Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 7601096, 9 pages
https://doi.org/10.1155/2022/7601096

Research Article

Optimal Model of Software Testing Path Selection Based on Genetic Algorithm and Its Evolutionary Solution

Lili Zhan

School of Information Engineering, Harbin University, Harbin, Heilongjiang 150086, China

Correspondence should be addressed to Lili Zhan; zhanlili@hrbu.edu.cn

Received 4 May 2022; Revised 17 May 2022; Accepted 25 May 2022; Published 15 June 2022

Academic Editor: Kalidoss Rajakani

In software testing, the selection of test data is a difficult problem in structural testing. Whether the test data is appropriate or not is directly related to whether the error can be expected to be detected. In the process of software testing, the generation of test data is not only the core problem but also the key and difficulty of software testing. Because of the huge number of test cases and low test efficiency, a powerful optimization algorithm is needed to optimize the initial test cases. As a robust search method, genetic algorithm shows unique advantages and high efficiency in solving high-complexity problems such as large space, multipeak, nonlinear, and global optimization. Based on the application of genetic algorithm, this paper analyzes the optimization path by classifying and calculating the objective function and introducing NSGA-II algorithm, measures the distance between each branch on the processing path sample set, and sorts the path set to obtain the optimal solution. On the basis of the designed model, the experimental results show that the error control rate of the model is 89.4%. Moreover, because of the superiority of NSGA-II algorithm, the probability of comprehensive cross mutation is increased by 56.7%.

1. Introduction

With the rapid development of computer science and technology, the continuous emergence of new computer software and hardware technologies, and the wide and in-depth application of computers in various industries of the national economy, computer software technology, as an indispensable part of computers, plays a more and more important role [1]. Ensuring software quality has become an important topic in the field of software in recent years. Some defects in the software will probably directly lead to software failure, especially in some key areas with large number of users or high safety factor [2]. Software testing is an indispensable part of the software life cycle, and it is also a very complex process, which may greatly extend the time of the software life cycle [3]. The rapid development of computer makes software face more and more requirements. These requirements often make the software system become huge and make the overall complexity of the software system higher and higher, and the demand for software testing and reliability is stronger and stronger [4]. However, people's ability to develop high-quality software lags far behind the increasing demand of the society for computer software, and there are many hidden faults and defects in the developed software system.

Undoubtedly, improving software quality, like improving software productivity, has become a problem that must be always concerned and solved in the whole software development process [5]. The purpose of software testing is to generate test data and find the errors in these test data. Therefore, an excellent system test environment should have the ability to differentiate good test data from bad test data. It should be able to detect good test data and generate them [6]. For general complex software, path coverage is an important test method, and it has been proved that many test methods can be transformed into path coverage test data generation. Most genetic algorithm-based path coverage test data generation methods generally cover one target path at a time [7]. Testers generally use manual methods to design test data. The automatic generation of test data will effectively reduce the labor intensity of testers and save the cost of software development [8]. For the multipath coverage test data generation problem, it is transformed into a function optimization problem, and a mathematical model is established [9]. In order to reduce the computational cost, the coarse-grained function is used as the objective function. The target paths are grouped according to the objective function, and each group of individuals conducts independent evolution [10]. As a technology, genetic algorithm is used in the process of automatic test cases. According to the existing research, evolutionary testing is often called genetic algorithm in the literature.

Software testing is one of the main feasible methods to increase programmers' great confidence in the correctness and reliability of software [11]. Contrary to static analysis, the software execution involved in dynamic testing tools is the feedback of generating test data according to software testing and relying on software implementation. Take the logical precautions of the original software to ensure that these additional instructions have no impact [12]. Optimization technology is a risk-based regression testing strategy, which aims at finding the most potential defects in the tested software with the least number of test cases, including test case selection technology, test case minimization technology, and test case prioritization technology. The optimization model of the above problem is established, and the multiobjective evolutionary algorithm is used to solve it [13]. The idea is as follows: first, take multiple paths as decision variables; establish a multiobjective optimization model based on the number of edges, paths, and their coverage difficulty contained in the decision variables; then, use multiobjective evolutionary algorithm to solve the model, and obtain the target path set [14]. However, in the above research and analysis, the problem of software test path selection optimization model and its evolutionary solution has not been well solved. Therefore, this paper puts forward the following innovations:

(1) For the software with random data and other uncertain parameters, the test adequacy criterion of path test with random data is given. According to the given test criteria, the optimization model is established. To solve the problem of test data generation, an evolutionary solution method based on genetic algorithm is adopted [15]. For the path coverage problem of this kind of software, establish an appropriate test adequacy criterion

(2) Aiming at the problem of increased testing difficulty caused by the existence of multiple processes and communication statements in parallel programs, this paper proposes a path test selection method suitable for this situation. In the process of establishing the design model, taking multiple parallel program paths as decision variables, the objective function is compared and analyzed through adaptive function, NSGA-II, and other algorithms, and the optimal solution is obtained under linear constraints

The chapters of this paper are arranged as follows: The first chapter of this paper is the introduction, which discusses the background and significance of the topic selection and expounds the innovation of the article. In the second chapter of this paper, the innovative achievements and the research ideas of this paper are put forward based on the research achievements in the field of optimization model and evolutionary solution of software testing path selection by genetic algorithm at home and abroad. The third chapter of this paper is the method part, which deeply discusses the application and principle of related algorithms, and based on the previous research results, combined with the innovation of this paper, a new software testing path selection optimization model and its evolutionary solution model are proposed. The fourth chapter of this paper mainly discusses the experimental part of the application of the algorithm. Through the experimental results, on the basis of sorting out the data, an optimization model is established. The fifth chapter is the summary part, which summarizes the research results and shortcomings of this paper, as well as the prospect of follow-up research.

