Approach for Test Profile Optimization in Dynamic Random Testing

Ye Li, Bei-Bei Yin, Junpeng Lv, Kai-Yuan Cai
Department of Automatice Control
Beihang University, Beijing 100191, China
yeahli@buaa.edu.cn

Abstract—**Dynamic Random Testing (DRT) is a feedback-based software testing strategy, which has been proved to be more effective than the traditional Random Testing (RT) and Random-Partition Testing (RPT) strategies. The major advantage of DRT is that the test profile is dynamically adjusted based on the previous test data. Since the frequency and range of the profile adjustment are fixed during the testing process, DRT might not react in time to the changes of the defect detection rates. In order to overcome these shortcomings, an approach for the test profile optimization in DRT, denoted as O-DRT, is proposed in this paper. In O-DRT, the test profile adjustment contains two parts. In addition to the original adjustment in DRT, O-DRT will change the test profile to a theoretically optimal one when the pre-defined criterion is satisfied. The theoretically optimal test profile is calculated based on an optimization goal of both maximizing the overall defect detection rate and minimizing its variance. Experiments on five real-life software subjects are conducted to validate the effectiveness of O-DRT. Experimental results demonstrate that O-DRT outperforms RPT and DRT in terms of the number of test cases required to detect and remove a given number of defects.**

Keywords-software cybernetics; random-partation testing; dynamic random testing; test profile optimization

I. Introduction

Random Testing (RT) and Partition Testing (PT) are two main forms of software testing. In RT, test cases are selected and executed from the entire test suite/input domain of the software under test (SUT) according to a uniform or nonuniform probability distribution [1, 2]. On the contrary, in PT [3], the input domain of SUT is partitioned into several overlapping or disjoint subdomains, and at least one test case is generated from each subdomain. PT is intuitively absorbing, and various coverage criteria, e.g., statement coverage, functional coverage, and dataflow coverage, can be used to generate test cases.

By combining RT with PT, random-partition testing (RPT) is proposed [4, 5]. Supposing that the test suite is partitioned into m subdomains, RPT first selects a subdomain according to the given test profile $\{p_1, p_2, \ldots, p_m\}$, where p_i is the selection probability of subdomain. Then, a test case from this chosen subdomain is randomly selected for execution.

In RPT, a subdomain C_i is selected with a constant probability p_i during the testing process. This might not be always desirable, because that defects tend to clusters [6, 7]. In the later stages of software testing, the defects proneness may be revealed by the previous testing history, which means the test cases in some classes may detect defects more easily than others. In order to improve RPT, Dynamic Random Testing (DRT) [8] is proposed in the context of software cybernetics, which aims to explore the interplay between software engineering and control theory [9]. The main advantage of DRT is that the test profile $\{p_1, p_2, \ldots, p_m\}$ is dynamically adjusted using the previous test data to improve the subsequent testing process. The main concept of DRT is to increase the selection probabilities of the classes which have higher defect detected rates. If a defect is detected by some test cases from subdomain C_i, then C_i is considered to have a higher defect detected rate and an increment, i.e. ε, is added to p_i. Otherwise, p_i is decreased to $p_i - \delta$. In the previous studies [10-12], DRT outperformed RT and RPT in the aspect of the number of test cases required to reveal given defects.

However, there are several aspects in original DRT which can be improved:

- The test profile is adjusted after executing each test case, which only depends on the latest history information. On one hand, such frequent adjustment may introduce random error since the current subdomain might not have a higher defect detection rate. On the other hand, this fixed adjustment might not react to the changes of failure detection rates after defects detection and removal in time, since the changes are instant and might be dramatic. Thus, the fixed adjustment may affect the effectiveness of DRT. Zhang et al. [11] estimated defect detection rates using more history information, but the frequency and range of the profile adjustment are still fixed.
- Such constant parameters might also delay DRT's convergence to the optimal test profile, since the changes of defect detection rates may be dramatic. Yang et al. [12] adjusted the parameters based on the theoretically optimized boundaries. However, this theoretical foundation of the estimation of defect detection rates was inexplicitly explained in the optimization process. Besides, the frequency of this adjustment is also fixed as in DRT.

