An Experimental Evaluation of the Reliability of Adaptive Random Testing Methods

Yu Liu
North China Institute of Computing Technology
Beijing, P. R. China, 100083
Email: liuyu.9855@yahoo.com.cn

Hong Zhu
School of Technology, Oxford Brookes University
Oxford, OX33 1HX, UK
e-mail: hzhu@brookes.ac.uk

***Abstract*—Adaptive random testing (ART) techniques have been proposed in the literature to improve the effectiveness of random testing (RT) by evenly distributing test cases over the input space. Simulations and mutation analyses of various ART techniques have demonstrated their improvements on fault detecting ability when measured by the number of test cases required to detect the first fault. In this paper, we report an experiment with ART using mutants to evaluate ART's reliability in fault detecting ability. Our experiment discovered that ART is more reliable than RT in the sense that its degree of variation in fault detecting ability is significantly lower than RT. It is also recognized from the experiment data that the two main factors that affect ART's reliability are the failure rate of the system under test and the regularity of the failure domain measured by the standard deviation of random test results.**

I. INTRODUCTION

Generally speaking, software testing methods can be classified into two types: random testing (RT) and selective testing, which is also called systematic testing in the literature. Selective testing methods require careful analysis of source code and/or the specification and design documents [1]. In contrast, random testing produces or selects test cases in the input domain through random sampling [2]. It is proven that RT can be as effective as selective testing methods to detect software faults using only a fraction of resource to generate a large number of test cases [3, 4, 5, 6]. It has been an active research topic in software testing [7, 8] and also possesses a niche in IT industry. Recently, adaptive random testing (ART) methods were proposed to further improve the effectiveness and efficiency of random testing [9,10,11,12,13, 14]. This paper is concerned with the comparison of ART to RT.

As Goodenough and Gerhart pointed out [15], a testing method needs to satisfy two criteria: validity and reliability. Informally, a test method's validity means that for all software under test (SUT) there is a test set generated by the method that can detect faults if any. The reliability of a testing method means that if one such test set detects a fault, then ideally any test set of the method will also detect the fault. The reliability of a testing method indicates how stable the testing method is and how consistent the test result can be. In the research on RT and ART methods, focus has been on fault detecting abilities. However, unfortunately, their reliabilities in fault detecting ability have not been investigated systematically.

This paper reports an experiment that uses mutation analysis to compare two ART methods and the traditional RT method to evaluate their reliabilities. Mutation analysis is used because, statistically speaking, mutants systematically generated by the application of a mutation testing tool seem capture the behaviour of real faults very well [16].

The remainder of the paper is organized as follows. Section 2 briefly reviews the basic concepts and related works and defines the notions and notations uses in the paper. Section 3 describes the process and design of the experiment. Section 4 reports the main findings of the experiment. Finally, Section 5 concludes the paper with a summary of the main findings of the experiment, a discussion of the threats to the validity of the conclusions and the directions for future work.

II. PRELIMINARIES AND RELATED WORK

In this section, we review the basic concept and define the notions and notations used in this paper.

A. *Random Testing and Adaptive Random Testing*

Generally speaking, *random testing* (RT) is a software testing method that selects or generates test cases through random sampling over the input domain of the SUT according to a given probability distribution [2]. As discussed in [1], RT techniques can be classified into two types, *representative random testing* and *non-representative random testing*. The former uses the probabilistic distribution on the input domain same as the input distribution in the operation of the SUT. A typical example of such testing techniques is to sample at random in an operation profile of the software under test [17]. Another example is to develop a Markov model of human computer interaction process and to use the model to generate random test cases [18]. In contrast, the non-representative RT uses a distribution irrelevant to the operation of the software. The ART methods studied in this paper belong to this type. Representative RT has the advantage that test results naturally lead to an estimate of software reliability, while the other aims at improve fault detecting ability. ART methods have been proven more effective to detect faults than RT with uniform distribution [14, 21].

The basic idea of ART is to generate and/or select random test cases that spread more evenly over the input space than the ordinary RT. The even spread is achieved through manipulations of randomly generated test sets. A number of such manipulation techniques have been developed and evaluated [9-13]. Before summarizing these manipulations, we first define some basic notions and notations.

Let D be the input domain, i.e. the set of valid input data to the SUT. A *test set* $T=<a_1, a_2, \dots, a_n>$, $n\geq 0$, on domain D is a finite sequence of elements in D, i.e. $a_i\in D$, for all $i=1, 2, \dots, n$.[(a)] The number n of elements in a test set is called the *size* of the test set. A SUT is tested on a test set $T=<a_1, a_2, \dots, a_n>$ means that the software is executed on inputs $a_1, a_2, \dots, a_n$ one by one with necessary initialization before each execution. When $n=0$, the test set is called *empty test*, which means the software is not executed at all. Without loss of generality, we assume that test sets considered in this paper are non-empty.

A test set $T=<a_1, a_2, \dots, a_n>$ is a *random test set* if the elements are selected or generated at random independently according to a given probability distribution on the input domain D. If the same input is allowed to occur more than once in a test set, we say that the random test sets are *with replacement*; otherwise, the test sets are called *without replacement*. In the experiment reported in this paper, the random test sets are with replacement.

Now, we formally define the manipulations on random test sets used in ART techniques.