2. Related Work

Aghabeig and Jaszkiewicz think that regression testing is an important stage in the process of software testing. When the code is modified, the software hardware platform is changed or the hardware configuration is changed; regression testing must be carried out. As an integral part of the software life cycle, regression testing occupies a large proportion of the workload in the whole software testing process, and multiple regression tests will be conducted in each stage of software development [16]. Hu et al. proposed that the test case optimization technique is a risk-based regression testing strategy. Its purpose is to use the fewest test cases to find the most potential defects in the software under test, and on the one hand, it can save computing resources and time and, on the other hand, can reflect the adequacy of the test [17]. The static analysis tool pointed out by Tong et al. can analyze the code not executed under the test software whether manually or automatically. It is a limited plan that contains array references, pointer variables, and other dynamic structure analysis techniques [18]. Martowibowo and Kaswadi put forward a theory and method of software testability transformation based on the dominance relationship of target statements. The basic idea is if there is another target statement that makes the target statement dominate the original target statement, the new target statement will replace the original target statement to generate test data, thus eliminating the adverse effects of the marked variables [19]. Research by Adame and Salau shows that static testing is the process of finding possible errors in program code or evaluating program code without executing program code. Dynamic testing examines the dynamic behavior and results of a program by running it on sample test data to find bugs. Dynamic testing consists of three core components: generating test cases, running programs, and verifying program results [20]. Liu et al. proposed how to reasonably and effectively select the test data as the input from the huge input, so as to meet the specific coverage criteria, and how to design good test data, which will affect the test quality and then the software quality. Generating test data reasonably is the basis of implementing efficient software testing [21]. WangPing et al. used interval operation to select the path

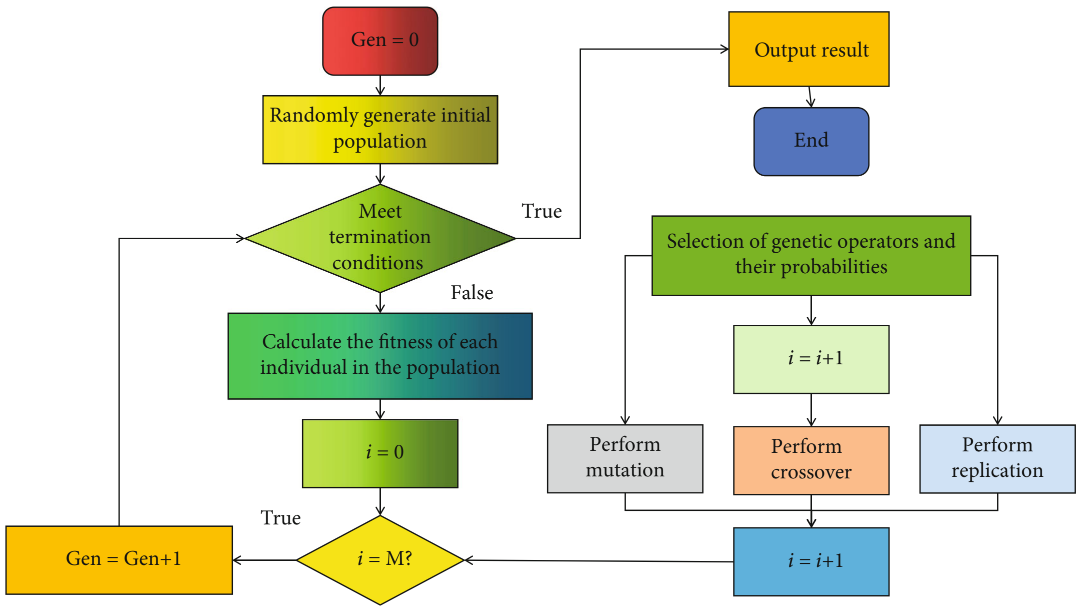
Gen = 0
Randomly generate initial population
Meet termination conditions
True
False
Output result
End
Selection of genetic operators and their probabilities
Calculate the fitness of each individual in the population
$i = i+1$
$i = 0$
Perform mutation
Perform crossover
Perform replication
True
Gen = Gen+1
$i = M?$
$i = i+1$

Figure 1: General flow chart of genetic algorithm.

