Parameter Adjustment of Test Profile in Dynamic Random Testing

Qiong-jie QU¹, Bei-bei YIN¹,a* and Han-yu PEI¹

¹. School of Automation Science and Electrical Engineering, Beihang University, Beijing, China

ayinbeibei@buaa.edu.cn

*Corresponding author

Keywords: Software cybernetics, Dynamic random testing, Testing profile, Parameter adjustment

Abstract. The basic motivation of Dynamic Random Testing (DRT), which is proposed in the context of software cybernetics, is to use test results as the feedback information to realize dynamic adjusting of selection probabilities of all the test cases. It is unreasonable that the profile adjustment in the original DRT is somehow contingent and only utilizes the latest execution result. Moreover, the parameter adjustment of test profile for different subdomains is uniform in original DRT, which may not be the best solution to improve the effectiveness of DRT. This paper presents an improved dynamic random testing that based on the function information of test cases (DRT-F) which makes use of correlation among the test cases to improve the performance of the original DRT. In DRT-F, the parameter adjustment is according to the function information of test case classification and makes the DRT adaptive to different classification of its influence on its performance.

Introduction

Software testing, as an inevitable activity to improve and ensure the reliability of a software system, has received significant attention and extensively studied. One of the purposes of software testing is to detect—as more as defects by using as less test cases as possible.

Many software testing techniques have been developed. In Random testing (RT) [3], test cases are selected and executed from the entire test suite randomly. In Partition Testing (PT) [1], the input domain of the software under testing is partitioned into several disjoint or overlapping subdomains, and then one or more test cases are generated from each subdomain. By combining RT and PT, a hybrid strategy, called Random-Partition Testing (RPT), is proposed [2]. RPT selects a subdomain according to the given test profile and then selects a test case from this subdomain randomly.

Since defects in software tend to clusters, Dynamic Random Testing (DRT) [5] is proposed in the context of software cybernetics [8]. The major idea of DRT is to make full use of the previous test data to dynamically adjust the test profile. If a defect is detected by a test case in a certain subdomain, then the subdomain is considered to have a higher defect detection rate and the selection probability of the subdomain should increase from $p_i^j$ to $p_i^j + \varepsilon$. Otherwise, the selection probability decreases to $p_i^j - \varepsilon$.

In previous study, DRT outperformed RT and RPT in terms of the number of test case required to reveal a given number of defects. However, there are several aspects in original DRT which can be improved:
1. The test profile is adjusted after executing each test case, and it only depends on the latest history information. On one hand, such frequent adjustment may introduce error randomly since the current subdomain might not have a higher defect detection rate. On the other hand, due to the random selection of the test cases and the occasional detection of defects, only using the current test data is not reasonable. Therefore, more information of the test cases should be considered in testing process. Zhang et al. [11] adjusted the test profile and estimated defect detection rates using more history information, however, the frequency and range of the profile adjustment are still fixed.

2. The parameter adjustment of test profile for different subdomains are uniform in original DRT, which may not be the best solution to improve the effectiveness of DRT. It is difficult for DRT to meet the real distribution of defect detection rate of the test case subdomain in time from the perspective of nature. Yang et al. [12] used the real-time defect detection rate to adjust the parameters during testing process, however, it just to improve the algorithm process and without the benefit of information of software, especially the information of test case. In software testing, the generation of testing data is one of the key steps and has great effect on software testing that include the selection and classification of test cases.

Therefore, it is natural to employ the information of test case classification to demonstrate the correlation among the test case subdomains and make the DRT adaptive to different classification. The function of the test cases obtained through the SIR website can be mapped to the information of test cases classification to a certain extent. Therefore, a new DRT strategy, named DRT-F, is proposed in this paper to realize the test profile parameter adjustment by employing the distance information among test cases. And the naive Bayesian classifier [14] is used to quantify the distances among the test case subdomains.

The remainder of this paper is organized as follows: related works are presented in Section II. Section III gives a brief introduction of DRT-F. Experiments setup on five real-life subject programs and experimental results are presented in Section IV. In Section V, evaluation of the effectiveness of DRT-F is discussed. Conclusions and future research plans are presented in section VI.