Mirror. Let $D_0, D_1, D_2, \dots, D_k$ ($k>0$) be a disjoint partition of the input domain D. The sub-domains $D_0, \dots, D_k$ are of equal size and there is an one-to-one mapping μ_i from D_0 to D_i, $i=1, \dots, k$. The sub-domain D_0 is called the *original* sub-domain. The other sub-domains are called the *mirror* sub-domains. The mappings μ_i are called the mirror functions. Let $T^0 =<a^0_1, a^0_2, \dots, a^0_n>$ be a random test set on D_0. The mirror of the original test sets T^0 in a mirror sub-domain D_i through μ_i is a test set T^i, written $\mu_i(T^0)$, defined as follows.

$$\mu_i(<a^0_1, a^0_2, \dots, a^0_n>)=<a^i_1, a^i_2, \dots, a^i_n>,$$

where $\mu_i(a^0_j) = a^i_j$, $i=1, \dots, k$, $j=1, \dots, n$. The *mirror test set* of T^0, written $Mirror(T^0)$, is a test set on D that

$$Mirror(T^0) = <a^0_1, a^1_1, a^2_1, \dots, a^k_1, a^0_2, a^1_2, \dots, a^k_2, \dots, a^0_n, a^1_n, \dots, a^k_n>.$$

Distance. Let $\| x, y \|$ be a distance measure defined on input domain D.[(b)] It is extended to the distance between an element x and a set Y of elements as $\| x, Y\| = min(\|x, y\|, y\in Y)$. A test set $T=<a_0, a_1, a_2, \dots, a_n>$ is *maximally distanced* if for all $i=1, 2, \dots, n$,

$$\| a_i, \{a_0, \dots, a_{i-1}\}\| \geq \| a_j, \{a_0, \dots, a_{i-1}\} \|, \text{ for all } j > i.$$

Let T be any given test set. The *distance* manipulation of T is a re-ordering of T's elements in the sequence so that it is maximally distanced, written $Distance(T)$.

Filter. Let $\| x , y \|$ be a distance measure defined on the input domain D. Let T be any given non-empty test set. Test set $T' =<a_1, a_2, \dots, a_n>$ is said to be *ordered according to the distance* to T, if for all $i=1, 2,\dots, n-1$, $\|a_i, T\| \geq \|a_{i+1}, T\|$. Let S and C be two given natural numbers such that $S > C > 0$, and $T_0, T_1, \dots, T_k$ be a sequence of test sets of size $S>1$. Let $U_0=T_0$. Assume that $U_i=<a_1, \dots, a_{ki}>$ and $T_{i+1}=<c_1, \dots, c_S>$. We define $T'_{i+1}=<b_1, \dots, b_S>$ be a test set obtained from T_{i+1} by permutation its elements so that T'_{i+1} is ordered according to the distance to U_i. Then, we inductively define $U_{i+1}=<a_1, \dots, a_{ki}, b_1, \dots, b_C>$, for $i=0, 1, \dots, k-1$. The test set U_k is called the test set obtained by *filtering C* out of S test cases.

It is worth noting that, first, strictly speaking, the results of the above manipulations are not random test sets even if the test sets before manipulation are random. Second, these manipulations can be combined together to obtain more sophisticated adaptive random testing. For example, in mirror adaptive random testing [10], the input domain is partitioned. A random test set is first generated on the original sub-domain. It is then manipulated by the Distance operation and then Mirrored into the mirror sub-domain. Third, the set of manipulations defined above may not be complete. There are also other manipulations on random test set; see [9-13]. Finally, in the literature, ART methods are usually described in the form of test case generation algorithms rather than manipulation operators. The filter manipulation is not used in the experiments reported in this paper. It is defined here to demonstrate that testing methods defined in the form of an algorithm can also be defined formally as manipulations. This not only simplifies the descriptions of ART testing methods concisely and formally, but also enables us to recognise easily the various different ways in which they can be combined.

B. *Measurements of fault-detecting ability*

A *fault* is a defect in a software system. The execution of faulty software on a test case may result in a *failure*, i.e. the output and/or the behaviour of the software do not meet user's expectation according to a given criterion (which is called test oracle). When a failure occurs, the test detects a fault in the SUT. Testing methods and techniques differ from each other in the way that test cases are selected as well as the correctness of the execution is judged. In this paper, we focus on the issue related to test case selection. It is assumed that the correctness is judged with a same given criterion in the comparison of different testing techniques.

In experimental evaluation and comparison of testing methods, the fault detecting ability of a testing method can be measured in a number of different ways; see, for example, [19]. The following are among the most widely used in the literature of software testing.

Let $\mathscr{M}$ be a testing method and S a given SUT. Let $T_1, T_2, \dots, T_k$ be $k>0$ non-empty test sets generated by the testing method $\mathscr{M}$ for testing S in the experiment.

P-measure -- Probability of detecting at least one fault

The *P-measure* of the k tests $T_1, T_2, \dots, T_k$ is defined as follows.

$$P(T_1, T_2, \dots, T_k)=d/k, \quad k>0$$

where $d = \left|\{T_i \middle| \exists a \in T_i \cdot (S \text{ fails on } a), i = 1\dots k\}\right|$.

Informally, d is the number of test sets T_x in $T_1, T_2, \dots, T_k$ that T_x contains at least one test case a such that S fails on a.

[a] Strictly speaking, it should not be called test set as elements are ordered. But, it is a convention in the literature of ART to call them test set.

[b] A distance measure on a domain D is a function $\| \ \|$ from D^2 to real numbers in $[0,\infty)$ such that for all $x, y, z\in D$, $\| x, x \| = 0$; $\| x, y\| = \|y, x \|$ and $\|x , y \| \leq \|x , z \| + \| z, y\|$.

It is obvious that when $k \to \infty$, $P(T_1, T_2, \ldots, T_k)$ approaches the probability that a test set T detects at least one fault. Therefore, P-measure is an estimation of the probability that a test set T detects at least one fault.

