A Strategy for Multi-target Paths Coverage by Improving Individual Information Sharing

Zhongsheng Qian*, Dafei Hong, Chang Zhao, Jie Zhu, Zhanggeng Zhu
School of Information Management, Jiangxi University of Finance & Economics,
Nanchang 330013, China
[e-mail: changesme@163.com, 1335822121@qq.com,
625091212@qq.com, 1967531979@qq.com, 251949387@qq.com]
*Corresponding author: Zhongsheng Qian

Received February 13, 2019; revised May 1, 2019; accepted May 26, 2019; published November 30, 2019

Abstract

The multi-population genetic algorithm in multi-target paths coverage has become a top choice for many test engineers. Also, information sharing strategy can improve the efficiency of multi-population genetic algorithm to generate multi-target test data; however, there is still space for some improvements in several aspects, which will affect the effectiveness of covering the target path set. Therefore, a multi-target paths coverage strategy is proposed by improving multi-population genetic algorithm based on individual information sharing among populations. It primarily contains three aspects. Firstly, the behavior of the sub-population covering corresponding target path is improved, so that it can continue to try to cover other sub-paths after covering the current target path, so as to take full advantage of population resources; Secondly, the populations initialized are prioritized according to the matching process, so that those sub-populations with better path coverage rate are executed firstly. Thirdly, for difficultly-covered paths, the individual chromosome features which can cover the difficultly-covered paths are extracted by utilizing the data generated, so as to screen those individuals who can cover the difficultly-covered paths. In the experiments, several benchmark programs were employed to verify the accuracy of the method from different aspects and also compare with similar methods. The experimental results show that it takes less time to cover target paths by our approach than the similar ones, and achieves more efficient test case generation process. Finally, a plug-in prototype is given to implement the approach proposed.

Keywords: multi-population genetic algorithm, individual information sharing, multi-target paths coverage, contact layer proximity, test case

http://doi.org/10.3837/tiis.2019.11.011 **ISSN : 1976-7277**

1. Introduction

Test data generation has been widely concerned in software testing [1-6], and it is very important to design corresponding test data according to software scenarios. Wang, et al. [7] adopted error propagation model to describe the evolution process of defects based on the defects existing in the test, and designs test data continuously through intelligent algorithms. In ref. [8], the authors introduced genetic algorithm and test process in detail. For the specified target path, they used genetic algorithm to generate test data. The target path coverage problem is regarded as the key research problem in software test, its purpose is to generate the test data covering target path. So far, there have been many methods to generate test data for path coverage problems.

Presently, evolutionary algorithms are widely applied for solving optimization problems [9-10] and genetic algorithm is regarded as the typical one of them. Due to its unique advantages, genetic algorithm has shown very good results in term of path coverage problems, and has become a choice with high tendency in dealing with such problems. The authors [11] proposed an evolutional test data generation method of path coverage based on node probability. Qian, et al. [12] presented a path coverage method based on contact layer proximity and node probability for single target path coverage problem. In this work, genetic algorithm is used as the main framework of the proposed method. To some extent, it improves the efficiency of the test cases generated, but it is just for a single path, similar to unit test. From a broad perspective, it appears relatively narrow, so it is necessary to do further research.

Although there are many researches on test data generation for path coverage[13-20], which solve various existing problems from different aspects and greatly improve the efficiency of test coverage, the efficiency of test data generation is still insufficient to some extent. The reason is that the above algorithms aim at a single target path, perhaps they are enough to meet traditional software requirements, but it is difficult to estimate the structural complexity and volume of current software, because there are many target paths to test. When target paths are small, it can be detected by manually editing the target paths and executing the algorithm several times. Once there are hundreds or thousands of target paths, the workload is very large and the efficiency is not high. In fact, a choice is to execute an algorithm program only one time to generate test cases for multiple target paths, which can greatly reduce test cost and improve the efficiency of test case generation.

Yao [21] believes that multi-population genetic algorithm is suitable for solving problems with multiple target solutions in running once, and discusses the shortcomings of multi-path coverage method proposed by Ahmed, et al. [22]. When it comes to specific problems, it is necessary to build corresponding mathematical models. Therefore, in the process of solving multi-path problems, a model containing multiple objective functions is firstly established, and then the evolutional solution of the multi-population genetic algorithm is adopted. In this model, each objective function is relatively independent, which usually corresponds to the maximum fitness value of each target path covered. In the multi-population genetic algorithm, individual migration is not required, and individual information sharing strategy is used to solve the established model, so as to obtain the solution set of multi-target paths coverage problems. Although information sharing strategy improves the efficiency of multi-population genetic algorithm to generate multi-target test data, there is still space for improvement. In this strategy, when a group covers other paths besides the corresponding target path, the group will stop. When a subpopulation covers its corresponding target path, it will stop before attempting to cover all the other target paths in the target data set. Thus, the

opportunity to cover other paths will be missed, which will affect the efficiency of covering the target path set. Therefore, the approach in this work makes some improvements for the individual information sharing, and the improved results are verified by experiments.

We will build a model of multi-target problem based on the single-target path coverage method [12], and then solve it using multi-population genetic algorithm, and adopt the idea of individual information sharing in ref. [21]. Regarding the shortcomings of individual information sharing in multi-population genetic algorithm, we make the following improvements in this work:

1) The execution strategy corresponding to the target path of the subpopulation coverage is improved, so that it can continue to try other sub-paths after covering its own target path, to find other paths suitable for the current population, so as to maximize the utilization of the current population and improve the running efficiency.

2) In order to reduce the influence of the randomized generation of individuals resulting in different effects of covering paths, and thus affect the efficiency, the sorting operation of each sub-population in the population set is carried out.

3) For the paths that may be difficult to cover, by directly comparing the relationship matrix of the individuals that have covered the difficult paths with the relationship matrix of the population individuals, the individuals that can cover the difficult paths are screened to avoid the large amount of calculation of fitness function values due to the existence of the difficult paths, so as to improve the coverage efficiency.

The remainder of this paper is organized as follows. Some related work are compared and analyzed in Section 2. After giving some related definitions, together with the multi-path coverage model, it designs a multi-population genetic algorithm of multi-path coverage test case generation in Section 3. The improved genetic algorithm is employed to cover multiple target paths. In Section 4, it conducts some experiments on several benchmark programs to elaborate on the accuracy verification of multi-path coverage and expound the comparative analysis with other similar methods. Based on the approach proposed in previous sections, a test case generation plug-in, which facilitates the test process, is developed in Section 5. It draws some concluding remarks and indicates our future work in last section.

2. Related work

Multi-population genetic algorithm has been widely used in multi-target paths coverage domain, and the large-scale path coverage problem has gradually become the focus of our attention, so there are a lot of gratifying research results. Yao [21] proposed a multi-path coverage method in her doctoral dissertation, which uses the multi-population genetic algorithm as the search strategy for the target path. The mathematical model and the data communication process in population evolution are different from the traditional processing methods. The difference of solving mathematical model is that multiple sub-problems in the model are optimized by their corresponding sub-populations, and not all sub-populations are involved in the same problem. The difference in the process of data communication during the evolution of a population is that there is no need for individual migration, only for individuals among the population to share information. These two aspects are much clearer than traditional processing methods and simplify the operations. However, there are still some deficiencies. The design of its fitness function is still based on layer proximity and branch distance.

The authors [23] proposed some test methods of path coverage, mainly including single-path coverage and multi-path coverage. Among them, multi-path coverage method

contains defect-oriented multi-path coverage test method and multi-path coverage test data evolutionary generation method based on search space reduction. The defect-oriented multi-path coverage method takes the target path coverage and the program defect as the target, transforms the model constraint into the penalty factor, and designs the fitness function. Multi-path coverage method based on search space reduction utilizes public independent sub-paths existing between multiple target paths, records the value the individual gets when passing the sub-paths and applies this value to other target paths, at the same time, deceases the scope of the crossover and mutation operation on the individuals (not including sub-paths), in order to avoid the redundancy for multiple paths to the sub-paths, improve the efficiency of test data generation.

Although the defect-oriented multi-path coverage method can obtain test cases that can both cover the target path and find defects, it still has some deficiencies. On the one hand, it will produce different results according to different defect degrees. Some defects will not continue to show in the target path once they occur, and this situation is not clearly stated. It is uncertain that the number of defects, on the other hand, is regarded as an important part of the objective function of the method, because when the program has multiple defects, the program will jump out when first coming into the fatal flaws, and then the rest of the program won't be able to execute, and there's only one defect that's exposed. Thus, it is difficult to determine the number of defects. Although the multi-path coverage method based on search space reduction can gradually reduce the search space as the sub-paths are covered, the value of fixed gene segments needs to be continuously accessed and modified in the search process, which increases the complexity of individual operations for each population.