Therefore, optimizing the test profile adjustment is a natural approach to improving DRT's effectiveness. In this paper, the optimization is realized by maximizing the benefit and minimizing the risk during the testing process, in which the overall defect detection rate and its variance denote the benefit and risk, respectively. More specifically, an objective function, in which the benefit and risk of using the current test profile are taken account, is constructed to quantify the performance of this test profile. If the performance do not meet the pre-defined expectation, the test profile will be adjusted immediately to the one that optimizes the objective

[1] This work was partially supported by the National Science Foundation of China and Microsoft Research Asia (Grant No. 61272164 and No.61402027).

function. Since the objective function needs estimated defect detection rates, the objective function evolves with the testing process. In this case, Bayesian estimation is adopted to estimate the defect detection rate of each subdomain [13] since it can incorporates prior assumptions into the analysis of testing results, which means it can adjust the probability of failure estimate when the prior test profile does not match the test profile. In addition, it provides a mathematical framework for incorporating information other than random testing result.

In this paper, a new DRT strategy, named O-DRT, is proposed to realize the optimization of the test profile adjustment. The advantage of O-DRT lies in two aspects. On one hand, O-DRT adjusts the test profile according to the objective function while retaining the original test profile adjustment in DRT. In this way, the error caused by frequency adjustment can be alleviated, and the response to the changes of defect detection rates should be timely. On the other hand, the test profile adjustment based on optimizing the objective function can adapt to the dramatic changes in the defect detection rates.

The remainder of this paper is organized as follows: Related studies are shown in Section Ⅱ. O-DRT is introduced in Section Ⅲ. In Section Ⅳ, the experiment setup and experimental results are presented. The experimental results are discussed in Section Ⅴ. Conclusions and future research plans are presented in section Ⅵ.

II. Related Studies

This study is related to the comparison of RT and PT in terms of defect removal efficiency. Tsoukalas et al. [14] 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 [3] 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 that PT outperforms RT when the probability of failure is not known with certainty.

Another related topic is testing strategies in the context of software cybernetics. Adaptive testing (AT) was proposed by Cai et al. in the context of software cybernetics [9]. From the perspective of software cybernetics, the issue of software testing is considered as a control problem, and history data are adopted to determine the strategy behavior. In AT, the SUT is modeled by a Markov chain. AT could significantly improve RT and RPT with respect to the number of test cases required to detect a given number of defects in experiments [9, 4]. AT with fixed-memory feedback [4] and DRT were proposed to reduce the high computational overhead in AT under some circumstance. Meanwhile, they do not lose ability of defect detection in AT. In the previous studies [10-12], DRT improved the effectiveness of both RT and RPT. DRT also has much lower computational overhead compared with AT and AT with fixed-memory feedback. A history-based DRT strategy, proposed by Zhang et al., further improves the DRT process using more testing history collected online [11]. 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 [10]. Based on this idea, Adjusted DRT 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 [12].

Bayesian estimation has been widely adopted in software testing researches. Lv et al. [15] proposed an enhanced AT to make a local optimal converge to the globally optimal solution. In that paper, the metrics were based on Bayesian estimation to validate the testing strategy's effectiveness. Rekab et al. [16] proposed a method to dynamically allocate test cases among partitions throughout the testing process for minimizing the expected loss incurred, in which Bayesian method is used to estimate the overall reliability due to its advantage of using the prior information. Benkamra et al. [17] proposed a risk-averse approach based on Bayesian analysis to estimate the reliability of a parallel-series system.

III. Approach forTest Profile Optimization in DRT

This section first presents three methods adopted in O-DRT. Then O-DRT algorithm is introduced in detail.

The first method is Bayesian estimation [13], which was used to estimate the overall defect detection rate and its variance in O-DRT as follows. The overall defect detection rate is denoted by θ. By adopting Bayesian estimation, we assume the prior distribution of θ is *Beta(a, b)*, the posterior (after test cases execution) distribution of θ is *Beta(x+a, t-x+b)*, where t is the number of test cases being chosen, x is the number of detected defects in t tests and a, b are the parameters of the prior distribution. Bayesian estimation of θ and its variance are given by

$$\hat{\theta} = \frac{x+a}{t+a+b}, \quad \hat{V} = \frac{(x+a)(t-x+b)}{(t+a+b)^2(t+a+b+1)} * t. \qquad (1)$$

The test suite is partitioned into m subdomains $\{C_1, C_2, \ldots, C_m\}$, and $\widehat{\theta_i}$ is allocated to each subdomain. The prior parameters for subdomain C_i are denoted as a_i and b_i. Then the defect detection rate of each class can be separately estimated by formula (1). Assuming these defect detection rates are independent, the estimation of overall θ is

$$\hat{\theta} = \sum_{i=1}^{n} p_i \widehat{\theta_i}, \quad \hat{V} = \sum_{i=1}^{n} {p_i}^2 \widehat{V_i}. \qquad (2)$$

where n is the number of subdomains, p_i is the selection probability of subdomain C_i.

