Random Testing with Dynamically Updated Test Profile

Kai-Yuan Cai[1], Hai Hu, Chang-Hai Jiang, Feng Ye
National Key Laboratory of Science and Technology on Holistic Control,
Department of Automatic Control
Beijing University of Aeronautics and Astronautics
Beijing 100191, China
kycai@buaa.edu.cn

Abstract

Random testing is popular in the area of software testing and often serves as a benchmark technique in comparison with other software testing techniques. Suppose that the input domain of the software under test is divided into m classes. Random testing often assumes that the underlying test profile is defined as $\{p_1, p_2, \ldots, p_m\}$, which may be a uniform or non-uniform probability distribution. That is, a test case is selected from class i with a constant probability p_i during software testing. In this paper, we follow the idea of software cybernetics and propose a concise yet effective technique of dynamic random testing (DRT). The test profile is dynamically updated during software testing by using feedback information. Case studies with two subject programs show that the proposed technique is encouraging in comparison with random testing technique and an adaptive testing technique.

1. Introduction

Random testing and partition testing are two major forms of software testing [1]. A natural idea is to combine random testing and partition testing to take advantages of both forms. This leads to the so-called random-partition testing [2, 3].

Many efforts have been made to improve conventional random testing. For example, the so-called anti-random testing and adaptive random testing techniques are proposed. On the other hand, a particular line of testing techniques, including the dynamic partitioning techniques [2] and the adaptive testing techniques [3], is proposed to improve the random-partition testing process by employing testing data collected online. This further improves conventional random testing as well as partition testing.

[1] Cai is supported by the National Science Foundation of China and Microsoft Research Asia (Grant No. 60633010)

This paper follows an alternative line and proposes a Dynamic Random Testing (DRT) technique to improve the existing random-partition testing process. The major advantage of the proposed DRT technique is that the test profile or the probability distribution $\{p_1, p_2, \ldots, p_m\}$ is dynamically updated during software testing according to the testing data collected online to improve the subsequent software testing process.

2. Dynamic Random Testing

Suppose the test suite is divided into m subdomains or classes $C_1, C_2, \ldots, C_m$, which may be disjoint or overlapping. It is assumed that C_i comprises k_i distinct test cases. A particular random-partition testing technique [2] follows two steps to select a test case for execution: first, a test case class, say C_i, is selected with probability p_i. Second, a test case is picked up from class C_i for execution, each with probability $1/k_i$. That is, the underlying test profile is described by $\{p_1, p_2, \ldots, p_m\}$, which may be a uniform or non-uniform probability distribution and remain invariant during software testing. In this paper, we propose to treat p_i as a variable so that it can be adjusted during the testing process by using feedback information. If a defect is detected by the test case picked up from C_i, then an increment is given to p_i. If the test case fails to detect a defect, then p_i is reduced to a certain extent. More specifically, we propose the following testing algorithm:

Algorithm DRT

Step 1 Partition the given test suite into m subdomains $C_1, C_2, \ldots, C_m$, which comprise $k_1, k_2, \ldots, k_m$ distinct test cases, respectively; $k_1 + k_2 + \cdots + k_m \geq k$, where k is the number of distinct test cases contained in the test suite.

Step 2 Initialize the parameter ε, $0 < \varepsilon < 1$.

Step 3 Select a subdomain in accordance with a given probability distribution $\{p_1, p_2, \ldots, p_m\}$; that is, the probability of the ith subdomain C_i being selected is p_i; suppose the lth subdomain C_l is selected;

Step 4 Select a test case from C_l at random in accordance with the uniform distribution; that is, each distinct test case in the ith subdomain has an equal probability of $1/k_l$ being selected;

Step 5 The selected test case is executed against the software under test and the testing results are recorded;

Step 6 If a defect is detected by the test case selected from C_l then the following sub-steps are taken:

(6-1). let $$p_j = \begin{cases} p_j - \dfrac{\varepsilon}{m-1} & \text{if } p_j \geq \dfrac{\varepsilon}{m-1} \\ 0 & \text{if } p_j < \dfrac{\varepsilon}{m-1} \end{cases} \quad \text{for } j \neq l$$

$$p_l = 1 - \sum_{j \neq l} p_j$$

(6-2).the detected defect is removed from the software under test.

