



Research on reducing fuzzy test sample set based on heuristic genetic algorithm

Zhihua Wang , Manman Cheng & Yongjian Wang

To cite this article: Zhihua Wang , Manman Cheng & Yongjian Wang (2020): Research on reducing fuzzy test sample set based on heuristic genetic algorithm, Systems Science & Control Engineering, DOI: [10.1080/21642583.2020.1843087](https://doi.org/10.1080/21642583.2020.1843087)

To link to this article: <https://doi.org/10.1080/21642583.2020.1843087>



© 2020 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group



Published online: 20 Nov 2020.



Submit your article to this journal [↗](#)






View related articles [↗](#)



View Crossmark data [↗](#)

Research on reducing fuzzy test sample set based on heuristic genetic algorithm

Zhihua Wang ^a, Manman Cheng ^a and Yongjian Wang ^b

^aSchool of Cyberspace Security, Zhengzhou University, Zhengzhou, People's Republic of China; ^bNational Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, People's Republic of China

ABSTRACT

Fuzzy testing is the most effective method for vulnerability mining, which can deal with complex programmes better than other vulnerability mining techniques and has strong scalability. However, in the large-scale vulnerability analysis test, the fuzzy test input sample set faces the challenges of low quality, high repeatability and low availability etc. Therefore, this paper studies the input sample set and proposes heuristic genetic algorithm. By using 0–1 matrix, the genetic algorithm is improved with a consideration of practical problems and the execution path for sample set is selected and compressed through approximation algorithm, thus obtaining a smallest sample set and accelerating the efficiency of fuzzy test. Experimental results show that the proposed method is effective.

ARTICLE HISTORY

Received 7 July 2020
Accepted 24 October 2020

KEYWORDS

Refining set; fuzzy test; set coverage; genetic algorithm; vulnerability analysis

1. Introduction

With the rapid development of information and communication technology, the network and information security issues have become increasingly prominent, causing widespread concern in all aspects of society. Software vulnerability mining technology (Lin et al., 2019) plays an important role in the field of information security, and fuzzy test (Peng & Tian, 2014) is one of the most effective solutions adopted by it. However, the traditional method of fuzzing test has problems such as low efficiency and blindness, so how to improve the defects of fuzzing has gradually become the current research hotspot. As far as the fuzzy test sample set is concerned, it can be optimized to reduce the number and improve the quality of the sample sets, thereby improving the efficiency and quality of the fuzzy test. This paper studies the reduction of the fuzzy test sample set. With the application of 0–1 matrix, we propose a solution to the sample set reduction based on the heuristic genetic algorithm.

2. Foreword

2.1. Research background


The data set is the foundation of information research in the current big data era. The quality of the data set directly affects the process and result of the experiment. However, the high repeatability makes the data set a prominent problem, which leads to a large amount of redundant work and low efficiency in the experiment. The sample

set of fuzzing in vulnerability mining technology is a typical example. In the fuzzing test, the data set obtained through multiple ways is applied directly without any processing in the traditional method, which results in a greatly reduced efficiency of the fuzzing test. Therefore, it is necessary to preprocess the sample set to optimize the fuzzy test, which is also one of the more intuitive methods.

2.2. Research status

Testing input is the first step of fuzzing. The quality of the test sample would affect the result of the entire fuzzing test. At present, some scholars have conducted research on the simplification of fuzzy test data sets, and realized the optimization to a certain extent.

Using the idea of set coverage to optimize data sets, Jinxin et al. (2016) designed a greedy approximation algorithm to optimize the input sample set of the fuzzy test. The minimum sample set is obtained on the premise that the sample set coverage is unchanged. However, considering the sample quality (high coverage), the results cannot achieve the optimal local meaning; Qian and Jiayong (2017) also designed a greedy approximation algorithm to simplify the fuzzy test input sample set, but the reduced set obtained by the algorithm is not the smallest sample set with mutual coverage and duplication. For the reduction or selection of sample sets, the precise algorithm can find the optimal solution of the set coverage problem, and the approximate algorithm can

CONTACT Zhihua Wang  12156719@qq.com

find the solution with quality assurance, but their execution speed is relatively slow, where only small-scale examples can be solved.

Use machine learning techniques to optimize data sets. Machine learning has been widely used in many fields such as image recognition and speech recognition, making a lot of breakthroughs. In recent years, researchers have also begun to use machine learning technology to solve problems in software vulnerability mining, assisting corresponding vulnerability mining tools and systems in experience and knowledge extraction from massive vulnerability-related data. Then based on the training the new samples are classified and predicted to improve the accuracy and efficiency of software vulnerability mining. Bhme et al. (2017) uses simulated annealing algorithm to allocate higher energy to the test input that can approach a specific target position, and prefers to select high-energy seed files for mutation, but the results are random during the search and are not targeted; Veggalam et al. (2016) uses context-free grammar as input and generates a parse tree, then extracts code fragments from the test set, and uses genetic evolution algorithms to reorganize the code fragments to generate new test input. However, this method is limited and local since it may not include the collection of code blocks of the initial sample set; Bhme et al. (2016) applies an approach of modelling the fuzzy test problem as a Markov model, and uses a specific strategy to guide AFL to select low-frequency paths and files for mutation, but variance of fuzzy testing should involve the path of the entire sample set, rather than a partial overview; Wang et al. (2017) proposed a quality-aware test case prioritization technique, which prioritizes the input of high-quality seeds. It is a good way to directly defines the high-quality seeds, but using quality perception to obtain them is relatively expensive; Guoqing et al. (2016) and others uses the fuzzy K-means algorithm to find similar test cases in order to reduce the duplication of test cases. The K-means algorithm itself is a clustering method within machine learning, where the data needs to be trained adequately and divided into several categories on the premise that there is no predefined rules, but the fuzzy test sample set itself belongs to a large category. The use of the K-means algorithm cannot play the role of clustering algorithm calling for more human interventions, thus reducing the efficiency of fuzzy test.