When the test cases in each test set T are generated at random independently using the same distribution of the operation profile, the probability that a test set of size w detects at least one fault is $1-(1-\theta)^w$, where θ is the failure probability of the software. In other words, the P-measure of random testing is an estimation of the value $1-(1-\theta)^w$, where w is the test size. Formally, for random testing, we have that

$$P(T) \approx 1-(1-\theta)^w, \text{ where } w = |\,T\,|.$$

E-measure -- Average number of failures

The *E-measure* of k tests $T_1, T_2, \ldots, T_k$ is defined as follows.

$$M(T_1, T_2, \ldots, T_k) = \frac{1}{k}\left(\sum_{i=1}^{k} u_i\right), \quad k>0$$

where $u_i = \left|\{a \in T_i \,|\, S \text{ fails on } a\}\right|$, $i=1,\ldots, k$.

Informally, u_i is the number of test cases in test set T_i on which S fails. It is obvious that when $k \to \infty$, $M(T_1, T_2, \ldots, T_k)$ approaches the expected number of test cases in a test set T on which S fails. Therefore, it is an estimation of the expected number of failures on a test set generated by the testing method $\mathcal{M}$.

When the test sets T_i are of the same size, say w, and the test cases are selected at random independently using the same distribution as the operation profile, the *E-measure* approaches θw, when $k \to \infty$. Here, θ is the failure probability of the SUT. In other words, the *E-measure* of random testing is the estimation of θw, where w is the test size. Formally, for random testing we have that $M(T) \approx \theta w$, where $w = |\,T\,|$.

It is worth noting that, when test size is 1, for random testing, *P-measure* equals to *E-measure.*

***F-measure – The First failure*.**

The F-measure is defined as follows.

$$F(T_1, T_2, \ldots, T_k) = \frac{1}{k}\left(\sum_{i=1}^{k} v_i\right), \qquad k>0$$

where $T_i = \langle a_1, a_2, \ldots, a_{ki} \rangle$, v_i is a natural number such that S fails on the test case a_{v_i} and for all $0<n<v_i$, S is correct on test case a_n. It is assumed that the sizes of the test sets T_i, $i=1,\ldots, k$, are large enough so that v_i exists.

Obviously, when $k \to \infty$, $F(T_1, T_2, \ldots, T_k)$ approaches the expected number of test cases to be executed before the software fails. Therefore, F-measure is an estimation of the number of test cases required to detect the first fault. This measure only applies when the test cases are ordered in a linear sequence as what we do in this paper.

As pointed out in [10], under the condition that the test cases a_i in each test set T are selected at random independently using the same distribution of the operation profile, the F-measure is in fact a random variable of geometric distribution, i.e. the distribution of first success Bernoulli trial. Therefore, when $k \to \infty$, $F(T_1, T_2, \ldots, T_k)$ approaches $1/\theta$, where θ is the probability of failure because the expectation of geometric distribution is $1/\theta$. In other words, for random testing, F-measure is an estimate of $1/\theta$. Formally, $F(T) \approx 1/\theta$, if $|\,T\,|$ is large enough.

Note that, we also use the same measurement defined above to measure the test result on one test set T. This is the case when $k=1$ in the above definition.

An important property of *P-measure* and *E-measure* is that, for all test sets T_1 and T_2, if T_1 is a permutation of T_2, then $P(T_1)=P(T_2)$, and $M(T_1)=M(T_2)$, provided that the software is initialized before every execution on each test case. In other words, they are independent of the order in which test cases are executed. This property is not true for the *F-measure*. Therefore, *F-measure* is suitable for the evaluation of testing methods that the order in which test cases are executed is important.

A common feature of the above measurements is that they are independent of the order in which test sets $T_1, T_2, \ldots, T_k$ are listed provided that the software is properly initialized after experiment on each test set.

C. Measurements of testing methods' reliability

As discussed in section 1, an important and desirable property of software testing methods is the reliability in their fault detecting abilities in the sense that its performance is consistent and stable. In other words, when the method is applied many times, the fault detecting abilities should not vary too much. However, the reliability of software testing methods in fault detecting ability has not been investigated in software testing literature. In this subsection, we propose to use standard deviation as a reliability measure because it is the most important indicator of the variation of random variables.

Let $T_1, T_2, \ldots, T_k$ be a number of test sets generated according to a testing method and ϕ be a given measurement of fault detecting ability, such as one of the above mentioned. Assume that $m_1, m_2, \ldots, m_k$ are the fault detecting abilities of $T_1, T_2, \ldots, T_k$ as measured by ϕ in testing SUTs $S_1, S_2, \ldots, S_k$, where S_i can be the same as or different from S_j.

S-measure – The variation of fault detecting ability

The S-measure $S(T_1, T_2, \ldots, T_k)$ of tests $T_1, T_2, \ldots, T_k$ is formally defined as follows.

$$S(T_1, T_2, \ldots, T_k) = \sqrt{\frac{1}{k-1}\sum_{i=1}^{k}(m_i - m)^2}, \quad k>1$$

where $m_i=\phi(T_i)$, $i=1,\ldots, k$, $m = \sum_{i=1}^{k} m_i / k$.

In other words, the S-measure is the sample standard deviation of the values $m_1, m_2, \ldots, m_k$ of the ϕ-measure on each test set. The smaller the S-measure, the more reliable the testing method is.

III. The Experiment

The main goal of the experiment is to evaluate ART methods on their reliabilities of fault-detecting abilities with respect to realistic faults. As reported in [16], mutants systematically generated by using mutation testing tools can represent very well the real faults. Thus, we use mutants in our experiments to achieve this goal rather than simulation, which is based on idealistic assumptions on the shapes of failure domains.

The secondary goal of the experiment is to identify the factors that influence the reliability of fault detecting ability. We identified two main candidate factors: the reliability (or equivalently the failure probability) of the SUT and the regularity of failure domains. This secondary goal is achieved through the design of experiment with repetition of testing on a number of test sets.