Related Works
This study is related to the comparison of RT and PT in terms of defect removal efficiency and make a list of merit and demerit of previous work. Tsoukalas et al. [2] held that PT is more attractive than RT in terms of upper confidence bounds for the cost weighted performance, which represents worst-case situations. Weyuker and Jeng concluded that compared with RT, PT is either a fantastic testing method or a poor one, which mainly depends on how a wrong output is revealed by the inputs that clustered in the subdomains. Gutjahr [1] first proposed PT outperforms RT when the probability of failure is not known with certainty, and further promoted the works of Weyuker and Jeng. Adaptive testing (AT) was proposed by Cai et al. in the context of software cybernetics [4]. AT with fixed-memory feedback and the high computational overhead and in order to balance the effectiveness and computational overhead, DRT were proposed.

DRT were proposed to reduce the computational overhead in AT without losing ability of defect detection. In previous study [5][6], DRT improved the effectiveness of both RT and RPT. A history-based DRT strategy was proposed by Zhang et al. to further improve the DRT process according to more testing history collected online. A
sufficient condition for estimation in DRT is proposed by Lv et al., in which the theoretical boundaries of parameters in DRT was discussed to guide parameters selection [8]. Based on this idea, DRT with parameter adjustment is proposed by Yang et al. to guarantee that theoretical boundaries are satisfied by updating the parameters according to testing history during the testing process [11].

There are plenty of defect location and defect prediction technologies, such as expert driver, static model, and machine learning. And these methods for ascension of Naive Bayes theorem is not clear [12]. Due to its good prediction effect and easy to understand, many researchers advocate using Naive Bayes theorem [13].

An Improved Dynamic Random Testing That Based on the Function Information of Test Cases

The main advantage of DRT-F is to use the function information of test cases to show the difference between them. In this paper using the naive Bayesian classifier [13] model technology to quantify the difference between the test case subdomains in the defect prediction, Bayesian classifier in the attribute value corresponding to the function of the test case subdomain, and the classifier instance corresponding to each of the functional configuration values in a test case subdomain. In DRT-F, the classification of function analogy for attributes and the function of the specific configuration analogy for the value of the attribute. Assume that the function of the subdomains that are selected is independent, so the posterior probability can be expressed as the multiplication of posterior probability of the different functional value. The example as be shown below.

Assuming the functions of X and Y are classified in test case, and the function of X have two configuration values that respectively are x1 and x2, and function Y also have two configuration values y1 and y2. Then the configuration values are divided into five test case subdomains: cases1 (x1, y1), cases2 (x1, y2), cases3 (x2, y1), cases4 (x2, y2), case5 (single and error). Set output for Z, if the defect is detected Z = 1, if not Z = 0. Based on the simple Bayesian network classifiers [13][14] can get formula (1)

\[
P(Z = 1, X = x_i, Y = y_j) = [P(c1) \cdot P(e1) \cdot 0.5 + P(c2) \cdot P(e2) \cdot 0.5] \cdot [P(c1) \cdot P(e1) \cdot 0.5 + P(c3) \cdot P(e3) \cdot 0.5] \\
P(Z = 0, X = x_i, Y = y_j) = [P(c1) \cdot Q(e1) \cdot 0.5 + P(c2) \cdot Q(e2) \cdot 0.5] \cdot [P(c1) \cdot Q(e1) \cdot 0.5 + P(c3) \cdot Q(e3) \cdot 0.5]
\]

DRT-F according to the relationship between the test case subdomains adjusting the selection probability of other test case subdomain according to a certain proportion. For example, assume that relationship between the test case subdomains can establish the following relationship matrix:

\[
M = \begin{bmatrix}
0 & 1 & 1 & 2 & 1 \\
1 & 0 & 2 & 1 & 1 \\
1 & 2 & 0 & 1 & 1 \\
2 & 1 & 1 & 0 & 1 \\
3 & 3 & 3 & 3 & 0
\end{bmatrix}
\]

Among them, \(M[i, j]\) suggest that when defect was found from the case \(i\) and the probability of case \(j\) could reduce the value in of in the meantime. On the contrary, increase the value in of in the meantime. Choose the first column in the matrix \(M\) to
interpret, for cases1, cases2 and cases3 shared a value with it, when found defects in cases1, the probability of cases2 and cases3 could be reduced smaller. And the function configuration values between cases4 and cases1 unlike, but involves the same function module, so in second place. Cases5 is for some special cases of test case subdomain and different from case1 on function module, so that rolled into last place.

