Configuration Strategies for Evolutionary Testing[1]

Xiaoyuan Xie[1,2] Baowen Xu[1,2,3,4] Changhai Nie[1,2] Liang Shi[1,2] Lei Xu[1,2]
[1] *Dept of Computer Science and Engineering, Southeast University, 210096 Nanjing, China*
[2] *Jiangsu Institute of Software Quality, 210096 Nanjing, China*
[3] *Computer School, National University of Defense Technology, 410073 Changsha, China*
[4] *Key Laboratory of Software Engineering, Wuhan University, 430072 Wuhan, China*

Abstract

Evolutionary Testing (ET) is a kind of efficient method of automatically test case generation. ET uses a kind of meta-heuristic search technique, the Genetic Algorithm, to convert the task of test case generation into an optimal problem. Nowadays, ET has been widely researched in many areas, especially in the GA configuration problem. In this paper, we suggest two strategies for the Genetic Algorithm configuration, to improve the performance of ET. One is Annealing Genetic Algorithm (AGA), which alters the mutation probability dynamically, and the other is Restricted Genetic Algorithm (RGA), which adds restrictions into fitness functions. The two strategies made ET hit the global optimal solution in fewer generations, and most offspring genes located in the legal domain.

1. Introduction

Evolutionary Testing (ET) refers to the use of a meta-heuristic search technique, Genetic Algorithm (GA) for test case generation. Because of its high automaticity, it improves the testing efficiency which results in the reducing of developing cost since about 40%~50% software production development cost is expanded in testing [1]. Additionally, ET can deal with various complex structures of programs and data, and generate efficient test cases that may detect more bugs [2].

ET uses the principle of the biological evolution theory to optimize test case generation. It utilizes a fitness function to evaluate an individual in an original population, and generates the offspring population from some genetic operation. Under the guide of the function, after several iterative generations, we could get the global optimal solution [2].

Recent researches have covered many areas, such as extending the application of ET, removing the state problem in ET [3] and etc.. And one of the kernel problems is the strategy for the GA configuration, since it greatly affects the performance of ET. Without a proper configuration strategy, the efficiency of ET may be worse than Random Testing. Nowadays many researches focus on this problem, and some experiments have been conducted to compare various strategies [2]. But for a certain kind of program or testing object, ET needs a unique strategy for the configuration, and sometimes with the evolutionary process, the strategy need to be altered dynamically to improve the performance of ET. Thus, in this paper, we suggest two configuration strategies, Annealing Genetic Algorithm (AGA), and Restricted Genetic Algorithm (RGA).

2. Problems for Traditional Configuration

Recently, we have conducted a series of experiments to inspect the traditional configuration method of Structural Evolutionary Testing for statement and branch coverage. In these experiments, we discovered two problems. One is that in the later phase of the evolution, some populations may not be stable around the global optimal solution as expected. Instead, they evolve without a consistent direction. This phenomenon may result in missing the global optimal solution.

[1] This work was supported in part by the National Natural Science Foundation of China (60425206, 60373066, 60403016), National Grand Fundamental Research 973 Program of China (2002CB312000), and National Research Foundation for the Doctoral Program of Higher Education of China (20020286004).
Correspondence to: Baowen Xu, Department of Computer Science and Engineering, Southeast University, 210096 Nanjing, China. Email: bwxu@seu.edu.cn

The other one is that if we code each parameter of a test case in the same length, we may get invalid solutions, which exceed their legal search domain.

3. New Configuration Strategies

In the traditional ET, we choose some proper configuration strategies to improve the performance of the algorithm. But during the evolution process, the characteristics of the population are degenerating. For example, in the population that converges too fast, the quantity of the same individuals may be too high in proportion to the size of the later generations, which makes the population lose its diversity. Usually the original configuration strategy made for the forefather population is no longer fit for the offspring population. Thus firstly we suggest the strategy AGA, which detects the status of the population and chooses new configuration to replace the present one dynamically.

The strategy considers about the parameter of mutation probability. Besides using GA, we introduce the simulated annealing (SA) into the optimization process. In the hybrid GA, the probability of mutation is no longer a constant. Instead, mutation probability is decided by a Boltzmann distribution according to the progress of annealing [4]. For example, we could set $exp(-\theta/T)$ as the mutation probability, in which θ is a constant and T is the annealing temperature that is related to the time of iteration. With this strategy we can change our parameter dynamically, to decrease the mutation probability with the evolution progress. Consequently, when the population evolves closer to the global solution, it becomes more stable and has lower mutation possibility.

Secondly, we suggest RGA, which introduces the restriction conditions to the algorithm. Commonly the input test cases have their restriction in searching domain. For example, we have a input vector $<a, b, c>$, which a has a rage of $[-100,100]$, b and c have a rage of $[-50,50]$. If all the three elements have been transformed into the same length codes, the element b and c may have fairly high possibility to exceed their valid rages when we do genetic operations. Thus, we bind the punishment factor $\theta*\Phi(x)$ with the original function like this:

$$ff(x)=f(x)+\theta*\Phi(x),\ (\theta<0)$$

To restrictions: $g_i(x)>=0$, $i=1,2 \ldots m$, and $h_j(x)=0$, $j=1,2 \ldots n$, we define the $\Phi(x)$ like this:

$$\Phi(x)=\sum_{i=1}^{m}\alpha_i \mid \min\{0, g_i(x)\} \mid + \sum_{j=1}^{n}\beta_j \mid h_j(x) \mid \ (\alpha_i,\ \beta_j > 0)$$

Replace the original function $f(x)$ with the function $ff(x)$.We can learn that the punishment factor $\theta*\Phi(x)$ gives a negative punishment value to invalid genes, and the far the gene is away from the valid domain the greater punishment value is. But it gives no punishment to valid genes. So the punishment factor confines the solutions into valid domain, and helps to generate legal solutions.

4. Conclusions

In this paper, we suggest to replace parts of the original configuration strategies with AGA and RGA. From experiments in structural testing we discovered that new strategies made ET hit the global optimal solution in fewer generations, and most offspring genes located in the legal domain.

In future work we will extend our experiments to other testing objectives to inspect the efficiency of AGA and RGA. Also, we will suggest some other configuration strategies and advanced genetic operations to improve the performance of ET.

5. References

[1] D.R Graham, "Software testing tools: A new classification scheme", *Journal of Software Testing, Verification and Reliability*, Vol.1, No.2, pp. 18-34, 1992.

[2] H.H. Sthamer, "The Automatic Generation of Software Test Data Using Genetic Algorithms", PhD Thesis, University of Glamorgan, Pontyprid, Wales, Great Britain, April 1996.

[3] ark Harman, Lin Hu, Rob Hierons, Joachim Wegener, Harmen Sthamer, Andre Baresel and arc Roper, "Testability Transformation", *IEEE Transactions on Software Engineering*, Vol.30, No.1, January 2004.

[4] D. Sirag and P. Weisser, "Toward a Unified Thermodynamic Genetic Operator", Proc. 2nd Int. Conf. Genetic Algorithms, Cambridge, A,1987.