International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.6 (2015), pp.113-122
http://dx.doi.org/10.14257/ijmue.2015.10.6.11

# Automatic Generation of Test Cases Based On Multi-population Genetic Algorithm

Na Zhang, Biao Wu and Xiaoan Bao*

*Zhejiang Sci-Tech University, Hangzhou, Zhejiang Sci-Tech University, Hangzhou, Zhejiang Sci-Tech University, Hangzhou*
*zhangna@zstu.edu.cn, 615567120@qq.com, baoxiaoan@zstu.edu.cn*

***Abstract***

*The design of automatic generation technology of test case is an important part of the software test automation implementation, having an important guiding role in testing of late work, which is the fundamental guarantee to improve the reliability of software. In this paper, considering the lack of adequacy of control flow testing, using the data flow testing as the test adequacy criteria, and then on the basis of the single population genetic algorithm search efficiency is not high, combining with previous methods on the improvement of the genetic algorithm, introducing the concept of multi-population, and then designs a kind of improved parallel evolutionary algorithm (IPEA) based on multi-population is used to automatically generate test cases. The algorithm defined the concept of external pressure which as the degree of competition between individuals. Fully considering the influence of coverage, branch condition and degree of competition between individual species of three aspects, and give different weights, we design a fitness function to evaluate the merits of the individual species. Experiments show that the IPEA has obviously improved in convergence speed, search time, coverage, scale of the test cases on key performance than the single population genetic algorithm and random search algorithm.*

***Keywords***: *Evolutionary algorithm; multi-population; diversity of the population; test case generation*

## 1. Introduction

Automatic test cases generation technique can not only reduce the time of the test cases generation, but also can reduce the test cost, shorten the software development cycle, so it is been a widely research problem [1]. A data generation technique is an algorithm that generates test cases, whereas an adequacy criterion is a predicate that determines whether the testing process is finished. Several test data adequacy criteria have been proposed, such as control flow criteria and data flow criteria. One of the major difficulties in software testing is the automatic generation of test data that satisfy a given adequacy criterion.

Lin Wan [2], Baowen Xu [3], ALSHRAIDEH M [4] put forward their own self test cases automatically generate algorithm successively in view of the control flow of test coverage criteria. Because of test coverage criteria based on the data flow has higher test adequacy and more able to detect the hidden mistakes and defects in software better. Therefore, many scholars proposed kinds of automatic test case generation approaches in view of the data flow test coverage criteria [5]. Which using Genetic Algorithm to implement automatic test case generation technology is one of the commonly used simple and efficient methods [6].

Nowadays, many scholars proposed respective method using genetic algorithm in view of the test cases automatically generation. Such as Moheb R.Girgis [7] put forward technology which is based on the guidance structure for automatic generation of test data, the method uses Genetic Algorithm (GA), structural analysis to the data flow, satisfying the requirement of the certain coverage of test cases. Ahmed S. Ghiduk [8] put forward a kind of automatic test cases generation technique based on genetic algorithm, which enhance test cases on the basis of certain data flow coverage criterion for achieving a higher coverage and test adequacy. But, the inherent some defects in the single population Genetic Algorithm (such as premature convergence, easy to fall into local optimum, algorithm search efficiency low lately) as well as the reasonable evaluation function design and consider not comprehensive enough, which restricting its use in the automatic test cases generation technique development inevitably.

In this paper, firstly, introducing the concept of multi-population, which enhancing the search ability and improving the search efficiency on the basis of improvement of the predecessors Genetic Algorithm in allusion to the above mentioned deficiencies. Secondly, with the adaptive design of fitness function, the evaluation method avoids the phenomenon of the premature and local optimum in Genetic Algorithm. Finally, we design a parallel genetic algorithm based on multi-population, the algorithm has high execution efficiency and coverage for the test cases generated automatically on the premise of meet the data flow coverage criteria, the set of test cases obtained by the algorithm is more ideal, reducing the time of software test and the test cost, shorten the software release time.

## 2. Data Flow Test Coverage Criteria

This paper presents a method for automatic test data generation that uses a genetic algorithm, which is guided by the data flow dependencies in the program. In the individual fitness evaluation function structure, data flow coverage criterion plays an important role to search for test data.