Step 7 If no defect is detected by the test case selected from C_l, then let

$$p_l = \begin{cases} p_l - \varepsilon & \text{if } p_l \geq \varepsilon \\ 0 & \text{if } p_l < \varepsilon \end{cases}$$

$$p_j = \begin{cases} p_j + \dfrac{\varepsilon}{m-1} & \text{if } p_l \geq \varepsilon \\ p_j + \dfrac{p_l}{m-1} & \text{if } p_l < \varepsilon \end{cases} \quad \text{for } j \neq l$$

Step 8 The executed test case is returned into C_l; the test suite and the partitioning keep invariant;

Step 9 The given testing stopping criterion is checked; if it is not satisfied, then go to Step 3;

Step 10 The software testing process is stopped.

3. Case Study and Conclusion

Two real life software programs: the Space program [3] and the SESD (Software Environment for Software Data collection) program are examined in the experiments. Two versions of the Space program with different number of defects are included. Table 1 tabulates more details about the subject programs examined in the experiment.

Four testing techniques are examined in the experiment: the conventional random testing technique-RT, the Dynamic Random Testing technique -DRT, and two Adaptive Testing techniques with different settings, namely AT-GA and AT-GA-200 [3]. We set $\varepsilon = 0.05$ tentatively in the experiments to alleviate this impact.

Table 1. Experiment Setup

Program	LOC	# of defects	# of test cases	# of partitions
SPACE-21	9,564	21	13,498	4
SPACE-36	9,564	36	13,498	4
SESD	3,559	26	5,938	17

Table 2. Experimental Results

Simulation Scheme	Testing Strategies	$\bar{T}$	*Percentage reduction*	D	CV
SESD	RT	514.13	-	265.18	51.58%
	DRT	451.24	-12.23%	249.89	55.38%
	AT-GA	516.43	0.45%	342.43	66.31%
	AT-GA-200	473.75	-7.85%	305.00	64.38%
Space-36	RT	2623.42	-	2476.84	94.41%
	DRT	1994.07	-23.99%	1487.50	74.60%
	AT-GA	2152.56	-17.95%	2079.70	96.62%
	AT-GA-200	1793.20	-31.65%	872.20	48.64%
Space-21	RT	2332.47	-	3222.35	138.15%
	DRT	1711.23	-26.63%	924.40	54.02%
	AT-GA	1975.68	-15.30%	1189.74	60.22%
	AT-GA-200	1386.63	-40.55%	918.50	66.24%

* $\bar{T}$: the average total number of test cases over 50 trials;
D : the standard deviation of T over 50 trials.
CV :the coefficient of variation over 50 trials.

The computational overhead of the testing strategies are also recorded. Table 3 tabulates the time required to select the next test cases:

Table 3. Computational Overhead (SPACE-36)

Time Consumption	RT	DRT	AT-GA	AT-GA-200
Total (ms)	104.26	153.14	79048.28	30540.34
Each test case(ms)	0.21	1.98	188.86	64.47

Experimental data suggest the DRT technique noticeably outperforms RT and AT-GA in terms of its defect detecting efficiency as well as its stability of performance over 50 trials. In comparison with the AT-GA-200, the DRT technique uses slightly more test cases to detect the same number of defects. However, the computational overhead incurred for test case selection in the adaptive testing technique is almost avoided.

References

[1]. W. J. Gutjahr, "Partition Testing vs. Random Testing: The Influence of Uncertainty", *IEEE Trans. Softw. Eng.* Vol.25(5), 1999, pp.661-674.

[2]. K.Y. Cai, B. Gu, H. Hu, and Y.C. Li, "Adaptive Software Testing with Fixed-Memory Feedback", *J. Syst. Softw.*, Vol.80, 2007, pp. 1328-1348.

[3]. K.Y. Cai, T. Jing, and C.G. Bai, "Partition Testing with Dynamic Partitioning", *Proc. COMPSAC'05*, pp. 113-116, Edinburgh, U.K., 2005.