Gong, et al. [24] thinks it is necessary to consider the individual's proximity criteria on multiple target paths at the same time and appropriate fitness function is the key to efficiency improvement. Huffman coding to the target path coding is presented in this paper, and then based on the fitness function to avoid redundancy and the relatively small amount of calculation, test data is successfully created to meet the requirements. Although Huffman coding method simplifies the coding and decoding process of the target path to some extent, the design process of its fitness function is somewhat complicated.

Suresh, et al. [25] proposes an approach to combining soft computing and genetic algorithm to generate test data based on base paths, which improves the generation ability of test data and improves the efficiency of test work. The authors [26] proposed a method to extract the information resulting in unreachable paths, determine the relevant conditions of dependence, then construct the automata of unreachable paths to detect unreachable paths, which saved time due to the existence of the unreachable path. The authors [27] mentioned the calculation of the out degree of the control flow diagram of the source code, and the application of genetic algorithm for testing can effectively simplify the calculation of fitness value to generate test data. Hermadi, et al. [28] believes that the search should be stopped when the program is not worth continuing to search test data, and proposes a method to determine when the search of test data is stopped, which can effectively reduce the computation and avoid the problems caused by the infeasible paths. These work [25-28] are all based on the idea of single target path and use the genetic algorithm to deal with path coverage problems, which has limitations when applied in multi-path problems

In view of this, we establish a multi-target paths coverage model based on single-target path coverage [12], and propose a multi-path coverage method combining with the way of individual information sharing among populations in ref. [21]. Firstly, it is improved for the way of terminating the execution of the subpopulation after its target sub-path is passed in information sharing proposed in ref. [21], so that the individual resources of the

subpopulation can be fully utilized. Secondly, prioritization is carried out for each sub-population in multiple populations. The priorities of sub-populations are determined according to the effect of covering target paths respectively. The sub-population with better coverage effect has the higher priority and is executed earlier, thus avoiding the problem that the sub-population with poor coverage effect delays the entire coverage process. Finally, for the difficultly-covered paths in the target path set, it proposes a method to extract the individual chromosome characteristics that can cover the difficultly-covered paths based on the data already generated, and then screen the individuals that can cover the difficultly-covered paths.

3.Covering multiple target paths by the improved multi-population genetic algorithm

In multi-population genetic algorithm, although multiple sub-problems can be solved separately by simply increasing the number of populations, the advantages of the algorithm cannot be shown. If the algorithm can provide certain help for solving other sub-problems while solving one sub-problem, then the process of solving the whole problem will be accelerated. This work will introduce the contact layer proximity [12] to design the multi-population genetic algorithm, and combine the strategy of sharing individual information among populations proposed in ref. [21] to accelerate the solution to the problem. In order to deepen the understanding of the method in this work, the following concepts are introduced.

3.1 Related definition

Definition 1. Control Flow Graph. Control flow graph, represented by $G=(D,E,st,ed)$, is one graphical representation of program structure, where D is the node set, d_i represents the node in D, and E is the edge set of the nodes. $e_{ij}=(d_i,d_j)\in E$, which represents the path from d_i to d_j, is the edge of G. In addition, st and ed are the start point and end point of G respectively.

Definition 2. Path. A path is a sequence composed of a series of nodes such as d_i (i = 1, 2,…,n). For example, $P=\{st,d_1,d_2,d_3,...,d_n,ed\}$ represents a path, where $d_i\in D$, and each edge corresponding to its nodes in the path belongs to E.

Definition 3. Consistency of Coverage. Coverage path is the execution path obtained by the population individual executing the tested program after being decoded. Target path is the execution target of the individual searched by the algorithm. When the coverage path is the same as the target path, it means that they are consistent.

Definition 4. Contact Vector. Given the path $P=\{\tau_1,\tau_2,\tau_3,...,\tau_n\}$, the corresponding contact vector $\rho=\{\rho_1,\rho_2,\rho_3,...,\rho_n\}$ is the n-dimensional vector corresponding to the path, whose component ρ_i (i=1,2,…n) represents the contact degree of the *ith* node of path P, which is encoded by integers from the outer layer to the inner layer. The contact degree of the outermost node is 1, which increases with the depth of the hierarchy.

Fig. 1 is the control flow graph of a program, where st and ed is the start and end points of the program, and the ellipses numbered 1, 2, 3, 4, 5, 6 and 7 are all program nodes. A sequence of nodes are a program path such that $P=\{st,1,2,3,5,ed\}$ is a path. The contact vector reflects how easy a program statement is to be executed. As can be seen from the

figure, nodes 1 and 2 are more likely to get the opportunity to be executed than nodes 3 and 4, because nodes 3 and 4 are more inner than nodes 1 and 2. They can only be executed by judging the corresponding statements of node 2. The contact vector is defined based on this situation. For example, the contact vector corresponding to the path P is $\rho = \{st, 1, 1, 2, 2, ed\}$, and nodes that are more difficult to contact are assigned with greater weights.

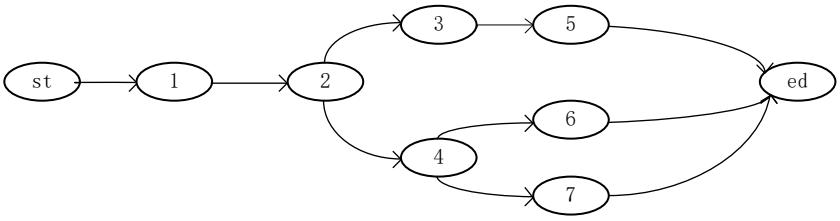


Fig. 1. An example of control flow graph

Definition 5. Individual Information Sharing [21]. In multi-population genetic algorithm, after every operation, it should not only judge whether the individual in this population is the best one, but also judge whether it is the solution to the corresponding problem of other populations.

Fig. 2 vividly shows the basic principle of information sharing. In addition to searching for the solution of the corresponding target path, each subpopulation also needs to search whether it contains those solutions to other target paths. Therefore, individuals in each population can be fully utilized.

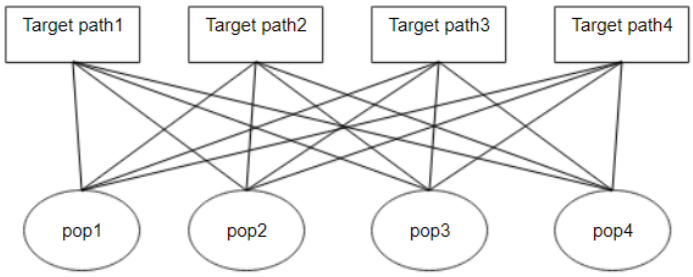


Fig. 2. Individual information sharing way among populations

Specifically, given the population set $POP=\{pop_1, pop_2, ..., pop_i, ..., pop_n\}$ $(i = 1,2,\ldots,n)$, any individual in the population pop_i, in addition to needing to be used to determine the optimal solution of $max(F_i)$, needs to make the decision on whether it is the optimal solution to other populations in the corresponding sub function $max(F_k)$ $(k \neq i)$, so that a population can be used for the solutions to n optimization functions, avoiding the situation that the individuals in the *ith* population can cover the *jth* $(j \neq i)$ populations, but not retained as a solution.

In addition, the criterion whether an individual e_{ij} (the *jth* individual of the *ith* population) is the optimal solution to the optimization function $max(F_i)$ $(k \neq i)$ is: comparing whether the coverage path $p(e_{ij})$ of the individual is consistent with the target path p_k; if so, the individual is the optimal solution to the *kth* optimization function; otherwise, it will no longer participate in the evolution process of the *kth* population, and the fitness function

value on the *kth* target path will not be calculated. Since e_{ij} is not an individual in the evolution population corresponding to the *kth* target path, it is not necessary to calculate the fitness value to carry out the selection, crossover and mutation operations of the genetic algorithm, which will greatly improve the efficiency of the algorithm.

3.2 Multi-path coverage model

In order to fully explore the value of each subpopulation, this work integrates the above information sharing improvement strategy and constructs a multi-path coverage model. The mathematical model of multi-target paths coverage problem is described in detail below.

