Research on automatic generation method of MC/DC test case based on improved drosophila optimization algorithm

Jiawei Song1,2*, Yunyun Ma2, Jinbo Wang2 and Yawen Yang2
1 University of Chinese Academy of Sciences, Beijing,100049, China
2 Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences, Beijing,100094, China
* songjiawei17@csu.ac.cn

Abstract. Based on the requirements of a certain embedded software evaluation task in China's space station, this paper proposes a MC/DC test case generation method based on improved drosophila optimization algorithm. Compared with other intelligent optimization algorithms, drosophila algorithm has the advantages of small computation, low complexity and few debugging parameters. MC/DC coverage is one of the logic coverage methods in white box test, it is widely used in defense and aerospace. In this paper, based on the MC/DC target path method, the iterative optimization ability of drosophila algorithm is used to find the test cases that meet the target path. At the same time, due to the instability of the drosophila algorithm and the tendency to fall into local optimum, the rate of change of the global optimal value is included in the weighting process in the iterative process to improve the drosophila optimization algorithm. Finally, we select a typical verification program to generate MC/DC test cases through simulation experiments. The effectiveness of the proposed method is verified, which provides a new method for coverage test in the space station software evaluation task.

1. Introduction
With the continuous expansion and increase of software application areas and scale, the quality of software becomes more and more important. Software testing is a key technical measure to ensure software quality. The focus of software testing lies in the writing of test cases. The automatic generation of test cases becomes the top priority of software testing[1].

In recent years, scholars have focused on transforming the generation of test cases into artificial intelligence algorithm optimization problems to complete the automatic generation of test cases, which can generate more efficient test cases for program coverage. At present, most scholars study the test case generation method based on path coverage[2,3]. The intelligent optimization algorithms used are genetic algorithm[4], ant colony algorithm[5], and simulated annealing algorithm[6]. However, these algorithms generally have large computational complexity, many debugging parameters and high complexity[7], and the path coverage criterion increases dramatically with the increase of program loop and judgment conditions. The path coverage method is more complicated in a large software project.

Based on the requirements of a certain embedded software test in China space station, this paper proposes a method for generating MC/DC test cases based on drosophila algorithm. Drosophila algorithm is a new intelligent algorithm proposed by Chinese Taiwan scholar Pan Wenchao in 2011. It has small calculation, low complexity and few debugging parameters. MC/DC is the correction conditional coverage. It is one of the white box test logic coverage methods. It is the “Aviation and Equipment System Software Certification Standard” jointly developed by aerospace/aerospace
manufacturers and users in Europe and America. Compared with other logic coverage, the MC/DC coverage strength is higher than the statement, decision and condition coverage, but the number of test cases is linearly increased much less than the index of multi-condition coverage. The increase in level is in line with the principle of streamlining and high efficiency, and is widely used in the fields of national defense and aerospace. In this paper, the drosophila algorithm is improved and modeled, the taste concentration function (fitness function) is designed. The tested program is preprocessed and instrumented, and the algorithm is applied to the MC/DC coverage test. Finally, the effectiveness of the method proposed in this paper is verified by simulation experiments.

2. Improved drosophila optimization algorithm

2.1. Standard drosophila optimization algorithm

The original idea of the drosophila optimization algorithm is derived from the flight behavior of the drosophila while searching for food. When drosophila is searching for food, it first uses the excellent olfactory ability to search for the smell of food, adjusts the direction of flight according to the smell, and reuses sensitive vision when it is closer to the food. According to this behavior, the basic idea of the drosophila algorithm is as follows:

1. Set the number of drosophila populations to NP, the maximum number of iterations is Maxgen, and arbitrarily give the two-dimensional starting position of the drosophila population as $X_{axis}, Y_{axis}$;
2. Give the random direction and interval of the individual movement to the drosophila, and the $Random Value$ is the interval of the movement;

\[
\begin{align*}
X_i &= X_{axis} + Random \text{ Value} \\
Y_i &= Y_{axis} + Random \text{ Value}
\end{align*}
\]