The experiment consists of the following activities.

A. Selection of subject programs

We selected two programs with integer input to simplify the random generation of test cases and the calculation of distances between the test cases. They are programs that calculate the greatest common divisor (GCD) and the least common multiplier (LCM). Both of them have a two dimensional input space on natural numbers. The programs are written in Java.

B. Generation of mutants of the subject program

For each of the subject program, mutants were generated systematically by using the MuJava testing tool [20]. A total of 187 mutants were generated. There are 49 mutants that are equivalent to the original program. To identify equivalent mutants, we first executed all the mutants on random test cases. When a mutant is still alive after being executed on 5000 random test cases, it is manually examined to see if it is equivalent to the original. There are also trivial mutants, which fail on every test case. Their failure domain is the same as the input domain, thus they are useless in our experiment. Therefore, both equivalent and trivial mutants were removed from statistical analysis. Table 1 gives the distribution of mutants. Readers are referred to [20] for how mutants are classified.

TABLE 1. DISTRIBUTION OF MUTANTS.

Mutant Type	GCD	LCM	Total
AOIS	54	52	106
AOIU	9	6	15
AORB	4	4	8
AORS	0	1	1
LOI	14	14	28
ROR	15	15	30
Total	96	92	187
Equivalent Mutants	28	21	49
Trivial Mutants	27	29	56
Used in experiment	**41**	**52**	**93**

C. Generation of test sets.

For each testing method, five test sets were generated according to the algorithm given below. Each test set consists of 1000 test cases.

Algorithm: *Test Set Generation*

Step 1. The input domain is divided into two sub-domains. The original sub-domain $D_0=\{1..5000\} \times \{1..10000\}$, The mirror sub-domain $D_1=\{5001..10000\} \times \{1..10000\}$.

That is, the input for variable x is an integer in the range from 1 to 5000 and the input for variable y is in the range from 1 to 10000. The total input domain is $D=D_0 \cup D_1 = \{1..10000\} \times \{1..10000\}$, which is of a size 10^8.

Step 2. Use pseudo-random function of the Java language with different seeds to generate 500 random test cases in the domain D_0. Let T_1 be the set of these test cases, where the elements are ordered in the order that they are generated.

Step 3. The mirror test set T_2 is generated by applying the following mirror mapping μ: $D_0 \rightarrow D_1$.

$$\mu(<x,y>)=<x+5000, y>. \qquad (*)$$

The test set RT is obtained by merging T_1 with T_2, i.e. adding the elements of T_2 at the end of T_1.

Step 4. Let $T=T_1 \cup T_2$. The test set $DMART$ is obtained by applying the *Distance* manipulation on T, i.e.

$$DMART=Distance(T).$$

Step 5. Let T_3 be the result of applying the distance manipulation on T_1, i.e. $T_3=Distance(T_1)$. The test set $MDART$ is obtained by applying the *Mirror* manipulation on T_3 with the mirror mapping μ as defined in (*) above, i.e.

$$MDART=Mirror(T_3)=Mirror(Distance(T_1)). \qquad \square$$

Note that, the test sets RT, DMART and MDART contain exactly the same elements, but in different order. As a consequence, the *P*-measure and *E-measure* of each RT test set *T* are identical to the *P*-measure and *E-measure* of the corresponding DMART and MDART test sets. Only the *F*-measure on these test sets will be different. Therefore, the *F*-measure reflects the ART testing methods' differences in fault detecting abilities; while the *E-measure* provides the background information about the mutants' failure probability. The *P*-measure is not used in this paper.

D. Test the mutants

The mutants are tested on each test case in each test set one by one in the order. The outputs of the mutants are compared with the output of the original program. If the output of a mutant on a test case is different from the output of the original program on the same test case, the mutant is killed by the test case. During this testing process, the first test case that kills the mutant and the total number of test cases on which a mutant fails are recorded for statistical analysis. In other words, the *F*-measures and *E-measure*s are taken during the test executions.

E. Analysis of the test results

The test data were analyzed statistically to compare the testing methods according to the F-measures and E-measures. The fault-detecting ability is measured by the F-measure. The reliabilities of the different testing methods are obtained with the *S*-measure of the results on the five repeating test sets. The factors that affect the fault detecting ability and reliability are also analyzed.

It is worth noting that, in the statistical analysis, no distinction is made between the mutants that are generated from GCD or LCM. This is because RT and ART methods are black box testing methods. Their fault detecting abilities only depend on the failure domain of the SUT regardless of the program's internal structure and/or the application domain. Each mutant represents one SUT with its own failure probability and its own regularity of the failure domain. All the mutants in our experiment have the same input domain. Thus, there is no need to treat them differently.

The main findings are discussed in the next section.

IV. MAIN FINDINGS

This section reports the main findings of the experiment.

A. *Characteristics of the mutants*

The mutants generated by the MuJava tool vary on their failure probabilities. In contrast to the experiments reported in [21], where a large proportion of mutants have a failure probability lower than 0.05, there are very few mutants in our experiment that have low failure probabilities. In fact, no mutant in our experiment has failure probability lower than 0.2, which is equivalent to the *E-measure* less than 200 for a test set of size 1000. Table 2 shows how the mutants are distributed according to their failure rates, i.e. the *E-measure*, where the test size is 1000. It is also visually depicted in Figure 1.