Assuming that the input vector is $ln=(i_1,i_2,i_3,...,i_m)$, where i_k $(1\le k\le m)$ is the node in a path, and its value is 0 or 1, which represents whether the node is executed (0 represents no execution, 1 represents execution); the target path set is $P=(p_1,p_2,p_3,...,p_n)$, where p_j $(1\le j\le n)$ represents the *jth* target path. Our goal is to find the test case set $C=(c_1,c_2,c_3,...,c_n)$ covering the path set *P*, where c_j is the input vector corresponding to the *jth* path covered.

For example, if an individual's chromosome decoded is a vector in_i, which is taken as input and the fitness value calculated after executing the tested program is not less than that of the covered path p_i, then it indicates that the individual successfully covers the target path p_i, and we will add in_i to the test set.

Let the target path set *P* contain *n* paths. For each path p_j (j=1,2,…,n), when $in_j=(i_{j1},i_{j2},i_{j3},...,i_{jn})$ covers the path p_j as input data, the target function $y_j=F(in_j)$ obtains the maximum value. Then, the problem of covering the target path set *P* will be converted into the optimization process of solving the maximum value of $y_1,y_2,...,y_n$, namely,

$$y_{best(j)}:\max(F(in_1,in_2,in_3,in_n)),\quad j=1,2,\ldots n \tag{1}$$

When solving a problem by traditional modeling, for the multi-target optimization problem, the solution must satisfy multiple conditions at the same time. According to the modeling method in ref. [5], in this work, each objective function corresponds to a separate target path and is independent of each other. That is, it should find test data to cover the path for each objective function. In order to understand and clarify the problem easily, Equation (1) above is transformed into Equation (2). That is, the final model of the proposed problem is:

$$\begin{cases}\max(F_1(in_1))\\ \max(F_2(in_2))\\ .....\\ \max(F_n(in_n))\end{cases} \tag{2}$$

As shown in Equation (2), the final model is composed of *n* functions, each of which corresponds to an optimization problem, and each optimization problem corresponds to its test data covering the target path. Since each subfunction is independent of each other and corresponds to a specific target path, the final problem to be solved is to find the solution corresponding to each subfunction (subproblem) and finally compose a solution set containing multiple solutions.

3.3 The genetic algorithm of multi-path coverage test case generation

The mathematical model in Section 3.2 shows that each subproblem is relatively independent, but not completely separated from each other. Subproblems are relatively independent because each subproblem represents the solution process of a target path for a subpopulation, and there is no other relation among them. Subproblems are not completely separated from each other, because subproblem solving not only resolves the corresponding target path, but also tries to cover the target path of other subproblems. The condition for the optimal solution of the subproblem holds if and only if the fitness value is the maximum when the subpopulation covers the corresponding target path. In this work, the fitness function is shown in Equation (3). The design of fitness function is based on the contact layer proximity combined with the similarity of branch condition, where the contact layer proximity is obtained by weighted calculation of contact vector (see Definition 4) and the execution path. See the detailed design process illustrated in [12].

$$F(x)=\frac{\sum_{j=R_{aim}} R_j\rho_j}{\sum R_j\rho_j}+\frac{n_c}{N_c} \tag{3}$$

where R_{aim} represents the target path, R_j represents the *jth* node of the current path, ρ_j is the contact degree of the *jth* node, N_c represents the number of conditional branches covered by the target path, and n_c represents the number of conditional branches which is equally covered by the current path and the target path. n_c / N_c represents the proportion of conditional branches executed by the current path to the same target path conditional branches. Different from ref. [12], n_c / N_c has not been standardized, because its value already ranges within [0, 1]. There is no need for standardization and it will not have a great impact on contact layer proximity.

In addition, the design of fitness function of each sub-problem in Equation (2) is the same in this work, and Equation (3) is taken as the standard. The implementation process of solving the model draws inspiration from the idea of individual information sharing strategy [21]. Due to some deficiencies of this strategy (it is impossible to avoid poor sub-population dragging down algorithm execution efficiency, insufficient utilization of individual population resources, and no specific measures for difficultly-covered paths), three improvements are made as follows:

1) Path coverage strategy for individual information sharing is improved. In the individual information sharing strategy in ref. [21], when the operation performed by the *ith* subpopulation after covering the *ith* path is determined (i≤n, where n is the number of populations), the *ith* subpopulation is then stopped. In this work, corresponding improvements are made for the process, that is, the stopping condition of population i is that the *ith* population continues to execute after covering the *ith* path until the last path is matched. The reason for this is that the *ith* population may still cover other paths after covering the *ith* path. Continuous traversal execution can make full use of sub-population resources and avoid waste of resources, thus affecting efficiency.

Although each population will be used in the solution of n target paths, it will not have a great impact on the efficiency. Because the number of n populations is not invariable, it will gradually decrease with the coverage of the target path set. For example, pop_i represents the *ith* population in the population set. When it covers the *kth* path, the population corresponding to the *kth* path will be removed from the population set. As the paths are

continuously covered by individuals, the populations corresponding to these paths will also be removed, so the efficiency of the algorithm will be constantly improved.

2) All kinds of populations in the population set were sorted according to the coverage effect. In initializing the population set, each subpopulation is produced randomly. Some populations may be less effective for the target path coverage, but some subpopulations may be better. In order to avoid that populations with poor coverage effect have great influence on the test data generation efficiency, outstanding subpopulations are executed with higher priorities, so the subpopulations in the population set are sorted according to their coverage effect.

3) An approach is proposed to dealing with the difficultly-covered paths. Specific process of this method is that after searching for the individual who has covered the difficultly-covered path in previous execution, it decodes the individual's chromosome, and obtains the relationship of the size between parameters and saves it as a matrix (The information stored in this matrix is the size relationship between each parameter and other parameters), the matrix is called target individual relationship matrix. After that, the way that the population chooses the individuals covering difficultly-covered paths is converted into the comparison between the individuals and target individual relationship matrix, which does not need to calculate the fitness value and greatly reduces the calculation effort.

The above three points are improvement measures for the deficiencies of individual information sharing [21]. The specific process can be seen in Algorithm 1.

Algorithm 1. The test genetic algorithm of multi-path coverage

step 1. Instrumentating the tested program and initializing the parameters

Set the parameters for the algorithm, including *n* for the population number, *m* for the number of individuals in the population, *n* for the number of target paths in the target path set, and the selection, crossover and mutation probability values required by population evolution.

step 2. Population initialization

In population set $P=\{ p_1, p_2, p_3, ..., p_n \}$, for any target path p_i belonging to *P*, it randomly generates a population pop_i with *m* individuals.

step 3. Coding individuals and executing the instrumentated program (without evolution process)

Use binary coding format and execute the program to obtain the effect of the target paths that each population tries to cover.

step 4. Collecting the results of each subpopulation covering the target path set and sorting those subpopulations

With the obtained coverage data, all populations in the population set are sorted according to the number of paths covered, and executed in the sorted order.

step 5. Calculating the fitness value of individual populations and determining the optimal solution

For the *ith* population pop_i, the maximum fitness value $\max(F_i(in_i))$ of individuals covering the *ith* path in the population was calculated. If there is an individual whose fitness value reaches the maximum, it means that the individual covers the target path p_i, and it will be removed from the target path set. Otherwise, it will move to step 9.

step 6. Determining whether pop_i is removed.

If i≠n, pop_i needs to try to cover the target paths from i+1 to n. If we find an individual that covers the *jth* path (j>i), we will remove pop_j and terminate the execution of pop_i until the attempt to cover the *nth* path is completed.

step 7. Covering through information sharing

In addition to determining whether the individual in pop_i is the optimal solution of $y_i = \max(F_i(in_i))$, it also needs to determine whether it is the optimal solution of y_k ($k \neq i$). If an individual in pop_i can cover the *kth* target path, then pop_k terminates.

step 8. Determining whether the algorithm terminates

If the target path P is completely covered, it indicates that the algorithm completes the task, then it terminates the program, or the population evolution time exceeds the threshold and moves to step 10.

step 9. Population evolution

Roulette wheel operators including selection, crossover, mutation and other genetic operations are performed on the population, and then it goes to step 5.

step 10. Terminating program execution

In this section, for the multi-path coverage problem, it constructs the corresponding problem model, designs the fitness function of multi-population genetic algorithms. For the deficiency of individual sharing strategy and overall multi-population genetic algorithm framework, we make some improvements. For some possible problems in the implementation process of the algorithm, we take some corresponding solution measures. And finally the calculation process of multi-path coverage algorithm is given. It compares and analyzes the effectiveness of the algorithm in the next section.