for the problem of unreachable path in cell coverage test. The method first selects a section of the path, then judges the reachability of the path through interval operation, and if it is unreachable, it modifies it and finally obtains a reachable path containing elements to be covered [22]. Han pointed out that software testing is the execution of a software system or component under given conditions and the act of observing or recording the results. A test case used in software testing is a collection of data used for a specific purpose, such as input data, program paths, execution conditions, and test requirements [23]. The research results of Cao et al. show that component-based software engineering is a technology to ensure efficient and high-quality software development. However, for component-based software, how to ensure software reliability is also a difficult problem for test engineers, and software testing is a necessary means to ensure software reliability [24]. Li et al. put forward that some representative operations or data will be selected in software testing, and they will form test cases. By using these operations and data on the tested program, the actual feedback information of the tested program will be obtained, and then, it will be compared with the expected results, and finally, the conclusion of whether the tested program meets the expected results will be obtained [25]. Shakya and Smys' studies show that each chromosome corresponds to a solution to the problem. The genetic algorithm starts from the initial population, adopts the selection strategy based on the proportion of fitness value to select individuals in the current population, and uses hybridization and mutation to generate the next generation population [26]. Al pointed out that if the GA algorithm does not make corresponding judgment on the test feedback and makes corresponding adjustment when necessary, the GA algorithm may derive illegal data that does not meet the definition of the program under test. Bryan F. Jones converts variables into bit strings and uses bits as nodes for mutation and crossover. In small populations, better optimization results can be obtained only by mutation [27]. Alphonse et al. think that by combining complex network with software testing, the program is constructed into a weighted complex network, and the key nodes and key test paths in the program are found out. Then, the software test paths are clustered, and the representative paths are selected from each class to get the minimum test path [28].

Based on the research of the abovementioned related work, this paper determines the positive role of genetic algorithm in the field of software testing path selection optimization model and its evolutionary solution and builds a genetic algorithm model that combines multiple algorithms. Genetic algorithm analysis conducts in-depth analysis and research, uses data more effectively, mines valuable knowledge hidden behind data, and discovers and finds potential problems that affect Ruan Jiyou's test path selection optimization model and evolutionary solution.

3. Methodology

3.1. Analysis and Research of Related Theories

3.1.1. Software Test Data Generation Based on Genetic Algorithm. A general genetic algorithm is an adaptive global optimization probabilistic search algorithm formed by simulating the genetic and evolutionary processes of organisms in the natural environment. GA search algorithm is an iterative process search algorithm of "survival + detection." In order to solve the problem of circular path, this paper proposes a new path exploration method, which reduces the number of states required to achieve high coverage. This method first determines the state priority and then prunes the new states created without changing the code coverage state. The genetic algorithm takes each chromosome in the population as the object and uses random method to guide the coded

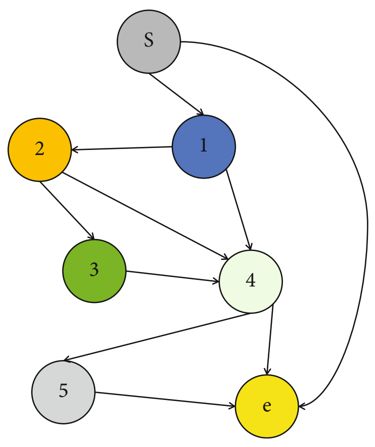

Figure 2: Example diagram of program control flow.

parameter space to perform evolutionary search to find the optimal solution. It has the characteristics of self-organization, parallelism, independence, and multisolution, which is in line with the analysis of the software test path experiment in this paper. Figure 1 is the general flow of the genetic algorithm.

As can be seen from the above figure, firstly, in the randomly generated initial population, the number of individuals is generally certain, and each individual is represented as chromosome-based coding. Then, the fitness of each individual is calculated to judge whether it meets the optimization criteria. If it does, the optimal individual will be output. At the same time, it also has the optimal solution. If it does not meet the requirements, the circular calculation will continue to screen. In the regenerated individuals, the high and low fitness screening, the high fitness will have a high probability of being selected; on the contrary, it will have a low probability. According to a certain method of variation and crossover, the optimal solution analysis is continued in the new generation of population selected by Shuai. The optimal solution set can be obtained by reciprocating calculation.

The genetic algorithm is superior to the traditional optimization method because its search space is a population, not a single solution; the genetic algorithm uses the fitness function in the evolution process, which can solve all kinds of fitness functions and constraints; the genetic algorithm does not use the determination The state transition rule adopts probability, so it can perform global search very well. Because the self-moderation should be considered when applying genetic algorithm to solve the optimal analysis of software path testing, it is necessary to design an appropriate fitness function to evaluate the performance of test data. By replacing the original samples and data on the template of genetic algorithm, the process of generating software test data using genetic algorithm can be expressed as follows: coding the input data to form individuals in the initial population; using random method to produce a specific number of individuals, so as to form the initial population; taking the generated individual as input, the tested program can be run after inserting, so as to obtain the individual fitness; genetic screening of individuals according to fitness to form the next

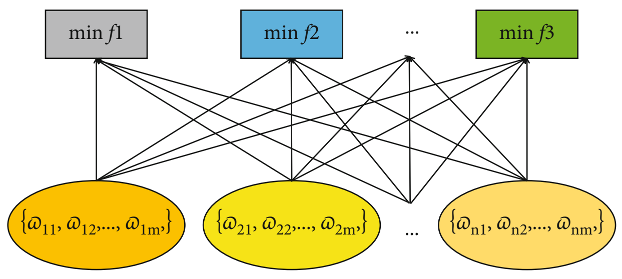

Figure 3: Schematic diagram of information sharing among individuals.

generation of population; and cycle the above process until an individual meeting the conditions is found, or the given termination criteria are met.