TABLE 2. THE DISTRIBUTION OF MUTANTS W.R.T THEIR FAILURE RATES

Ranges of Avg E-measure	Number of Mutants		
	GCD	LCM	Total
1--49	0	0	0
50-99	0	0	0
100-149	0	0	0
150-199	0	0	0
200-249	1	0	1
250-299	3	0	3
300-349	7	0	7
350-399	1	4	5
400-449	4	0	4
450-499	2	7	9
500-549	1	7	8
550-599	0	0	0
600-649	1	0	1
650-699	12	4	16
700-749	0	2	2
750-799	1	0	1
800-849	2	0	2
850-899	1	0	1
900-949	3	0	3
950-999	1	17	18

In the existing research on ART techniques, the subject samples (i.e. the SUT) are always selected with very low failure rates. For example, in [21, 22], mutants with a failure rate higher than 0.05 were dismissed. All the mutants of failure rates at most 0.05 were treated equally in statistical analysis in the comparison of RT and ART techniques. In the simulation experiments on ART, the focuses are also on low failure rate situations. For example, in [10], the performances of ART testing techniques were evaluated by simulating the failure rates of 0.01, 0.005, 0.001 and 0.0005. In [14], a more systematic simulation of the fault detecting abilities of ART techniques is reported. However, most of the simulations are for failure rates less then 25%. There are only three sets of simulation data for failure rates higher than 25%, i.e. the failure rates of $\sqrt{2}/2$, 1/2 and $\sqrt{2}/4$. Thus, it is an open question that how ART perform on SUT that have higher failure rates.

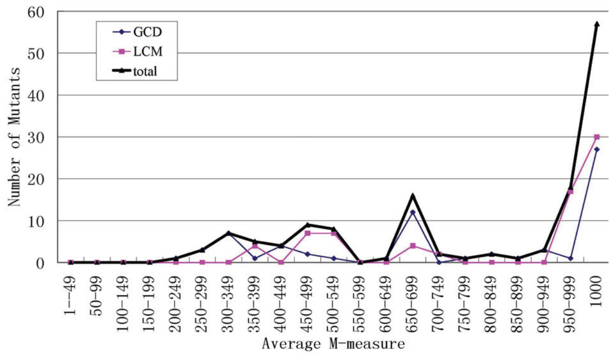

Figure 1. The distribution of mutants w.r.t. to the E-measure

The second statistical property of the mutants used in our experimentations is characterized by the distribution of the variations of the E-measures of the mutants in five different tests. As shown in Figure 2, this distribution is not uniform.

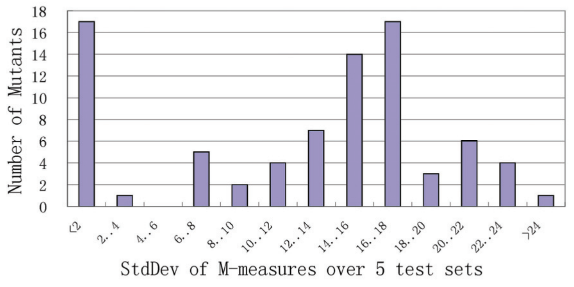

Figure 2. The distribution of mutants w.r.t. to the standard deviation of E-measures over five test sets

There is also a relationship between the average E-measures and the standard deviation of the E-measures on the mutants. As shown in Figure 3, when the average E-measures is very small or very large, the standard deviations of the E-measures tend to be small. In the middle, the standard deviations are larger than those on the two ends.

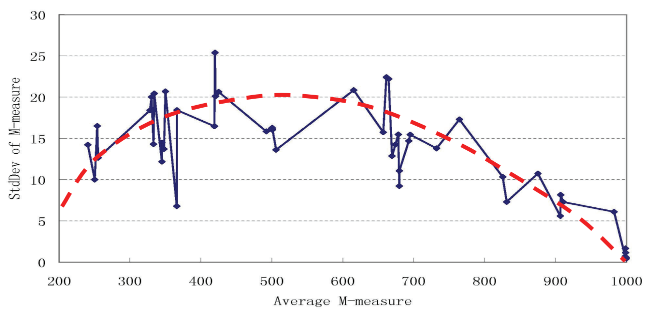

Figure 3. The relationship between average E-measures and the standard deviation of the E-measures

These statistical characteristics of mutants were not given in existing literature in the evaluation of ART methods [21, 22]. Therefore, we cannot compare our samples with the existing work on these properties.

B. *ART methods' fault detecting abilities*

To compare the fault detecting ability of ART methods with RT, we use the F-measure. The distribution of the data is shown in Figure 4.

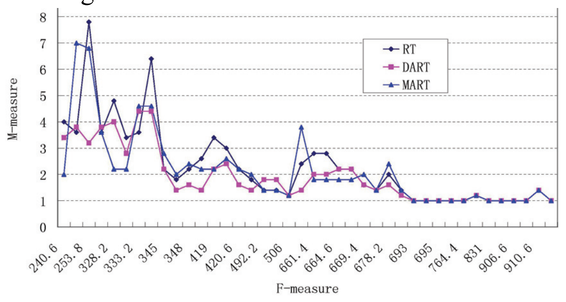

Figure 4. Distribution of F-measures w.r.t. to E-measures

As stated in the previous sub-section, there are a large number of trivial mutants that fail on all inputs. As in [21, 22], we calculated the average F-measures of RT, MDART and DMART on non-trivial mutants. The result is shown in Figure 5. Using these average F-measures one can infer that DMART improves RT by 17.64%, which is much more than MDART does, which is 3.75%; see equations below.

$$\frac{F(RT)-F(MDART)}{F(RT)}=\frac{1.973-1.901}{1.973}=3.65\%\ ,$$

$$\frac{F(RT)-F(DMART)}{F(RT)}=\frac{1.973-1.625}{1.973}=17.64\%$$

In [10, 14], MART and DART were of similar fault detecting abilities in terms of average F-measures. They only differ from each other in the time needed to generate test cases. The results we obtained seem inconsistent with their simulation results.