4. Experimental design and analysis

In order to verify the effectiveness of the improved multi-path coverage model for individual information sharing proposed in this work, we take three benchmark programs, namely triangle-classifying[21], binary search (link to https://www.cnblogs.com/coderising/p/5708632.html) and bubble sorting[21], as an example for experiment. The triangle-classifying is used to verify the effectiveness of the method proposed in generating test data with difficultly-covered paths, while the binary search and bubble sorting are utilized to illustrate the effectiveness of the method without a difficultly-covered path. This part is mainly divided into 3 subsections. Section 4.1 is the experimental environment and parameter configuration. Section 4.2 verifies the accuracy of the algorithm. The triangle-classifying program is a very common and concise benchmark program, which is conducive to the presentation of experimental details, hence it is used as an example program for verifying the accuracy of the algorithm. Section 4.3 analyzes the efficiency of the proposed method and applies it to three benchmark programs (triangle-classifying, binary search and bubble sorting). The advantages of this approach are illustrated by comparison with other similar methods.

4.1 Experimental environment and parameter configuration

The relevant parameters of system environment required for the experiment are as follows: operating system, programming language, *jdk* version, programming tool, system memory, etc. See **Table 1** for details.

Table 1. The parameters of system environment

Parameter	Value
Operating system	win7
Programming language	java

Jdk version	1.8
IDE	eclipse
RAM	8G
Arch	X64
bw	3.30GHz

The relevant parameters of genetic algorithm are as follows: population number (*Population num*), target path number (*TargetPath num*), individual number (*Individual num*), and so on. Where, the number of population individuals is a set. In the following experiments, each number of individuals will all be within this set. See **Table 2** for the specific parameters, which are described in ref. [21] in detail.

Table 2. The parameters of algorithm

Parameter	Value
Population num	4
TargetPath num	4
Individual num	{20,50,100,150,200}
MaxGeneration num	10000
Crossing-over rate	0.9
Mutation rate	0.2
Input value	(0,255]
Chromosome length	24

4.2 Accuracy verification of multi-path coverage

In verification, we employ triangle-classifying program as the tested program. The triangle-classifying program is a relatively classic benchmark program, and it is also simple. It is suitable for such an experiment and easy to manifest the data results during the experiment [11, 21].

In this experiment, the parameter setting of the genetic algorithm is consistent with that in ref. [21]. The crossover probability is 0.9, the mutation probability is 0.1, the number of individuals in the population is set to 200, and the maximum termination times of population evolution are 5000. Besides, since the roulette wheel selection method [29-30] is usually used as a selection operation in genetic algorithm, this method is also utilized in this work.

The triangle-classifying program requires three input parameters, whose range is consistent with that in ref. [21], both are (0, 255]. And the individuals are binary-encoded. Hence, each input parameter can be derived as a binary string of length 8. Evidently, the length of the individual chromosome is 24.

To illustrate the accuracy of the proposed method, four target paths $P=\{ p_1, p_2, p_3, p_4 \}$ to be covered are randomly selected, where

$$p_1 =\{ st.d_1, d_5, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{15}, d_{16}, d_{18}, ed \},$$
$$p_2 =\{ st, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, ed \},$$
$$p_3 =\{ st, d_1.d_2, d_3, d_4, d_5, d_9, d_{13}, d_{15}, d_{16}, d_{18}, d_{19}, ed \},$$
$$p_4 =\{ st, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{15}, d_{16}, d_{18}, ed \},$$

here, d_i (i = 1,2,…,19) is the statement node in the program. Kindly refer to ref. [21] for the concrete program and its corresponding control flow graph. The improved genetic algorithm is repeated 10 times for the four target paths, and the test cases for covering the target paths

are shown in **Table 3**.

Table 3. Test situation of covering target paths

	p_1	p_2	p_3	p_4
1	38,96,64	249,128,89	203,171,203	185,178,95
2	108,246,197	136,39,1	202,36,202	219,128,93
3	80,242,185	170,32,14	217,1,217	214,167,164
4	123,254,140	218,191,6	137,37,137	208,172,162
5	58,144,108	252,232,13	202,69,202	214,167,111
6	40,254,222	147,91,26	225,136,225	218,215,141
7	66,190,181	207,112,75	249,119,249	110,98,86
8	99,202,148	255,94,15	255,127,255	229,142,131
9	54,158,153	222,91,10	175,21,175	235,201,146
10	55,241,196	242,79,31	178,79,178	247,235,126

From **Table 3**, it is clear that the test cases can cover the target path completely. The test data of each path are found, which shows the accuracy of this method in covering the target path set.

In order to illustrate the improved effect of information sharing of populations by using the method of this work, **Table 4** lists the detailed data of the five experimental results, including the target path (*Target path*), the population number covering the target path (*Pno*), and population evolution time(*EvoluNum*), etc. It can be seen that the information sharing of individuals between populations plays a significant role in path coverage.

Table 4. Detailed path coverage based on individual information sharing

Frequency	Target path	Pno	EvoluNum
1	p_1	1	0
	p_2	1	0
	p_3	3	68
	p_4	1	0
2	p_1	3	0
	p_2	1	0
	p_3	3	66
	p_4	1	0
3	p_1	1	2
	p_2	1	0
	p_3	1	0
	p_4	1	0
4	p_1	3	0
	p_2	1	0
	p_3	3	2
	p_4	1	0

5	p_1	3	0
	p_2	1	0
	p_3	3	60
	p_4	1	0

As shown in **Table 4**, in the first experiment, the order of four paths are covered is p_1, p_2, p_4, p_3. The three paths p_1, p_2, p_4 are all covered by the first population, while p_3 is covered by the third population after 68 evolution times. It can be seen from the evolution time that the first population covers p_1, p_2, p_4 without additional genetic operation. This can be further deduced that it covers the first target path for the first time (i.e. the target path which is corresponding to the solution of the first population) and covers the two paths p_2, p_4 at the same time. In the second experiment, the first population covers two paths p_2, p_4 without any evolution. The results of the third, fourth, and fifth experiments also show that the population covers multiple paths in just one process. First of all, it can be concluded that the case where the population covers multiple paths at one operation is not uncommon. The advantages of the individual sharing of the population are reflected. Secondly, whether the population can continue to be executed can be affirmed from the above table after the population covers its corresponding target path. This can make full use of the individual resources of the population.

In the above experimental results, **Table 3** shows the accuracy of the proposed method for multi-target paths coverage problems. From the results of columns *Pno* and *EvoluNum* in **Table 4**, we can infer that the *ith* population should not be terminated immediately after it covers the corresponding *ith* path, but it should continue to execute until it reaches the end of the target path set. As can be seen from the first experiment in **Table 4**, the first population also covers other paths efficiently after covering its own corresponding paths.

In order to recognize more intuitively the importance of individual information sharing among populations in multi-population genetic algorithm, we calculate the number of shared individuals on each target path based on 100 and 1000 duplicate experiments. Specific data are shown in **Table 5**.

Table 5. The effect of individual information sharing

Target path	100 repeated experiments		1000 repeated experiments	
	Number of shared individuals	Percentage of total	Number of shared individuals	Percentage of total
p_1	42	10.5%	513	12.825%
p_2	100	25%	1000	25%
p_3	45	11.25%	317	7.925%
p_4	100	25%	1000	25%

Firstly, from the results of 100 repeated experiments in **Table 5**, according to the number of shared individuals in the process of covering paths from p_1 to p_4, we can see that for covering p_1, the role of shared individuals reaches 10.5%; for covering p_2, p_3, p_4, the proportions of shared individuals in the total number of individuals are 25%, 11.25% and 25%, respectively. Then, from the results of 1000 repeated experiments, for covering p_1, p_2, p_3, p_4, the ratios of shared individuals to total individuals are 12.825%, 25%,

7.925% and 25% respectively. When there is a 10-fold difference between 100 and 1000 experiments, it is very obvious about the proportion of shared individuals in the experiment. In the process of multi-population evolution, individuals in the population can be significantly reused, and resources consumed can be reduced in generating new individuals. Therefore, the individual sharing strategy is effective in covering multi-path problems.

This subsection verifies the accuracy of the method in this work. It can also be seen from the experimental results that it has certain advantages and can maximize the use of population information. However, the target paths selected in the above experiment are relatively common and they are easily covered by the program. For the triangle-classifying program, determining whether or not an input is a triangle or the triangle type involves comparing the three input parameters. These three parameters are formed by chromosome decoding. The probability of generating three different numbers by chromosome decoding is much higher than that of generating three identical numbers. Therefore, the difficulty of generating the target paths corresponding to these two types of parameters is quite different. The previous experiment uses a common triangle (with three different parameter values) for the target path, so it is easy to get the test data, as is exactly illustrated by the small values listed in *EvoluNum* column in **Table 4**. In order to more fully clarify the superiority of the approach, we will choose the paths that are difficultly covered as the target paths, and demonstrate the above conclusions through the results, and compare and analyze with the similar methods to verify that the method proposed has higher efficiency than other similar methods.