Research on fuzzy testing based on genetic algorithm. The traditional fuzzing test method needs to mutate all bytes of the original input data when generating test cases, resulting in a large number of invalid test cases. In response to this shortcoming, Xiao (2019) proposed an improved method of dynamic taint analysis, using taint analysis and other techniques to obtain the execution

path of data, genetically coding the fields related to code coverage, performing selection mutation process of genetic algorithm and boundary value assignment on the relevant fields of dangerous operations, and finally generating new test cases; Longlong et al. (2018) proposed a binary programme fuzzy test method based on genetic algorithm to solve the problem of low code coverage caused by high execution path repetition rate of test data generated from mutation in current binary programme fuzzy test. At present, the most direct way to optimize fuzzy testing is to compress its sample set, and there are many ways to do so. References (Xiao, 2019) and (Longlong et al., 2018) deal with it mainly from the perspective of code coverage, using the analysis of taints and other related methods to optimize the fuzzing test.

The above literatures show diversity of fuzzy test optimization, and also illustrate that not only the approximation algorithm can reduce the fuzzy test sample set, but also that different optimization algorithms can be used from different angles to improve the efficiency of fuzzy test. Therefore, this paper considers adding some methods on the basis of genetic algorithm to obtain better data sets.

2.3. Research content

The heuristic genetic algorithm is used to study the fuzzy test sample set. By analysing the test cases generated by the fuzzy test mutation, it is found that the sample is realized by destroying its basic code block when the mutation is performed, and different samples will have the same basic code block, Therefore, it is considered to transform the study of sample set into the problem of minimum set coverage from the source of its variation or mutation. For general samples with many attributes, it is relatively simple to construct a matrix. However, in most cases, each fuzzing sample is actually a single and indivisible data set. Considering the characteristics and problems of the fuzzy test sample set, the research contents of the sample reduction based on genetic algorithm are as follows:

Simplification of the sample set: the process of reducing a large number of data sets. For the samples in the fuzzing test set, the sample repetition not only appears in the exact same samples, but also from the root of the variation/mutation, the mutual coverage of the basic blocks between the samples causes the mutation to produce a large number of repeated test cases. Therefore, in the process of sample selection, this article stipulates that the sample with the largest number of basic blocks of code (the samples with more code blocks can get more test cases when mutating) is selected first. The selection is then based on sample performance until the base code

block of the new sample set can override the base code block of the original sample set.

Set coverage problem (SCP): In fuzzy testing, any sample set can be transformed into a minimum set coverage problem, and the minimum cover set is a typical NP-hard problem (Bergantiños et al., 2020; Khulood & Jonathan E, 2018), which is difficult to find the optimal solution. The set coverage problem can be described in the following form: Let $A = (a_{ij})$ be a 0–1 matrix with m rows and n columns. $C = (c_j)$ is an n -dimensional integer vector. Let $M = \{1, \dots, m\}$, $N = \{1, \dots, n\}$ denote the row vector and column vector of matrix A . The value $c_j (j \in N)$ represents the cost of a column. Without loss of generality, assuming $c_j > 0, j \in N$. if $a_{ij} = 1$, it is considered that column $j \in N$ covers row $i \in M$. The set coverage problem (SCP) requires a minimum cost subset $S \subseteq N$, so that each row $i \in M$ is covered by at least one column $j \in S$. A natural mathematical model of SCP can be described as $v(SCP) = \min \sum_{j \in N} c_j x_j$ Subject to $\sum_{j \in N} a_{ij} x_j \geq 1, i \in M, x_j \in (0, 1) (j \in N)$, if $x_j = 1 (j \in S)$, else $x_j = 0$.

The traditional approximation algorithm can be used to reduce the sample set, but the applicability of the sample in practical problems is not considered. Genetic algorithm has high search efficiency, while approximate algorithm can obtain the optimal solution of the effective sample set as much as possible. This article will extend on the basis of genetic algorithm, combining approximation algorithm and genetic algorithm to form heuristic genetic algorithm. Specifically, the approximation algorithm is the core of heuristic calculation. Genetic algorithm, as a search algorithm, is mainly used to accelerate the search speed.

3. Related algorithms and evaluation indicators

3.1. Approximation algorithm

Approximation algorithms are algorithms used to find approximation methods to solve optimization problems. Approximation algorithms are usually related to NP-hard problems. Since it is impossible to solve the NP-hard problem by an effective polynomial time exact calculation, a polynomial time suboptimal solution is obtained. Unlike heuristic algorithms, only reasonable solutions are usually found fairly quickly, requiring a provable solution quality and a provable run time range, even if the approximation algorithm usually yields a quality-assured solution. The approximation algorithm refers to the scenario or accuracy of using the relevant algorithm to solve some practical problems, and the solution given is the theoretical optimal solution. In the reduction of the sample set, the approximation algorithm refers to giving the smallest sample set of samples as accurately as possible, and

obtaining high-quality samples is the only standard of the approximation algorithm (Liu et al., 2020).