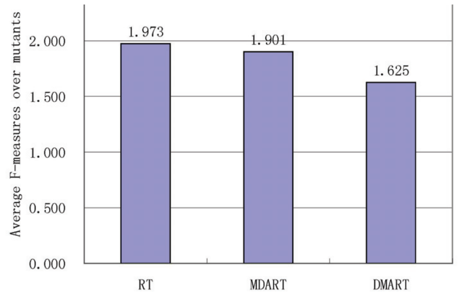

Figure 5. Average F-measures over all non-trivial mutants.

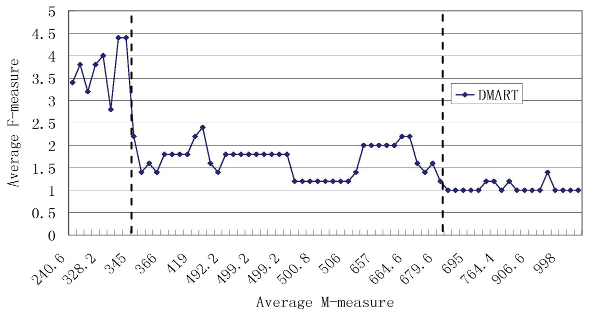

Figure 6. Distribution of DMART's F-measure

One of the main factors that affect the fault detecting ability of ART techniques is the failure rate of the SUT. This is also confirmed in our experiments. Figure 6 shows the distribution of DMART's F-measures against the average of E-measures.

From Figure 6, it is apparent that there are three levels of F-measures (>2.5, 1.5 ~ 2.5 and <1.5) and the E-measures can be divided into three areas (240~340, 340 ~ 680, >680). The average F-measures in each area is shown in Figure 7.

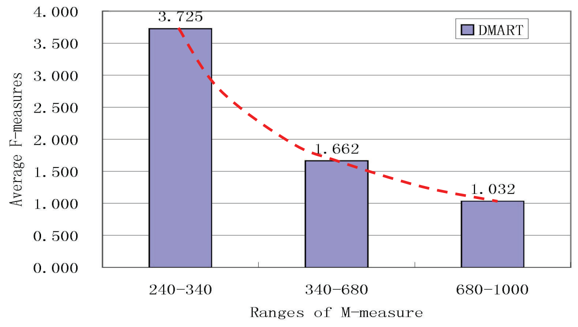

Figure 7. F-measures of DMART in different ranges of E-measures

A similar phenomenon can be observed on the distribution of F-measures on MDART tests, but less obvious and the areas for each level of F-measures are wider; see Figure 8.

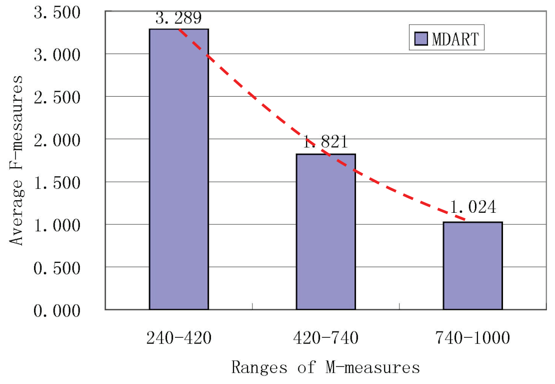

Figure 8. F-measures of MDART in different ranges of E-measures

These results are consistent with existing work on ART although the scales are different, which is probably because our mutants have higher failure rates. In all studies including ours, ART's fault detecting abilities increase as SUT's failure rate decreases.

C. *Reliability of ART methods*

Now, let's analyze the reliability of ART methods and the main discovery of the experiment.

Similar to the calculation of average F-measures for each testing methods, we calculate the average standard deviation for each testing methods, which reveals the differences in the reliabilities of the methods as shown in Figure 9.

From Figure 9, DMART made a significant improvement in reliability over RT, which is 32.70%. In contrast, the improvement that MDART made is much smaller, which is only 6.30%. The formulas for the calculation are given in eq. (1) and (2) below. In other words, DMART not only has better fault detecting ability than MDART, but also more consistent.

$$\frac{S(RT)-S(MDART)}{S(RT)}=\frac{1.413-1.324}{1.413}=6.30\% \qquad (1)$$

$$\frac{S(RT)-S(DMART)}{S(RT)}=\frac{1.413-0.951}{1.413}=32.70\% \qquad (2)$$

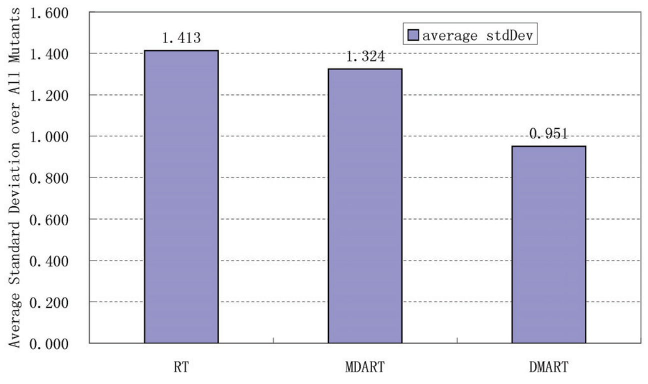

Figure 9. Average S-measures over all non-trivial mutants

To further identify the factors that affect the reliability of ART methods, the relationship between the variations of F-measure and the variation of the corresponding mutants' failure rates on the five test sets are investigated. Figure 10 shows the relationship between the standard deviation of the F-measures on DMART tests and the average failure rate. Here, the standard deviations are calculated from the test results on five different test cases on each mutant.