The second method is to construct the objective function as below. The estimated value of θ_i and V_i are normalized by

$$\widehat{\theta_i}' = \frac{\widehat{\theta_i}}{\widehat{m\theta}}, \quad \widehat{V_i}' = \frac{\widehat{V_i}}{\widehat{mV}}. \qquad (3)$$

where $\widehat{m\theta}$ and $\widehat{mV}$ are the estimated value of *max(θi)*

$max(\theta_i)$ and $max(V_i)$ ($i \in \{1, 2, \ldots, m\}$), and they are selected based on prior information about software before testing. Therefore overall defect detection rate and its variance are calculated by

$$\hat{\theta}' = \sum_{i=1}^{m} p_i \widehat{\theta_i}', \quad \hat{V}' = \sum_{i=1}^{m} {p_i}^2 \widehat{V_i}'. \qquad (4)$$

The objective function is

$$f\left(\widehat{m\theta}, \widehat{mV}, p_1, \ldots, p_m, \widehat{\theta_1}, \ldots, \widehat{\theta_m}, \widehat{V_1}, \ldots, \widehat{V_m}\right) = V' - \theta'. \qquad (5)$$

The optimization goal is maximizing $\hat{\theta}'$ and minimizing $\hat{V}'$, i.e., minimize function value.

The third method is the rules of optimizing the test profile adjusted by DRT strategy as below. There are several restrictions of parameters in this function: the range of p_i is [0, 1], the sum of p_i is 1, and the range of both θ_i and V_i are [0, 1]. According to these conditions, the range of the objective function can be obtained as $[lb, ub]$, where lb=0, ub=1.

After $\hat{\theta}'$ and $\hat{V}'$ are substituted into the objective function, the value of the objective function is calculated. If this value is greater than fb, then the test profile will be adjusted to a theoretically optimal one. This theoretically optimal test profile is obtained by minimizing the objective function (formula (5)). The parameter fb is a pre-defined criterion to determine whether the test profile (adjusted by DRT) should be adjusted.

$$fb = lb + (ub - lb) * R, \tag{6}$$

where R represents the location of fb in the range of the objective function.

A. The O-DRT Algorithm

Step 1 The test suite is partitioned into m subdomains $\{C_1, C_2, ..., C_m\}$, which include $k_1, k_2, \ldots, k_m$ distinct test cases;
Step 2 The values of ε, δ, a_i, b_i, $\widehat{m\theta}$, $\widehat{mV}$, R are initialized;
Step 3 A subdomain is selected according to a given probability distribution $\{p_1, p_2, \ldots, p_m\}$; suppose the C_i subdomain is selected;
Step 4 A test case is selected from C_i randomly;
Step 5 The selected test case is executed against the SUT and the testing results are recorded;
Step 6 The estimation of the defect detection rate $\hat{\theta}_i$ and its variance $\hat{V}_i$ for the subdomain are calculated by formula (1);
Step 7 If the selected test case from C_i detect a defect, then

$$p_j = \begin{cases} p_j - \dfrac{\varepsilon}{m-1} & if\ p_j \ge \dfrac{\varepsilon}{m-1} \\ 0 & if\ p_j < \dfrac{\varepsilon}{m-1} \end{cases} \quad for\ j \ne l$$

$$p_l = 1 - \sum_{j \ne i} p_j.$$

Step 8 If there is no defect detected by this test case, then

$$p_l = \begin{cases} p_l - \delta & if\ p_l \ge \delta \\ 0 & if\ p_l < \delta \end{cases}$$

$$p_j = \begin{cases} p_j + \dfrac{\delta}{m-1} & if\ p_l \ge \delta \\ p_j + \dfrac{p_l}{m-1} & if\ p_l < \delta \end{cases} \quad for\ j \ne l.$$

Step 9 The objective function value can be obtained on the basis of parameter values above. If this value of the objective function is less than fb, then go to *step10*; if this function value is greater than fb and there is no adjustment by optimizing the objective function in q selection ($q \ge 1$), then the test profile is adjusted to $\{p_1', p_2', \ldots, p_n'\}$ by minimizing the formula (5);
Step 10 The given testing stopping criterion is checked; if it is not satisfied, then go to *step 3*;
Step 11 The testing process is stopped.