The DRT-F Algorithm

The above example is described in detail that how to establish the relationship between the test case subdomains and matrix M in DRT-F algorithm. The DRT-F algorithm is introduced as follows:

Step 1: The test suite is partitioned into m subdomains: \( \{C_1, C_2, \ldots, C_m\} \) which include \( k_1, k_2, \ldots, k_m \) distinct test case respectively, establish the test case subdomain relationship matrix \( M \);

Step 2: Initialize the test \( p_1, p_2, \ldots, p_m \) profile and the initial value of parameters \( \varepsilon \) and \( \delta \) is 0.05;

Step 3: A subdomain is selected in accordance with the given probability distribution; suppose the subdomain \( i \) is selected;

Step 4: A test case is selected from subdomain randomly;

Step 5: Execute the selected test case, and record the test results;

Step 6: If this test case \( i \) from subdomain detects a defect, record and eliminate defects that was found, and modify the test profile according to the formula (2):

\[
\delta_j = \frac{M[j, i]}{\sum_{j=1}^{k} M[j, i]}, \quad \eta_j = 1 - \sum_{j=1}^{k} \delta_j
\]

\[
P_i = \begin{cases} 
\delta_j - \eta_j & \text{if } R_j \geq \delta_j, \text{ for } f \neq i \\
0 & \text{if } R_j < \delta_j, \text{ for } f \neq i
\end{cases}
\]

(2)

Step 7: If this test case \( i \) from subdomain does not detect any defect, then modify the test profile according to the formula (3):

\[
\delta_j = \frac{M[j, i]}{\sum_{j=1}^{k} M[j, i]}, \quad \eta_j = 1 - \sum_{j=1}^{k} \delta_j
\]

\[
P_i = \begin{cases} 
\delta_j - \eta_j & \text{if } R_j \geq \delta_j, \text{ for } f \neq i \\
0 & \text{if } R_j < \delta_j, \text{ for } f \neq i
\end{cases}
\]

(3)

Step 8: Check if the test termination conditions are met; or go to Step 3;

Step 9: Terminate the test.

Experiment Setup

Subject Programs

The experiments were conducted on four-real subject programs (see Table 1). There are some modified versions and test suites for each program, except several versions without a defect or test case, and all defects were injected by software developers.
Table 1. Subject program.

<table>
<thead>
<tr>
<th>Subject Programs</th>
<th>Size (LOC)</th>
<th>Versions of Defects of Test Cases of Subdomain</th>
</tr>
</thead>
<tbody>
<tr>
<td>flex</td>
<td>10, 459</td>
<td>v1-v2-v3</td>
</tr>
<tr>
<td>grep</td>
<td>10, 068</td>
<td>tsl1:v1-v2-v3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tsl2:v1</td>
</tr>
<tr>
<td>gzip</td>
<td>5, 680</td>
<td>v1-v2-v3</td>
</tr>
<tr>
<td>make</td>
<td>35, 545</td>
<td>v1-v2</td>
</tr>
</tbody>
</table>

Testing Strategies

These three testing strategies are used in the experiments, i.e., RPT, DRT and DRT-F, the detailed settings are shown as follows: if a defect is detected, it will be removed immediately from the software; the test will be stopped when all defects are detected and removed; experiment for each strategy repeats 1000 times.

Measures

In order to reflect both defect detection effectiveness and the robustness of testing strategies, five metrics were measured in the experiment:

T: The total number of test cases consumed to detect and remove all defects. The average value of T is denoted by $\bar{T}$.

D: The standard variance of T over 1000 iterations. The following equations give the mathematical definitions of D.

$$D = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (T_i - \bar{T})^2}$$  \hspace{1cm} (4)

P-value: The p-value indicates whether there is a significant difference between the efficiency of two strategies. If the p-value is equal to or smaller than the significance level (traditionally 5%), it suggests that the observed data are inconsistent with the assumption that the null hypothesis is true (there is the significant difference).

Effect size: The effect size is a quantification of the strength of a strategy better than another one. The formula of is

$$A_{12} = \frac{\bar{D}_1 - \bar{D}_2}{2}\sqrt{n}$$  \hspace{1cm} (5)

Where $A_{12}$ is the rank sum of the former data group, and are the number of observations in the former and later data samples. The range of is [0, 1], and it reflects the extent of strength or shortcoming. If $A=0.5$, it means the two strategies are equivalent. If $A>0.5$, it expresses the former is superior to the later. More details about the effect size presented in 0.