[Weyuker, 8] has carried on the detailed analysis to the include relations between all kinds of data flow coverage criteria. [Mathur A P, 9] shows that under the most of data flow coverage criterions such as ALL EDGES, ALL NODES, the test data can’t test the program enough and can't fully exposes code defects through experimental analysis. Therefore, this paper adopts the ALL-DU-PATHS coverage with high requirements of rule as our test adequacy criteria to generate test data.

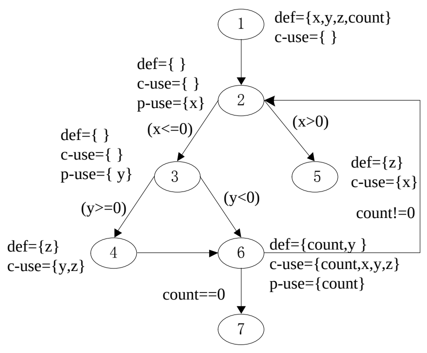


**Figure 1. Data Flow Graph for a Program**

Data flow graph (DFG) can be represented by a directed graph G (N, E) with a set of nodes and a set of edges. N is the set of nodes which represents a group of consecutive statements include the start node and the end node, which together constitute a basic block. E is the set of edges of the graph are then possible transfers of control flow between the nodes. A path is a finite sequence of nodes connected by edges. Considered the variables x in data flow graph G as shown in Figure 1, x is defined in the node 1 and used in the node 3. A path is def-use with respect to a variable if it contains no new definition of that variable, as a result, the node 1 to node 3 is an effective du-path to the variables x. If the test case set T executed a program, covers all the DU-PATHS, T satisfies ALL-DU-PATHS coverage criteria.

## 3. Multi-Population Parallel Genetic Algorithm

### 3.1 The Design of Multi-Population Model

In order to further improve the global search ability of genetic algorithm, the improved methods emerge in endlessly, in which multi-population genetic algorithm is presented in recent years, as a kind of improved genetic algorithm has good performance of the method, and has produced many model of genetic algorithm based on multi-population [10-12,16]. On the one hand, the multi-population model can avoid a fitness larger individual quickly spread resulted in missing some key information and search premature convergence before the global optimal solution are found, which can’t get the ideal optimal solution set; On the other hand, the multi-population model can improve the quality of the solutions and improve the evolution speed of genetic algorithm.

This paper makes use of the framework of multi-population parallel evolutionary, different populations are given different control for achieving the goal of different search finally. Will more child populations instead of the original single-population in the feasible solution domain to search, and each child population respectively parallel evolution, consequently, designed a kind of parallel population model. The model is shown in Figure2:

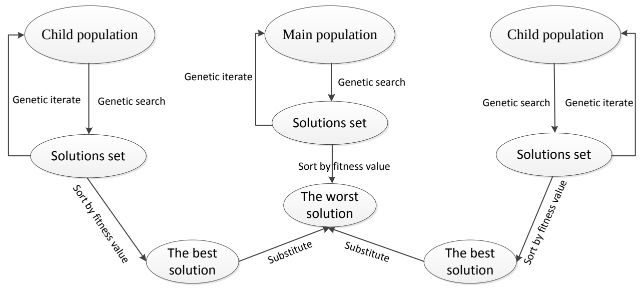


**Figure 2. Parallel Multi-Population Model**

In this model, there are three populations, a main population and two child populations. Two child populations are independent in the process of evolution, and doesn’t exchange information between each other, only provide the best individual evolution to the main population to replace the worst individual in each generation. The independence of main population between two populations each other is embodied in their respective evolution and genetic operation is not affected. The main population is random initialization, and the initialization method of two child population is one of the random and another child population is determined by the negation of the previous child population via finite invert. The two child populations are realized complementary, so that can reach the goal of more excellent individuals sharing and the main purpose of high species diversity in the entire population model. The individual similarity and the calculation of population diversity are defined as follows.

**Definition 1**: similarity among individuals, SAI: Quantitative descript the similarity between individual genes, and hamming distance is used to depict the similarity values.

