2011 35th IEEE Annual Computer Software and Applications Conference Workshops

A Sufficient Condition for Parameters Estimation in Dynamic Random Testing*

Junpeng Lv, Hai Hu, Kai-Yuan Cai
Department of Automatic Control
Beijing University of Aeronautics and Astronautics
Beijing 100191, China
Email: reallјp@buaa.edu.cn

Abstract – **Dynamic Random Testing (DRT) is a testing strategy to improve traditional random testing and random-partition testing strategies. The essential idea of DRT is that the testing profile, based on which test classes are chosen to select next test case, should be dynamically adjusted according to the testing history. DRT can have various forms depending on parameters settings, application scenarios and test suite partitioning schemes. This paper discusses the theoretical boundary of the parameters in DRT and their impact on the actual effectiveness in practice. A sufficient condition for the parameters estimation in DRT is proposed and further examined through experiments on three subject programs. Data indicates that the proposed sufficient condition provides a reasonable guideline for choosing parameters when applying the DRT strategy.**

Keywords-Dynamic Random Testing; parameters estimation

I. INTRODUCTION

Random testing and partition testing are two major forms of software testing. In traditional random testing (RT) [5], the test suite/input domain of the software under test comprises a large number of distinct test cases and each time test cases are selected and executed in accordance with a uniform or non-uniform probability distribution. No partitioning is applied to the test suite. On the other hand, partition testing [4] tries to partition the input domain of the software under test into a number of disjoint or overlapping subdomains and one or more test cases are generated from each subdomain. The resulting test suite comprises a modest number of test cases, which are exhaustively executed against the software under test.

By combining these two testing strategies, a new testing strategy named random-partition testing (RPT), is proposed [8, 9]. Suppose the input domain of the software under test is partitioned into m subdomains $C_1, C_2, \ldots, C_m$, RPT first selects a subdomain according to a testing profile $\{p_1, p_2, ..., p_m\}$, where p_i denotes the probability that subdomain C_i is selected, then a test case from the selected subdomain is randomly chosen for execution.

Dynamic random testing (DRT) is a software testing strategy proposed to improve traditional random testing and random-partition testing strategy [3]. The essential idea of DRT is the testing profile $\{p_1, p_2, ..., p_m\}$ should not be fixed but needs to be dynamically adjusted according to testing history during test execution. DRT can have various forms depending on parameters settings, application scenarios and test suite partition. In our previous study [3], a simple parameter setting is adopted and experiments on two real-life subject programs are reported indicating that DRT outperforms RT in terms of defect removal efficiency.

However, from the experimental data collected in the previous case studies, several unanswered questions arise:

- Defect removal is immediate in the previous case studies. A more commonly adopted defect removal scheme is batch debugging, which removes several defects simultaneously after a certain number of failures have been observed or after a certain number of test cases have been executed. Will DRT manage to maintain its advantage over RT if batch debugging is adopted?
- While we let the increase/decrease on the probability of selected class after the execution of one test case be the same, i.e. ε in [3], DRT outperforms RT in terms of defect removal efficiency. One may wonder whether it is a necessary condition required by DRT to outperform RT, or whether there are any boundary values for the increase, i.e. ε, and the decrease, i.e. δ, to guarantee DRT's advantage.

In this paper, we followed a theoretical-experimental approach to investigate the above two questions. More specifically, theoretical analysis is carried out to identify the boundary values of the parameters ε and δ. Then the impact of batch debugging is investigated by experiments on three real-life subject programs. Moreover, experiments with various combinations of ε and δ are done to verify whether the boundary values for ε and δ are of practical meaning. Experimental data indicates that the impact of parameter settings are notable and the boundary values can provide a basic guideline for choosing ε and δ in practice.

The rest of this paper is organized as follows: related studies are presented in Section II. Section III gives a brief introduction of the DRT strategy and formulates the problem investigated in this paper. The theoretical analysis part of this paper is presented in Section IV and Section V reports experiments setup on three real-life subject programs. Experiment results and some general discussions are provided in Section VI. Conclusions and future studies are presented in Section VII.

*Supported by the National Natural Science Foundation of China (Grant Number 60973006) and the Beijing Natural Science Foundation (Grant Number 4112033)

II. RELATED STUDIES

The effectiveness of random testing and partition testing are compared and the viability of random testing are confirmed [1, 5, 10]. Gutjahr et al. [2] compared partition testing and random testing by modeling the failure rates with random variables. Weyuker et al.[4] investigated the conditions that affect the efficacy of partition testing, and compared the fault detection capabilities of partition testing and random testing. Hamlet and Taylor improved the negative results published about partition testing[11]. Bonland et al. [12] investigated the relative effectiveness of partition testing versus random testing through the technique of majorization.

