Analysis of an Intelligent Optimization Algorithm for Automatic Generation of Computer Software Test Data

Liping Li and Xiaoyan Zhang(✉)

Yunnan College of Business Management, Kunming 650300, Yunnan, China
z316819595@163.com

Abstract. Software testing can guarantee the quality of software products, but it also takes up nearly half of the cost and resources of the entire software development cycle. The traditional test data acquisition requires manual design, but as the scale and complexity of software increases, manual design of test data can no longer meet the requirements of testing, therefore, automatic test data generation has become a hot spot and focus of many scholars' research. In this paper, we will study and analyse the automatic generation of computer software test data based on intelligent optimisation algorithms.

Keywords: Software testing · Automatic data generation · Intelligent optimisation algorithm

1 Introduction

Software testing is a method and means of evaluating software products by designing test data and using it to find defects or errors in the software in order to guarantee the quality of the product. The importance of software testing is evident in the fact that it runs through almost the entire software development process, and the design of test data is the most central and important part of testing, which determines the quality and efficiency of testing. The traditional way of designing test cases and test data is to design them manually, relying mainly on the experience of testers, but However, as the size of software increases, the traditional manual data design mode of generating test data is inefficient, laborious, prone to omissions, long testing cycles and high testing costs, so more and more researchers are beginning to study the automatic generation of test data.

In this paper, the research of automatic test data generation is carried out by optimising some shortcomings of the algorithm in the application of the problem, improving the performance of the algorithm so as to achieve the goal of fast and effective test data generation, reducing the time cost and resources consumed in software testing due to the design of test data, and improving the efficiency of test data generation while reducing unnecessary costs in testing. This paper is not only of theoretical significance, but also of practical application.

Achieve usability and ease of use; The core production management software has leading advantages, and the material base, knowledge base, model base, process base, algorithm database and other basic elements are constantly improved; Cultivate new industrial software platform; Strengthen the enabling of industrial data and improve the intelligent level of industrial software. As shown in Fig. 1, the intelligent level of industrial software.

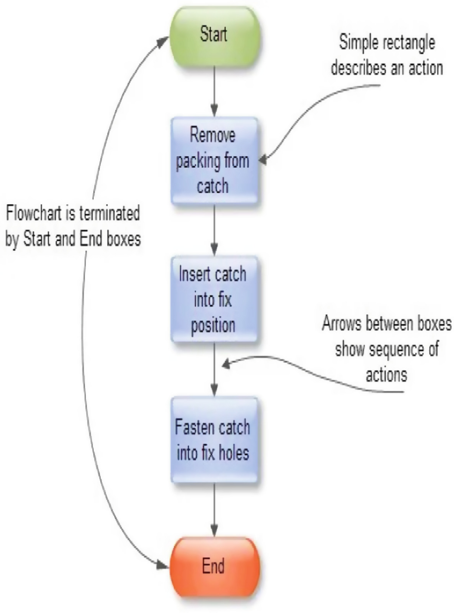
Start
Simple rectangle describes an action
Remove packing from catch
Flowchart is terminated by Start and End boxes
Insert catch into fix position
Arrows between boxes show sequence of actions
Fasten catch into fix holes
End

Fig. 1. Intelligent level of industrial software

Focus on the development of core industry software. For the process industry, support the research and development of planning and scheduling, production scheduling and real-time optimization software, overcome the deep integration of scheduling and planning and the development of rolling optimization technology, and promote the application of petrochemical, chemical and non-ferrous metals. Applications in light industry and other fields accelerate the breakthrough of core technologies such as industrial software and hardware, intelligent algorithm and industrial mechanism model, focusing on the whole process of optimization design, production, operation and maintenance, quality improvement, intelligent manufacturing, intelligent detection, real-time scheduling, optimization decision-making, predictive protection and other industry scenarios.

Focus on industrial Internet, intelligent manufacturing and production services. This paper consists of the following parts. The first part introduces the relevant background and significance of this paper, the second part is the related work of this paper, and the third part is automatic generation of single-path test data. The fourth part is automatic generation of multi-path software test data. The fifth part is conclusion.