Define the hamming distance between two individuals and is as follows:

$$d_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2} \tag{1}$$

Among them, $i, j = 1,2,\cdots,N$ and $i \neq \mathrm{j}$, $x_i$ represents the input variable, $x_{ik}$ represents the first $k$ symbols ("0" or "1") in the binary string of the $i$-th input variables, $N$ is the length of binary string.

**Definition 2**: population diversity, PD: Describe the diversity and universality of the individuals genetic in a population, which measured by the hamming distance between the individuals and the variance of fitness value.

The hamming distance between individuals in a population matrix of the:

$$D = \begin{bmatrix} d_{12} & \cdots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & d_{nn} \end{bmatrix}$$

The $i$-th hamming distance of an individual in a population are as follows:

$$d_i = \sum_{j=1}^{n} d_{ij} \tag{2}$$

The hamming distance of populations is:

$$D_k = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \tag{3}$$

The average fitness of population is as follows:

$$\overline{f} = \frac{1}{M}\sum_{i=1}^{M} f_i \tag{4}$$

The population fitness value variance as:

$$F_k = \frac{1}{M}\sum_{i=1}^{M}(f_i - \overline{f})^2 \tag{5}$$

Among them, $f_i$ is the fitness value of the $i$-th individual in the $k$-th population, $M$ is the total number of individual in the population.

The diversity value of the $k$-th population is:

$$V_k = \alpha D_k + (1-\alpha)F_k \tag{6}$$

The $\alpha(0 < \alpha < 1)$ is the weighting parameters.

### 3.2 The Design of Fitness Function

The structure of fitness function is the most critical part of genetic algorithm [13, 14], that is to design a standard to judge the pros and cons of a feasible solution. How to evaluate a test case (individual) under the guidelines of the data flow testing path coverage? The test case must be input into the program to execute and record coverage path. If so, according to the structural analysis for the data flow, the total number of paths is $n$, and a test case t article covers $m$ testing path, so the fitness function of the test case $t$ can use the formula as follows:

$$f_i(t) = \frac{m}{n} \tag{7}$$

The meaning of this formula is that a test case covers the more the number of testing paths its fitness value was greater, but the calculation of the fitness function completely ignores the influence of the branch predicate to the testing path. With different of the branch condition is taken the true value or the false value, the test path covered by the test cases is different. And that not considering the actual situation of the execution of the test cases, such as the influence of the change of the degree of competition between individuals on the test case execution. So this paper designed an improved calculation method of the individual fitness. This method not only considered the size of the coverage of test cases, but also the contribution of the branch conditions to the program execution is considered, and taken the influence of the change of outside pressure on the result of test case execution procedures into consideration. Now the analysis and the calculation of two factors are as follows:

(1) The instrumentation of branch function

Program instrumentation is insert some probes (also known as the "instrument") into the program on the basis of keeping the original logic and integrity of the test program, and throwing program running characteristics of the data through the execution of the probe [15]. Through analyzing the characteristics of data, obtaining the data flow information of the program, and the dynamic information, such as logic coverage. Therefore, this paper considered the effect of branch conditions on the fitness function, the branch function as the probe method is used. Using this method to obtain the necessary coverage path information and the executive branch conditions information on the time to the test case drives the program executed.

Assuming that the program $P$ gets the number of paths which need to be covered is $n$ after structural data flow analysis, and the number of paths which are already covered according to the article probe information is $m$, the number of executed predicate is $s$ in $n-m$ paths which are not covered after the test case $t$ driver the program execution, the branch function value of the $i$-th branch is $f_i(x)$, so the influence value of the branch conditions on the execution of a program for the test case $t$ is defined as follows:

$$Q(t)=\frac{\sum_{i=1}^{s}\frac{1}{1+f_i(x)}}{s} \tag{8}$$

(2) The influence of external pressure

**Definition 3**: external pressure, EP: Describe a degree of competition between individuals in a population, and measure it with population hamming distance and population quantity.