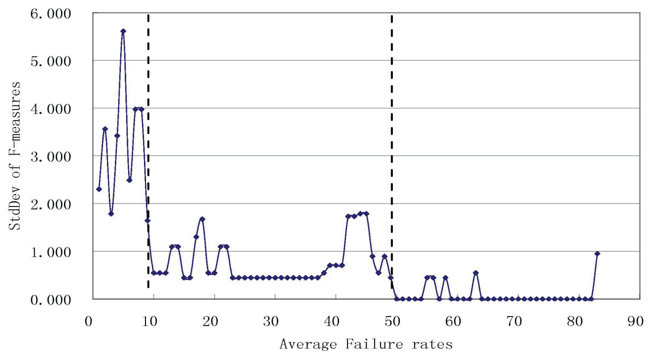

Figure 10. The Standard Deviation of F-measures vs Average Failure Rates

From Figure 10, it is clear that the variation of DMART tests' F-measure decreases as the average failure rates increases. That is, when the software under test is of high reliability (or, equally, low failure rate), the F-measure is more consistent. Again, we can identify three levels of the standard deviations of F-measures on three areas of average failure rates as shown in Figure 11. The correlation coefficients between the average E-measure and the average standard deviations of F-measure of RT, MDART and DMART on all mutants are -0.735, -0.645 and -0.615, respectively. Therefore, it is clearly that the reliabilities of these testing methods depend on SUT's failure rate.

Figure 12 shows the relationship between the standard deviations of F-measures and the standard deviation of E-measures (i.e. failure rates). Note that, when a number of random tests of the same size are applied to a SUT, the standard deviation of the E-measures of these tests reveals a property, i.e. the variation, of the SUT's failure domain. The more regular the failure domain is, the smaller the standard deviation of the E-measure should be. Therefore, the standard deviation of E-measure reflects the regularity of the SUT's failure domain.

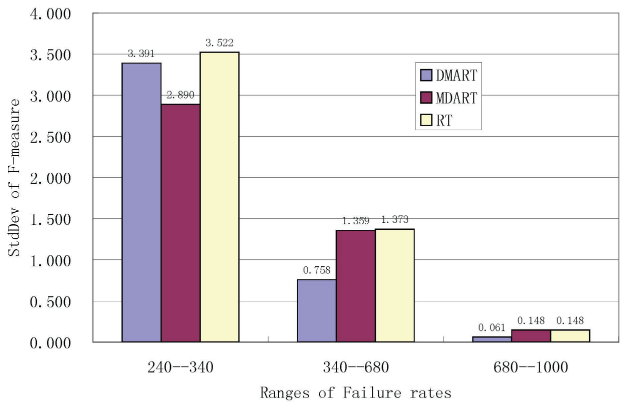

Figure 11. StdDev of F-measures on different ranges of E-measures

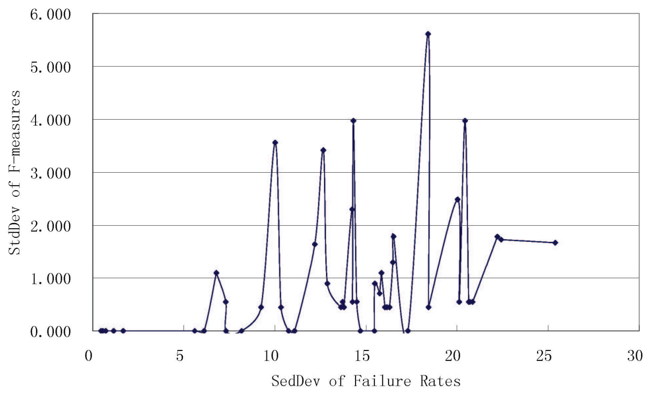

Figure 12. The Standard Deviation of F-measures vs Standard Deviation of Failure Rates over five test sets

Figure 12 shows that as the standard deviations of E-measures increases, the standard deviations of F-measures tend to increase. In other words, when a SUT's failure domain is more regular (the standard deviation of the E-measures is small), the consistency of F-measure is higher. This relationship becomes clearer when the average standard deviation is calculated on mutants in various ranges of standard deviations of E-measures, as shown Figure 13. Moreover, the correlation coefficients between the standard deviations of F-measures of RT, MDART and DMART and the standard deviations of E-measures for all mutants are 0.574, 0.464 and 0.430, respectively. Therefore, we can conclude that the reliabilities of these testing methods are related to the standard deviation of E-measures, i.e. the regularity of failure domains.

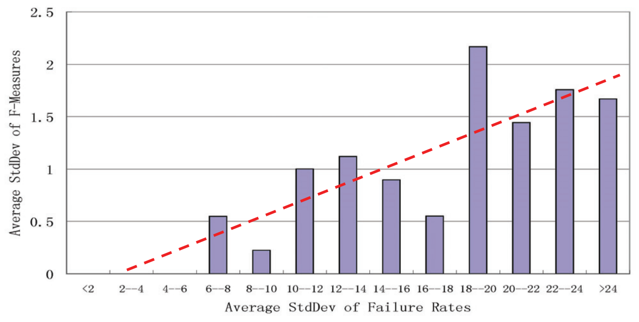

Figure 13. Average Standard Deviations of F-measure over Mutants in Various Ranges of the Standard Deviation of Failure Rates

V. Conclusion

Our experiment results described above confirm the main results of existing work on ART methods [9~14, 21~22]. In particular, we confirm that ART method can improve the fault detecting ability over random testing. However, the scale of improvement is much smaller than that demonstrated in existing work, which could be up to 40% increase in F-measures while we have shown that the average improvement is less than 20%.

Our experiments differs from the existing work in that our mutants have higher failure rates than those used in [21~22]. As far as we know, the performances of ART methods have not been systematically investigated for SUTs that have high failures rate as those in our experiment. In [21~22], only a subset of mutants that have very low failure rates are used in the experiment. We believe that, for mutation analysis of testing methods, the whole set of mutants should be used because the set of mutants collectively models the real faults in software systems.