3.2. Genetic algorithm

Genetic algorithm is a bionic algorithm for searching optimal solutions based on the principle of biological evolution. It simulates the natural process of gene recombination and evolution, programming the parameters of the problem into binary codes or decimal codes (also can be compiled into other hexadecimal codes), that is, genes, which then form a chromosome (individual). The operations (generation inheritance) similar to natural selection, pairwise crossover and mutation would be performed iteratively on chromosomes until the final optimization result is obtained. The steps of the genetic algorithm are as follows:

- (1) Initialize the 0th generation population P_0 .
- (2) Perform steps (i)–(v) on the iteration P_i of the i -th population until the stopping criterion is satisfied:
 - (i) Calculate the fitness value of each individual in P_i , and sort all individuals according to fitness value;
 - (ii) Add the individual with the best fitness value in P_i to P_{i+1} ;
 - (iii) Select two parent bodies in P_i according to the order of the fitness value;
 - (iv) Select the crossover operator or mutation operator according to the probability to perform genetic operations on the two parents, and add the generated individuals to P_{i+1} ;
 - (v) If the scale of P_{i+1} is equal to P_i , then $i \leftarrow i + 1$ and go to (2), otherwise go to (iii).
- (3) The individual with the best fitness in the final population is used as the result of the genetic algorithm.

3.3. Evaluation index

The evaluation index of the algorithm is a measure of the pros and cons of the algorithm, generally considered from the time complexity and space complexity.

- (1) Time complexity refers to the time complexity of the algorithm. Suppose the initial sample set of the fuzzy test is n , and the fuzzy test time function is $f(n)$, so the time complexity of the algorithm is also written as $T(n) = O(f(n))$. As the number of fuzz tests increases, the growth rate of fuzz test execution time is positively related to the growth rate of $f(n)$.
- (2) The space complexity of the algorithm refers to the memory space consumed by the algorithm. The calculation and representation methods are similar

to those of time complexity, which are generally expressed by asymptotic complexity.

For the heuristic genetic algorithm proposed in this paper, the optimization process uses matrices and sets as an auxiliary, which theoretically occupies more space than the traditional genetic algorithm, leading to a higher time complexity is higher than the general genetic algorithm. Therefore, the algorithm proposed in this paper is mainly measured from the time complexity.

4. Sample set reduction based on heuristic genetic algorithm

The traditional genetic algorithm determines the choice of the parent according to the calculated fitness value when performing chromosome selection. For general problems, the optimal chromosome will eventually be selected as the optimal solution in the theoretical sense of a function or algorithm. However, starting from the practical problem, not only should we choose the appropriate parent to generate the next generation, but we should also choose the optimal solution that meets the conditions of practical problem (Wei et al., 2020). Therefore, for the problem of simplifying the fuzzy test sample set, the heuristic algorithm to be defined in this paper is an algorithm for chromosome selection and final chromosomes set generation. The genetic algorithm itself is a search algorithm, and what this article does is to integrate the approximation algorithm into the genetic algorithm. The genetic algorithm is regarded as an algorithm framework, and the heuristic algorithm obtained is to improve it.

4.1. Chromosome representation

In order to ensure that better samples are obtained on the basis of the minimum number of sample sets (the number of basic code blocks is large and the mutation ability is good), during the implementation process, the 0–1 matrix (Shitov, 2018) is introduced, and the elements of the vector x are 0 or 1. The chromosome structure is represented by an n -bit binary string, where n is the number of columns in matrix A and the value '1' at the i -th bit means that the i -th column is selected.

During the fuzzing test, each programme has its own basic code blocks, and the content found according to the address of the basic blocks is the corresponding code block. It is more convenient to obtain the address information of the basic block, so the basic block address is taken as the research object (each basic address block is equivalent to the gene in the genetic algorithm). Each sample is regarded as a set of elements with basic code

block addresses (samples are equivalent to chromosomes in genetic algorithms). If there is a basic address block in the sample, it is represented by '1', otherwise it is represented by '0', and all the samples form a 0–1 matrix with 0 or 1 as the elements. The '1' in the matrix represents the genes of the chromosome, and the set of genes in each column is equivalent to one chromosome.

4.2. Heuristic genetic algorithm to improve chromosome

The selected chromosome will have both its own unique genes and countless overlapping genes across chromosomes. When performing set coverage, the smallest coverage set is easy to find. However, the difficulty is not only to obtain the smallest set of chromosomes but also to ensure that the redundancy of the set is minimum. In other words, the key of the algorithm design is how to find the chromosomes containing the same genes and ensure the minimum redundancy in the meanwhile given that a gene could be included in multiple chromosomes.

In the actual genetic process, due to the fact that the chromosomes generated by the mutation operation are not suitable for the problem set, this article no longer repairs the mutant genes of new individuals, but discards the new individuals generated by the mutation directly (Since the original data are similar, the probability of new gene appearing is very small). After the genetic algorithm is executed, all the generated chromosomes will be retained, so the role of the heuristic algorithm includes the following two points:

- (1) Eliminate the redundancy caused by gene duplication;
- (2) Choose better quality chromosomes (more genes and richer gene combinations).