B. Remarks

Compared with DRT algorithm, there are some extra steps in O-DRT algorithm as below. In *step 6*, the estimation of overall variance is taken account since the optimization goal of both maximizing the overall defect detection rate and minimizing its variance. In *step 9*, the test profile can be optimized by minimizing the objective function, and the frequency of this optimization depends on the pre-defined parameter fb. In addition, some new parameters (a_i, b_i, R, $\widehat{m\theta}$, $\widehat{mV}$) are required to be initialized in *step 2*.

The given values of ε and δ are based on the DRT specific sufficient condition [10], which is described as follows. θ_i denotes the defect detection rate of the test case class C_i, and $\theta_M = max(\theta_i \mid i$=1, 2, …, $m)$, $\theta_\Delta = max(\theta_i \mid i$=1, 2, …, m, $i \ne M)$. Assume n is the number of test cases executed. To make sure $E(p_M(n+1) > p_M(n))$, the following equation needs to be hold:

$$1/\theta_M - 1 < \varepsilon/\delta < 1/\theta_\Delta - 1. \tag{7}$$

The experiments in [10] indicated that the effectiveness of DRT improves with the increase of the values of ε/δ. Thus, set $\varepsilon/\delta = r_M + (r_\Delta - r_M)*K$, where $r_M = 1/\theta_M - 1$, $r_\Delta = 1/\theta_\Delta - 1$. The effectiveness of DRT is configured in a higher level by setting K as 0.8. Conservatively, ε is set relatively small (ε=0.05). In this paper, θ_M and θ_Δ are generated from true value of defect detection rates.

Initial profile $\{p_1, p_2, \ldots, p_m\}$ and initial variables a_i, b_i of Bayesian estimation are determined by test engineers' previous experience. All the previous information is neglected in this paper. Therefore, both a_i and b_i are set to 1, and p_i should be $1/m$. That means the selection probabilities of all subdomains are same. In effect, due to the test profile updated dynamically during the testing process, DRT and O-DRT allow an arbitrary initial condition. If the initial condition is favorable, the effectiveness of DRT and O-DRT can be further improved.

Assume that the estimated value $\widehat{m\theta}$ may have some errors, which is used to normalize the estimated values of defect detection rates and their variance. Thus the value of $\widehat{m\theta}$ is set as twice as the maximal defect detection rate in practice, and the estimation of its variance $\widehat{mV}$ is calculated by:

$$\widehat{mV} = \max\left(\hat{\theta} * \left(1 - \hat{\theta}\right)\right), \quad \hat{\theta} \epsilon \left[0, \widehat{m\theta}\right]. \tag{8}$$

As the O-DRT is proposed to increase the DRT, set R<1 (R=1 means the maximum value of the objective function) in experiments. In fact, the upper bound of normalized defect detection rates $\hat{\theta}_i'$ is 0.5 rather than 1 due to the assumption about $\widehat{m\theta}$. In this case, the range of the objective function is [-0.5, 1]. Then, R is set as 0.6 tentatively, which based on the assumption of both moderate DRT's effectiveness and the lower frequency of the optimization. This assumption also can be validated in experiments. The reason for limited frequency is that the computational overhead is considered. Besides, the frequency of the optimization and the computational overhead in the O-DRT are also affected by experimental subjects.

In *step 9*, the test profile optimization is limited to be conducted at most one time in q selection for two reasons. One is the optimized test profile also cannot meet the pre-defined expectation in some cases, therefore successive adjustments may not work and result in waste of time. The other one is to avoid frequent readjustments, which may cause more random errors. In this paper, q is set 5 tentatively.

IV. EXPERIMENT SETUP

A. Subject Programs

TABLE I. SUBJECT PROGRAMS

Subject Programs	Size (LOC)	Versions	# of Defects	# of Test Cases	# of Class
bash	59,846	v1-v2-v3-v4-v5-v6	5-3-6-4-4-9	1061	5
flex	10,459	v1-v2-v3	16-13-9	525	5
grep	10,068	tsl1:v1-v2-v3-v4	3-2-7-3	477	6
		tsl2:v1	3	148	7
gzip	5,680	v1-v2-v3-v4	5-3-3-4	214	4
make	35,545	v1-v2	4-5	795	5

The experiments are conducted on five-real subject programs (see Table Ⅰ). The applications are commonly used as UNIX utilities developed by GNU. For each subject program, a testing environment is developed to automate experiments. There are some modified versions and tests suites for each program, except several versions without a defect or test case, and all defects are injected by software developers.