In addition, this is the first time that the reliabilities of ART methods are investigated. We discovered that ART not only improves the fault detecting ability, but also significantly improves the reliability of fault detecting ability in the sense that the variations on F-measures are significantly smaller than random testing, where variation is measured by the standard deviation of F-measures. We also analyzed the factors that affect ART's reliability. The main factors are the failure rate of the software under test and the regularity of the failure domain when measured by the standard deviation of E-measures. The experiment data demonstrated that when the SUT's failure rate increases, the test method's reliability increases. Moreover, when the regularity increases, the reliability also increases.

The main threats to the validity of our experiment and its conclusions come from two aspects. First, our subject programs were selected at random. They are of small scale and are of rather low computation and structural complexity. They may or may not represent the real software systems. As a consequence, the failure domains represented by their mutants could be simpler than the real faults. However, since ART and RT are black box testing methods, the impact of the complexity of the program is limited. Moreover, the validity of the failure domains represented by mutants should be no less than simulations where ideal assumptions on failure domains were made. We plan to repeat our experiment with more subject programs, especially those more complicated and more complex. By doing so, the validity of the experiment will be validated.

Second, the number of repeated test sets for each test method is small. We only used 5 test sets for each test method. This is due to the restriction on the resource available. We are now repeating the experiment with more test sets. However, although using more test sets will improve our confidences in the conclusions, we don't see any reason why the conclusions drawn from the experiment could be significantly different. In particular, we believe that the qualitative conclusions should not be threatened by the sample size.

Another direction of future work is to further investigate the factors that affect the fault detecting ability as well as their reliability. In [23], the compactness of failure domains is proven to be an important factor that affects fault detecting ability. An open question is how to measure the compactness of failure domains that are not in regular geometry shapes.

Acknowledgement

The work reported in this paper was carried out while Y. Liu is visiting the Oxford Brookes University, which was funded by the China Scholars Council.

References

[1] H. Zhu, P. Hall, and J. May, Software unit test coverage and adequacy, ACM Computing Surveys, Vol. 29, No. 4, Dec. 1997, pp366~427.
[2] D. Hamlet, Random testing, in Encyclopedia of Software Engineering, J. Marciniak, ed., Wiley, 1994, 970-978.
[3] S. Ntafos, On random and partition testing, Proc. of ISSTA '98, ACM SIGSOFT Software Engineering Notes, Vol. 23 Issue 2, pp42-48.
[4] J.W. Duran, & S. Ntafos, An evaluation of random testing, IEEE TSE, Vol. SE_10, No. 4, pp438-444, July 1984.
[5] D. Hamlet, and R.Taylor, Partition testing does not inspire confidence, IEEE TSE, Vol. 16, pp206~215, Dec. 1990.
[6] M.Z., Tsoukalas, J.W. Duran, and S.C. Ntafos, On some reliability estimation problems in random and partition testing, IEEE TSE, Vol. 19, No.7, July 1993, pp687~697.
[7] RT 2006, Proceedings of the 1st international workshop on random testing 2006, Portland, Maine, July 20 - 20, 2006, ACM Press.
[8] RT 2007, Proceedings of the 2nd international workshop on random testing 2007, Atlanta, Georgia, November 06 - 06, 2007, ACM Press.
[9] T. Y. Chen, H. Leung, and I. K. Mak. Adaptive random testing. Proc. of the 9th Asian Computing Science Conf., LNCS 3321, pp320–329, 2004.
[10] T. Y. Chen, F.-C. Kuo, R. G. Merkel and S. P. Ng, Mirror Adaptive Random Testing, Inf. & Software Tech. 46(15), pp1001-1010, 2004.
[11] K. P. Chan, T. Y. Chen, D. Towey, Adaptive Random Testing with Filtering: An Overhead Reduction Technique, pp292-299
[12] T. Y. Chen, De Hao Huang, F.-C Kuo, Adaptive random testing by balancing, in [7], pp2 – 9.
[13] J. Mayer, Lattice-based adaptive random testing, Proc. of ASE '05, November 2005, pp333-336.
[14] T. Y. Chen, F.-C. Kuo: Is adaptive random testing really better than random testing. In [8], pp64-69.
[15] J.B. Goodenough & S.L. Gerhart, Toward a theory of test data selection, IEEE TSE, Vol.SE_3, June 1975.
[16] J. H. Andrews, L. C. Briand, Y. Labiche, Is mutation an appropriate tool for testing experiments? Proc. of ICSE 2005, pp 402 – 411.
[17] G. J. Myers. The Art of Software Testing. Wiley, 2nd edition, 1979.
[18] J.A. Whittaker, and M.G. Thomason, A Markov Chain Model for Statistical Software Testing, IEEE TSE 20(10), 1994, pp812-824.
[19] H. Zhu, A formal analysis of the subsume relation between software test adequacy criteria, IEEE TSE 22(4), 1996, pp248~255.
[20] Y-S Ma, J. Offutt and Y-R. Kwon. MuJava: An Automated Class Mutation System, *JSTVR* 15(2), June 2005, pp97-133.
[21] J. Mayer, C. Schneckenburger, An Empirical Analysis and Comparison of Random Testing Techniques. Proc. of ISESE 2006, pp105-114.
[22] J. Mayer, T. Y. Chen, D. H. Huang, Adaptive Random Testing with Iterative Partitioning Revisited, Proc. of SOQUA 2006, pp22-29.
[23] T.Y. Chen, F.-C. Kuo and C.A. Sun, The impact of the compactness of failure regions on the performance of adaptive random testing, Journal of software, 17 (12) , 2006, pp2438-2449.