is built on a processor with Intel (R) Core (TM) i5-8300H CPU @ 2.30Ghz, 8GB of memory, and Windows 10 64bit operating system. The testing and verification platform was conducted in

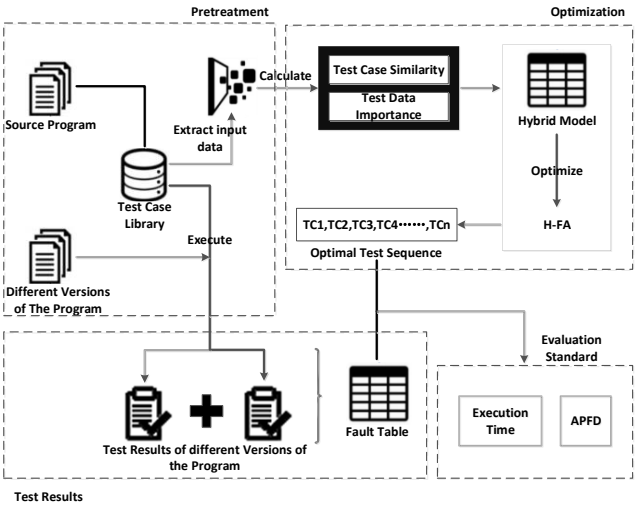

Figure 2. Overview of the overall experimental process.

a Linux environment built in VMware Workstation. The overall process is shown in the figure 2.

B. Display and Analysis of Experimental Data

In this experiment, a hybrid model is established for the test case edit distance and importance metric (TFIDF), and then a test sequence is generated based on the improved firefly algorithm. The test verification was performed in three benchmark testing programs, respectively, and compared with the particle swarm algorithm (PSO), greedy algorithm (GREEDY), and the original firefly algorithm that have achieved better performance in the test case ranking technology.

As shown in the figure 3-5, a box plot is used to show the overall APFD experimental results of different algorithms on three benchmark tests. From the graphs and tables shown in the figure below, it can be found that although the scales of the three benchmark tests are different, the proposed algorithm and the FA algorithm both show good performance.

Where, in the FLEX data set, except for the standard deviation of H-FA, which is slightly worse than FA, other indicators perform best. In the GZIP data set, although H-FA is slightly inadequate in terms of mean, median, and standard deviation, it is significantly better than the other two algorithms. It is worth mentioning that in the largest test set GREP (with 809 test cases), H-FA has achieved good performance in various indicators.

In order to easily compare the performance of each algorithm on the benchmark test, iterate 20 times for each algorithm and calculate the APFD value. As shown in figure 6-8, the algorithm proposed in this paper does not show large fluctuations on the three benchmark test procedures, showing good adaptability and stability. This can also be confirmed from the standard deviation index. The search strategy proposed in this paper performs well on APFD in general, because the choice of the firefly flight path of this method depends on the candidate set $Set_{candidate}$, which avoids the large search space problem caused by global search, which can more Quickly determine the flight path.

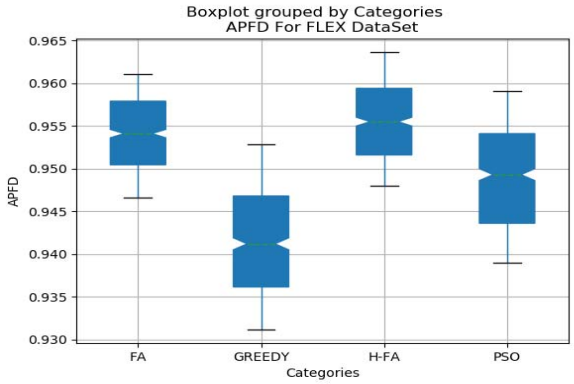

Figure 3. APFD for FLEX program.