Cai et al. [8, 9] proposed the random-partition testing to combine random testing and partition testing to take advantages of both forms. Chen et al. [6] examined analytically the possible effect of failure patterns on software testing and obtained an upper bound on the potential effectiveness improvement attainable by assuming some extra knowledge about the regions. Chan et al.[7] presented Restricted Random Testing (RRT), which offered a significant improvement over random testing by the F-measure. Cai et al.[3] followed the idea of software cybernetics, which is aimed to explore the interplay between software and control, and proposed Dynamic Random Testing, in which feedback mechanism is used to adjust the testing profile during the test process.

III. PROBLEM FORMULATION

In this section, we first revisit the DRT strategy proposed in [3] and then formulate the boundary values of the parameters using mathematical deductions.

Suppose the input domain or the test suite is divided into m subdomains or classes C_1, C_2, …, C_m, which may be disjoint or overlapped. Each class contains k_i distinct test cases. Dynamic Random Testing is executed as two steps: first, a test class C_i is selected with probability p_i; second, a test case is selected from class C_i for execution uniformly, that is with probability $1/k_i$. After the execution of a test case, there are two scenarios: one is this test case detects a defect; the other is the test case is executed without any failure. For the first condition, we consider that this class C_i has a higher probability of causing a failure, so an increment, i.e. ε, should be added to p_i; on the contrary, the second condition implies that this class may cause less failure, so p_i should reduce to an extent, i.e. p_i-δ. More specifically, we have the following strategy:

- *DRT Strategy*

Step 1 partition the test suite into m classes C_1, C_2, …, C_m,which comprises k_1, k_2, …, k_m distinct test cases respectively; $k_1 + k_2 + \cdots + k_m \geq k$, where k is the number of test cases contained in the test suite;

Step 2 initialize the number ε, δ and $\varepsilon > 0, \delta > 0$.

Step 3 initialize the testing profile $\{p_1(0), p_2(0), ..., p_m(0)\}$.

Step 4 select a class in accordance with the testing profile;that is the probability of class C_i being selected is p_i; suppose the C_l class is selected;

Step 5 select a test case from class C_l at random in accordance with a uniform distribution, that is each distinct test case in the lth class has an equal probability of $1/k_l$ being selected;

Step 6 the selected test case is executed against the software under test and the test result is recorded;

Step 7 when the test case from class C_l detects a defect, then

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

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

Step 8 when the test case from class C_l did not detect a defect, then

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

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

Step 9 the executed test case is returned into C_l, the partition and the test suite keeps invariant;

Step 10 the given test criterion is checked; if it is not satisfied, go to step 4;

Step 11 the testing process is stopped.

According to the above strategy, several factors affect DRT: the number of test case classes, the initial testing profile and two parameters ε and δ used in the DRT strategy. Among these factors, the former two are external factors determined by test engineers. ε and δ are internal parameters of the DRT strategy needed to be properly set to achieve improved testing performance. In the previous study [3], the parameters are simply set as ε=δ=0.05. The DRT strategy investigated in this paper adopts a different parameter setting by assigning different values to ε and δ. The major motivation is to investigate what their impacts on the testing performance are.

- *Assumptions*

In order to make the problem mathematically tractable, we have following assumptions:

(1) The test suite has an overall failure detection rate θ, which is the probability of detecting a failure by randomly selecting a test case from the test suite that is not partitioned. Further let the failure detection rate of each class of partitioned test suite be θ_i (i=1, 2,…, m). This could be arguable because in practice we cannot find out their precise values. In this paper, the failure detection rates are used just for theoretical analysis and in practice they can be estimated. The robustness of DRT on this assumption needs futher study.

(2) No matter how the test suite is partitioned, there is at least one test class that has the highest failure detection rate θ_M, whereas $\theta_M > \theta$ and $\theta_M \geq \theta_i$ ($i \neq M$).

(3) The failure detection rates θ_i (i=1, 2,…, m) remain invariant, that is, no defect removal is conducted during

testing. As any defect removal may cause an unclear impact on the failure detection rates, any particular assumption on such impact will not be sufficient to reflect the situations encountered by test engineers in practice. The impact of defect removal is not included in the problem formulation.