B. Test Cases

All specification-based tests were written by TSL language [18], which is a category-partition method for creating functional test suites. Based on the various characteristics of the parameters, the initial test suite was divided into several subdomains.

For *bash*, test suites were divided into classes according to the partitions in its manual; for *flex*, different settings of the parameter "fast scanner with table" and "align" were allocated to each classes; for *greptsl1*, each subdomain involved different settings of the parameter "miscellaneous"; for *greptsl2*, this partition relied on the form of quoting pattern and whether pattern occurrence on target line; for *gzip*, it depended on the type of input file; for *make*, the use of "-s" or "–k" flag was the criterion of partition.

C. Testing Strategies

The three testing strategies are used in the experiments, i.e., RPT, DRT and O-DRT. In the course of these three test strategies, 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. The initial settings of ε and δ in DRT are the same as the O-DRT initial settings (shown in Ⅲ.*B*).

D. Measures

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

1) T, The number of test cases required to detect and remove all defects. The mean of T is denoted by $\bar{T}$.

2) D, The standard variance of T over 1000 iterations. The mathematical definitions of $\bar{T}$ and D are given by

$$\bar{T} = \frac{1}{1000}\sum_{i=1}^{i=1000} T_i \text{ , } \mathrm{D} = \sqrt{\frac{1}{999}\sum_{i=1}^{i=1000}(T_i - \bar{T})^2}. \quad (9)$$

3) p-value: The *p-value* indicates whether there is a significant difference between the efficiency of two strategies. If the *p-value* is smaller than or equal to the significance level (traditionally 5%), it suggests that there is the significant difference between two strategies.

4) effect size: The *effect size* is a quantification of the strength of a strategy better than another one. In the literature [19], a nonparametric effect size $\hat{A}_{12}$ was presented. The formula of $\hat{A}_{12}$ is

$$\hat{A}_{12} = (R_1/m - (m+1)/2)/n. \quad (10)$$

where R_1 is the rank sum of the former data group, m and n are the number of observations in the former and later data sample. The range of $\hat{A}_{12}$ is [0, 1], and it reflects the extent of strength or shortcoming. If $\hat{A}_{12}$ =0.5, it means the two strategies are equivalent. If $\hat{A}_{12}$> 0.5, it expresses that the former is superior to the later. More details about the *effect size* presented in [19, 20]. In the experiments, the values of $\hat{A}_{12}$ for DRT and O-DRT were calculated by these two strategies being compared with RPT, respectively.

E. Experiment Results

Table Ⅱ shows the experimental results and comparisons among RPT, DRT and O-DRT in terms of $\bar{T}$, D, *p-value* and $\hat{A}_{12}$.

TABLE II. EXPERIMENTAL RESULTS

Software Subject	Testing Strategies	$\bar{T}$	D	*p*-value with RPT	*p*-value with DRT	$\hat{A}_{12}$ with RPT
bashv1	RPT	2916.6	2146.2			
	DRT	2457.6	1968.3	0.000		0.570
	O-DRT	1667.2	953.9	0.000	0.000	0.680
bashv2	RPT	1071.9	1007.0			
	DRT	1010.0	890.6	0.466		0.509
	O-DRT	835.0	676.8	0.000	0.001	0.551
bashv3	RPT	771.0	685.7			
	DRT	49.0	50.3	0.214		0.968
	O-DRT	44.5	49.8	0.000	0.023	0.970
bashv4	RPT	1973.4	1931.5			
	DRT	1805.1	1781.1	0.038		0.527
	O-DRT	1255.7	937.2	0.000	0.000	0.586
bashv5	RPT	1820.4	1753.3			
	DRT	1719.3	1738.8	0.409		0.511
	O-DRT	1238.7	1045.9	0.000	0.000	0.577
bashv6	RPT	193.0	205.7			
	DRT	180.1	176.4	0.517		0.508
	O-DRT	182.1	181.4	0.754	0.740	0.504
flexv1	RPT	36.4	31.4			
	DRT	35.3	31.3	0.382		0.511
	O-DRT	40.7	56.6	0.399	0.966	0.511
flexv2	RPT	188.2	141.1			
	DRT	175.7	143.1	0.001		0.541
	O-DRT	183.3	167.9	0.001	0.904	0.541
flexv3	RPT	235.5	160.9			
	DRT	201.9	151.3	0.000		0.576
	O-DRT	121.0	74.0	0.000	0.000	0.755
greptsl1v1	RPT	38.6	36.3			
	DRT	40.5	39.6	0.483		0.491
	O-DRT	36.8	40.0	0.021	0.003	0.530
greptsl1v2	RPT	91.3	91.8			