Experiment Results

Table 2 provides results and comparisons in terms of T, D, p-value and effect size.
<table>
<thead>
<tr>
<th>Software Subject</th>
<th>Testing Strategies</th>
<th>T</th>
<th>D</th>
<th>p-value with RPT</th>
<th>p-value with DRT</th>
<th>A12 with RPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>flexv1</td>
<td>RPT</td>
<td>24.3</td>
<td>19.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT</td>
<td>23.8</td>
<td>19.0</td>
<td>0.382</td>
<td>0.966</td>
<td>0.511</td>
</tr>
<tr>
<td></td>
<td>DRT-F</td>
<td>23.0</td>
<td>18.0</td>
<td>0.399</td>
<td>0.966</td>
<td>0.511</td>
</tr>
<tr>
<td>flexv2</td>
<td>RPT</td>
<td>151.1</td>
<td>111.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT</td>
<td>150.9</td>
<td>98.8</td>
<td>0.001</td>
<td>0.904</td>
<td>0.541</td>
</tr>
<tr>
<td></td>
<td>DRT-F</td>
<td>115.1</td>
<td>76.3</td>
<td>0.001</td>
<td>0.904</td>
<td>0.541</td>
</tr>
<tr>
<td>greptsl1v1</td>
<td>RPT</td>
<td>263.0</td>
<td>215.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT</td>
<td>246.0</td>
<td>209.5</td>
<td>0.483</td>
<td>0.530</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT-F</td>
<td>193.4</td>
<td>135.6</td>
<td>0.021</td>
<td>0.003</td>
<td>0.966</td>
</tr>
<tr>
<td>greptsl1v2</td>
<td>RPT</td>
<td>106.9</td>
<td>100.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT</td>
<td>98.2</td>
<td>98.3</td>
<td>0.271</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT-F</td>
<td>81.7</td>
<td>73.0</td>
<td>0.001</td>
<td>0.027</td>
<td>0.543</td>
</tr>
<tr>
<td>greptsl2v1</td>
<td>RPT</td>
<td>194.0</td>
<td>191.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT</td>
<td>181.4</td>
<td>84.1</td>
<td>0.579</td>
<td></td>
<td>0.507</td>
</tr>
<tr>
<td></td>
<td>DRT-F</td>
<td>140.0</td>
<td>123.8</td>
<td>0.000</td>
<td>0.000</td>
<td>0.566</td>
</tr>
<tr>
<td>gzipv1</td>
<td>RPT</td>
<td>428.2</td>
<td>310.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT</td>
<td>309.4</td>
<td>257.7</td>
<td>0.000</td>
<td></td>
<td>0.593</td>
</tr>
<tr>
<td></td>
<td>DRT-F</td>
<td>257.9</td>
<td>189.9</td>
<td>0.000</td>
<td>0.000</td>
<td>0.645</td>
</tr>
<tr>
<td>gzipv2</td>
<td>RPT</td>
<td>16.2</td>
<td>16.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT</td>
<td>17.4</td>
<td>15.5</td>
<td>0.910</td>
<td></td>
<td>0.499</td>
</tr>
<tr>
<td></td>
<td>DRT-F</td>
<td>17.0</td>
<td>16.2</td>
<td>0.102</td>
<td>0.088</td>
<td>0.521</td>
</tr>
<tr>
<td>makev1</td>
<td>RPT</td>
<td>57.3</td>
<td>44.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT</td>
<td>49.8</td>
<td>37.4</td>
<td>0.182</td>
<td></td>
<td>0.517</td>
</tr>
<tr>
<td></td>
<td>DRT-F</td>
<td>50.8</td>
<td>39.1</td>
<td>0.000</td>
<td>0.002</td>
<td>0.557</td>
</tr>
<tr>
<td>makev2</td>
<td>RPT</td>
<td>75.5</td>
<td>64.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DRT</td>
<td>66.1</td>
<td>53.5</td>
<td>0.000</td>
<td></td>
<td>0.559</td>
</tr>
<tr>
<td></td>
<td>DRT-F</td>
<td>63.8</td>
<td>53.2</td>
<td>0.000</td>
<td>0.001</td>
<td>0.604</td>
</tr>
</tbody>
</table>

**Analysis and Discussion**

**Data Analysis**

The test case can be divided into 5 ~ 9 classes generally. To acquire the matrix in DRT-F include a parameter: the k of function number that partition test case. The light color suggests that the p-value is less than 0.5.