(4) After a test case is selected and executed, it is returned into its original class, that is, test case replacement is enabled in this study. It is true that the same test case may be selected again in the test process. However, we do *not* consider selected test cases will detect any other defect no matter whether they have detected one or not. This is also required to guarantee assumption (1).

For DRT, the expected failure detection rate is

$$\theta_{DRT} = \sum_{i=1}^{m} p_i^{DRT} \theta_i ,$$

where p_i^{DRT} $(i = 1, 2, \cdots, m)$ is dynamically updated.

Recall that there is at least one test case class C_M with the highest failure detection rate θ_M. The rationale behind the DRT strategy is that subdomains or classes that have higher failure detection rates should also have higher probability to be selected. On the other hand, the value of ε and δ will affect the evolution of the testing profile as testing proceeds. Therefore, a natural criterion to determine whether ε and δ are properly set is whether they lead the DRT strategy to select the class C_M. In other words, if as testing proceeds, the probability of selecting C_M, denoted by p_M, increases, θ_{DRT} will also increase. An extreme case is when the probability of selecting C_M is maximized, that is $p_M = 1$, then we have $\theta_{DRT} = \theta_M$. In this case, DRT performs best without failure detection rates changes.

To sum up, the problem of determining appropriate parameters ε and δ is now transformed into an inequation problem: whether the expectation of the probability of selecting class C_M, i.e. p_M, is increasing as testing proceeds:

$$E(p_M(n+1)) > p_M(n)$$

where n denotes the number of test cases executed.

IV. THEORETICAL ANALYSIS

Suppose that we started with the initial testing profile $\{p_1(0), p_2(0), ..., p_m(0)\}$, and n test cases have already been executed. The testing profile has evolved to $\{p_1(n), p_2(n), ..., p_m(n)\}$. Without considering the situation that $p_i < \varepsilon$ or $p_i < \delta$ $(i = 1, 2, \ldots, m)$, which is also very rare in practice, then generally the expected value of the testing profile after the execution of the next test case is as follows:

$$\begin{aligned} E(p_i(n+1)) &= p_i(n)\theta_i(p_i(n)+\varepsilon) + p_i(n)(1-\theta_i)(p_i(n)-\delta) + \\ &\sum_{j\neq i} p_j(n)\theta_j(p_i(n) - \frac{\varepsilon}{m-1}) + \sum_{j\neq i} p_j(n)(1-\theta_j)(p_i(n) + \frac{\delta}{m-1}) \\ &= p_i(n) + Y_i(n) \end{aligned}$$

where $Y_i(n) = \frac{1}{m-1}(1 - mp_i(n))\delta + (p_i(n)\theta_i - \frac{1}{m-1}\sum_{j\neq i} p_j(n)\theta_j)(\delta+\varepsilon)$

$$\because \sum_{i=1}^{m} E(p_i(n+1)) = 1 \therefore \sum_{i=1}^{m} Y_i(n) = 0 \qquad (1)$$

And $\forall i \in \{1, 2, \cdots, m\} \,\&\, i \neq M$

$$Y_M(n) - Y_i(n) = \frac{m}{m-1}(p_i(n) - p_M(n))\delta + \frac{m}{m-1}(p_M(n)\theta_M - p_i(n)\theta_i)(\delta+\varepsilon)$$

According to Equation (1), if it always holds that $Y_M(n) > Y_i(n)$ for any $i \in \{1, 2, \cdots, m\} \,\&\, i \neq M$, then we guarantee $Y_M(n) > 0$, so that $E(p_M(n+1)) > p_M(n)$.

Note that $Y_M(n) - Y_i(n) > 0$

$$\Leftrightarrow (p_i(n) - p_M(n))\delta > (p_i(n)\theta_i - p_M(n)\theta_M)(\delta+\varepsilon) \qquad (2)$$

There are still three scenarios, which should be considered to guarantee $Y_M(n) - Y_i(n) > 0$:

- $p_i(n) = p_M(n)$

In this scenario, there is some class C_i whose selected probability, i.e. $p_i(n)$, is equal to $p_M(n)$. In this case, $Y_M(n) - Y_i(n) > 0$ is satisfied when $\varepsilon > 0, \delta > 0$ according to (2). This means that in this scenario, $\varepsilon > 0, \delta > 0$ is required.