1. Calculate the interval value $Dist_i$ of the drosophila individual from the origin, the taste density determination value $S$, and the taste density value $Smell_i$;

\[
\begin{align*}
Dist_i &= (X_i^2 + Y_i^2)^{1/2} \\
S_i &= Dist_i^{-1} \\
Smell_i &= \text{Function}(S_i)
\end{align*}
\]

1. Obtain the individual with the highest Smell value in the population and record as the current optimal drosophila individual;

\[
[bestSmell \ bestIndex] = \max(Smell)
\]

1. Determine whether the current best individual concentration is higher than the last iteration's best taste concentration. If so, execute (6); if not, execute (7);

1. Save the taste concentration value of the current optimal individual and its two-dimensional coordinates $X, Y$, and other drosophila individuals manipulate the direction to the optimal position.

\[
\begin{align*}
SmellBest &= bestSmell \\
X_{axis} &= X(bestIndex) \\
Y_{axis} &= Y(bestIndex)
\end{align*}
\]

1. Determine whether the termination condition is satisfied, if so, end the iteration; conversely, execute (2).

2.2. Improve the weight change strategy

The drosophila optimization algorithm does not control the flight direction and distance of the drosophila in the iterative optimization process, ignoring the difference of the individual samples, which leads to the instability of the algorithm and is easy to fall into local optimum. In this paper, we propose the improvement idea refer to \cite{8}. In the iterative process, the rate of change of the global optimal value is included in the weight consideration. The main strategies are as follows:

\[
r = \left| \frac{\text{fitness}(t) - \text{fitness}(n - t)}{\text{fitness}(n - t)} \right|
\]
\[ w = \begin{cases} 
0.2 + r, & r \leq 0.3 \\
0.5 + r, & 0.3 < r \leq 0.6 \\
0.6 + r, & r > 0.6 
\end{cases} \]

\[ X_i = X_{\text{axis}} + w \times \text{Random Value} \]

\[ Y_i = Y_{\text{axis}} + w \times \text{Random Value} \]  

(5)

Where \( r \) is the rate of change of the optimal fitness value in \( n \) generations, \( \text{fitness}(t) \) is the optimal fitness value of the \( t \)-th generation, and \( \text{fitness}(n - t) \) is the optimal fitness value of the \((n-t)\)-th generation. When the optimal fitness value of the population changes greatly, it indicates that the drosophila population is expanding into the new space, and increasing the weight is beneficial to its global search. Conversely, when the optimal fitness value changes little, the algorithm is in the local search stage, reducing the weight can speed up the acquisition of the optimal solution.

3. MC/DC test case generation method based on improved Drosophila algorithm

3.1. MC/DC coverage criteria

The MC/DC coverage requires that each entry and exit in the program be executed at least once, and all possible outcomes of each decision statement in the program appear at least once, and all possible outcomes of each condition in the decision occur at least once, most importantly Each condition must independently and correctly affect the outcome of the decision.

3.2. MC/DC test case generation

In this paper, the MC/DC target path based method is used to generate test cases. In order to obtain the set of MC/DC target paths in the program, we first need to understand the structure of the program, this paper uses third-party software (Code Visual to Flowchart), which can automatically generate the control flow chart of the program. In the code flow chart, the MC/DC target path set can be constructed and the specified path can be selected.

When it is determined that there are many conditions in the branch node, it will excessively consume much time and difficult to realize if the truth table of the node is listed to obtain the MC/DC test case set one by one. In order to improve the efficiency of automatic generation of MC/DC test cases, this paper uses the recursive block matrix algorithm proposed in [9] to generate test case sets for each decision node in the program.

The test model in this paper is for the MC/DC path oriented to the structure test, so all the branch points in the path need to be covered, and the branch point to complete the overlay is not a simple left and right branch, but the left and right branches of all the conditions in the branch.Only in this way, coverage can be completed. So this paper chooses the modified branch distance function plus layer proximity function as the fitness function of the program refer to [10].