Individual living in a population, when the number of individuals is small and the population genetic diversity was not enough in a population, so the degree of competition between individual species is relatively small, the corresponding fitness value is lower. When the number of individuals is huge and the individual diversity of population is large in a population, the competition ability between individuals is relatively larger; the corresponding population individual fitness value is higher. Assuming that the outside pressure of the $i$-th individual in the $k$-th population is $P_{ki}$, its calculation formula is:

$$P_{ki}=\frac{d_i}{D_k}\cdot M_k \tag{9}$$

Among the formula, $d_i$ is the hamming distance of the $i$-th individual in the $k$-th population, which can be obtained by the formula (2), $D_k$ is the hamming distance of the $k$-th population, which can be obtained by the formula (3), $M_k$ is the number of individuals in the $k$-th population. This formula accords with the growth law of community in the nature.

Based on the previous analysis, we can get the individual fitness function of current population which is designed for:

$$fit(t)=\alpha\cdot\frac{m}{n}+\beta\cdot\frac{\sum_{i=1}^{s}\frac{1}{1+f_i(x)}}{s}+\gamma\cdot\frac{d_iM_k}{D_k} \tag{10}$$

Among the formula, $\alpha$ is the weighting factor of the influence of coverage to the fitness function, $\beta$ is the weighting factor of the influence of the branch predicate to the fitness function, $\gamma$ is the weighting factor of the influence of the external pressure on the fitness function, and $\alpha+\beta+\gamma=1$. After early research and many experiments, we have $\alpha,\beta,\gamma$ value with its own methods [17, 18].

## 3.3. IPEA Implementation of the Algorithm

In this paper, Figure 3 Shows the Implementation of the Pseudo Code of IPEA:

```
Input:
  Instrumented version $P'$ of the program to be tested $P$; List of det-use paths to be covered;
```

```
          Number of program input variables; Domain and precision of input data;
          Population size ($M$); Maximum number of generations (Max_Gen) ;
          Probability of crossover; Probability of mutation;
      Output:
         Set of test cases for $P$; Set of def-use paths covered by each test case;
          The coverage of each test cases; Number of last generation; The searching time when the end;

    Begin
       Step 1:Initialization
            Create three Initial_Populations;      //a main population and two child populations
            Current_population ← Initial_Populations;
            Set of test cases for $P$ ←∅ ;
            Coverage_percent ← 0;
            No. of Generations ← 0;
       Step 2:Generate test cases
             nEffective ← 0;      // the number of effective test cases
             for the main populations do;    // the other two iterate parallel
                   for each individual of current population do;
      Begin
           Compute the current individual populations external pressure values according to the formula (9);
           execute  $P'$  with this data set as input;
           compute the branch function value influence on program execution according to the formula (8);
           if(some def-use paths are covered) then
                     nCases ← nCases+1;        // number of test cases plus 1
                     add effective test case to set of test cases for $P$;
                     update coverage_percent coverage vector;
                     nEffective ← nEffective+1;
                     compute the coverage_percent according to the formula(8);
                     compute the current population fitness variance according to the formula(5);
                     compute the diversity of population according to the formula (6);
           end if
        end for
           If(three termination conditions) do
              Begin
                 Sorting test cases according to the fitness value;
                 If(the best test case of the child population > the worst test case of the main population
        or the diversity value of population < the fixed threshold value )
                         Exchange the two test cases;
                 End If
         Else
                Genetic operations:
                Select operation using roulette wheel method;
                Create New_Population using crossover and mutation operators
                Current_Population ← New Population;
                Return step2;
        End If
        Step3:Produce output
                set of test cases for P, and set of def-use paths covered, and coverage_percent;
                Report on uncovered def-use paths, if any;
                  The completing searching times;
                  The No. of Generations;
    End.
```

**Figure 3. The Implementation of the Pseudo Code of IPEA**

In the IPEA, when to evaluate individual species to calculate the fitness value, three influenced factors are quantitative calculated according to the formula (7-9) respectively. When to replace individual species, there are two decision conditions.

1) The fitness value of the size of the comparison of individuals in the three populations after sorted;

2) The diversity value of the size of the comparison of the current population.

Population will evolve according to genetic algorithm strategy constantly, until the optimal solution appears. This paper uses three conditions to stop genetic algorithm:

1) Recording the number of test paths covered. When all the test paths are covered, and then find the optimal solution, the end of the algorithm;
2) Because of some test path node may be difficult to cover or can't cover, you need to set certain evolution algebra, when the number of iterations of the evolution is reached the preset value, the algorithm is end;
3) The average fitness value of population reaches the preset value, the end of the algorithm.