Next, we will give the idea of the heuristic algorithm, as follows:

- (1) Chromosome and gene representation in the matrix

In the 0–1 matrix, one column represents a chromosome, '1' in the matrix means that a certain chromosome contains this gene, and '0' means that this chromosome does not contain this gene. The algorithm uses the priority idea of performance-cost ratio: Performance refers to the number of columns containing '1' in the set corresponding to a gene in the chromosome. The more the number, the more rows covered by the column, indicating the higher performance of the chromosome. Suppose that the set of all chromosomes generated when using a genetic algorithm for a group of chromosomes is: $S = s_1,$

s_2, s_3, s_4, s_5 , where $s_1 = 2, 4, 6$ and $s_2 = 1, 4, 6, s_3 = 2, 3, 5, 6, s_4 = 2, 3, 7, s_5 = 1, 7$ (use numbers to directly represent the genes and positions they contain). The following example will be used to illustrate the idea of combining approximation.

(2) Set coverage eliminates redundancy

Suppose that A is the initial set (corresponding to the initial population in the genetic algorithm), and B is the final set that can cover set A . Set C , is used to store rows numbers that have no genes and are not covered after a chromosome selection. Set D is used to store all the unique genes in set A that are not repeated. Based on the above example, then $A = s_1, s_2, s_3, s_4, s_5$, sets B and C are both empty sets, and set $D = 1, 2, 3, 4, 5, 6, 7$.

- (i) Calculate the performance η of each chromosome in set A , that is, the number of '1' is contained in each column. The more '1', the more rows the chromosome covers, the more genes it contains, and the higher its performance. At this time, the performance of each chromosome is: $\eta_{s_1} = \eta_{s_2} = \eta_{s_4} = 3, \eta_{s_3} = 4, \eta_{s_5} = 2$.
- (ii) Select the chromosome with the highest performance and put it into set B , count the genes contained in the chromosome, delete it in set D , calculate the genes it does not contain, and store the row numbers of these genes in set C . Therefore, the performance of each chromosome has been given in step (i), where $\eta_{s_3} = 4$ is the highest. Then s_3 is put into set B , and the genes contained in s_3 are deleted in set D , then set $B = s_3$, set $D = 2, 3, 5, 6$, set $C = 1, 4, 7$.
- (iii) After each chromosome selection, set B will increase a chromosome, the row number recorded in set C will decrease with the increase of chromosomes in B , until it is empty. The genes in set D will also decrease with the increase of genes in set B until it is empty.
- (iv) When set C and set D are empty, it means that set B contains the set of all genes in set A . At this time, the set of chromosomes in set B is the minimum set required. Also, set C and set D are not empty, then we continue to the previous step, where $\eta_{s_1} = \eta_{s_2} = \eta_{s_4} = 3$. Between s_1 and s_2 , s_2 is selected (see the next section for the reason), and now s_4 contains new genes which is then added to set B . At this time, set $B = s_2, s_3, s_4$, set C and D are empty sets. There is no need to consider the genes contained in chromosome s_5 , because the genes selected in set B have already s_5 coverage. Then set

$B = s_2, s_3, s_4$ is the minimum set coverage of chromosome set S .

(3) The process of high-quality chromosome selection

In the process of chromosome selection, some chromosomes may have the same performance, the key of algorithm design is to select the best chromosome among the chromosomes those with the same performance. For example, suppose that there are two chromosomes with the same performance in set A . $s_1 = 2, 4, 6, s_2 = 1, 4, 6$, and currently set B contains a chromosome $s_3 = 2, 3, 5, 6$. Between the candidates s_1, s_2 to be selected. s_2 is more suitable than s_1 according to the variation of code block in fuzzy test. Here, row numbers are used instead of rows themselves that can be covered by genes. When the fuzzy test samples are mutated, they will randomly generate test cases according to their own code block random mutation. In chromosome s_1 , genes 2 and 6 have appeared in set B . When the code block is mutated, it is equivalent to a combination of genes. Specifically, when genes 3 and 5 are not mutated, the remaining combination of genes 2 and 6 has the same effect as chromosomal S_1 on the basis of no variation of gene 4. In contrast, except for gene 4, gene 1 and 6 of chromosome s_2 will generate new combinations when they are mutated, and then generate new test cases. On the basis of eliminating redundancy, when selecting the next chromosome, chromosomes with the same performance will appear, but the genes they contain are different. Therefore, it is suggested to make a 'difference' between the chromosomes with the same performance waiting for selection and the selected chromosomes in set B , and record the number of elements in the 'difference' set before making the selection decision. According to the above rules, the 'difference' of the number of elements is more if the number of chromosomes in the set is different from the selected chromosome, and then the corresponding chromosome will be selected as the next candidate chromosome.

Compared with the traditional genetic algorithm, the heuristic genetic algorithm needs to preprocess the population. To be specific the chromosomes needed by the heuristic genetic algorithm are obtained on the basis of the original population with the help of 0–1 matrix. The improvement of our heuristic process is reflected in optimizing the search conditions on the basis of the genetic algorithm, which has little to do with the process of the genetic algorithm itself. The implementation process of the heuristic genetic algorithm is shown in Algorithm 1.