In the MC/DC coverage test, we can use each individual as a test data, Let \( T_0 = (f_1, f_2, \cdots, f_m) \) denote the target path identification sequence in the MC/DC coverage test, so the problem is transformed into finding the optimal vector \((x_1, x_2, \cdots, x_m)\) to cover the target path identification sequence. The specific implementation process is as follows:

(1) Code

The improved drosophila algorithm uses integer coding. Each integer represents the component position of the corresponding drosophila individual in this dimension, and its range is \([\text{min}, \text{max}]\), where \(\text{min}\) is the minimum value in the dimension and \(\text{max}\) is the the maximum value in the dimension.

(2) Drosophila individual

Drosophila individual \( G_i = (x_1, x_2, \cdots, x_m) \), where \( x_j \) is an integer representing the position of the i-th drosophila individual in the j-th dimension, and \( N \) drosophila individuals constitute a drosophila population, record as \( G = \{G_1, G_2, \cdots, G_N\} \).

(3) Taste concentration function
The taste concentration function represents the utility of the drosophila individual, expressed in terms of fitness, it can distinguish the advantages and disadvantages of different drosophila individuals, and then select the best individual. The taste concentration function of this paper is divided into two parts: layer proximity and branch distance, which is the fitness function mentioned above. The specific form will be presented in the simulation experiment.

(4) Algorithm step

Assuming that the MC/DC target path to be covered is \( T_0 \), the specific process for the improved drosophila algorithm to generate test cases is as follows:

Input: The number of groups is NP, the maximum number of iterations is Maxgen, and the m-dimensional starting random position of the drosophila group is \( X_{1\_axis}, X_{2\_axis}, \ldots, X_{m\_axis} \).

Output: target test data \( x_1, x_2, \ldots, x_m \).

1) Carry out the first iteration, random direction and interval is given to the drosophila individuals \( G_i \). \( x_i = X_{i\_axis} + \text{Random Value}, (i = 1, 2, \ldots, m) \);

2) Run the program after program instrumentation;

3) Calculate the concentration value of each drosophila individual according to fitness function and find the highest concentration among them in the first iteration;

4) All individuals fly toward the optimal value of the first iteration and give the random direction and distance of the flight according to formula (5). The subsequent iterative process is performed according to formula (5);

5) Determine whether the concentration of the current optimal individual is higher than that of the previous iteration, and if so, preserve the optimal individual \( GBest(x_1, x_2, \ldots, x_m) \), if not, move on to the next step;

6) Determine whether to coverage the target path \( T_0 \) or reach the maximum number of iterations, and if so, end the evolution of the population, save and output the corresponding test results, if not, jump to 4);

In summary, the flow chart of the MC/DC test case generated in this paper is shown in figure 1.
4. Simulation and result analysis

Due to the confidentiality of the space station task, this paper selects a typical triangle decision program to verify the algorithm. Experimental conditions: Windows 10 operating system, Visio Studio 2017 simulation environment, the experiment is implemented in C language. The triangle determination procedure after instrumentation as follows:

```c
if (a + b > c)
{
    p[0] = 1;
    if (a == b || b == c)
    {
        p[1] = 1;
        if (a == c)
        {
            p[2] = 1;
            str = "Equilateral triangle";
        } else
            p[3] = 1;
        str = "Isosceles triangle";
    } else
        p[4] = 1;
    str = "Ordinary triangle";
} else
    p[5] = 1;
str = "Non-triangle";
```

The block matrix algorithm is used to generate a set of test cases for the determine branch nodes by the triangle decision logic. The node 1 is determined such that \( a+b>c \) is a Boolean variable \( A \), and the decision node 2: \( a==b \) is a Boolean variable \( B \), and \( b==c \) is Boolean variable \( C \), decision node 3: \( a==c \) is a Boolean variable \( D \), then the MC/DC test case set of decision point 1 is \( (1)(0) \), and the MC/DC test case set of decision point 2 is \( (00)(01)(10) \), the MC/DC test case set of decision point 3 is \( (1)(0) \), and the path of the program as shown in table 1:

| Path | Passed MC/DC branch | Program result/Test case set |
|------|---------------------|------------------------------|
| P1   | (1)(10)(0)          | Isosceles triangle (1 10 0)  |
| P2   | (1)(01)(0)          | Isosceles triangle (1 01 0)  |
| P3   | (1)(00)(0)          | Ordinary triangle (1 00 0)   |
| P4   | (0)                 | Non-triangle (0)             |