2 Related Work

Koparan aimed to examine the effect of dynamic data analysis software-supported learning environments on secondary school students' achievement and attitude [1]. Su et al. take the touch screen characteristic test as an example,through the multi-dimensional automatic motion platform, the test pen is driven to move on the touch screen according to the set trajectory [2]. Transformer routine tests have been analyzed by using the generated firefly algorithm [3]. The subject of Ref [4] was to improve students understanding of introductory oceanography with the aid of a computer program. Rahmawati et al. aim to find out if computer self-efficacy, learning motivation, and accounting knowledge affect the computer anxiety of accounting students in using accounting software [5]. To cooperate with the research on the fragmentation pre-conditioning technology of large hard rock in natural caving mining, Jingjie et al. perform a large flow hydraulic fracturing test in Tongkuangyu Copper Mine to investigate the relationship between the occurrence and expansion of fractures in ore bodies and the pressure and flow of water injection [6]. The research results show that the system constructed has certain practical effects [7]. Huang et al. introduce the secondary development process of simcenter Test [8]. Lab software. CBCT images of twenty patients with UCCLP were included Ref [9]. Other influential work includes Ref [10]. Relying on universities and scientific research institutes, build industrial software adaptation, testing, verification and pilot test platforms for production, University and research. Vigorously promote the promotion of new technologies for industrial software integration; Promote the autonomy of industrial software of industrial Internet base, support micro service architecture such as aggregation tools, algorithms and models, promote industrial software components and services, and improve the comprehensive integration, test and verification, quality control, life cycle management and service capabilities of industrial software.

2.1 Overview of Intelligent Optimisation Algorithms

The so-called intelligent optimisation algorithm is a kind of intelligent search calculation method developed according to some principles of the real phenomena in nature, and the intelligent optimisation algorithm can be regarded as a kind of reference and simulation of the laws of nature. At present, in addition to traditional algorithms such as genetic algorithms and particle swarm algorithms, there are also algorithms such as firefly algorithms and firework explosion algorithms. The genetic algorithm is an algorithm invented by simulating biological genetics and biological evolution in nature, which is characterised by its adaptive structure and global optimisation effect. The particle swarm algorithm is based on the interaction between particles and is able to find optimal regions in a complex search space. In the process of computer software testing, these two more basic intelligent optimisation algorithms have been used in large numbers.

$$\begin{cases} E(t)\dot{x}_k(t) = f(t, x_k(t)) + B(t)u_k(t) \\ y_k(t) = C(t)x_k(t) \end{cases} \tag{1}$$

$$u_{k+1}(t) = u_k(t) + \Gamma_{l1}\dot{e}_k(t) + \Gamma_{l2}\dot{e}_{k+1}(t) + \Gamma_{p1}\Delta\dot{e}_k(t) + \Gamma_{p2}\Delta\dot{e}_{k+1}(t) \tag{2}$$

2.2 Applications of Genetic Algorithms

Genetic algorithms have a global probability search function. In using this algorithm, the problem is transformed by replacing the problem of generating software test data with a functional optimisation problem, and then designing the fitness function, while the population of software test data is coded to facilitate the application of genetic operations. The populations of software test data are evolved over many generations to obtain the corresponding test data results. In the automatic generation of software test data, the first step in the application of genetic algorithms is to encode the data entered through the software program in order to facilitate the formation of different individuals.

$$\dot{x}_{k+1}(t) = f(t, x_d(t)) - f(t, x_{k+1}(t)) \tag{3}$$

$$\begin{aligned}\Delta x_{k+1}(t) &= \int_0^t Q^{-1}(f(t, x_d(\tau)) - Q^{-1} f(t, x_{k+1}(\tau)))d\tau \\ &+ \int_0^t Q^{-1} B \Delta u_k(\tau)d\tau - \int_0^t Q^{-1} Z \dot{x}_d(\tau)d\tau + \int_0^t Q^{-1} F \dot{x}_{k-1}(\tau)d\tau\end{aligned} \tag{4}$$

The second step is to generate the initial population. The initial population is composed of N individuals, each of which is made up of N initial strings of structural data, the production of which is random. The initial population is an important starting point for iterative updating of the genetic algorithm.

The third step is to select the population. The selection operation is a simulation by the genetic algorithm of the survival of the fittest principle. Selection begins by selecting a certain number of individuals in the initial population that meet the criteria of being well adapted. These selected individuals will become the new parents and from this a new generation of individuals will be generated.

The fourth step is to carry out crossover. The main purpose of the crossover operation is to facilitate the exchange of data and information. Without crossover, the new generation of individuals would not be available to the parents selected by the algorithm. Almost every individual of the new generation has some of the characteristics inherited from its parents.