3.1.2. Software Test Path and Data Filtering. In order to improve the generation efficiency of multipath coverage test data, a suitable mathematical model must be established. In fact, in the process of evolution, the generated test data may not cover the current target path but may cover other target paths. Therefore, this paper chooses an appropriate method to improve the utilization rate of test data. The success of the fitness function directly affects the success of the algorithm. The function value of fitness function represents the test efficiency of test cases in the parameter domain corresponding to an individual, and the test cases corresponding to individuals with high fitness have higher test efficiency. Such test cases can cover as many component-based software paths as possible in a shorter time and find more component-based software errors.

It is necessary to introduce the evaluation function. The path crossed by the test data X is $P(X)$. It is assumed that the branch distance of $P(X)$ from the target path is $B_p(X)$. Then, for the test data X of the target path, the defined evaluation function is expressed as follows:

$$g_P(X) = f_P(X) + \left(1 - 1.01^{-B_P(X)}\right). \quad (1)$$

If and only if the path X traversed by $P(X)$ is the target path, that is, $P(X) = P$, then $g_P(X) = 0$. Obviously, the smaller the value of $g_P(X)$, the closer X will be to the test target. Figure 2 shows an example of program control flow.

If there are three test data, and the second and third test data will execute condition 2, the first test data will deviate from the branch of the original objective function after completing condition 1, so the fitness of the second and third test data will be much better than test data 1. At this time, this paper assumes that the condition is $if\, a \geq 8$. When inputting data ϖ, the value of a after running the assumption is recorded as $a(\varpi)$, and the calculation formula of branch distance corresponding to this input is as follows:

$$\mathrm{dist}(\varpi) = \begin{cases} 0, a(\varpi) \geq 8, \\ 8\text{-}a(\varpi), a(\varpi) < 8. \end{cases} \quad (2)$$

At this time, this paper uses fitness(ϖ) to represent the fitness function of individual ϖ. In general, fitness(ϖ) is the

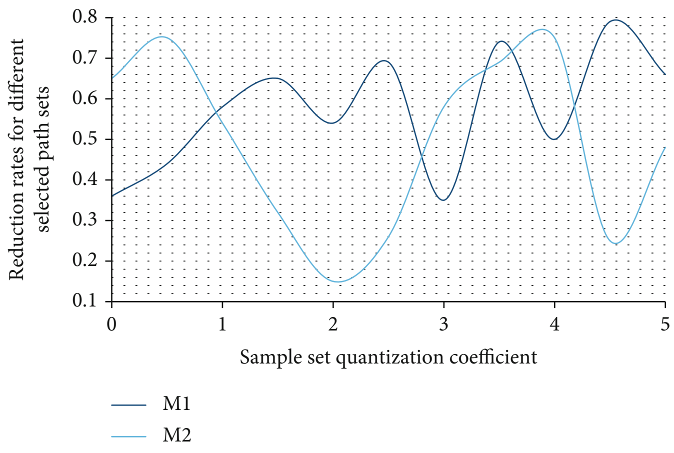

Figure 4: Reduction rates for different selected path sets.

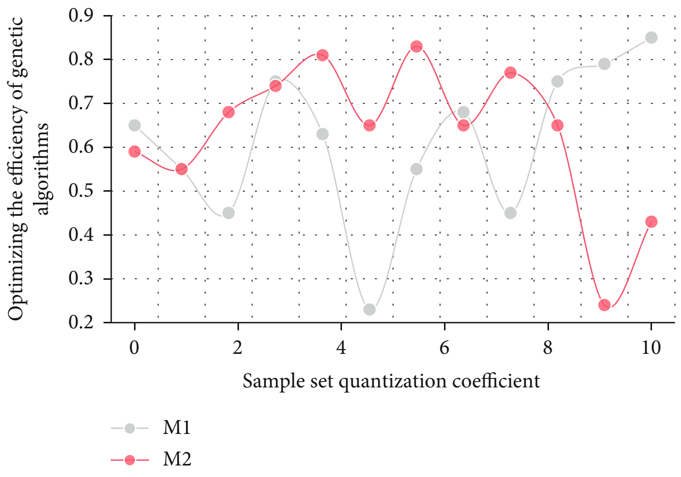

Figure 5: Optimizing the efficiency of the genetic algorithm.