Algorithm 1: Heuristic genetic algorithm implementation process.

```

Input: S //Initial population
        A //Initial sample set
        a,X,P,D,Q //The name of chromosome, a set, etc.
Output: B //Reduced sample set
1. X ← peach(s) //Peach is used to obtain each address block of the sample in the initial population s and store it in X
2. P ← createZeroOneMatrix(X) //Construct a 0 minus 1 matrix based on P
3. A ← pretreatment(P) //Preprocessing the matrix P, obtain the genetic algorithm initial chromosome set
4. Calculate  $\eta$  //Calculate the properties of chromosomes
5. B ← select a ( $\eta$ ) //Take high performance chromosome a from set A and you put it into set B
6. delete D (a) //Set D deletes the genes contained in chromosome a
7.  $a_1(\eta) = a_2(\eta)$ 
   Q-a1 > Q-a2
   Select a1 insert B //*: The two chromosomes had the same performance, and the chromosomes with more new gene combinations
   were selected */
8. Q-a = null //The chromosomes in set B contain all the rows of the original population
9. return B //Return the set B

```

5. Key implementation of heuristic genetic algorithm

There is no difference between heuristic genetic algorithm and traditional genetic algorithm theoretically. The following aspects should also be considered:

- (i) Population size and population initialization: the size of population sample set can be controlled artificially, but not selected randomly;
- (ii) Parent selection: select two sample sets as parents in millions of sample sets, which is the reflection of bionics in genetic algorithm;
- (iii) Crossover rate: make sure that the parent is cross-breeding at some point on the chromosome;
- (iv) Variation rate: variation is also one of the most important sources of new species beyond cross-breeding;
- (v) Stop rule: the traditional genetic algorithm usually has its set limit, but the sample set coverage problem can only complete the selection of chromosome in the process of generating offspring by setting numbers of generations.

(1) Population size and population initialization

When using genetic algorithm to solve practical problems, the size of population has a great influence on the quality of the solution obtained by genetic algorithm. According to the research results in literature (Zhang et al., 2009), if the number of populations is too large, the efficiency of the algorithm will be affected; if the number of populations is too small, the whole algorithm process will not play a role, but the set problem will be complicated. In the problem of sample set reduction, the algorithm in this paper does not limit the number of the population. The number could be self-defined according to the actual situation, but a better population

range could be obtained through tests across different populations.

(2) Selection of Parents

The selection of parents is the opportunity given to each individual in the population to produce offspring. General selection methods include proportion selection, tournament selection and roulette betting. In this paper, the proportion selection method is applied with the basic idea that the probability of individual being selected is directly proportional to their fitness value. The specific implementation is as follows:

- (i) The fitness $f_i (i = 1, 2, 3, \dots, M)$ of each individual was calculated, M is the total population numbers.
- (ii) Calculate the probability that each individual is passed on to the next generation population $p_i = \left(\frac{f_i}{\sum_{i=1}^M f_i} \right)$.
- (iii) Calculate the cumulative probability of each individual $q_i = \sum_{j=1}^i p(x_j)$.
- (iv) A uniformly distributed pseudo-random number r is generated in the $[0, 1]$ interval. If $r < q_1$, then select individual 1, otherwise, select individual K , so that $q_{[k-1]} < r \leq q_{[k]}$ is established.
- (v) Repeat (4) and (5) form times.

(3) Selection of crossover rate

Crossover is the primary method to generate new individuals. Determining a crossover point in the selected parents enables the gene recombination at the crossover point. The crossover rate is used to set the number of chromosomes involved in crossing in the crossing pool. A reasonable crossover rate can ensure that new individuals will be generated continuously in the crossing pool, and

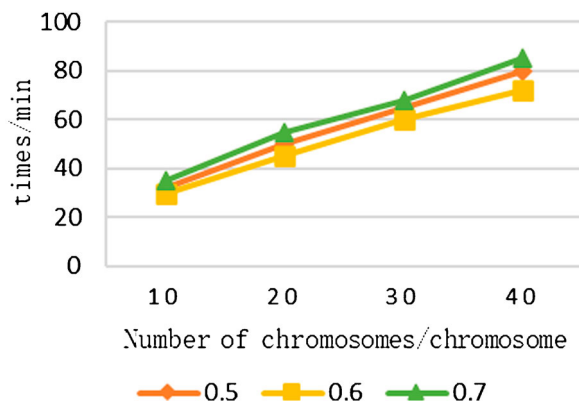


Figure 1. Chromosome time line chart with different mutation rates.

will not generate too many new individuals, leading to the destruction of genetic order. Pereira et al. (2020) Different cross methods correspond to different crossover rates. In view of the current mainstream adaptive methods, no separate experiment would be implemented to determine the value of crossover rate.

(4) Selection of variation rate

Variation is another way to generate new individuals, which can set the number of randomly mutated genes or the rate of variation. Variation rate refers to the ratio of the number of genes mutated in a population to the total number of genes (Gupta et al., 2020). In the sample set reduction, the chromosome containing unique genes could be included into the search rules we set for mutation in advance, so as to ensure the global nature of genetic algorithm. Therefore, a simple experiment is designed to set the appropriate value of mutation rate. The last chromosome with unique genes is mixed with other chromosomes and the heuristic algorithm proposed in this paper is used for selection.