4.3 Comparative analysis

The comparative experiments will be carried out on three benchmarks, which are triangle-classifying, binary search and bubble sorting. The following is a detailed comparison of the experimental results on each program.

4.3.1 Comparison on triangle-classifying program

In previous experiment, the target path set is extracted from an ordinary triangle. In order to increase the difficulty of coverage and illustrate the advantages of the method in covering the target path set, the following experiments use a special target path set, which also contains four paths corresponding to an equilateral triangle, two isosceles triangles and an ordinary triangle respectively, i.e.

p_1 ={ $st, d_1, d_5, d_9, d_{13}, d_{15}, d_{16}, d_{17}, d_{18}, ed$ },

p_2 ={ $st, d_1, d_5, d_6, d_7, d_8, d_9, d_{13}, d_{15}, d_{16}, d_{18}, d_{19}, ed$ },

p_3 ={ $st, d_1, d_5, d_9, d_{13}, d_{15}, d_{16}, d_{18}, d_{19}, ed$ },

p_4 ={ $st, d_1, d_5, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{15}, d_{16}, d_{18}, ed$ }.

Under the premise that the experimental condition is basically the same as the previous experimental parameters (such as crossover rate, mutation rate and individual selection method, etc.), the number of individuals is 200, and the experiment is repeated 5 times. The results are shown in **Table 6**.

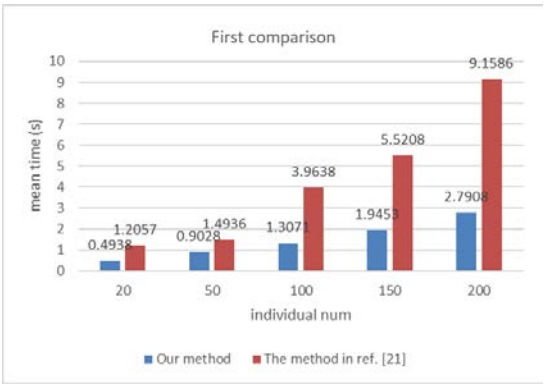
Table 6. The result of covering difficultly-covered paths

Experimental number	Execution time	p_1		p_2		p_3		p_4	
		Evolution times	Population number	Evolution times	Population number	Evolution times	Population number	Evolution times	Population number
1	8.473*s*	7300	1	5	1	0	2	0	1
2	2.375*s*	2192	1	2	1	0	2	0	1
3	3.467*s*	3291	1	3	1	2	1	0	1
4	2.238*s*	2058	1	0	1	0	1	0	1
5	2.313*s*	2132	1	2	1	0	1	0	1

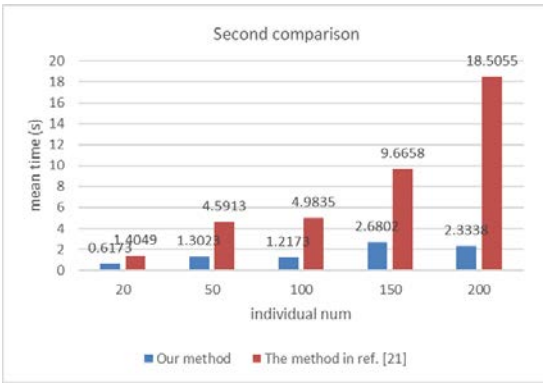
As can be seen from **Table 6**, in the first experiment, the first population covers p_4 without evolution, the second population covers p_3, and the first population covers p_2 and p_1 successively after evolving; in the second experiment, the sequence of covering target paths is p_4, p_3, p_2 and p_1; in the third experiment, the first population covers p_4, p_3, p_2 and p_1 successively; in the fourth experiment, the first population covers the three paths p_2, p_3 and p_4 in a single operation, and then covers the path p_1 after evolving, at this time, other populations do not cover any path. In the fifth experiment, the first population firstly covers p_3 and p_4, and then evolves to cover paths p_2 and p_1 successively.

In addition, it is found from **Table 6** that the time spent on path coverage in target path set is basically on path p_1 (evolution times). It takes thousands of evolutions for the population to successfully obtain the optimal individuals, because path p_1 corresponds to an equilateral triangle. The probability of generating an equilateral triangle by individual chromosome decoding is very small, hence it takes more time to find the optimal solution of the path. Since the population of each program execution is generated randomly, the time to cover the target path set is very unstable and may be very time-consuming. From the results of the second to fifth experiments, we can see that the efficiency of each population covering each path tends to be stable, which greatly improves the efficiency of covering the target path set by the algorithm.

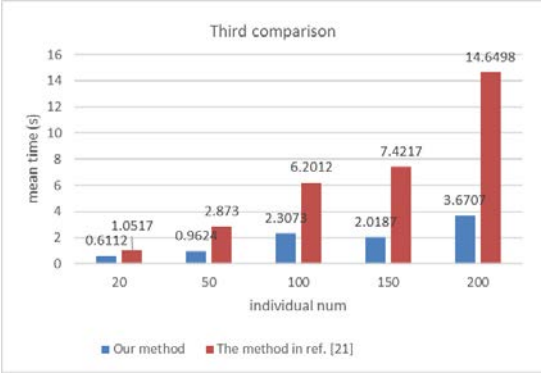
When the individual numbers are 20, 50, 100 and 200, the methods in this work and in ref. [21] are carried out respectively. The results are compared and are divided into five batches. Each batch is repeated 10 times to obtain the average execution time. The experimental results are shown in **Fig. 3**.



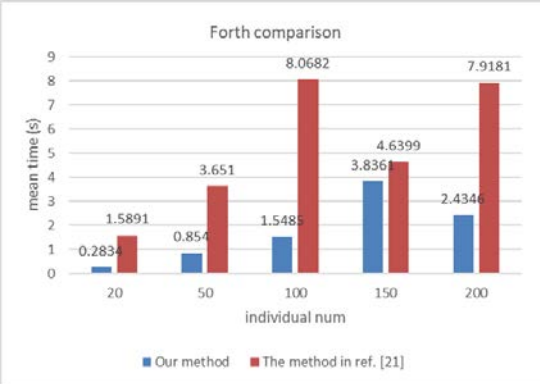
a) The first comparison



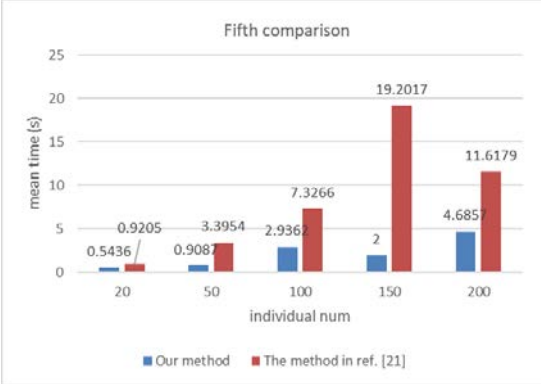
b) The second comparison



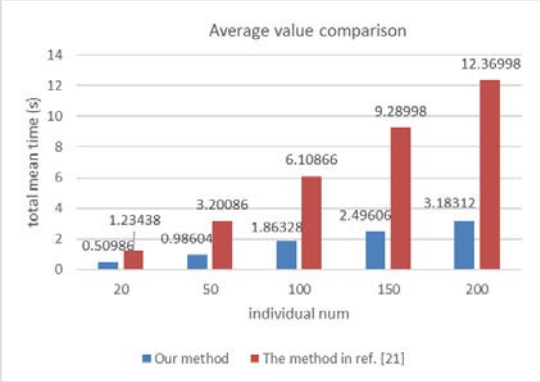
c) The third comparison



d) The fourth comparison



e) The fifth comparison



f) Comparison of total mean time

Fig. 3. Comparison of average coverage time for difficultly-covered paths

In different numbers of population individuals, **Fig. 3** shows the experimental results of the time consumed to obtain the optimal solution of each target path by using the method presented in this work and that in ref. [21] when the program is executed separately. The results are divided into five batches. Each batch is repeated 10 times by the current method to obtain the optimal solution time. Firstly, we comprehensively observe the mean consumption time corresponding to any individual number, and find that the advantages of this method are evident. Secondly, seen from the total average time in the figure (that is, the average execution time of five batches of experiments), the average time of the method is less than that of ref. [21]. Therefore, the experimental results show that our approach is more efficient.