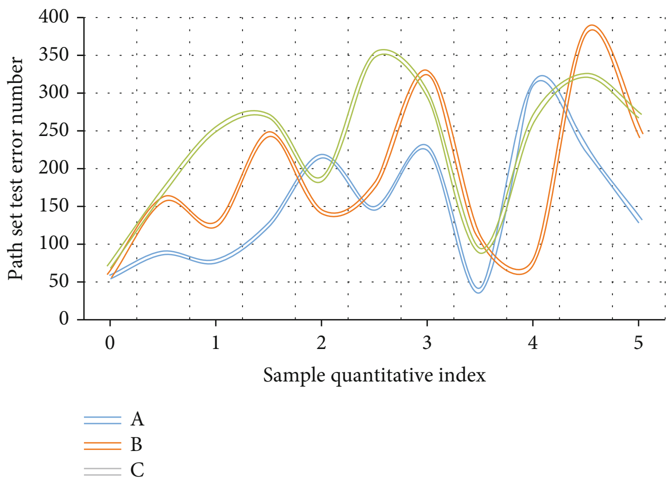

Figure 6: Path set test error number.

sum of layer proximity and branch distance, as follows:

$$\mathrm{fitness}(\varpi) = \mathrm{Appr}(\varpi) + \mathrm{Normal}(\mathrm{dist}(\varpi)). \tag{3}$$

Among them,

$$\mathrm{Normal}(\mathrm{dist}) = 1 - 1.001^{-\mathrm{dist}}. \tag{4}$$

In the above formula, it is the calculation formula of standardized branch distance. ϖ contains exactly all test data, which is a necessary and sufficient condition for individual fitness function $\mathrm{fitness}(\varpi) = 0$. Therefore, when the individual fitness function ϖ is smaller, it means that the distance between $\mathrm{fitness}(\varpi)$ and the target is smaller. In this way, the genetic algorithm can be used to solve the problem from the perspective of minimizing the problem.

With the solution of subproblems, the number of population will decrease correspondingly. However, in the traditional multipopulation genetic algorithm, the number of population is always the same. Finally, there are differences in communication methods among populations. Therefore, this paper proposes an individual information sharing between different populations, so it is also necessary to share information between individuals. Using the form of simultaneous evolution of multiple groups, each subpopulation optimizes a subproblem. In this way, individual information is shared among different populations during evolution to increase the solution range of various groups, so as to improve the solution efficiency. Generally, a random method is selected to generate a subpopulation with a population size of i for the $\min f_i(\varpi i)$th suboptimization problem m as follows:

$$M(P_i) = \{\varpi i1, \varpi i2, \cdots, \varpi im\}. \tag{5}$$

Including $i = 1, \cdots, n$. ϖ_{ij} represents the i individual in the j subpopulation. All subspecies evolve independently. Therefore, $f_i(\varpi_{ij})$ represents the fitness of individual ϖ_{ij}. Figure 3 is a schematic diagram of information sharing among individuals.

Therefore, when it is necessary to judge whether the individual ϖ_{ij} is the optimal solution of the kth optimization subproblem $\min f_i$, it is not necessary to calculate $f_k(\varpi_{ij})$, and it is only necessary to judge whether the path ϖ_{ij} traversed by the individual $P(\varpi_{ij})$ is equal to the target path P_k. If $P(\varpi_{ij}) = P_k$, then ϖ_{ij} is the optimal solution of the k subproblem; otherwise, ϖ_{ij} will not participate in the evolution of the k subpopulation. At the same time, there is no need to calculate the fitness value ϖ_{ij} of k to the $f_k(\varpi_{ij})$ optimization problem at this time.

3.2. Optimization Algorithm Design and Evolutionary Computation. We want to get as many test path sets as possible, so that more sample sets can be included in the calculation, so as to meet the adequacy criterion of the test. Set the path set corresponding to the decision variable as X, and the

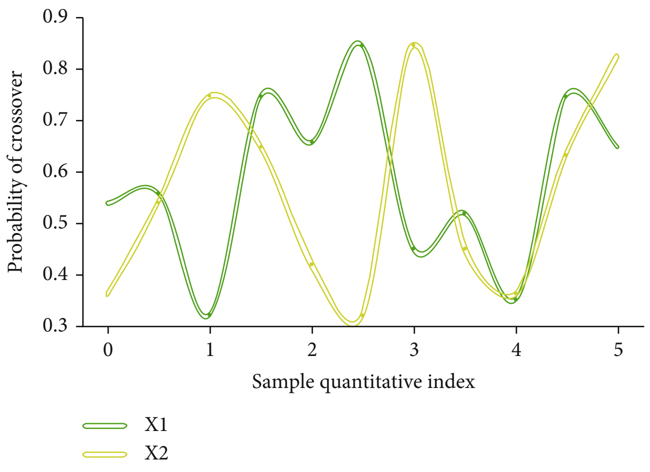

FIGURE 7: Probability of cross-variation.