The fifth step is the mutation operation. When the mutation operation is performed, the elite individuals are kept out of the mutation operation. The first step is to select individuals at random in the population and to change them in a random way.

The sixth step is to calculate the value of the fitness function. If the result of the calculation satisfies the termination condition of the algorithm, the operation of the algorithm is terminated, and if the result of the calculation does not meet the requirements of the termination condition, the genetic algorithm is repeated until the result of the data satisfying the termination condition is found.

2.3 Firefly Algorithm

The proposed firefly algorithm is based on the luminous properties of fireflies, and summarises the relevant laws of their luminous activity in order to find the optimal solution.

The first step is to determine the population size of the fireflies, and then to initialise the population to determine the initial position of the individuals.

The second step is to calculate the absolute brightness of individual fireflies by using the objective function based on their position. As shown in Fig. 2 below, initialize the population to determine the initial position of the individual.

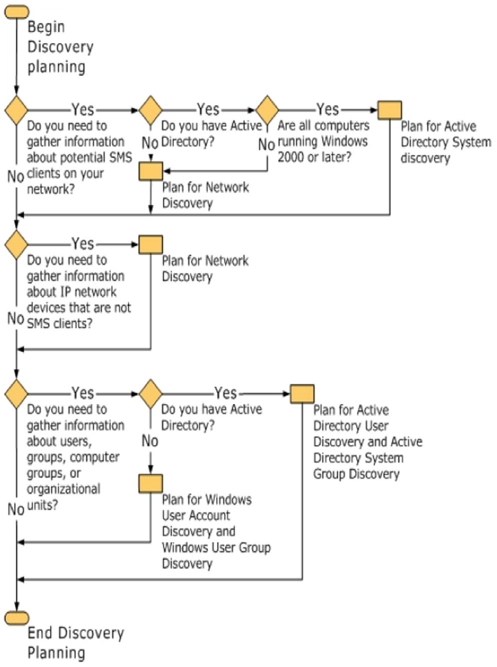
Begin Discovery planning
Yes
Do you need to gather information about potential SMS clients on your network?
No
Yes
Do you have Active Directory?
No
Yes
Are all computers running Windows 2000 or later?
No
Plan for Active Directory System discovery
Plan for Network Discovery
Yes
Do you need to gather information about IP network devices that are not SMS clients?
No
Plan for Network Discovery
Yes
Do you need to gather information about users, groups, computer groups, or organizational units?
No
Yes
Do you have Active Directory?
No
Plan for Active Directory User Discovery and Active Directory System Group Discovery
Plan for Windows User Account Discovery and Windows User Group Discovery
End Discovery Planning

Fig. 2. Initialize the population to determine the initial position of the individual

The third step is to calculate the relative brightness and attraction of each of the two individuals based on the formula for the relative brightness of the individuals and the formula for the attraction between the individuals.

In the fourth step, according to the individual position change formula, the update of individual positions in the population can be calculated. The brightest individual firefly in the population, however, moves irregularly and its position change can be calculated using a separate formula.

In the fifth step, the luminous intensity of each new individual firefly is recalculated for each individual firefly in the population at the end of the update.

Step 6, determine if the algorithm meets the conditions for the end of the algorithm, if so, end the algorithm and produce the final result, if not, repeat step 3 until the algorithm is finished.

3 Automatic Generation of Single-Path Test Data

There are many classical optimization algorithms, such as genetic, ant colony, particle swarm, simulated annealing, etc., which have been used in single-path applications, so there is less and less room for development and it is difficult to further improve the efficiency of data generation by improving classical algorithms such as genetic and particle swarm algorithms. In solving the single-path testing problem, this paper chooses to apply the Firefly algorithm (FA) to it and optimise it in order to find new ideas and solutions for the study of single-path testing.

3.1 FA Algorithm Mathematical Model

For the establishment of the FA algorithm mathematical model, the first thing to understand is the concept of absolute luminance and relative luminance, the absolute luminance of fireflies, is for a firefly individual *i*, its initial luminance value is called absolute luminance, can be recorded as I_i. And for the definition of relative luminance, and the absolute luminance I_i is different, it refers to for two individuals *i* and *j*, firefly *i* in firefly *j* position of the luminous intensity, can be be expressed as Iij.