Fig. 4 is a comparative histogram about the coverage ratios of each path in the target path set by using the method in this work and that in ref. [21] under the condition that the evolution time is limited to 10000.

Since the multi-population genetic algorithm randomly generates individuals, the coverage effect is fluctuating in the process of covering the target path. Sometimes the individuals generated are relatively good, so the coverage effect is good and the paths in the target path set can be quickly covered. Sometimes the individuals generated are poor, and need to evolve more times to cover all the target paths. In this case, it is likely to exceed the limit of the maximum evolution time set by the program, as results in unsuccessful coverage. Therefore, there is no specific rule between the effect of successfully covering the paths and

individual numbers, but the effect of the method can be illustrated by observing the coverage effect through multiple experiments.

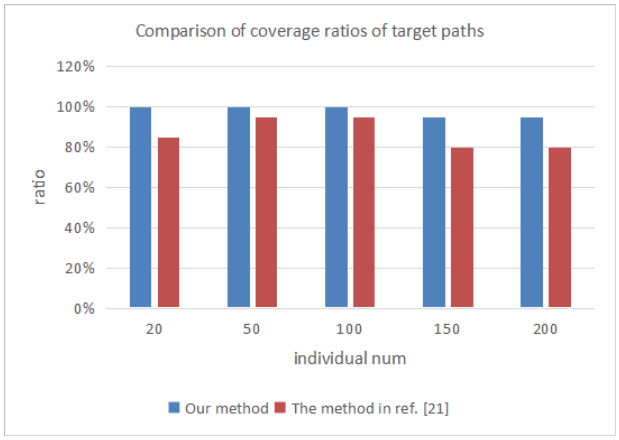


Fig. 4. The ratio of covering target paths

As can be seen from **Fig. 4**, there are still uncovered paths in the target path set under different numbers of individuals. Although it may not be successfully covered due to the limitation of the evolution time of populations, it can still be seen that the number of paths covered by this method is more stable and more than that covered by ref. [21]. It is shown that the average coverage effect of the proposed method is better than that of ref. [21].

4.3.2 Comparison on binary search program

In order to further demonstrate the superiority of the method, we apply the method into binary search and bubble sorting programs. At this time, the parameter setting of the triangle-classifying program is not suitable, so the parameters are adjusted to be used in these two new benchmark programs, and the individual chromosome length in **Table 2** is changed to 32.

The program is configured according to the parameters of the algorithm, and then is instrumented. The path length after instrumentation is 21, that is, there are 21 nodes. The experimental results of the binary search program are shown in **Fig. 5**.

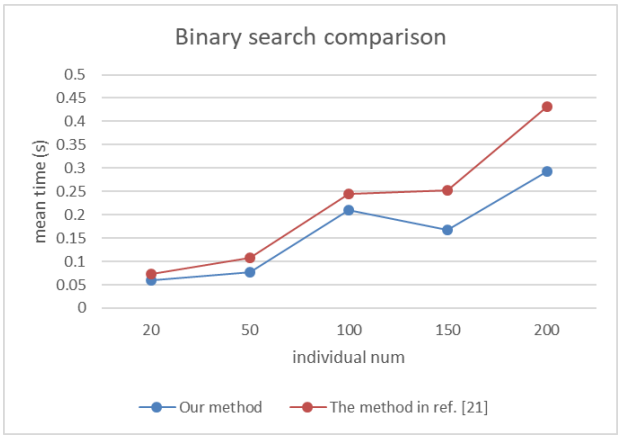


Fig. 5. The comparison of average time for binary search program

It can be seen from **Fig. 5** that when the number of individuals in the population is 20, 50, 100, the average time-consuming curve of the method is below that of ref. [21], indicating that the time consumption is less. When the number of individuals in the population is 150 and 200, the difference between the time consumption curves of the two methods is large, and the advantage of this method are more obvious. Therefore, the application on binary search program shows that the method in this work is more efficient.

The coverage ratio experiment of the target path set for binary search program is carried out below. Just like the experiment on triangle-classifying program where the given maximum population evolution time is also 10000, the coverage effect of the method proposed and ref. [21] covering the target path set is easily observed, as shown in **Fig. 6**.

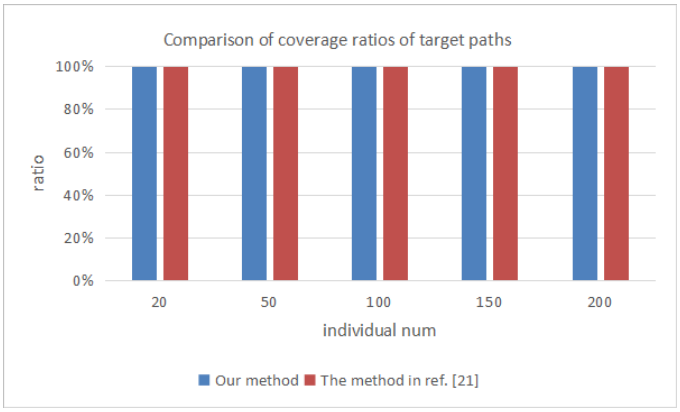


Fig. 6. The ratio of covering target paths

As can be seen from **Fig. 6**, in the experiment with binary search as the tested program, the method in this work and that in ref. [21] have successfully covered all paths in the target path set. The function of binary search program is to find the position of the target element in an ordered array. Its execution does not encounter any difficultly-covered target path, so the population can cover the target path with less evolution times.

4.3.3 Comparison on bubble sorting program

The bubble sorting program uses the same parameter settings as binary search program. In this program, the length of nodes after instrumentation is 50, and the experimental results are shown in **Fig. 7**.

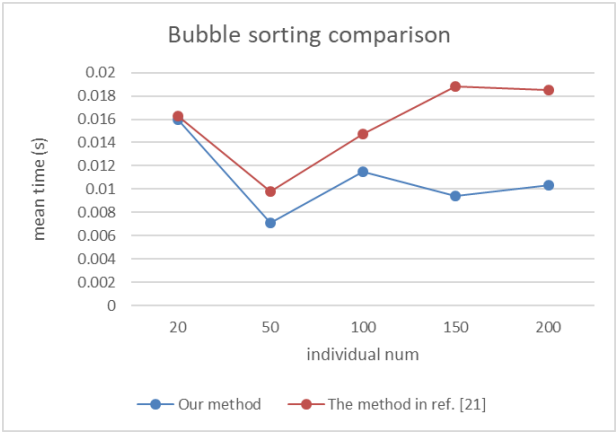


Fig. 7. The comparison of average time for bubble sorting program

It can be seen from **Fig. 7** that when the individual number is 20, the time difference between the method and that in ref. [21] is small, and when the individual number is 50, the average time-consuming gap between these two methods becomes larger. During the process of the individual number rising from 50 to 200, the time difference between the two methods gradually increases and tends to be gentle. The method is less time-consuming than that in ref. [21], indicating that the application of this method in bubble sorting program is better.

Then, the comparison experiment of target path coverage ratio is carried out for the algorithms with bubble sorting as the tested program. Under the constraint that the maximum evolution time is 10000, the effect of the method proposed in this work and ref. [21] covering the target path set is illustrated. The experimental results are shown in **Fig. 8**.

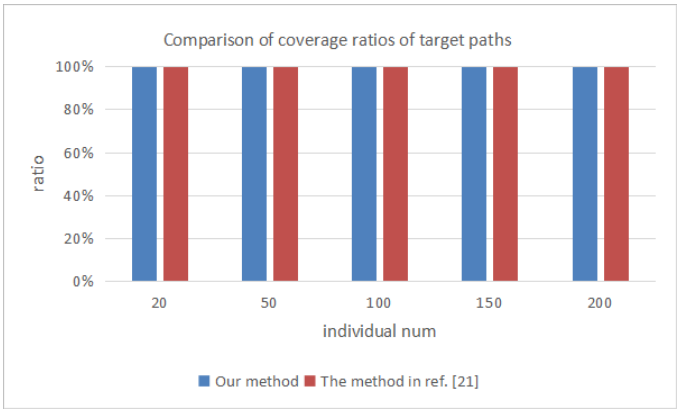


Fig. 8. The ratio of covering target paths

It can be seen from **Fig. 8** that under the condition of maximum evolution time, both the method in this work and that in ref. [21] have successfully covered the target path set, showing a very good effect. This result is due to the fact that the bubble sorting program does not have a difficultly-covered target path, for its purpose is simply to sort the target array. Although the sorting time is related to the original order of the target array, the difference of time consumed is very small and there is no situation that is difficult to cover. Therefore, both methods can successfully cover the target path set with small evolution times.