- $p_i(n) > p_M(n)$

There may still exist some other class C_i whose selected probability, i.e. $p_i(n)$, is larger than $p_M(n)$. According to (2), a sufficient condition is proposed,

$$(p_i(n) - p_M(n))\delta > (p_i(n) - p_M(n))\theta_i(\delta+\varepsilon) \qquad (3)$$

because $(p_i(n) - p_M(n))\theta_i(\delta+\varepsilon) > (p_i(n)\theta_i - p_M(n)\theta_M)(\delta+\varepsilon)$.

According to condition (3), we can reach the simplified sufficient condition as follows:

$$\frac{\varepsilon}{\delta} < \frac{1}{\theta_i} - 1 \qquad (4)$$

In this scenario, $Y_M(n) - Y_i(n) > 0$ is satisfied as long as the sufficient condition (4) is promised.

- $p_i(n) < p_M(n)$

The remaining classes are ones whose selected probabilities are less than $p_M(n)$. In this scenario, a sufficient condition to keep (2) satisfied is as follows:

$$(p_M(n) - p_i(n))\theta_M(\delta+\varepsilon) > (p_M(n) - p_i(n))\delta \qquad (5)$$

because $(p_M(n)\theta_M - p_i(n)\theta_i)(\delta+\varepsilon) > (p_M(n) - p_i(n))\theta_M(\delta+\varepsilon)$

So the simplified condition for in equation (5) is

$$\frac{\varepsilon}{\delta} > \frac{1}{\theta_M} - 1 \qquad (6)$$

Inequation (6) is a sufficient condition to satisfy $Y_M(n) - Y_i(n) > 0$ for this scenario.

To sum up, one sufficient condition by combining (4) and (6) for $E(p_M(n+1)) > p_M(n)$ is as follows:

CONDITION I: For all $\theta_i (i \in \{1, 2, \cdots, m\} \,\&\, i \neq M)$, $E(p_M(n+1)) > p_M(n)$ is guaranteed as long as

$$\frac{1}{\theta_M} - 1 < \frac{\varepsilon}{\delta} < \frac{1}{\theta_\Delta} - 1 \qquad (7)$$

where, $\theta_\Delta = \max\{\theta_i \mid i = 1, 2, \cdots, m, i \neq M\}, \theta_\Delta < \theta_M$.

$R = (1/\theta_M - 1, 1/\theta_\Delta - 1)$, which is the interval that ε/δ lies in.

V. EXPERIMENT SETUP

In this section, experiments on nine versions of three real-life subject programs are reported to validate the sufficient condition presented in Section 4. The objective of the experiments is to verify two hypotheses:

H1. If ε and δ satisfy Condition I, then the failure detection effectiveness of using DRT is better than that of RT.

H2. If ε and δ do *not* satisfy Condition I, then the failure detection effectiveness of using DRT is *no* better than the performance of using DRT under the scenario in which ε and δ satisfy Condition I.

- *Testing Scenarios*

Each test scenario is determined by the following factors: the subject program under test, the test suites, the values of ε and δ, the initial testing profile, the number of test steps, and the values of the failure detection rates. In real life, we do not have precise data associated with the last factor. However, the values of failure detection rates are used to calculate the boundary values of the parameters.

Subject Programs

In our study, three different subject programs are selected for testing: *tcas*, *replace* and *Space*. *tcas* and *replace* are chosen from the Siemens programs while the *Space* program is from NASA.

Test Suites

For each of the three subject programs, both partitioned and unpartitioned test suite are prepared The former test suite is used as the test suite of DRT, and the latter one is for RT. The DRT test suites for *tcas* and *replace* are both partitioned into five disjointed classes randomly. For *Space*, it is partitioned into four disjoint classes at random. Different partition in three subject programs is to validate the effectiveness of DRT under different number of classes.

Failure Detection Rates

In this study, three different scenarios are set for each of the subject programs. Subject programs in these three scenarios are injected with 3, 2 and 1 defect respectively to have three versions of each subject program with different failure detection rates in a descending manner.

The failure detection rates of each class are obtained by running all the test cases in each class against the subject program and computing the ratios of the number of failures observed, respectively, to the total number of test case in that class. Table 1 tabulates the exact failure detection rates of each class.

Parameter Settings

In our study, three parameters need initialization: the initial testing profile, the parameters ε and δ. Recall that, there are no constraints on initial testing profile in the theoretical analysis, so it can be set to any value. Here we set the initial testing profile uniformly, that is, {0.2, 0.2, 0.2, 0.2, 0.2} for *tcas* and *replace* as they both have 5 classes, and {0.25, 0.25, 0.25, 0.25} for *Space* with only 4 classes.