	DRT	84.0	83.3	0.271		0.514
	O-DRT	76.6	82.0	0.001	0.027	0.543
greptsl1v3	RPT	98.6	93.6			
	DRT	102.1	100.5	0.802		0.497
	O-DRT	114.3	149.4	0.940	0.799	0.501
greptsl1v4	RPT	345.9	252.5			
	DRT	335.2	254.3	0.191		0.517
	O-DRT	252.7	168.8	0.000	0.000	0.607
greptsl2v1	RPT	194.0	191.2			
	DRT	181.4	171.9	0.579		0.507
	O-DRT	140.0	123.8	0.000	0.000	0.566
gzipv1	RPT	196.5	136.1			
	DRT	153.1	98.7	0.000		0.593
	O-DRT	130.2	69.6	0.000	0.000	0.645
gzipv2	RPT	10.8	10.3			
	DRT	11.1	10.3	0.910		0.499
	O-DRT	11.1	11.8	0.102	0.088	0.521
gzipv4	RPT	192.1	136.2			
	DRT	159.4	116.7	0.000		0.574
	O-DRT	122.4	83.9	0.000	0.000	0.666
gzipv5	RPT	460.2	376.0			
	DRT	440.1	376.2	0.133		0.519
	O-DRT	377.3	309.2	0.000	0.000	0.569
makev1	RPT	55.6	44.2			
	DRT	51.4	38.3	0.182		0.517
	O-DRT	44.1	29.0	0.000	0.002	0.557
makev2	RPT	76.7	67.7			
	DRT	63.2	54.2	0.000		0.559
	O-DRT	52.1	36.1	0.000	0.001	0.604

V. Analysis and Discussion

A. Data Analysis

1) Both DRT and O-DRT outperform RPT in the aspect of *p-value* and *effect size*. O-DRT is better than RPT in 16 out of 20 SUTs with a higher *effect size*. There are no significant difference between O-DRT and RPT for remainder SUTs (*bashv6*, *flexv1*, *greptsl1v3*, *gzipv2*). As for DRT, it is better than RPT; however, its advantage over RPT is less than that of O-DRT. Compared with DRT, O-DRT is a excellent one in 15 out of 20 by evaluating the *p-value* and *effect size*. O-DRT is also at least the same as DRT in effectiveness. In terms of *D*, the proportions of O-DRT and DRT outperforming RPT are 15/20 and 14/20, respectively. Moreover, the proportion of O-DRT outperforms DRT based on *D* are 14/20. Hence O-DRT is more effective and robust than DRT and RPT in most real testings. The reasons for those "exceptional" subjects will be discussed in *2)*.

2) Compared with RPT and DRT, O-DRT has no distinct advantage under two circumstances. (1) There is not enough information collected on-line. From the experiment results, for *flexv1* and *gzipv2*, the total number of test cases consumed by RPT are 36.4 and 10.8, respectively. The total number of test cases in these two subjects are 525 and 214, respectively (seen from the Table Ⅰ). Hence the test cases revealing all the defects are less than 7% (i.e. 36.4/525 or 10.8/214). Since all defects can be revealed easily, the information collected on-line is quite limited. Moreover, time is not enough to show the advantage of DRT and O-DRT on such programs. The fault matrix of *bashv6* implies that one and only one test case is able to detect all the nine defects. Under this circumstance, the information about detected defect is not used. The modes of the test profile adjustment in O-DRT and DRT, according to the knowledge of undetected defects, are similar to that in RPT. These two examples are unusual in practice. (2) The test profile changed dramatically during the testing process. For *greptsl1v3*, its five "easy" defects are probably detected and removed at first, but only one subdomain can detect the "difficlut" defect, which is probably the last detected one. Therefore, after removing such "easy" defects, the real defect detection rates will change a lot. As for *flexv2*, its features are similar to those of *greptsl1v3*. In addition, the effectiveness of DRT is the same as RPT in both cases, which also impacts the effectiveness of O-DRT. Both cases are caused by the characters (partition and fault matrix) of SUTs. Though the effect of such characteristics on the effectiveness of strategies can not be confirmed according to current experimental results, O-DRT should be at least as effective as RPT and DRT, and much more effective in most experiments, which also confirmed conclusions *1)* in this section.