4.3.4 Experimental discussion

From the experimental results of triangle-classifying, binary search and bubble sorting programs, we can see that the advantages of this method in covering multi-target paths are quite obvious, which is due to some improvements put forward in this method for the shortcomings of individual information sharing.

Although individual sharing can improve the efficiency of multi-population computing without significantly increasing the computational power, there are still some shortcomings. On the one hand, there is a waste of resources in dealing with sub-populations that have covered the corresponding target path. On the other hand, there are no good measures on the problem of difficultly-covered target paths. In some extreme cases, for example, if there is only one difficultly-covered path in the target path set, then in the later stage of program execution, it may evolve into the situation that only one sub-population continuously tries to cover the path (other target paths have been covered). This becomes a single difficultly-covered target coverage problem, and may even face the case where the evolution

time of the sub-population exceeds the threshold. For this situation, in the method of this work, it is solved by extracting the chromosomal features of individuals that have covered the difficultly-covered path in previous experiments. Concretely, It first searches for the individuals who have covered the difficultly-covered paths in previous execution, and then decodes the individual chromosome, and finally calculates the size relationship between decoded parameters and saves it as a relationship matrix (the information stored in the matrix is the size relationship between each parameter and other parameters), which is named as target individual relation matrix. After that, the way that the population chooses the individuals covering difficultly-covered paths is converted into the comparison between the individuals and target individual relationship matrix, which does not need to calculate the fitness value and greatly reduces the calculation effort, thereby improves the efficiency of covering path set.

4.3.5 Limitations and threats to the validity of the proposed approach

In the experiments, three benchmark programs are employed to verify the accuracy of the proposed method, and the comparison with similar methods proves that our method is more efficient in covering multi-target paths. However, the proposed method also has some limitations, and its validity is also affected by some factors, which can be summarized as follows:

1) The selection of parameters of genetic algorithm in the experiment refers to the empirical values of relevant researches. However, the relevant parameters of genetic algorithm such as mutation rate and crossover rate may have a better choice of values.

2) In the multi-population genetic algorithm, the initialization of all the populations is generated randomly, accompanied by large randomness, which will affect the execution efficiency of the algorithm to a certain extent. The randomness of the population initialization may be improved by combining with other relevant strategies.

3) The length of individual chromosome encoding given in the genetic algorithm should be moderate. Excessively long chromosome length will cause more resources to be consumed in the coding and decoding process of population individual, resulting in a bad effect on the execution of the algorithm and low efficiency.

4) The proposed method for the processing strategy of the difficultly-covered target path is designed only according to the characteristics of the triangle-classifying program. Generally, for other types of tested programs that contain difficultly-covered paths, it is necessary to analyze specific problems in a specific way, and design how to extract individual feature matrix from the perspective of specific problems.

5) Covering the target path to generate test data is closely related to the tested program, and the programs with different functions have different characteristics. As a result, the effects on various types of programs may change significantly (in fact, the same is true for other similar methods). In terms of the programs with difficultly-covered target paths, the coverage efficiency is much lower than that of the simple programs containing no difficultly-covered target paths, which is inevitable in the automatic generation of test data.

The above points are some problems found in the experiments and also some aspects that affect the validity of the approach. When we design an algorithm to generate test data for the target program, we should analyze the program under test and adopt appropriate relevant strategies according to the important characteristics to achieve efficient test data generation.

5. Prototype implementation

Based on the method proposed in this work, a test case generation plug-in for triangle-classifying program is developed. To perform best practices in software process, it is essential to choose a good development pattern [31]. Eclipse is selected as the development platform for this plug-in, and the configuration of plug-in functions is realized through XML files in the way of plug-in project development pattern. The prototype of test case generation plug-in is shown in **Fig. 9**.

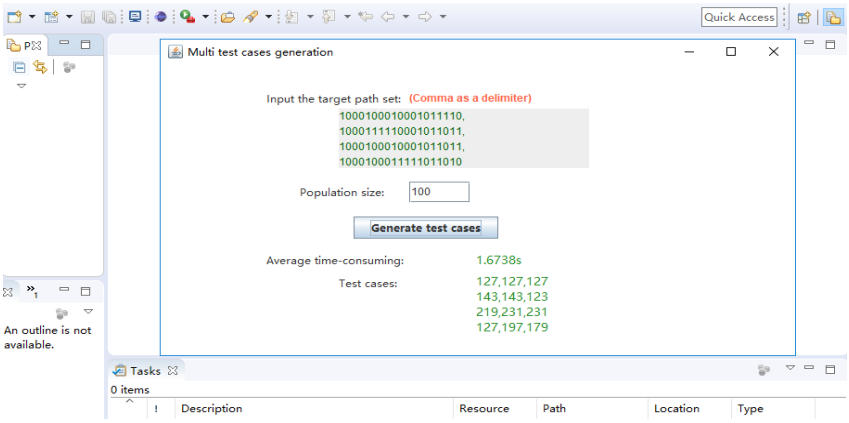


Fig. 9. A plug-in for test case generation

The plug-in, written in Java, implements the functionality to obtain test cases based on a specified set of target paths and population size. The user inputs the target path set under test in the input box *Input the target path set*, with a comma as a separator between each target path, each of which is a string whose length is 19, denoting that the program has a total of 19 nodes. Each node has a value of 0 or 1, with the 0 representing that the node is not executed, and the 1 indicating that the node is executed. In **Fig. 9**, the goal of the plug-in is to find the test cases corresponding to the four target paths entered in the target path set. In the algorithm, we utilize the input box *Population size* to set the number of individuals, and then run the plug-in by clicking on the button *Generate test cases*. The algorithm will automatically verify the contents of the target path set and cut it for execution. The average time-consuming label and the test cases label below are used to exhibit the results of test case generation. The test cases covering the target paths and the corresponding average running time after the execution of the plug-in are displayed close to the related labels.

6. Conclusions and future work

For the low efficiency of traditional multi-target paths coverage method to generate test cases, an improved approach to multi-target path test data generation is proposed in this work. In solving the multi-target problem, an optimization model of multi-target path coverage is established, and then the problem is solved by the multi-population genetic algorithm. Different from the traditional one, the multi-target problem model in this work has independent solutions of each sub-problem corresponding to each subpopulation in the population set, but the individuals between the populations can be employed to find the optimal solution of other paths. The subpopulation of population set can be reduced with the decrease of the number of target paths, so that the computation amount can be greatly

reduced and the efficiency of test case generation can be improved. The main contributions of this work are as follows:

1) The method of single-target contact layer proximity is borrowed for multi-population genetic algorithm, and an approach to multi-target paths coverage test data generation is proposed, which greatly improves the efficiency of test data generation.

2) For the problem that the efficiency of multi-target paths coverage is unstable due to the random generation of each sub-population in the population set, a method to sort the sub-populations according to the coverage effect of each sub-population is proposed in order to give higher priority to the better sub-populations and improve the efficiency of population coverage.

3) In the multi-population genetic algorithm, the strategy of individual information sharing among populations is applied, and the process of information sharing is improved, that is, when a subpopulation finds the corresponding target path covered, the population will not stop executing immediately, but will stop after traversing all the target path set. It can be seen from the experimental results that the improved individual information sharing strategy significantly improves the efficiency of covering multi-target paths set.

4) To solve the problem caused by difficultly-covered paths, an approach is proposed to extracting chromosome features of individuals who have covered the difficultly-covered paths in the early stage. The coverage problem is then converted into that of the comparison and screening of chromosome features, to obtain the individuals covering this path. Experimental results show that the efficiency of covering multi-target paths has been improved to some extent.

Compared with the traditional one, the multi-population path coverage strategy proposed in this work has better coverage rate, accuracy and time. By means of individual information sharing among populations, the problem of the computation amount increased by the existence of multiple subpopulations is greatly reduced. In dealing with the difficultly-covered paths, the method proposed improves the efficiency of covering the target path set. However, when the number of paths in the target path set is relatively large, it is not obvious to improve the efficiency of test case generation by using the simple multi-target paths coverage algorithm. The next step is to introduce the idea of concurrent programming into the algorithm, so as to further improve the efficiency of test generation through the collaborative execution of multiple threads.

Acknowledgments

Thanks go first to the anonymous referees for their sound comments and suggestions. This work is partly supported by National Natural Science Foundation of China under Grant No. 61762041, Jiangxi Provincial Natural Science Foundation of China under Grant No. 20181BAB202009, and Key Project of Science and Technology of Jiangxi Provincial Department of Education of China under grant no. GJJ180250.