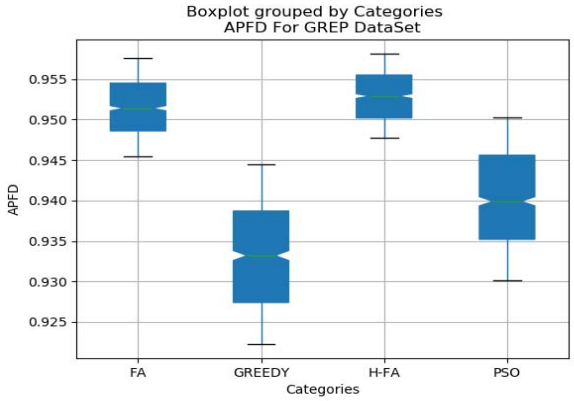

Figure 4. APFD for GREP program.

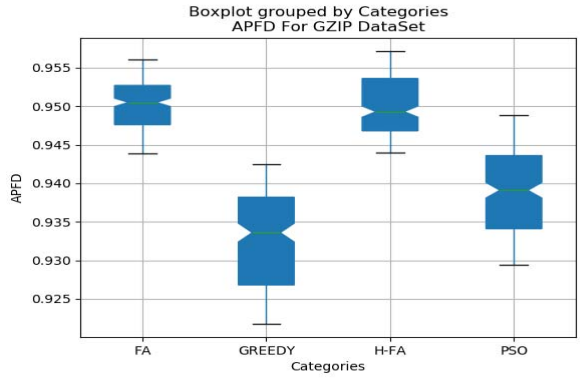

Figure 5. APFD for GZIP program.

In addition, Table 3 shows the execution time of different algorithms on each benchmark test program. According to the data in the table, it can be seen that the execution time of the Firefly algorithm is better than the other two algorithms. The improved firefly algorithm proposed in this paper performs better in large-scale data sets such as Flex and Grep, and does not perform as well as the original firefly algorithm in smaller

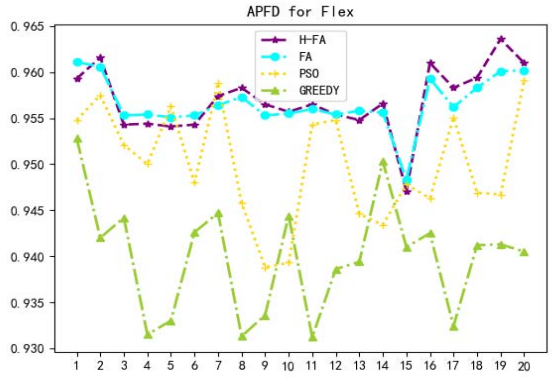

Figure 6. APFD graph for Flex program.

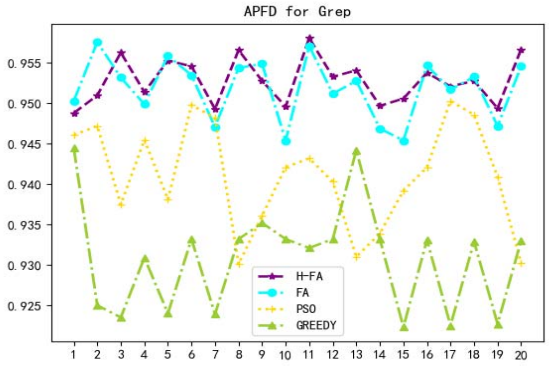

Figure 7. APFD graph for Grep program.

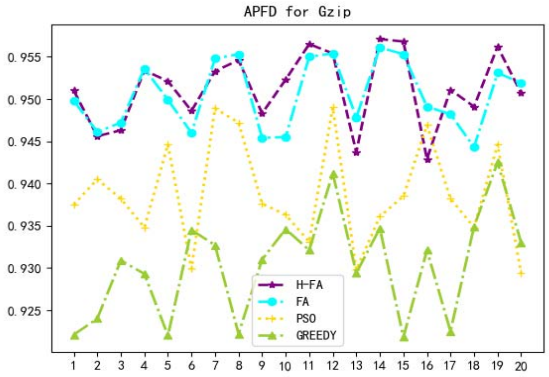

Figure 8. APFD graph for Gzip program.

data sets just like Gzip. The reason for this may be that in the case of a small number of test cases, the effect of the candidate set strategy proposed in this paper is not obvious enough, but after the number of use cases increases to a certain extent, the advantages of this method can be reflected.