set including edges is

$$E = x_1 \cup x_2 \cup \cdots \cup x_m. \tag{6}$$

Then, the number of edges $|E|$ contained in the path at this time can be expressed as follows:

$$f_1(X) = -|E| = -|x_1 \cup x_2 \cup \cdots \cup x_m|. \tag{7}$$

In the set of selectable paths, it is hoped that redundant paths can be reduced, so the number of paths needs to be as small as possible. This is because the fewer paths, the less test data need to be generated, thus reducing the time consumption of path testing. Halstead measure is introduced here, which can effectively express the complexity of the program, and it is expressed as follows:

$$D(x) = \theta_i n^* + \theta_j \widehat{N}^*. \tag{8}$$

In the above formula, $n^*, \widehat{N}^*$ is obtained after the linear normalization of $n, \widehat{N}$. When the influence on Halstead measure is different, the given θ is also different. In the processing of the above indicators, it is found that the smaller the difficulty of path coverage, the fewer functions and expressions in the path set, and the fewer edges and paths may be included. Therefore, there is a linear relationship $f(x)$, and it is in a positive correlation. So the optimization model of the objective function is

$$\min\left(f_i(X), f_j(X)\right). \tag{9}$$

Generally speaking, if a path set contains more edges, then more paths should be needed, and the path set coverage is difficult. However, we expect that the path set contains fewer paths and the coverage is less difficult. Therefore, including $f_i(x), f_j(x)$ two conflicting goals is a typical two-goal minimization problem. At this time, it is necessary to

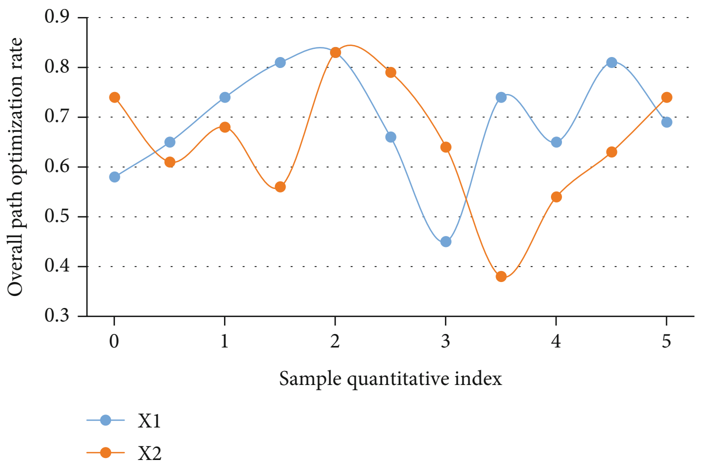

Figure 8: Overall path optimization rate.

adopt targeted methods to efficiently obtain the optimal solution set on the objective function of the problem.

Since NSGA-II is one of the most widely used and best effective methods for solving continuous optimization problems, it is quite reasonable to use this method to solve evolutionary problems in this paper. In the process of finding the target path set, the individual scale can be changed through the crossover operation, and new individuals can be generated. As can be seen from the above section, the individuals in this chapter are a set, and there are two ways to generate new individuals through crossing, the crossing between sets and the crossing within sets. Because of the crossover operation within the set, the newly generated individual will contain unreachable or incomplete paths. At this time, you can get:

$$X_1 = \left(x_1^1, x_1^2, \cdots, x_1^\alpha\right), \tag{10}$$

$$X_2 = \left(x_2^1, x_2^2, \cdots, x_2^\beta\right). \tag{11}$$

This is for individuals in a single set, but it also has a good effect in the whole changeable genetic process. Judge the individuals who need to perform mutation operation according to the individual mutation probability: then, randomly select a gene position that needs mutation, that is, the mutation point; finally, a path is randomly selected from the initial path set to judge whether the path is repeated with the path in the current individual. If the formula is repeated on this path, a new path needs to be selected. If it is not repeated, the path needs to be replaced by the path on the mutation point. If no such path exists in the initial set of paths, no replacement is required.

4. Result Analysis and Discussion

In order to establish a scientific, feasible, and practical model system, it is the prerequisite and foundation for correctly calculating the optimization path and the software testing path of evolutionary analysis. Therefore, the principles of integrity, scientificity, and accuracy should be followed when establishing the model structure. This paper designs the optimization and evolutionary calculation model of software test path based on genetic algorithm. In the experiment, the reliability and accuracy of the model will be described from several important parameters: the reduction rate of different selected path sets, the efficiency of optimized genetic algorithm, the number of path set test errors, the probability of cross mutation, and the overall path optimization rate. In the following, M1 and M2 are selected as the experiments on the reduction rate and the optimization of the efficiency of the genetic algorithm for different path sets, as shown in Figures 4 and 5.

For each selected experiment path set, the selected method is path set, which can greatly reduce the number of paths. Among them, the average reduction rate is the largest M1, up to 57.33%, and the smallest M2. Compared with the same multiobjective function, the number of paths will be less, and the reduction rate will be higher. However, the algorithm designed in this paper can not include all edges, including reachable edges, which means that it can not meet the adequacy criterion of the test. Therefore, when selecting the path set, we should pay attention to the number of redundant paths in order to affect the final result. In terms of the efficiency of the genetic algorithm, it can be clearly found that the optimized calculation process has been greatly simplified. With the increase of the number of sample sets, the overall trend is upward, which also shows that the efficiency is continuously increasing. However, according to the actual calculation, it will not increase when it reaches the average value of 75.8%. Although there is infinite growth in theory, in practice, due to the constraints of the site and energy, there will be no extreme growth. But overall, the efficiency of the optimized genetic algorithm is improved by 70.9%. Let A, B, and C be the experimental set of the selected unprocessed path set on the number of path set test errors, as shown in Figure 6.