As shown in Figure 1, the time change chart of the genes to be selected by the heuristic algorithm with the increase of the initial chromosome number when the mutation rate is 0.5, 0.6 and 0.7 is given respectively. It takes the least time when the variation rate is 0.6. Too small mutation rate will lead to too few chromosomes participating in the mutation, and cannot input the chromosomes with unique genes into the set in advance; too high mutation rate will lead to too many chromosomes participating in the mutation, which further leads to the increase of mutation time. Therefore, the chosen mutation rate of heuristic genetic algorithm is 0.6.

(5) Stop criteria

In practical operation, genetic algorithm needs to be executed through multiple generations of evolution until the

fitness value tends to be stable, the optimal solution is found or the specified generation numbers are reached. In the process of sample set reduction in this paper, there is no relevant objective function served as the stop criterion. Therefore, when the algorithm obtains the minimum sample set, then we stop the algorithm.

6. Experimental evaluation

6.1. Experiment preparation

(1) Experimental environment

In this experiment, based on the Windows system, MATLAB and peach are used to simplify the fuzzy test sample set of genetic algorithms.

(2) Test environment and tested software

In the fuzzy test experiment, this paper chooses peach as the fuzzy test tool with several reasons to support. First, it is easy to install and configure. Second, it has open sources for instructions. Third, a variety of operations could be realized due to its abundant commands. It is convenient for obtaining the execution path of the sample without referencing other tools. In addition, Mspaint is chosen as the software to be tested.

(3) Data source

The experimental data in this paper come from the pictures of PNG format provided by peach. The files from peach include many duplicate pictures, and there are many intersections between different pictures, which will not only increase the input time of the fuzzy test sample set, but also produce more repetitive and useless test cases during the variation.

6.2. Experimental results and analysis

Section 2.3 of this paper describes the evaluation index of the algorithm. The time complexity function is used to show the degree to which the algorithm changes with the change of the initial data. In order to show the effectiveness of the algorithm more intuitively, the following different experiments are designed.

(1) number of sample sets

Heuristic genetic algorithm operates on the basis of 0–1 matrix. The element '1' in the matrix is regarded as the chromosome of genetic algorithm. In the process of genetic algorithm, new chromosomes are generated by

Table 1. number of sample sets.

Sample set name	Initial sample set	Reduced sample set
Png1	50	39
Png2	100	39
Png3	100	68
Png4	200	147

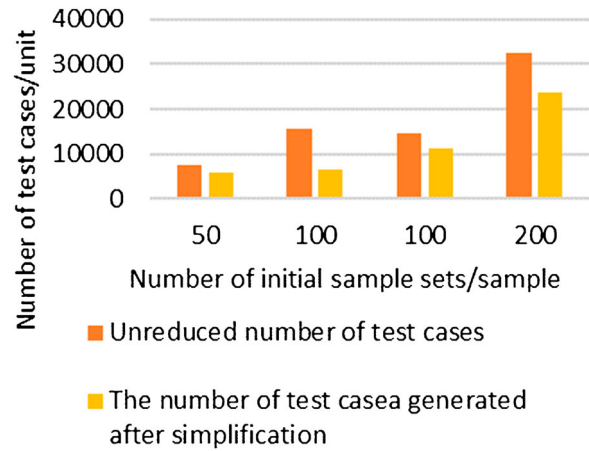
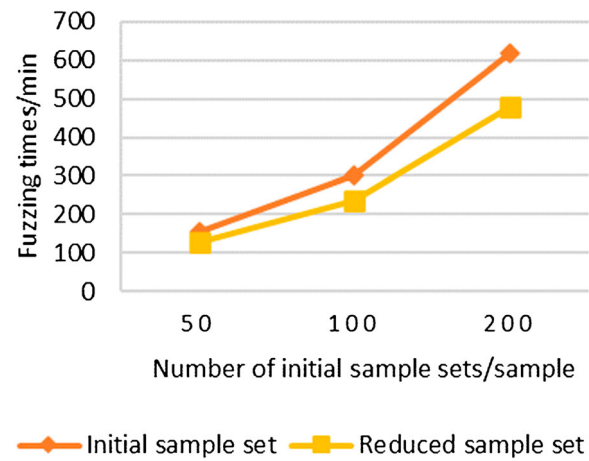
the variation and crossing of the parent chromosomes. The selection of the parent is also a process of sample reduction. Only the chromosomes with good performance are selected as the parent until 500 generations of evolution. In the initial sample set, oriented selection is made: png1–random selection; png2–two png1; png3 and png4–random selection. The relationship between the number of reduction sets and the number of initial sample sets is shown in Table 1. It can be seen from the first two groups of data that the number of the second group of data is significantly reduced after the heuristic genetic algorithm. However, png2 is a special data group, which cannot reflect the practicability of the algorithm. From the data of png2 and png3 we can see that the heuristic genetic algorithm, indeed reduces the number of sample sets; and from the data of png3 and png4, it is evident that, with the increase of the number of sample sets, the number of reduced sample sets also increases but not in proportion, indicating that the larger the number of sample sets, the greater the probability of duplicate samples occur.