TABLE III. BENCHMARK TEST PROGRAMS

Algorithm	Average Time Execution(sec)					
	Flex		Gzip		Grep	
	Mean APFD	Time	Mean APFD	Time	Mean APFD	Time
H-FA	0.9556	230	0.9498	195	0.9528	239
FA	0.9540	235	0.9501	192	0.9515	252
PSO	0.9489	350	0.9389	321	0.9402	361
GREEDY	0.9415	265	0.9325	232	0.9331	279

In general, the algorithm proposed in this paper has verified the effectiveness of the H-FA algorithm on the test suite through experiments on three benchmark test programs, while showing good efficiency. Compared with other optimization algorithms, a higher APFD score is obtained.

VI. CONCLUSION

Aiming at the priority of test cases in software testing, this paper studies the existing meta-heuristic optimization methods, and aims to improve the test suite test efficiency and reduce the test cost, and proposes a meta-heuristic test based on a hybrid model Use case ranking method. The average fault coverage of the three benchmark test programs reached 95.56%, 94.98%, and 95.28%, respectively. The method achieved the best results on both the Flex and Grep datasets, and compared with the best algorithm execution time. The algorithm is 5s and 13s less, respectively. The experimental results prove that H-FA has a better performance in the test case priority ranking problem.

ACKNOWLEDGMENT

This work was financially supported by 2019 Industrial Technology Foundation Public Service Platform Project "Public Service Platform Construction for Standard Verification and Testing in the Field of Internet of Things" (No.2019-00894-1-1).

REFERENCES

[1] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, ‘‘Test case prioritization: An empirical study,’’ in *Proc. IEEE Int. Conf. Softw. Maintenance(ICSM)*, Aug./Sep. 1999, pp. 179–188.

[2] S. Elbaum, A. G. Malishevsky, and G. Rothermel, ‘‘Test case prioritization:A family of empirical studies,’’ *IEEE Trans. Softw. Eng.*, vol. 28, no. 2,pp. 159–182, Feb. 2002.

[3] D. Gao, X. Guo, and L. Zhao, ‘‘Test case prioritization for regressiontesting based on ant colony optimization,’’ in *Proc. IEEE Int. Conf. Softw.Eng. Service Sci.*, Sep. 2015, pp. 275–279.

[4] A. Shahbazi and J. Miller, ‘‘Black-box string test case generation througha multi-objective optimization,’’ *IEEE Trans. Softw. Eng.*, vol. 42, no. 4,pp. 361–378, Apr. 2016.

[5] P. R. Srivatsava, B. Mallikarjun, and X. S. Yang, ‘‘Optimal test sequence generation using firefly algorithm,’’ *Swarm Evol. Comput.*, vol. 8,pp. 44–53, Feb. 2013.

[6] Y. Singh, ‘‘Systematic literature review on regression test prioritization techniques,’’ *Informatica*, vol. 36, pp. 379–408, Dec. 2012.

[7] Praveen Ranjan Srivastava, Optimal software release using time and cost benefits via fuzzy multi-criteria and fault tolerance, Journal of Information Processing System 8 (1) (2012) 21–24.

[8] Y. Ledru, A. Petrenko, S. Boroday, and N. Mandran, ‘‘Prioritizing test cases with string distances,’’ *Autom. Softw. Eng.*, vol. 19, no. 1, pp. 65–95, 2012.

[9] W. H. Gomaa and A. A. Fahmy, ‘‘A survey of text similarity approaches,’’ *Int. J. Comput.*, vol. 68, no. 13, pp. 13–18, 2013.

[10] M. Khatibsyarbini, M. A. Isa, and D. N. A. Jawawi, ‘‘A hybrid weightbased and string distances using particle swarm optimization for prioritizing test cases,’’ *J. Theor. Appl. Inf. Technol.*, vol. 95, no. 12, pp. 2723–2732, 2017

[11] Amir Hossein Gandomi, Xin She Yang, Amir Hossein Alavi, Mixed variable structural optimization using Firefly algorithm, Computers and Structures 89 (23–24) (2011) 2325–2336.

[12] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, H. N. A. Hamed and M. D. Mohamed Suffian, "Test Case Prioritization Using Firefly Algorithm for Software Testing," in *IEEE Access*, vol. 7, pp. 132360-132373, 2019.