Since there are always errors in the actual test, it is necessary to control the number of error individuals, which is also necessary to obtain accurate results. In the figure above, it can be found that sample set A has a good stability and is at a low level on the whole quantization axis. The other two sample sets B and C, although compared with each other, have a large number of error individuals, but in the whole sample set, their error individuals are extremely few. Because the cyclic calculation in genetic algorithm often produces repeated results, there are still repeated errors in the error individuals, but they will also be counted as the second error individuals or even multiple error individuals in model detection. Therefore, but overall, the error control of the model designed in this paper has been in a reasonable range, and the comprehensive error control rate is 89.4%. Let $X1$ and $X2$ be the test sample set on the probability of cross mutation and the overall path optimization rate. The experimental analysis are shown in Figures 7 and 8.

In the calculation of genetic algorithm, crossover and mutation often occur, which is also determined by the characteristics of genetic algorithm. In the general sense of inheritance, the increase of the probability of crossover and mutation means that the number of samples that can be generated is greatly increased, which is conducive to the emergence of excellent samples, and an optimized sample set can be formed as soon as possible. After the experiment, the cross mutation rate of the model designed in this paper basically keeps a high state, which will be more powerful than the selection of excellent subsets and facilitate the optimization calculation. Because NSGA-II algorithm is embedded in the analysis, it also strengthens the efficiency and accuracy of continuous optimization. With the support of optimization calculation, the probability of comprehensive cross-mutation is increased by 56.7%. Figure 8 shows the analysis of the overall model, which basically maintains the stability in the optimization scheme and calculation, which plays a good role in a large number of repeated calculations to prevent large differences in the processing of different sample sets, resulting in the failure of sample sets.

5. Conclusions

The path generation and selection method in the existing path test have low test efficiency, high test overhead, and easy generation of redundant test paths, which makes the test efficiency difficult to meet the requirements. This paper studies the path selection optimization model for software testing and its evolutionary solution. Test case prioritization is implemented with genetic algorithm. Genetic algorithm is a method to search the optimal solution by simulating the natural evolution process. This paper proposes a genetic algorithm suitable for the optimal path prioritization of test software. In the selection stage, in order to select individuals more accurately, we use fine-grained fitness function to select individuals. By using local evolutionary genetic algorithm, we can generate test data more efficiently, and by selecting different evaluation functions at different stages of the algorithm, the calculation cost is greatly reduced. The model is based on genetic algorithm, combined with the fitness function formulated according to the actual situation; uses the selection, replication, crossover, mutation, and other operations in genetic algorithm; takes the methods of data flow chart analysis and selection, program insertion, and so on as the auxiliary; and calculates the test data of the optimal path according to the evolution of the selected path. But overall, the model designed in this paper has been in a reasonable range for error control, and the comprehensive error control rate is 89.4%. Moreover, due to the superiority of the NSGA-II algorithm, it also strengthens the efficiency and accuracy of continuous optimization. With the support of optimization calculations, the probability of comprehensive crossover variation is improved by 56.7%.

Data Availability

The figures used to support the findings of this study are included in the article.

Conflicts of Interest

The author declares that there are no conflicts of interest.

References

[1] S. Wang, J. Li, and X. Gao, "Optimization of cutting parameters for complex surface NC machining based on genetic algorithm," *Boletin Tecnico/Technical Bulletin*, vol. 55, no. 12, pp. 86–92, 2017.

[2] M. Nabil and E. A. Hamra, "A hybrid approach based on genetic algorithm and particle swarm optimization to improve neural network classification," *Journal of Information Technology Research*, vol. 10, no. 3, pp. 48–68, 2017.

[3] E. Chiarito, F. Cigna, G. Cuozzo et al., "Biomass retrieval based on genetic algorithm feature selection and support vector regression in Alpine grassland using ground-based hyperspectral and Sentinel-1 SAR data," *European Journal of Remote Sensing*, vol. 54, no. 1, pp. 209–225, 2021.

[4] S. Sui, H. Ma, H. W. Chang, J. F. Wang, Z. Xu, and S. B. Qu, "Optimization design of metamaterial absorbers based on an improved adaptive genetic algorithm," *Applied Computational Electromagnetics Society Journal*, vol. 34, no. 8, pp. 1198–1203, 2019.

[5] V. Jamshidi, V. Nekoukar, and M. H. Refan, "Analysis of asynchronous distributed multi-master parallel genetic algorithm optimization on CAN bus," *Evolving Systems*, vol. 11, no. 4, pp. 673–682, 2020.

[6] C. Jiang, K. Hao, W. Pedrycz, L. Chen, and X. Cai, "Optimization control method for industrial Internet of Things based on biological adaptive coevolutionary," *Wireless Networks*, vol. 27, no. 8, pp. 5145–5160, 2021.

[7] A. A. Taleizadeh, R. Askari, and I. Konstantaras, "An optimization model for a manufacturing-inventory system with rework process based on failure severity under multiple constraints," *Neural Computing and Applications*, vol. 34, no. 6, pp. 4221–4264, 2022.