As can be seen from the table above, on the whole, DRT-F use fewer test cases to find all the defects rather than RPT and DRT, and the value of D, T of DRT-F is smaller, so the stability is better. In addition, the classification of different parameters (k) has less effect on the DRT-F, it can be seen under the different parameter values DRT-F still has good performance, which means DRT-F also has strong adaptive parameters.

Exceptions are gzipv2, flexv1, greptsl2v1, on the four subjects, there is no obvious advantage relative to the DRT in the DRT-F. On the one hand, because of the smaller test step size (< 30) in gzipv2 and flexv1, there is not enough time to reflect the relationship between the test case subdomains and can't play to the advantages of DRT-F; On the other hand, due to the classification of the test cases of various kinds of defect detection rate is not big difference, makes the RPT, DRT and DRT-F were no obvious difference of the effectiveness of these three methods.

As can be seen from the table 3 the statistics between DRT-F and DRT, RPT has a significant difference in most cases (10/14), and the detection of defects has a better efficiency. Exceptions are still gzipv2 flexv1, greptsl2v1, its reason in the period of the
From the experimental results, while DRT-F improved the classification of adaptability relative to the DRT to some extent, but to the adaptive classification on improvements remains limited in the subjects greptsl2v1. Therefore, DRT-F found the trend that the test case subdomain better gathered and improved the DRT defect detection efficiency.

Table 3. The result of effect sizes.

<table>
<thead>
<tr>
<th></th>
<th>greptsl1v1</th>
<th>greptsl1v2</th>
<th>greptsl1v3</th>
<th>greptsl1v4</th>
<th>greptsl1v5</th>
<th>makev1</th>
<th>makev2</th>
<th>gzipv1</th>
<th>gzipv2</th>
<th>gzipv3</th>
<th>gzipv4</th>
<th>gzipv5</th>
<th>flexv1</th>
<th>flexv2</th>
<th>flexv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRT-F vs. RPT</td>
<td>0.570</td>
<td>0.535</td>
<td>0.519</td>
<td>0.503</td>
<td>0.480</td>
<td>0.545</td>
<td>0.549</td>
<td>0.689</td>
<td>0.480</td>
<td>0.635</td>
<td>0.577</td>
<td>0.514</td>
<td>0.602</td>
<td>0.680</td>
<td></td>
</tr>
<tr>
<td>DRT vs. RPT</td>
<td>0.526</td>
<td>0.528</td>
<td>0.477</td>
<td>0.488</td>
<td>0.483</td>
<td>0.517</td>
<td>0.518</td>
<td>0.661</td>
<td>0.464</td>
<td>0.608</td>
<td>0.543</td>
<td>0.502</td>
<td>0.541</td>
<td>0.577</td>
<td></td>
</tr>
<tr>
<td>DRT-F vs. DRT</td>
<td>0.543</td>
<td>0.506</td>
<td>0.543</td>
<td>0.516</td>
<td>0.512</td>
<td>0.528</td>
<td>0.532</td>
<td>0.597</td>
<td>0.597</td>
<td>0.597</td>
<td>0.597</td>
<td>0.597</td>
<td>0.597</td>
<td>0.597</td>
<td></td>
</tr>
</tbody>
</table>

Although DRT-F has a high cost in the test case subdomain relationship matrix $M$, because of DRT、 RPT according to the function index should be considered in the test case classification and established on the basis of function configuration values to matrix $M$ value strategy is also very simple, so the computational complexity relative to RPT and DRT is negligible.

Conclusions and Future Works

Dynamic random testing strategy is to use test result as feedback information in realizing dynamic correction of all the test cases selection possibilities so that the test subclass with higher defect rate can have a higher probabilities to be selected. However, each profile correction is only based on the present execution result, and the profile correction is somehow contingent. This is unreasonable. The paper analyzes its flaws and suggests a kind of improved DRT strategy (DRT-F). Its advantage lies in its use of test case subdomain information to modify the parameters value of DRT, making the parameter adjustment more accurate and more quickly close to the real defect detection section and to greatly improve the defect detecting ability.

Adopting software control theory as the framework of the thought and control theories and approaches as the theoretical basis, the paper applies the feedback control mechanism to the software test process. It is of great practical significance to study the software test strategies that guide the test case selection. However, DRT-F still has its own flaws. In the future controller design, how to integrate the information above to design a test strategy closer to the reality forms a direction of research is a direction.

References