At position Xi(xi1,xi2,…,xid), the expression for the absolute luminance

$$I_i = f(X_i) \tag{5}$$

The brightness between fireflies is not fixed. It will be gradually weakened with the increase of the interval between them and the absorption of some substances in the air. Therefore, the relative brightness formula of firefly individual *i* to firefly individual *j* can be expressed as

$$I_{ij}(r_{ij}) = I_i e^{-\gamma_{ij}^2} \tag{6}$$

3.2 Basic Flow of FA Algorithm

Step l: Assume that the whole population of fireflies is *N*, and initialize the positions of individuals in the whole population;

Step 2: According to the position of firefly X_i, we can know the value of the objective function $f(X_i)$, so as to calculate the absolute brightness $I_i(X_i)$ of firefly *i* according to the formula (1);

Step 3: Calculate the relative luminance I_i; and attractiveness of fireflies *i* and *j*, respectively;

Step 4: Update the positions of the individuals in the population, and for the brightest fireflies, move them randomly in an irregular manner;

Step 5: After the iterative update of the individuals in the population, the luminous intensity of each individual firefly is recalculated;

Step 6: Determine whether the algorithm meets the conditions of the end, if not, then return to Step3, if so, the algorithm will end and the results will be output.

FA algorithm basic flow chart is shown in Fig. 3.

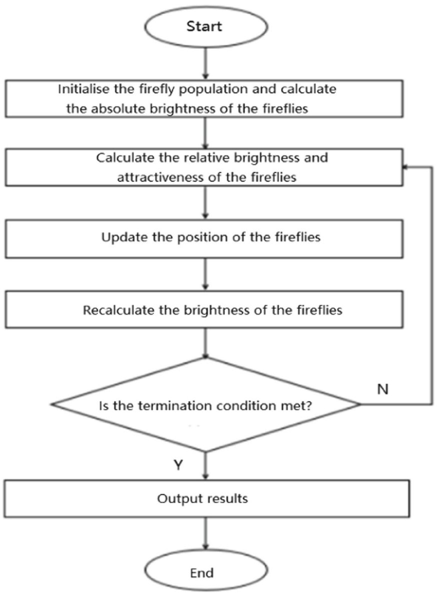
Start
Initialise the firefly population and calculate the absolute brightness of the fireflies
Calculate the relative brightness and attractiveness of the fireflies
Update the position of the fireflies
Recalculate the brightness of the fireflies
Is the termination condition met?
N
Y
Output results
End

Fig. 3. Basic flow chart of the FA algorithm

3.3 Construction of the Fitness Function

The bridge between algorithms and real application problems is the fitness function, which can be used to assess the quality of the data generated. A well-designed fitness function can better guide the algorithm towards the optimal solution region, covering the target in fewer iterations and less time path and find the most optimal solution. Therefore, it is important to construct a suitable fitness function, which is related to the efficiency of data generation.

In this chapter, the fitness function is constructed using the branching function superposition method devised by Korel, which converts branching predicates into branching functions, and then uses these functions, which are formed by superposition of branching functions, as the final objective function to be optimised.

This is the concept of a branch predicate, which indicates when a branch can be overwritten. A branch predicate can be expressed in the form of "E1 op E2", where E_1 and E_2 are mathematical expressions and op is a relational operator containing $<$, $\leq$, $=$, $>$, $\geq$, $\neq$. When the predicate does not contain logical operations, then a branching predicate in the form of "E1 op E2" can be converted to "f rel0", where f is the mentioned "branching function" and f is used to quantitatively evaluate the degree to which the input value satisfies the target function, when the branching predicate is true, the function value is negative. When the branch predicate is true, the function value is negative, when $f = 0$; conversely, when the branch predicate is false, the function value is positive, $f > 0$.

The relationship between branch predicates and branch functions is shown in Table 1.

Table 1. Relationship between branch predicates and branch functions

Branch predicates	Branching functions
$E1 > E2$	E2-E1
$E1 \geq E2$	E2-E1
$E1 < E2$	E1-E2
$E1 \leq E2$	E1-E2
$E1 = E2$	|E1-E2|
$E1 \neq E2$	|E1-E2|

4 Automatic Generation of Multi-path Software Test Data

The automatic generation of computer software test data for multiple paths involves two techniques. The first is path similarity. In computing, path similarity refers to the degree to which the path of software test data matches the target path. In order to determine path similarity more accurately, a more appropriate measure of path similarity is needed. At present, there are three main factors that affect path similarity: the number of identical nodes, the number of consecutive identical nodes, and the weight of different nodes. These three factors must be fully considered in designing the path similarity measure. The second is the design of the multi-path fitness function. In multi-path testing, the design of the fitness function is usually based on the idea of mean value. The first step is to calculate the value of the similarity of all paths according to a professional calculation method, and then take the average value. This mean value can be used as the fitness value of the data for the target set of paths. By comparing the matching of different paths with the mean value, it is more intuitive to see the strengths and weaknesses between the test data.