[8] H. Moayedi, M. A. Mu'Azu, and L. K. Foong, "Swarm-based analysis through social behavior of grey wolf optimization and genetic programming to predict friction capacity of driven piles," *Engineering with Computers*, vol. 37, no. 2, pp. 1277–1293, 2021.

[9] N. Charhouni, M. El Amine, M. Sallaou, and K. Mansouri, "A preference-based multi-objective model for wind farm design layout optimization," *International Journal on Interactive*

Design and Manufacturing (IJIDeM), vol. 16, no. 1, pp. 323–337, 2022.

[10] T. Cui, W. Zhao, and C. Wang, "Design optimization of vehicle EHPS system based on multi-objective genetic algorithm," *Energy*, vol. 179, no. 15, pp. 100–110, 2019.

[11] A. Davahli, M. Shamsi, and G. Abaei, "Hybridizing genetic algorithm and grey wolf optimizer to advance an intelligent and lightweight intrusion detection system for IoT wireless networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 11, pp. 5581–5609, 2020.

[12] E. Gunpinar and S. Khan, "A multi-criteria based selection method using non-dominated sorting for genetic algorithm based design," *Optimization and Engineering*, vol. 21, no. 4, pp. 1319–1357, 2020.

[13] Z. Xie, L. Li, and X. Luo, "A foot-ground interaction model based on contact stability optimization for legged robot," *Journal of Mechanical Science and Technology*, vol. 36, no. 2, pp. 921–932, 2022.

[14] J. Wang, Z. Cheng, O. K. Ersoy, F. Wang, P. Zhang, and Z. Dong, "Crop planting structure optimization based on improved real genetic algorithm," *International Agricultural Engineering Journal*, vol. 27, no. 2, pp. 169–185, 2018.

[15] M. Kamalinejad, H. Arzani, and A. Kaveh, "Quantum evolutionary algorithm with rotational gate and H_ϵ-gate updating in real and integer domains for optimization," *Acta Mechanica*, vol. 230, no. 8, pp. 2937–2961, 2019.

[16] M. Aghabeig and A. Jaszkiewicz, "Experimental analysis of design elements of scalarizing function-based multiobjective evolutionary algorithms," *Soft Computing*, vol. 23, no. 21, pp. 10769–10780, 2019.

[17] Q. Hu, J. Jia, Y. Zhu, J. Cao, and J. Li, "A multisource PNT fusion algorithm based on a variance genetic model," *International Journal of Control, Automation and Systems*, vol. 20, no. 4, pp. 1294–1304, 2022.

[18] X. Tong, J. Shen, and S. Yu, "Double optimization of the mesoscopic geometric parameters of ball-end milling cutters based on interactive experiment," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 16, no. 1, pp. 37–47, 2022.

[19] S. Y. Martowibowo and A. Kaswadi, "Optimization and simulation of plastic injection process using genetic algorithm and moldflow," *Chinese Journal of Mechanical Engineering*, vol. 30, no. 2, pp. 398–406, 2017.

[20] B. O. Adame and A. O. Salau, "Genetic algorithm based optimum finger selection for adaptive minimum mean square error rake receivers discrete sequence-CDMA ultra-wide band systems," *Wireless Personal Communications*, vol. 123, no. 2, pp. 1537–1551, 2021.

[21] R. Liu, R. Wang, R. Bian, J. Liu, and L. Jiao, "A decomposition-based evolutionary algorithm with correlative selection mechanism for many-objective optimization," *Evolutionary Computation*, vol. 29, no. 3, pp. 1–36, 2020.

[22] Z. WangPing, J. Min, Y. JunFeng, L. KunHong, and W. QingQiang, "The design of evolutionary feature selection operator for the micro-expression recognition," *Memetic Computing*, vol. 14, no. 1, pp. 61–76, 2022.

[23] X. Han, "A study of performance testing in configurable software systems," *Journal of Software Engineering and Applications*, vol. 14, no. 9, pp. 474–492, 2021.

[24] B.-w. Cao, X.-h. Liu, W. Chen, Y. Zhang, and A.-m. Li, "depth optimization analysis of articulated steering hinge position based on genetic algorithm," *Algorithms*, vol. 12, no. 3, pp. 55–55, 2019.

[25] G. Li, H. Wang, L. Xu, M. Wu, and N. Cao, "A hybrid optimisation algorithm based on genetic algorithm and ACO algorithm improvements for routing selection in heterogeneous sensor networks," *International Journal of Embedded Systems*, vol. 1, no. 1, p. 1, 2019.

[26] S. Shakya and S. Smys, "Reliable automated software testing through hybrid optimization algorithm," *Journal of Ubiquitous Computing and Communication Technologies*, vol. 2, no. 3, pp. 126–135, 2020.

[27] C. Al, "Genetic algorithm for test suite optimization: an experimental investigation of different selection methods," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 3, pp. 3778–3787, 2021.

[28] S. Alphonse, B. Jacques, N. Kitmo, R. Djidimbele, P. Andre, and K. Cesar, "Optimization PV/batteries system: application in Wouro Kessoum Village Ngaoundere Cameroon," *Journal of Power and Energy Engineering*, vol. 9, no. 11, pp. 50–59, 2021.