## 4. The Experimental Simulation and Algorithm Evaluation

This paper uses 10 C language program as test object, part of the data and information from the present [7].

This group program contains some widely used in the study of test case generation process, such as triangle classification, find the median value, calculate the value of $x^y$ and calculate the remainder and so on simple programs, other programs are comprehensive program of complex structure. Table 1 shows the details of these programs.

**Table 1. The Detail Information of 10 Programs**

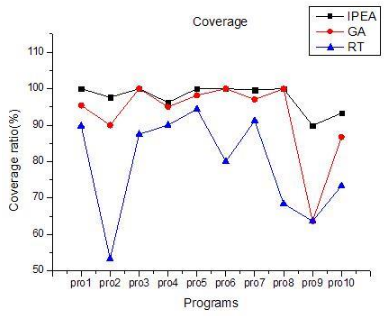
| Program code | Program rows | du-pairs | Program description |
|---|---|---|---|
| Pro1 | 61 | 88 | classify the triangle type |
| Pro2 | 37 | 30 | compute the average value |
| Pro3 | 35 | 24 | compute the value of $x^y$ |
| Pro4 | 41 | 40 | compute the remainder |
| Pro5 | 45 | 54 | synthetic of do-while,ifs |
| Pro6 | 37 | 50 | synthetic of while,for,ifs |
| Pro7 | 40 | 34 | synthetic of while,do-while,ifs |
| Pro8 | 21 | 19 | sorts,finds smallest,largest in arry |
| Pro9 | 20 | 11 | sort,finds even numbers in arry |
| Pro10 | 38 | 19 | find the solution of quadratic equation |

In accordance with the requirements of the above 10 programs, every programs are structural analyzed and instrumented function respectively, and according to IPEA algorithm, classic genetic algorithm (GA), random search algorithm (RT) three test strategy to generate test cases. Finally, using ALL-DU-PATHS data flow testing standard criteria as cover standard, will the three algorithms in the coverage, genetic algebra, the convergence of the execution time and the size of test cases compared. In the set of genetic parameter, the crossover rate and mutation rate are assigned 0.8 and 0.15 respectively [7]. In order to maximize the test efficiency, testers need to select the appropriate values $(\alpha,\beta,\gamma)$ according to the actual situation of the program code. In this experiment, according to previous research results, take $(\alpha,\beta,\gamma)$ to (0.5, 0.3, 0.2) can achieve an ideal effect [18].

(1) Coverage

The coverage comparison results of the 10 tested programs with three test strategies as shown in Figure 4:

From Figure 4, when the genetic algebra is fixed, IPEA test strategy is higher than the other two in coverage, such as pro1, IPEA may reach 100% coverage, but correspondingly GA coverage rate is only 95.42%, while RT is only 89.79%. Therefore, IPEA algorithm's searching ability is stronger and the coverage is more advantage than the other two algorithms in our experiment.



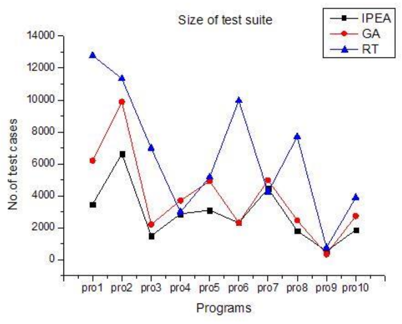


Figure 4. The Coverage

Figure 5. The Size of Test Suite

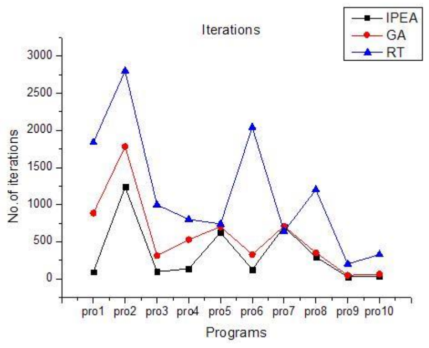
(2) The size of test cases comparison

The size of test cases comparison results of the 10 program with three test strategy as shown in Figure 5:

From Figure 5, we can see that the size of test cases obtained through the IPEA algorithm in the 10 tested programs is less than the other two algorithms mostly. This means that under the condition of achieving the same test coverage, IPEA will be more save test cost and shorten the testing time. Thus it can be seen that IPEA algorithm has more advantages on test case size than the other two kinds of test strategy, and reduced the tester’s test time in actual, improved the working efficiency.