3) These three strategies were compared in terms of variance by Levene's Test. The results are shown in Table Ⅲ. In Table Ⅲ ">" or "<" denotes that the former strategy in the first column is better or worse than the latter based on variance, "=" represents that there is no significant difference between these variances of two strategies. The numbers in the table present the number of subject programs.

TABLE III. The Result of Levene's Test

	>	=	<
O-DRT vs. RPT	13	5	2
DRT vs. RPT	5	15	0
O-DRT vs. DRT	12	4	4

As seen from the Table Ⅲ, the variance of O-DRT is better than RPT and DRT, and RPT is similar to DRT in terms of variance. O-DRT is worse than RPT based on variance for *flexv1*, *flexv2*, *gzipv2* and *greptsl1v3*. The major factors are the features of SUTs, which were discussed in 2).

4) Considering the effectiveness and robustness of these three strategies, the *O*-DRT is a good alternative. If the test cases consumed by RPT is not very few (>50) or the percentage of defects detected by test cases in most partitions on the total number of defects is not that high (<50%), the effectiveness of O-DRT will be better. The former indicates O-DRT has obvious advantage of detecting "hard-to-reveal" defects, and the latter might be affected by different category-partition criteria and test cases.

B. Computational Overhead

There are several testing methods (AT, DRT, etc.) that can improve the effectiveness of RPT, while they increase the computational overhead in RPT. AT-200 serves 200 recent data (rather than using all history in AT) as the feedback to reduce the computational overhead of AT. Table Ⅳ records computational overhead of five strategies on *bashv1*, where the difference among various computational overhead can be observed clearly by the large number of test cases required. To avoid the bias brought by the subject, the recorded time does not include the execution time of test cases.

TABLE IV. COMPUTATIONAL OVERHEAD（BASHV1）

	RT	RPT	DRT	O-DRT	AT-200
$\overline{T}$	1676.6	2916.6	2457.6	1667.2	799.47
Total(ms)	1.4972	47.528	49.035	209.929	973692.6
Each test case(ms)	0.0008	0.01629	0.01995	0.12592	1217.9

This experiment is executed on a 32-bit Windows 7 PC, which is powered by a 3.20GHz Intel Core i5-3740 Quad processor and has 4GB RAM. These test programs are written by MATLAB. As shown in Table Ⅳ, the time of each test case selection in O-DRT is approximately 10 times as long as that in DRT. However, the total time to execute test cases in O-DRT is less than 5 times as long as that in DRT, because the value of $\overline{T}$ in O-DRT is much smaller than that in DRT. In terms of computational complexity, O-DRT is worse than DRT; while, O-DRT is far superior than AT-200. Besides, the execution time of test cases cannot be ignored in practice, in which case the selection time of test cases may be less important. Therefore the computational overhead of O-DRT is acceptable.

VI. CONCLUSIONS AND FUTURE WORKS

DRT is a feedback-based software testing strategy, which has been proved to be more effective than RT and RPT strategies. Since the frequency and range of the test profile adjustment are fixed during the testing process, DRT might not timely react to the changes of defect detection rates. The O-DRT is proposed to overcome such shortcomings in this paper. In O-DRT, the test profile adjustment contains two parts. Besides of the original adjustment in DRT, O-DRT changes the test profile to a theoretically optimal one when the give criterion is satisfied. This optimization is obtained by maximizing the overall defect detection rate and minimizing its variance. Experiments on five real-life software subjects are conducted to validate the effectiveness of O-DRT. Experimental results on five SUTs show that O-DRT outperforms RPT and DRT strategies in 15 out of 20 testing scenarios. Even in the other 5 SUTs with special characters, O-DRT is still as effective as DRT and RPT. Though the time of each test case selection in O-DRT is approximately an order of magnitude higher than that in DRT, it is more acceptable since the value of $\overline{T}$ in O-DRT is much smaller than that in DRT. Besides, the execution time of test cases might not be ignored in practice, in which case the selection time of test cases may be less important. In conclusion, O-DRT could be a good choice to improve the effectiveness of DRT and RPT.

Future work will include the analysis of the adjusted criterion and the objective function, which determine the frequency and efficiency of adjustment. The computational overhead can also be reduced by improving the adjusted criterion and the objective function. In addition, the impact of parameter settings deserves further research.