In order to validate the hypotheses, three pairs of the parameters ε and δ are examined: proper, lower and upper, corresponding to ratio between ε and δ that lies within, on the left and on the right of the interval R calculated based on Condition I, respectively. For each of the subject programs, three ratios are selected as shown in Table 2.

Table 1. Subject programs and failure detection rates

Subject programs		**Subdomain failure detection rates**					**Overall failure detection rate θ**
tcas	3 defects	0.01	0.0056	0.021	0.105	0.02	0.033
	2 defects	0.01	0	0.021	0.105	0.02	0.032
	1 defects	0	0	0.021	0.08	0	0.023
replace	3 defects	0.03	0.028	0.041	0.09	0.05	0.046
	2 defects	0.03	0.018	0.037	0.052	0.04	0.036
	1 defects	0	0	0	0.012	0	0.0025
Space	3 defects	0.0042	0.0046	0.0017	0.006		0.0042
	2 defects	0.0017	0.0017	0.0004	0.0038		0.0019
	1 defects	0	0.001	0	0.0008		0.0005

As pointed out in [3], the parameter ε in the DRT strategy has an impact on its performance. According to our experiments before, if the value of ε is too small or too large, the effect of DRT is not as good as DRT with a "proper" ε, i.e. 20%-40% of initial testing profile, because a too small ε will diminish the "benefit" brought by detecting a defect while a too large one will enlarge that. Several experiments with different "proper" ε are set and ε=0.05 is presented in this paper. Table 2 shows the corresponding δ.

Table 2 DRT parameter setting

subject programs		**boundary values R**	**Proper**		**Lower**		**Upper**	
			ratios	**δ**	**ratios**	**δ**	**ratios**	**δ**
tcas	3 defects	(7.5,45.6)	25	0.002000	5	0.010000	100	0.000500
	2 defects	(7.5,45.6)	25	0.002000	5	0.010000	100	0.000500
	1 defect	(10.5,45.6)	25	0.002000	5	0.010000	100	0.000500
replace	3 defects	(9.1,20.1)	15	0.003300	5	0.010000	50	0.001000
	2 defects	(18.1,23.5)	20	0.002500	5	0.010000	50	0.001000
	1 defect	(83.8,535.5)	100	0.000500	50	0.001000	600	0.000083
Space	3 defects	(168.4,215.7)	200	0.000250	100	0.000500	1000	0.000050
	2 defects	(262.4,576.8)	500	0.000100	100	0.000500	1000	0.000050
	1 defect	(1039.0,1184.5)	1111	0.000045	500	0.000100	2000	0.000025

Remarks:

1. The two Siemens programs are relatively small, and the Space program is medium size. Three scenarios for each of the subject program are set in the aspect of different failure detection rates of same subject program. So we can validate the effect of our testing strategy setting.
2. The three defects for each subject program are relatively hard to detect by using the given test suites. For same program, we keep same defects, that means, the defects in the other two versions are some of the defects using in the 3-defects version.
3. The number of test steps has a special impact. If the number is too small, DRT will not get enough information to "learn"; on the contrary, if the number is too large, we will waste some testing resources. In this paper, we set this according to our experience.

- *Testing Strategies and Performance Measurements*

In our experiments, the traditional random testing provides a baseline for performance comparison. We will compare the number of failures encountered after executing same number of test cases.

The performance of testing strategies is measured by three metrics: T, the total number of failures after executing all the test steps; and D, the standard deviation of T over a number of iterations. Each experiment is repeated 40 iterations to obtain the average value of T, denoted by $\overline{T}$, to avoid possible bias. Moreover, because the standard deviation D is closely related to the scale of T, we also included another metric CV to denote the coefficient of variation over 40 iterations. The following equations give the mathematical definitions of $\overline{T}$, D, and CV:

$$\overline{T} = \frac{1}{40}\sum_{i=1}^{40} T_i, \quad D = \sqrt{\frac{1}{39}\sum_{i=1}^{40}(T_i - \overline{T})^2}, \quad CV = \frac{D}{\overline{T}} \times 100\%$$

VI. DATA ANALYSIS & GENERAL DISCUSSIONS

▪ *Data Analysis*

Table 3 (a), (b), (c) give the performance comparison in terms of average value $\overline{T}$, standard deviation D and coefficient of variation CV of the four testing strategies on nine scenarios. The percentages of improvement over RT are calculated by dividing the corresponding data by that of the RT. For example, at column 4 of the third row of Table 3 (a), 40.68% denotes that DRT-Lower detects 40.68% more failures than RT.