(3) The search time of algorithm

The comparison final results of searching time cost achieving the algorithm convergence of 10 programs with three kinds of test strategy as shown in Figure 6:



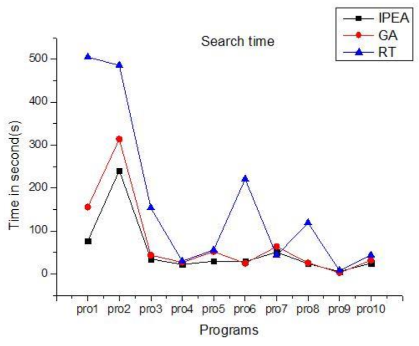


Figure 6. The Search Time

Figure 7. The Iteration

From Figure 6, we can see that in the 10 tested program, most IPEA algorithm has less searching time than the other two algorithms, and saves the software testing time, shorten the software development cycle, saving the time for the release of the software costs.

(4) Iterations of the genetic

The comparison results of iteration steps in 10 tested programs with three algorithms as shown in Figure 7:

From Figure 7, we can see that when the algorithm convergence, IPEA algorithm had fewer iterations compared with other two kinds of algorithm mostly in 10 programs. This means that the IPEA algorithm's searching ability is stronger, and is more close to the ideal test case set in the actual operation.

From the above the analysis of the results of four aspects we can see that the proposed improved evolutionary algorithm based on multiple populations IPEA, in our experiment simulation, compared with classic genetic algorithm (GA) and the random search algorithm (RT), IPEA has more advantages reflected in: have a higher coverage, can reduce the time to obtain the required test cases, can reduce the size of the test cases and reduce genetic algebra to obtain the required test cases. Therefore IPEA has the very high performance, also has a good practicability in practice.

## 5. Conclusions

This paper puts forward a kind of automatic test case generation technology based on improved evolutionary algorithm to meet the data flow testing standard. This technology is put forward based on the concept of multi-population, and our experiments show that the technology also has high practicability in practice. In the process of improving evolutionary algorithm, when considering the influence of the individual similarity in species and population diversity to automatically generate test cases, introducing the concept of multi-population; when considering the influence of the branch predicate path and the fitness function of the test, introducing the concept of branch function; Finally adding these factors to improve the fitness function, and give different weights factor to weight the distribution which influence and help the dynamic adaptive adjustment of test cases generation. Based on the above research, we finally design a comprehensive consideration of various aspects of the fitness evaluation function. Through the experimental simulation results of analysis, compared with other algorithms the improved evolutionary algorithm based on multi-population has the advantage of high whether in the coverage, or on the size of the test cases, or genetic algebra and search time and so on. In view of the above research results, we will do the deeper research of the algorithm, have the algorithm more practical in practice, at the same time considering the software defect detection [19], introduces the concept of multi-objective algorithm, make the search more targeted, reducing search time, and study the genetic operator for adjusting the adaptive technology in order to achieve better testing results.

## Acknowledgments

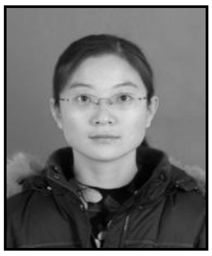
This research was supported in part by the National Natural Science Foundation of China (Grant No.61202050, No.61379036), Sion-D. K. cooperation project of the National Natural Science Foundation (Grant No.61361136002);the Natural Science Foundation of Zhejiang Province, China (No.Y13F020175), 521 ta-lent project of Zhejing Sci-Tech University; Qian jiang Talents Program of Zhejiang Province, China (Grant No.2013R10005,No.2013R10015).