(2) Fuzzy test time

Fuzzy testing is one of the vulnerability mining techniques. The experiment is to verify and evaluate the effectiveness of the heuristic algorithm and the benefits to fuzzy testing. Table 2 shows the time spent before and after fuzzy testing the four groups of data in Table 1. The change in fuzzy testing time is used to explain whether the heuristic genetic algorithm can improve the efficiency of fuzzy testing. Figure 2 shows a comparison between the number of samples and the number of test cases. Due to the particularity and specialty of data, the test cases generated by the second group of data are almost the same as the first group. From the first two groups of data, we can draw a conclusion that the generation of test cases has a great relationship with the input samples. To a certain extent, it mainly depends on the minimum sample

Table 2. Fuzzy test times.

Number of initial sample sets	Initial sample set test cases	Initial sample fuzzy test time (min)	Reduce the number of sample sets	Reduce the number of test cases in the sample set	Reduced sample set fuzzy test time (min)
50	7524	156	39	5869	127
100	15,652	292	39	6334	206
100	14,563	302	68	11,263	236
200	32,384	617	147	23,586	478

**Figure 2.** Number of samples and number of test cases.**Figure 3.** Fuzzy test time and sample size.

set of the original samples. Figure 3 shows the line chart of fuzzy test time with sample number, excluding the second group of data. As shown in the two lines, the fuzzy test time of the simplified sample is lower than that of the initial sample. It can be concluded that the simplified sample set can indeed improve the fuzzy test time (Table 2).

(3) Test case

Both the compression process and quality of samples are considered in the design of heuristic algorithm. Each sample has its own corresponding code block, based on

Table 3. Test case data.

Test time (hours)	The number of test cases generated by unreduced samples	Number of marked test cases generated by unreduced samples	Number of test cases resulting from reduced samples	The number of labelled test cases generated by the reduced sample
1	3075	2356	2954	2879

the variation of which, test cases are generated and would directly affect the fuzzy test. In the case without sample simplification, useless test cases will be generated due to repeated samples. In order to further prove that the algorithm can take into account the selection of sample quality, after the same group of initial population is input, a simple mark is made on the test cases generated to ensure the uniqueness and then the number of test cases generated during mutation before and after reduction is counted. Table 3 shows the specific data of the experiment. It can be seen that when the test time is fixed, the total test cases generated by the unsimplified sample set in a short period of time are larger than those generated by the simplified sample, but the number of marked test cases is smaller. It can be concluded that the quality of the sample set has been improved after the heuristic algorithm implementation. The generated test cases have little repeatability, which shows that the algorithm does take the quality of samples into account.

(4) comparative experiment

The optimization method of fuzzy test sample set mainly starts from the following two aspects: (a) approximation algorithm; (b) machine learning. The implementation of this paper involves the path and code block used in the approximation algorithm. In terms of machine learning, the algorithm related to genetic algorithm is still selected. The difference is that there is no targeted improvement on genetic algorithm with only a little modification applied to the simplification of sample set. Another method, proposed in this paper, is heuristic genetic algorithm. Combined with Figure 4, the design of the approximation algorithm is roughly the same as heuristic genetic algorithm in our paper. Therefore, the final simplification results of the two methods are close while the speed of approximation algorithm is far slower than that of the heuristic genetic algorithm. It shows that the combination of the traditional algorithm and the genetic algorithm can reduce the time of sample set simplification, thus reducing the time of fuzzy test. Compared with the heuristic algorithm, the genetic algorithm can reduce the time of fuzzy test but the acquired sample set is not very valuable. Genetic algorithm is only a search algorithm obtaining the final result with no corresponding constraint. Heuristic algorithm consumes a slight longer time than genetic

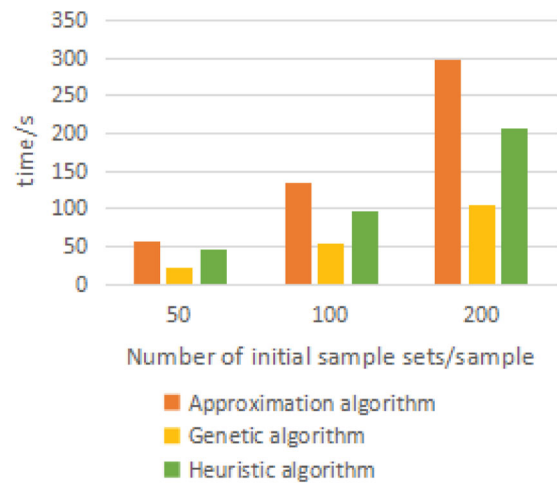


Figure 4. Comparison graph of time (in seconds) required for different algorithms to produce a simplified sample set.

algorithm, which just shows the improvement that it is effective and does not take a part of time to implement the constraint.

7. Summary and outlook

The set coverage problem is NP hard, and the execution time of fuzzy test increases consistently with the increase of initial sample set. Genetic algorithm is a kind of search algorithm with the advantage of comparing multiple individuals at the same time due to its potential parallelism. This paper mainly introduces the current research status of fuzzy test sample set simplification, the sample set simplification model, the detailed idea of genetic algorithm improvement and summary of the experimental results. The heuristic algorithm is improved on the genetic algorithm and applied to the set coverage problem. It uses chromosomes to represent the covering situation in the set. The high-quality chromosomes are used as basis to be modified heuristically to enable the frequent evolution in population. The results of our experiments also provide evidence for the effectiveness of the heuristic algorithm from different angles. However, the algorithm in our paper does not study the mutation rate and crossover rate of genetic algorithm whose values are set according to preexisting experimental results which may not necessarily be the best choice for this experiment. Therefore, our next plan is to study the mutation rate and crossover rate of genetic algorithm,

so as to further improve the speed of heuristic genetic algorithm.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Zhihua Wang  <http://orcid.org/0000-0003-1732-6245>