Table 3 (a), (b), (c) shows the following observations:

(1) Compared with RT, DRT-Proper provides a clear improvement in $\overline{T}$. More specifically, DRT-Proper has an increase (9.65 to 206.33%) of $\overline{T}$ in all the nine scenarios. This proves that the first hypothesis is right. When considering D, DRT-Proper has an increase (45.18 to 187.64%) in eight out of nine scenarios. This is because that the standard deviation D is closely related to the scale of T. On CV, DRT-Proper has an increase in six out of nine scenarios. This can be explained that in actual testing process the profile does *not* always evolve exactly following the expectation in the theoretical analysis. The increase in D and CV does *not* mean DRT is worthless; in fact, we care $\overline{T}$ much more than D and CV.

(2) Compared with DRT-Proper, DRT-Lower shows a decrease in $\overline{T}$ in all the nine scenarios. This means that if we selected a relatively smaller ε (or a relatively larger δ) other than a "proper" one, we would receive a decrease in the effectiveness of failure detection. Fortunately, from table 3 (a), (b) and (c) DRT-Lower is still better (0.92 to 78.82%) than RT in terms of $\overline{T}$ in all the nine scenarios.

(3) Compared with DRT-Proper, DRT-Upper shows an increase in $\overline{T}$ in seven out of nine scenarios. This can be explained as follows: firstly, the theoretical analysis just supplies a sufficient condition. The first observation shows the theoretical analysis does work, but that does *not* mean there are no other settings to make that DRT outperforms RT; secondly, the other two of nine scenarios indicate that there are indeed values of ε and δ which lie on the right of the interval R but make worse performance than DRT-Proper.

To sum up, from the data analysis, we can have following conclusions:

1. If ε and δ satisfy Condition I, then the failure detection effectiveness of using DRT is better than that of RT, that is, hypothesis H1 is right.
2. If the ratio of ε and δ lies on the left of the interval R, then the failure detection effectiveness of using DRT is no better than that of the scenario in which ε and δ satisfy Condition I; if the ratio lies on the right of the interval R, it has a mixed impact on the failure detection effectiveness of DRT. Hypothesis H2 needs future study.

Table 3 Experimental results (a)

Subject programs	Testing Strategies	average value	Improvement Over RT	standard deviation	Improvement Over RT	coefficient of variation	Improvement Over RT
tcas-3 defects	RT	31.65	-	5.24	-	16.56%	-
	DRT-Proper	83.78	164.69%	13.60	159.50%	16.23%	-1.96%
	DRT-Lower	44.53	40.68%	10.68	103.74%	23.98%	44.82%
	DRT-Upper	86.88	174.49%	11.51	119.71%	13.25%	-19.96%
tcas-2 defects	RT	33.33	-	5.41	-	16.23%	-
	DRT-Proper	84.20	152.66%	14.88	175.12%	17.67%	8.89%
	DRT-Lower	48.08	44.27%	10.81	99.93%	22.49%	38.58%
	DRT-Upper	88.68	166.09%	12.57	132.47%	14.18%	-12.64%
tcas-1 defect	RT	24.05	-	5.11	-	21.24%	-
	DRT-Proper	57.00	137.01%	14.70	187.64%	25.78%	21.37%
	DRT-Lower	30.50	26.82%	8.33	62.99%	27.30%	28.52%
	DRT-Upper	65.73	173.28%	11.19	119.00%	17.02%	-19.86%

Table 3 Experimental results (b)

Subject programs	Testing Strategies	average value	Improvement Over RT	standard deviation	Improvement Over RT	coefficient of variation	Improvement Over RT
replace-3 defects	RT	91.48	-	9.35	-	10.22%	-
	DRT-Proper	148.83	62.69%	17.91	91.52%	12.03%	17.72%
	DRT-Lower	102.23	11.75%	12.21	30.61%	11.95%	16.87%
	DRT-Upper	155.63	70.13%	17.27	84.74%	11.10%	8.59%
replace-2 defects	RT	72.20	-	7.84	-	10.86%	-
	DRT-Proper	79.18	9.65%	11.38	45.18%	14.38%	32.40%
	DRT-Lower	72.87	0.92%	8.42	7.33%	11.55%	6.35%
	DRT-Upper	88.43	22.48%	13.56	72.90%	15.33%	41.17%
replace-1 defect	RT	24.90	-	5.33	-	21.42%	-
	DRT-Proper	76.28	206.33%	14.94	180.14%	19.59%	-8.55%
	DRT-Lower	44.53	78.82%	11.66	118.59%	26.19%	22.24%
	DRT-Upper	73.31	202.45%	15.53	180.83%	21.18%	-1.12%