## References

[1] Sofokleous A. A. and Andreou A. S. "Automatic, evolutionary test data generation for dynamic software testing. Journal of Systems and Software, vol. 81 no. 11, **(2008)**, pp. 1883-1898.
[2] Wan L., Zhang W. and Ma X. Y., "Automatic test cases generation technique based on path. Fault-tolerant computing professional committee of China computer society," Proceedings of the 10th national conference on fault-tolerant computing academic,**(2003)**, June; China.
[3] Xie X. Y., Xu B. W. and Shi L., "For the evolution of path coverage test cases generation technique (English)," Journal of software, vol. 12, **(2009)**, pp. 3117-3136.
[4] Alshraideh M., Mahafzah B. A. and AL-SHARAEH S. "A multiple-population genetic algorithm for branch coverage test data generation," Software Quality Journal, vol. 19 no. 3, **(2011)**, pp. 489-513.
[5] E. J. Weyuker, "The applicability of program schema results to programs, Internet. J. Comput.Inform.Sci. vol. 8, **(1979)**, pp. 387-403.
[6] Zhou M. and Sun S., "The principle and application of genetic algorithm (ga)," Beijing: national defence industry press, **(1999)**.
[7] Girgis M R. Automatic test data generation for data flow testing using genetic algorithm. Journal of Universal Computer Science, vol. 11 no. 6, **(2005)**, pp. 898–915.

[8] A. S. Ghiduk, M. J. Harrold and M. R. Girgis, "Using genetic algorithms to aid test-data generation for data-flow coverage," In 14th Asia-Pafic Software Engineering Conference(APSEC), December **(2007)**, pp. 41–48.
[9] Mathur A. P., "Foundations of Software Testing:Fundamental Algorithm and Technique," New Delhi: Dorling Kinder-sley (India) Pvt. Ltd, **(2008)**.
[10] D. W. Gong and X. Y., "Sun. Multi-population genetic algorithm with variational search areas," Control Theory and Applications, vol. 23, no. 2, **(2006)**, pp. 256-260.
[11] Angelova M, Atanassov K and Pencheva T., "Multi-population genetic algorithm quality assessment implementing intuitionistic fuzzy logic," Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on, **(2012)**, pp. 365-370.
[12] Alshraideh M., Mahafzah B. A. and AL-Sharaeh S., "A multiple-population genetic algorithm for branch coverage test data Generation," Software Quality Journal, vol. 19 no. 3, **(2011)**, pp. 489-513.
[13] Rauf A., Jaffar A. and Shahid A. A., "Fully automated gui testing and coverage analysis using genetic algorithms," International Journal of Innovative Computing, Information and Control, vol. 7 no. 6, **(2011)**, pp. 3281-3294.
[14] Lin J. and Yeh P., "Automatic test data generation for path testing using Gas," Information Sciences, vol. 131 no. 1-4, **(2001)**, pp. 47 – 64.
[15] Huang J. C., "Program Instrumentation and Software Testing," IEEE Computer, vol. 11, **(1978)**, pp. 25-32.
[16] Alshraideh M., Mahafzah B. A. and AL-Sharaeh S., "A multiple-population genetic algorithm for branch coverage test data generation," Software Quality Journal, vol. 19 no. 3, **(2011)**, pp. 489-513.
[17] Bao X. A., Yao L., Zhang N. and Song J. Y., "Adaptive Software Testing Based on Controlled Markov Chain.," Journal of Computer Research and Development, vol. 49 no. 6, **(2012)**, pp. 1332-1338.
[18] Bao X. A., Yao L., Zhang N., Dong M. and Gui L., "Based on multi-objective optimization adjustment strategy study online test case priority," Journal of Software. (Has hired in 2014).
[19] Bao X. A., Xie X. M., Zhang N. and Cao J. W., "Optimization testing strategy based on correlation of defect Markov modeling software," Journal of Software. (Has hired in 2014).

## Authors



**Na Zhang**, she was born in 1977, M.S. She mainly researched software engineering and software testing technology.

**Biao Wu**, he was born in 1989, M.S He mainly researches software engineering and software testing technology.

**Xiaoan Bao**, he was born in 1973, M.S. He is a professor and mainly researches adaptive software, software testing and intelligent information processing.