Then we can choose a feasible path, such as selecting the path P1 \( (1 10 0) \). The branch function of the decision point 1 is \( f_1 = a + b > c?0:c-(a+b) \), and the branch function of the decision point 2 is \( f_2 = f_B + f_C, f_B = abs(a-b), f_C = b == c?k:0 \), the branch function of the decision point 3 is \( f_3 = a == c?k:0 \), where \( k \) is a positive number with a small value. The branch distance function of this target path is constructed as:

\[
branch\_distance = \frac{(100)}{f_1+1} + \frac{100}{f_2+1} + \frac{100}{f_3+1} / 3
\]

The P1 path identification sequence traversed by drosophila individuals is \([1,1,0,1,0,0]\), and the actual traversing path identification sequence is assumed to be \([1,0,0,0,1,0]\), then the number of different elements between the two identification sequences is 3, \( lay\_proximity \= \text{the number of different elements/the total number of elements}=3/6 \).

\[\text{Fitness} = branch\_distance + layer\_proximity\]
The initialization parameters of the improved drosophila algorithm are set to be: the population of the drosophila is 500, the maximum number of iterations is 2000, and the dimension is 3, the value of k is 0.1. The experimental results are shown in the figure 2:

![Figure 2. Test case generation result.](image)

5. Conclusions

From the experimental results, it can be seen that through the analysis of MC/DC feasible paths, test cases of all feasible paths are generated by drosophila algorithm. However, from the choice of feasible paths, we can also know that even if all feasible paths can be completed, the complete coverage of MC/DC will not necessarily be achieved, such as the equilateral triangle case of the program, because the minimum test case set at the decision branch point 2 does not contain the type of (11), so it makes impossible to reach (1) in the decision branch point 3. Because the variables are related, that is, the values of variables $B$ and $C$ have an impact on $D$. To achieve complete MC/DC coverage of the program, it is necessary to determine that the variables between branch nodes have no relationship to achieve. If we want to achieve full coverage of the program, we can add missing test cases manually. Because the number of missing cases is small, the method proposed in this paper has feasibility and practicability in future space station software evaluation.

References

[1] Li L S, Guo Z M. (2018) Optimization Techniques Research on Testing Data Through Path Coverage with Chaotic Fruit Fly Algorithm. Journal of Chinese Computer Systems, 39: 362-366.

[2] Fan H L, Zeng G B. (2016) Automatic generation of test data for paths coverage using evolutionary algorithm with self-adaptive immigrate. Application Research of Computers, 33, 3743-3747.

[3] Xia C Y, Zhang Y, Song L. (2016) Evolutionary Generation of Test Data for Paths Coverage Based on Node Probability. Journal of Software, 27, 802-813.

[4] Yu B, Jiang S J, Zhang Y M. (2012) Multiple Paths Test Case Generation Based on Complex System Genetic Algorithm. Computer Science, 39, 139-141.

[5] Lao T, Ma Y. (2018) Generation of software interface use case in ant colony algorithm. Computer Engineering and Design, 39, 79-85.

[6] Zheng J Q, He Y, Wang C W. (2015) A Method of Test Case Optimization Based on Improved Simulated Annealing Genetic Algorithm. Modern Computer, 32, 3-6.

[7] WU X W, Li Q. (2013) Research of Optimizing Performance of Fruit Fly Optimization Algorithm and Five Kinds of Intelligent Algorithm. Fire Control & Command Control, 38, 17-20.

[8] Ding G S, Zou H. (2016) New improved fruit fly optimization algorithm. Computer Engineering and Applications, 52, 168-174.

[9] Ge H Q. (2011) Generating Algorithm of Recursive Blocks Matrix for Minimum Test Case Set on MC/DC. Computer Systems & Applications, 20, 195-198.

[10] Nigel T, John C, Keith M, John M. (1998) An Automated Framework for Structural Test-Data Generation. In: IEEE International Conference on Automated Software. Washington, DC. 285-288.