Table 3 Experimental results (c)

Subject programs	Testing Strategies	average value	Improvement Over RT	standard deviation	Improvement Over RT	coefficient of variation	Improvement Over RT
Space-3 defects	RT	85.83	-	8.47	-	9.87%	-
	DRT-Proper	97.53	13.63%	15.15	78.90%	15.53%	57.44%
	DRT-Lower	89.70	4.52%	8.40	-0.78%	9.37%	-5.07%
	DRT-Upper	96.78	12.76%	15.07	78.01%	15.57%	57.87%
Space-2 defects	RT	38.55	-	5.46	-	14.16%	-
	DRT-Proper	60.13	55.97%	11.23	105.83%	18.69%	31.97%
	DRT-Lower	43.10	11.80%	7.00	28.20%	16.24%	14.67%
	DRT-Upper	62.13	61.15%	11.62	112.82%	18.70%	32.06%
Space-1 defect	RT	8.60	-	3.77	-	43.81%	-
	DRT-Proper	10.80	25.58%	3.72	-1.20%	34.47%	-21.33%
	DRT-Lower	9.18	6.69%	3.52	-6.52%	38.39%	-12.38%
	DRT-Upper	11.88	38.08%	4.42	17.35%	37.23%	-15.01%

▪ *General Discussions*

Based on the theoretical analysis and experimental results, several discussions are presented in this section to discuss the hints we learned from the above analysis.

1. Is it reasonable to have the test without defect removal?

For assumption 1, one may argue that the purpose of software testing is to find defects and remove them. As soon as a defect is removed, the failure defection rates usually change. However, batch debugging is a common strategy, and before a defect is removed, the more information about the bugs we find out, the more convenient it is to debug.

2. Does the way of partitioning the test suite affect the effectiveness of the strategy?

In our theoretical analysis, the way of partitioning only affects the value of θ_i (i=1, 2,…, m). No matter how the detection rate distributes, our DRT strategy with appropriate ε and δ can perform better than RT in terms of number of failures detected by executing same amount of test cases. However, we can find that the improvement over RT for *tcas* and *replace* is larger than that of *Space*. One reason is that the failure rate distributions of the former two subject programs are more skewed than the latter one, which is affected by the partition. This needs further investigation.

3. **Is the theoretical analysis "stable"?**

Condition I aims to provide a sufficient condition to ensure that the class with the largest failure detection rate has a positive increase on the expected value of selecting probability before next test. One may argue whether this method is "stable". Actually, this method only supplies a sufficient condition to find a pair of good parameters for DRT to outperform random testing. Moreover, the stable condition will be considered in future research.

4. **What do we learn from the experiments?**

Firstly, the DRT-Proper *does* improve the effectiveness of failure detection than RT. Despite the *CV* of DRT-Proper is sometimes larger than RT, it is not certain to say that DRT is less stable than RT. Secondly, if we cannot know the failure rates precisely, we still have a chance to have DRT outperform RT, especially when we choose a relative larger ε or a relative smaller δ. The robustness of DRT will be studied in future. Thirdly, a more skewed failure rates distribution may improve the performance of DRT. As shown by *tcas*, *replace* and *Space*, the performance of DRT on the former two subjects are better than that of the latter one. The more skewed the failure detection rates distribution is, the more "benefit" we can get from choosing the class with largest failure rate, which is our goal.

5. **Is DRT still effective if $\theta_i = \theta (i=1, 2,\ldots, m)$ after a partition?**

$\theta_i = \theta(i=1, 2,\ldots, m)$ means the failure detection rates of all classes are equal to each other after the partition. In this case, the probability of detecting a failure for DRT, i.e. θ_{DRT}, is equal to θ. In other words, using DRT is "equal" to randomly selecting a test case from the unpartitioned test suite. Therefore, it is fair to conclude that DRT can perform as well as traditional random testing in this case.