The path selector selects the least number of paths in the program control flow diagram for use by the test data generator. The path selection must meet certain coverage principles. Here are some common principles:

(1) Statement coverage: any statement can be covered by the selected path.
(2) Branch coverage: all conditional branches in the procedure must be covered. For example, in an IF statement, the condition predicates that are true or false must be covered by the selected path.
(3) Conditional coverage: when each clause in the condition of a branch statement is true or false (single condition coverage) and all combinations of the true values of each clause must be covered by the selected path (multi condition coverage).
(4) Path coverage: traverse all paths in the program control flow graph.

Among the above four principles, multi condition coverage and path coverage are difficult to achieve, because with the growth of program scale, the combination of clauses in conditions and the number of paths in program control flow graph will generally increase exponentially. Therefore, statement coverage and branch coverage are widely used as the basic measures of software testing.

Static test data generation is not based on the input data of the program, but adopts the method of program symbol execution and expression digestion and transformation. Dynamic test data generation is a method of executing the program by using the actual input data of the program. In the early research of automatic generation of program test data, the symbol based method is basically used. Because the problem of program test data generation is NP hard, the symbol based method occupies a lot of computer resources and has certain restrictions on the program. The advantage is that there is no need to check the truth value of the branch predicate.

The method based on the actual operation of the program is to take the value of the input variable to actually execute the program, and determine whether the value of the selected input variable can traverse the path selected by the path selector by observing the data flow in the program. Using different search algorithms, we can find the value of the input variable traversing a path, but it often takes a lot of time. It combines the symbol based method and the actual operation method based on program to generate test data, which saves the workload.

5 Conclusion

Software testing is an important part of the computer software development industry and a major need for the industry. The use of intelligent optimization algorithms to automatically generate computer software test data can be studied to obtain important results to improve the efficiency of automatic software testing, which is of great value to ensure the quality of software testing, improve the efficiency of software development, and help the development of China's software development industry.

References

1. Koparan, T.: Examination of the dynamic software-supported learning environment in data analysis. Int. J. Math. Educ. Sci. Technol. **83**, 80–98 (2018)
2. Su, M., Zhu, N., Huang, L., Xu, H.: Research on multi-degree-of-freedom and high-precision touch screen characteristic test instrument. J. Test Syst. **16**(8), 1204 (2019)
3. Zile, M.: Routine test analysis in power transformers by using firefly algorithm and computer program. IEEE Access **26**, 401–417 (2019)
4. Firdaus, M.L., Parlindungan, D., Elvia, R., Swistoro, E., Sundaryono, A., Rahmidar, L.: Teaching oceanography using ocean data view software. In: Proceedings of the International Conference on Educational Sciences and Teacher Profession (ICETEP 2018), pp. 3177–3184 (2019)
5. Rahmawati, A., Abidin, F.I.N.: The influence of computer self-efficacy, learning motivation, and knowledge of accounting on accounting students computer anxiety in using accounting software. Acad. Open **93**, 395–422 (2021)
6. Jian, J., Peng, H., Ma, X., Sun, Y.: Software design of a data acquisition system for the hydraulic fracturing experiment of rock at Tongkuangyu copper mine. J. Earth Environ. Sci. **282**, 232–247 (2021)
7. Narengerile, L., Di, L.: Framework and performance analysis of college English testing system based on data mining technology. J. Intell. Fuzzy Syst. **26**, 401–417 (2021)

8. Huang, J., Chen, B., Tan, M., Liu, M., Jia, C.: Exploration and research of noise automatic processing algorithm based on multi-scale convolution neural network. J. Neurnalnet Appl. **209**, 16–36 (2021)
9. Phienwej, K., Chaiworawitkul, M., Jotikasthira, D., Khwanngern, K., Sriwilas, P.: Comparison of preoperative measurement methods of alveolar cleft volume using cone beam computed tomography between computer simulation and water displacement methods. Cleft Palate-Craniofac. J. **34**, 2677–2684 (2021)
10. Zhang, H., Lin, F., Zhang, X., Wen, X.: Design of torque motor characteristic test system. J. Exp. Manag. **1786**, 108322–108326 (2021)