REFERENCES

[1] W. J. Gutjahr, "Partition Testing vs. Random Tesing: The Influence of Uncertainty,"IEEE Transactions on Software Engineering, vol. 25, issue 5 pp: 661-674, September/October 1999.

[2] T. Y. Chen and Y. T. Yu, "A more general sufficient condition for partition testing to be better than testing," Information Processing Letters, vol. 57, issue 3, pp. 145-149, February 1996.

[3] E. J. Weyuker and B. Jeng, "Analyzing Partition Testing Stragegies,"IEEE Transactions on Software Engineering, vol 17, no. 7, pp.7.3-711,July 1991.

[4] K. Y. Cai, B. Gu, H. Hu, and Y. C. Li, "Adaptive Software Testing with Fixed-Memory Feedback," Jounal of Systems and Software, vol.80, issue 8, pp.1328-1348, August 2007.

[5] K. Y. Cai, T. Jing, and C. G. Bai, "Partition Testing with Dynamic Partitionint," in Proceedings of the 29th COMPSAC, Edinburgh, Scotland, July 2005, pp. 113-116.

[6] G. B. Finelli, "NASA Software Failure Characterization Experiments," Reliability Engineering and System Safety, vol 32, pp. 155-169, 1991.

[7] A. G. Koru, K. E. Emam, D. S. Zhang, H. Liu, D. Mathew, "Theory of relative defect proneness," Empirical Software Engineering, vol.13, no. 5, pp. 473-498, 2008

[8] K. Y. Cai, H. Hu, C.H. Jiang, and F.Ye, "Random testing with dynamically updated test profile," in Proceedings of the 20th ISSRE 2009, Fast Abstract 198, Mysuru, Karnataka, India, November 2009.

[9] K. Y. Cai, "Optimal software testing and adaptive software testing in the context of software cybernetics,"Information and Software Technology,vol 44, pp. 841-855, November 2002.

[10] Junpeng Lv, Hai Hu, and Kai Yuan Cai, "A sufficent condition for parameters estimation in Dynamic Random Testing," International Computer Software and Applications Conference, 2011 IEEE 35th Annual, Munich, 2011, pp. 19-24.

[11] L. Zhang, B.B. Yin, J.P. Lv, K.Y. Cai, S.S. Yau, and J. Yu, "A History-Based Dynamic Random Software Testing," International Computers,Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th Annual, 2014, pp. 31-36.

[12] Z.J. Yang, B.B. Yin, J.P. Lv, K.Y. Cai, S.S. Yau, and J. Yu, "Dynamic Random Testing with Parameter Adjustment," COMPSACW, 2014 IEEE 38th Annual, 2014, pp.37-42.

[13] K.W. Miller, et al, "Estimating the Probability of Failure When Testing Reveals No Failures." IEEE Transactions on Software Engineering, vol. 18, issue 1, pp: 33-43, January 1992.

[14] M. Z. Tsoukalas, J. W. Duran and S. C. Ntafos, "On some reliability estimation problems in random and partition testing," IEEE Transactions on Software Engineering, vol. 19, issue 7 pp:194 – 201, July 1991.

[15] J. P. Lv, B. B. Yin, and K. Y. Cai, "On the Asymptotic Behavior of Adaptive Testing Strategy for Software Reliability Assessment", Software Engineering, IEEE Transactions on, vol 40, issue: 4, pp. 396 – 412, April 2014

[16] K. Rekab, H. Thompson, W. Wei, "A Multistage Sequential Test Allocation for Software Reliability Estimation", Reliability, IEEE Transactions on, vol. 62, issue 2, pp. 424 – 433, June 2013

[17] Z. Benkamra, T. Mekki, and T. Mounir, "Bayesian sequential estimation of the reliability of a parallel-series system," Appl. Math. Comput., vol. 219, no. 23, pp. 10842–10852, Aug. 2013.

[18] T. J. Ostrand, M. J. Balcer, "The Category-Partition Method for Sepcifying and Generating Functional Tests," Communications of the ACM, vol. 331, no. 6, pp. 676-686, June 1988.

[19] A. Arcuri, Andrea, and L. Briand, "A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering." Software Testing, Verification and Reliability,vol. 24, issue 3 pp: 219-250, May 2014.

[20] A. Vargha and H. D. Delaney, "A critique and improvement of the CL common language effect size statistics of McGraw and Wong," Jonural of Educational and Behavioral Statistics, vol.25, no.2, pp. 101-132,2000