6. **Does the number of test steps affect the effectiveness of DRT?**

The number of test steps has a special impact on the performance of DRT. On one hand, too few test steps may lead to little difference between the performance of DRT and RT because DRT may need more "learning". Take *Space* programs for example, tests for all the three *Space* programs are executed with same number of test cases, but the result of *Space-2 defects* is better than that of *Space-1 defect* in spite that the former one has a less skewed failure distribution than the latter one. On the other hand, too many test steps may force DRT to provide redundant information. For *tcas* and *replace* DRT performs more than 100% better than RT in terms of $\bar{T}$, which may be a redundant condition. These impacts need future investigation.

VII. CONCLUSION & FUTURE WORKS

Dynamic Random Testing is a software testing strategy proposed to use feedback information of failure detections to dynamically update the testing profile during testing. The major advantage of DRT is that the testing profile is never invariant thus test case classes that have higher failure detection rates will have higher probabilities to be selected. However, the effectiveness of DRT is affected by the parameters used to adjust the testing profile. In this paper, a sufficient condition on the parameters choice is obtained through theoretical analysis and experiments are conducted to verify it. Data indicates that in most cases, when these parameters do not satisfy the condition, the improvement of using DRT over RT is reduced, especially when we choose a relative larger δ or a relative smaller ε. On the other hand, for all scenarios with different parameter settings, DRT always outperforms RT regardless of whether the sufficient condition is satisfied. This implies that the DRT strategy is robust in terms of parameters, and engineers can follow the sufficient condition to further fine tune the testing strategy to obtain greater improvement over RT.

Future works on this topic will include the "stable" problem of the theoretical analysis, the robustness of the theoretical analysis, the impact of the value of parameter ε and the "proper" test steps, which determines the minimum of test cases to make DRT outperform RT.

REFERENCES

[1] S. C. Ntafos, "On Random and Partition Testing," in *Proceedings of the 1998 ACM SIGSOFT International Symposium on Software Testing and Analysis*, 23(2), pp. 42-48, Clearwater Beach, Florida, USA, March 1998.

[2] W. J. Gutjahr, "Partition Testing vs. Random Testing: The Influence of Uncertainty," *IEEE Transactions on Software Engineering*, 25(5): 661-674, September/October 1999.

[3] K. Y. Cai, H. Hu, C. H. Jiang, and F. Ye, "Random testing with dynamically updated test profile," in *Proceedings of the 20th International Symposium on Software Reliability Engineering(ISSRE 2009)*, Fast Abstract 198, Mysuru, Karnataka, India, November 2009.

[4] E. J. Weyuker and B. Jeng, "Analyzing Partition Testing Strategies," *IEEE Transactions on Software Engineering*, 17(7): 703-711, July 1991.

[5] J. W. Duran and S. C. Ntafos, "An Evaluation of Random Testing," *IEEE Transactions on Software Engineering*, 10(4): 438-444, July 1984.

[6] T. Y. Chen and R. G. Merkel, "An Upper Bound on Software Testing Effectiveness," *ACM Transactions on Software Engineering and Methodology*, 17(3): Article No. 16, June 2008.

[7] K. P. Chan, T. Y. Chen, and D. Towey, "Restricted Random Testing," *Software Quality - ECSQ 2002*, Helsinki, Finland, June 2002, *Lecture Notes in Computer Science*, vol. 2349, pp. 321-330, 2006.

[8] K. Y. Cai, B. Gu, H. Hu, and Y. C. Li, "Adaptive Software Testing with Fixed-Memory Feedback," *Journal of Systems and Software*, 80(8): 1328-1348, August 2007.

[9] K. Y. Cai, T. Jing, and C. G. Bai, "Partition Testing with Dynamic Partitioning," in *Proceedings of the 29th International Computer Software and Applications Conference*, pp. 113-116, Edinburgh, Scotland, July 2005.

[10] S. C. Ntafos, "On Comparisons of Random, Partition, and Proportional Partition Testing," *IEEE Transactions on Software Engineering*, 27(10): 949-960, October 2001.

[11] D. Hamlet and R. Taylor, "Partition Testing does not Inspire Confidence [Program Testing]," *IEEE Transactions on Software Engineering*, 16(12): 1402-1411, December 1990.

[12] P. J. Boland, H. Singh, and B. Cukic, "Comparing Partition Testing and Random Testing via Majorization and Schur Functions," *IEEE Transactions on Software Engineering*, 29(1): 88-94, January 2003.