References

[1] Sharma, Akshat, R. Patani, and A. Aggarwal, "Software Testing Using Genetic Algorithms." *International Journal of Computer Science & Engineering Survey*, vol. 7, no. 2, pp. 21-33, 2016. Article (CrossRef Link)

[2] Harman M, Jia Y and Zhang Y, "Achievements, open problems and challenges for search based software testing," in *Proc. of International Conference on Software Testing, Verification and Validation. IEEE Computer Society*, pp.1-12, April, 2015. Article (CrossRef Link)

[3] Chang R, Sankaranarayanan S, Jiang G, et al., "Software testing using machine learning," US, US8924938, 2014. Article (CrossRef Link)

[4] Androutsopoulos K, Clark D, Dan H, et al., "An analysis of the relationship between conditional entropy and failed error propagation in software testing," in *Proc. of International Conference on Software Engineering*, pp.573-583, May, 2014. Article (CrossRef Link)

[5] Lv J, Hu H, Cai KY, et al., "Adaptive and random partition software testing," *IEEE Transactions on Systems Man & Cybernetics Systems*, vol. 44, no. 12, pp. 1649-1664, November, 2014. Article (CrossRef Link)

[6] Zamir T, Stern R and Kalech M, "Using model-based diagnosis to improve software testing," in *Proc. of AAAI Conference on Artificial Intelligence*, pp.1135-1141, June, 2014. Article (CrossRef Link)

[7] Wang K and Wang Y, "Software test case generation based on the fault propagation path coverage," in *Proc. of 2016 Annual Reliability and Maintainability Symposium*, pp.1-4, January, 2016. Article (CrossRef Link)

[8] Nuntanee Chuaychoo and Supaporn Kansomkeat, "Path coverage test case generation using genetic algorithms," *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, no. 2, pp. 115-119, November, 2017. Article (CrossRef Link)

[9] R. Ayachi, H. Bouhani and N.Ben Amor, "An evolutionary approach for learning opponent's deadline and reserve points in multi-issue negotiation," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 5, no. 3, pp. 131-140, 2018.Article (CrossRef Link)

[10] R. Kaur, S. Arora, "Nature inspired range based wireless sensor node localization algorithms," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 6, pp. 7-17, 2017. Article (CrossRef Link)

[11] Xia CY, Zhang Y and Song L, "Evolutionary generation of test data for paths coverage based on node probability," *Journal of Software*, vol. 27, no. 4, pp. 802-813, 2016.(in Chinese with English abstract). Article (CrossRef Link)

[12] Qian ZS, Hong DF and Wang XJ, "A plug-in test case generation method based on contact layer proximity and node probability coverage," *International Journal of Performability Engineering*, vol. 13, no. 8, pp. 1281-1292, 2017. Article (CrossRef Link)

[13] Qin XJ, Zhou L, Chen ZN and Gan ST, "Software vulnerable trace's solving algorithm based on lazy symbolic execution," *Chinese Journal of Computers*, vol. 38, no. 11, pp. 2290-2300, 2015. (in Chinese with English abstract). Article (CrossRef Link)

[14] Wen S, Xu J, Yuan LY, et al., "A test case generation approach for exploiting access control vulnerability based on policy inference," *Chinese Journal of Computers*, vol. 40, no. 12, pp. 2659-2670, 2017. (in Chinese with English abstract). Article (CrossRef Link)

[15] Tang EY, Zhou Y, Ou JS, and Chen X, "Test generation approach guided by linear fitting for Condition/Decision coverage criteria," *Journal of Software*, vol. 27, no. 3, pp. 593-610, 2016. (in Chinese with English abstract).

[16] Wang Y, Yu H and Zhu ZL, "A class integration test order method based on the node importance of software," *Journal of Computer Research & Development*, vol. 53, no. 3, pp. 517-530, 2016. (in Chinese with English abstract). Article (CrossRef Link)

[17] Pan WF, Li B, Zhou XY, et al., "Regression test case prioritization based on bug propagation network," *Journal of Computer Research & Development*, vol. 53, no. 3, pp. 550-558, 2016. (in Chinese with English abstract). Article (CrossRef Link)

[18] You F, Zhao RL and Lv SS, "Output domain based automatic test case generation," *Journal of Computer Research & Development*, vol. 53, no. 3, pp. 541-549, 2016. (in Chinese with English abstract). Article (CrossRef Link)
[19] Wang KC, Wang TT, Su XH, et al., "Test case selection for improving the effectiveness of software fault localization," *Journal of Computer Research & Development*, vol. 51, no. 4, pp. 865-873, 2014. (in Chinese with English abstract).
[20] Mao CY, Yu XX and Xue YZ, "Algorithm design and empirical analysis for particle swarm optimization-based test data generation," *Journal of Computer Research & Development*, vol. 51, no. 4, pp. 824-837, 2014. (in Chinese with English abstract).
[21] Yao XJ, *Theory of evolutionary generation of test data for complex software and applications*. PhD Dissertation. Jiangsu: China University of Mining and Technology, 2011 (in Chinese). Article (CrossRef Link)
[22] Ahmed M. A. and Hermadi L, "GA-based multiple paths test data generator," *Computer & Operations Research*, vol. 35, no. 10, pp. 3107-3127, 2008. Article (CrossRef Link)
[23] Zhang Y, "Theories and methods of evolutionary generation of test data for path coverage," *PhD Dissertation. Jiangsu: China University of Mining and Technology*, 2011 (in Chinese). Article (CrossRef Link)
[24] Gong DW and Zhang Y, "Novel evolutionary generation approach to test data for multiple paths coverage," *ACTA ELECTRONICA SINICA*, vol. 38, no. 6, pp. 1299-1304, 2010. (in Chinese with English abstract). Article (CrossRef Link)
[25] Suresh Y and Rath S K, "A genetic algorithm based approach for test data generation in basis path testing," *Computer Science*, vol. 3, no. 3, pp. 326-332, 2014.Article (CrossRef Link)
[26] Delahaye M, Botella B, and Gotlieb A, "Infeasible path generalization in dynamic symbolic execution," *Information & Software Technology*, vol. 58, no. 6, pp. 403-418, 2015. Article (CrossRef Link)
[27] Irfan S and Ranjan P, "A concept of out degree in CFG for optimal test data using genetic algorithm" in *Proc. of International Conference on Recent Advances in Information Technology. Dhanbad, India*, pp.436-441, March, 2012. Article (CrossRef Link)
[28] Hermadi I, Lokan C and Sarker R, "Dynamic stopping criteria for search-based test data generation for path testing," *Information & Software Technology*, vol. 56, no. 4, pp. 395-407, 2014. Article (CrossRef Link)
[29] Jung D, Suh T, Yu H and Gil J M, "A workflow scheduling technique using genetic algorithm in spot instance-based cloud," *Ksii Transactions on Internet & Information Systems*, vol. 8, no. 9, pp. 3126-3145, 2014.
[30] Thammano A and Teekeng W, "A modified genetic algorithm with fuzzy roulette wheel selection for job-shop scheduling problems," *International Journal of General Systems*, vol. 44, no. 4, pp. 499-518, 2014. Article (CrossRef Link)
[31] SJB Castro and RG Crespo, VHM García, "Patterns of software development process," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 1, no. 4, pp. 33-40, 2011. Article (CrossRef Link)

Zhongsheng Qian graduated from the School of Computer Engineering and Science, Shanghai University, China, for the degree of PhD in 2008. He entered the Post-doctoral station of Computer Science & Technology Department in Jiangxi University of Finance & Economics, China, from 2013 to 2015. He visited Aalborg University, Denmark as a guest professor in 2016. Now he is a professor and PhD supervisor of the School of Information Management, Jiangxi University of Finance & Economics, Nanchang, China. His current research interests include algorithmic design, software testing, software engineering, formal verification.

Dafei Hong is a master student from the School of Information Management, Jiangxi University of Finance & Economics, Nanchang, China. His research interests include software testing, plug-in testing & verification, software engineering.

Chang Zhao is a master student from the School of Information Management, Jiangxi University of Finance & Economics, Nanchang, China. Her research interests include recommender system and deep learning.

Jie Zhu is a master student from the School of Information Management, Jiangxi University of Finance & Economics, Nanchang, China. Her research interests include software testing, software engineering, and automatic test data generation.

Zhanggeng Zhu is a master student from the School of Information Management, Jiangxi University of Finance & Economics, Nanchang, China. His research interests include recommender system and social network analysis.