Manman Cheng  <http://orcid.org/0000-0002-3278-903X>

Yongjian Wang  <http://orcid.org/0000-0002-7875-1395>

References

- Bergantiños, G., Gómez-Rúa, M., Llorca, N., Pulido, M., & Sánchez-Soriano, J. (2020). Allocating costs in set covering problems. *European Journal of Operational Research*, 284(3). <https://doi.org/10.1016/j.ejor.2020.01.031>
- Bhme, M., Pham, V. T., Nguyen, M. D., et al. (2017). *Directed grey-box fuzzing*. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA: ACM, October 2017:2329–2344.
- Bhme, M., Pham, V. T., & Roychoudhury, A. (2016). Coverage-based greybox fuzzing as Markov chain 45(5), 489–506. <http://10.1109/TSE.2017.2785841>
- Guoqing, Y., Lanrong, Z., & Ke, L. (2016). Application of fuzzy k-means algorithm in test case set reduction. *Journal of Huaqiao University (Natural Science Edition)*, 37(6), 778–781. <https://10.11830/ISSN.1000-5013.201606024>
- Gupta, S., Deep, K., Mirjalili, S., & Kim, J. H. (2020). A modified Sine Cosine Algorithm with novel transition parameter and mutation operator for global optimization. *Expert Systems With Applications*, 154. <https://doi.org/10.1016/j.eswa.2020.113395>
- Jinxin, M., Tao, Z., Zhoujun, L., & Jiangxiao, Z. (2016). Some optimization methods in fuzzy process. *Journal of Tsinghua University (Natural Science Edition)*, 56(5), 478–483. <http://10.16511/j.cnki.qhdxxb.2016.25.004>
- Khulood, A., & Jonathan E, R. (2018). Landscape analysis of a class of NP-hard binary packing problems. *Evolutionary Computation*, 27(1), 47–73. https://10.1162/evco_a_00237
- Lin, D., Wu, X., Wu, Q., Liu, Y., Zeng, J., & Tan, J. (2019). A survey of the key technology of software vulnerability mining. *Journal of Physics: Conference Series*, 1187(5). <https://10.1088/1742-6596/1187/5/052031>
- Liu, Y., Sid-Lakhdar, W., Rebrova, E., Ghysels, P., & Li, X. S. (2020). A parallel hierarchical blocked adaptive cross approximation algorithm. *The International Journal of High Performance Computing Applications*, 34(4), 394–408. <https://doi.org/10.1177/1094342020918305>
- Longlong, J., Linlin, L., Wangtong, L., Limin, P., & Gu, Z. (2018). Fuzzy test method of binary program based on genetic algorithm. *Journal of Zhejiang University (Engineering Edition)*, 52(5), 1014–1019. <https://10.3785/j.issn.1008-973X.2018.05.023>
- Peng, S. Q., & Tian, Z. Y. (2014). Design and realization of IE vulnerabilities mining based on fuzz testing. *Applied Mechanics and Materials*, 3512, 2032–2035. <https://doi.org/10.4028/www.scientific.net/AMM.651-653.2032>
- Pereira, A. G. C., Campos, V. S. M., de Pinho, A. L. S., Vivacqua, C. A., & de Oliveira, R. T. G. (2020). On the convergence rate of the elitist genetic algorithm based on mutation probability. *Communications in Statistics – Theory and Methods*, 49(4), 769–780. <https://doi.org/10.1080/03610926.2018.1528361>
- Qian, L., & Jiayong, L. (2017). Research on sample set reduction and optimization based on fuzzy technology. *Network Security Technology and Application*, 1, 62–64. <https://10.3969/j.issn.1009-6833.2017.01.039>
- Shitov, Y. (2018). On the determinant of a sparse 0–1 matrix. *Linear Algebra and Its Applications*, 554. <https://doi.org/10.1016/j.laa.2018.05.019>
- Spandan Veggalam, Sanjay Rawat, Istvan Haller, & Herbert Bos (2016). *Ifuzzer: An evolutionary interpreter fuzzer using genetic programming*. European Symposium on Research in Computer Security, Heraklion, Greece: Springer, pp. 581–601.
- Wang, S., Nam, J., & Tan, L. (2017). *QTEP: Quality-aware test case prioritization*. Proceedings of the 2017 11th Joint Meeting on Foundation Software Engineering, Paderborn, Germany: ACM, pp. 523–534.
- Wei, X., Wang, X., & Chen, S. (2020). Research on parameterization and optimization procedure of low-Reynolds-number airfoils based on genetic algorithm and Bezier curve. *Advances in Engineering Software*, 149. <https://doi.org/10.1016/j.advengsoft.2020.102864>
- Xiao, T. (2019). Improved feedback fuzzy testing method based on dynamic taint analysis. *Information Security Research*, 5(2), 145–151. <https://10.3969/j.issn.2096-1057.2019.02.006>
- Zhang, G. L., Liu, X. X., & Zhang, T. (2009). The impact of population size on the performance of GA. *2009 International Conference on Machine Learning and Cybernetics. IEEE*, 4, 1866–1870. <https://doi.org/10.1109/ICMLC.2009.5212113>