Software Qual J (2008) 16:303–327
DOI 10.1007/s11219-008-9047-6

Enhancing adaptive random testing for programs with high dimensional input domains or failure-unrelated parameters

Fei-Ching Kuo · Tsong Yueh Chen · Huai Liu · Wing Kwong Chan

Published online: 8 March 2008

Abstract Adaptive random testing (ART), an enhancement of random testing (RT), aims to both randomly select and evenly spread test cases. Recently, it has been observed that the effectiveness of some ART algorithms may deteriorate as the number of program input parameters (dimensionality) increases. In this article, we analyse various problems of one ART algorithm, namely fixed-sized-candidate-set ART (FSCS-ART), in the high dimensional input domain setting, and study how FSCS-ART can be further enhanced to address these problems. We propose to add a filtering process of inputs into FSCS-ART to achieve a more even-spread of test cases and better failure detection effectiveness in high dimensional space. Our study shows that this solution, termed as FSCS-ART-FE, can improve FSCS-ART not only in the case of high dimensional space, but also in the case of having failure-unrelated parameters. Both cases are common in real life programs. Therefore, we recommend using FSCS-ART-FE instead of FSCS-ART whenever possible. Other ART algorithms may face similar problems as FSCS-ART; hence our study also brings insight into the improvement of other ART algorithms in high dimensional space.

Keywords Software testing · Random testing · Adaptive random testing · Fixed-sized-candidate-set ART · High dimension problem · Failure-unrelated parameters

A preliminary version of this article was presented at the 22nd Annual ACM Symposium on Applied Computing (SAC'07) (Kuo et al. 2007).

F.-C. Kuo (✉) · T. Y. Chen · H. Liu
Faculty of Information and Communication Technologies, Swinburne University of Technology, Hawthorn, Victoria 3122, Australia
e-mail: dkuo@ict.swin.edu.au

W. K. Chan
Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Hong Kong, China

1 Introduction

Software testing is a major software engineering activity to assure the quality of software under test. It assures software quality by actively detecting bugs before serious software failures actually take place in operation. Inputs used for testing are called *test cases*, and those that lead to software failures are called *failure-causing inputs*. Software often cannot be completely tested due to limited testing resources and its huge set of inputs (known as *input domain*). Thus, one focus of software testing is to select test cases that can cost-effectively reveal failures.

Test case selection is a critical task in software testing. Many testing methods (Myers et al. 2004) have been developed to guide the selection of test cases. One simple method is *random testing* (RT), in which test cases are selected in a random manner from the input domain (Hamlet 2002; Myers et al. 2004). There are many merits of using RT in software testing. For example, it can generate numerous test cases automatically at low cost. Its generation of test cases needs not to involve software specifications or source code. It brings "randomness" into the testing process, so it can detect certain failures unable to be revealed by deterministic approaches (those designing test cases to target certain faults or test objectives). Because of these merits, RT has been widely used for detecting failures (Bird and Munoz 1983; Cobb and Mills 1990; Miller et al. 1990, 1995; Slutz 1998; Forrester and Miller 2000; Yoshikawa et al. 2003; Dabóczi et al. 2003; Godefroid et al. 2005; Miller 2005; Regehr 2005; Sen et al. 2005, Nyman), and has been incorporated into many industrial software testing tools, such as RAGS (Random Generation of SQL) used by the Microsoft SQL Server testing group (Slutz 1998) as well as those developed by IBM (Bird and Munoz 1983) and Bell Laboratories (Godefroid et al. 2005). Miller et al. (1990) used RT to test UNIX utilities, and observed that 25–30% of these utilities had been crashed. Five years later, they repeated and extended their study of testing UNIX utilities, and continued to find a lot of failures revealed by RT (Miller et al. 1995). Regehr used RT for testing embedded systems because RT can "create a large number of uncorrelated test cases automatically. These can be used to drive a system into interesting states, with the goal of eliciting failure modes that cannot be found using other testing methods or static analysis" (Regehr 2005). In brief, RT is particularly desirable if complete specifications and source code are unavailable (as a result, some testing methods may not be applicable) or automation of other testing methods is expensive.

In spite of the popularity, some people criticised RT for utilizing little or no information to guide its test case selection. It had been observed that failure-causing inputs tend to cluster together (Ammann and Knight 1988; Finelli 1991; Bishop 1993). This observation inspired Chen et al. to improve the effectiveness of RT by enforcing a more even-spread of random test cases. They referred to this testing approach as adaptive random testing (ART) (Mak 1997; Chen et al. 2001). ART aims for generating random test cases (same goal as RT), at the same time, evenly spreading them (not concerned by RT). This approach of testing can be implemented in various ways. Previous studies (Mak 1997; Chen et al. 2001, 2004, 2005; Mayer 2005; Chan et al. 2006) showed that their ART algorithms can outperform RT when failure-causing inputs do cluster into contiguous regions (known as *failure regions*). In addition to such an improvement, ART can be automated and its test case selection involves randomness like RT. Therefore, it is strongly recommended to consider ART as an alternative to RT.

It has been recently observed that the effectiveness of some ART algorithms may deteriorate as the number of program input parameters (dimensionality) increases (Chen et al. 2005). It should be noted that the *curse of dimensionality* (defined by Bellman 1957 as the remarkable growth in the difficulty of problems due to an increase in dimensionality) is a well-known problem in many disciplines. For example, it is more difficult to generate a truly uniform distribution of points in higher dimensions (Matsumoto and Nishimura 1998). It is worthwhile to study the problems of ART in high dimensional space (referred to as *high dimension problems of ART* in this article).

In this article, we investigate the high dimension problems of one particular ART algorithm, namely *fixed-sized-candidate-set ART* (FSCS-ART) (Mak 1997; Chen et al. 2001) and propose a solution, namely FSCS-ART with filtering by eligibility (FSCS-ART-FE) algorithm, to address these problems. Our study shows FSCS-ART-FE can improve FSCS-ART not only in the case of high dimensional space, but also in the case of having failure-unrelated parameters. Both cases are common in real life programs. Therefore, we recommend that FSCS-ART-FE should be used instead of FSCS-ART whenever possible. FSCS-ART is not the only one ART algorithm that encounters high dimension problems. Our study also brings insight into the improvement of other ART algorithms that face similar problems as FSCS-ART.

This article is organized as follows. Section 2 introduces the algorithm of FSCS-ART and the experimental setup related to the study of ART. Section 3 discusses several problems of FSCS-ART when dealing with high dimensional space. Section 4 details our approach to enhancing FSCS-ART with respect to high dimension problems. Section 5 reports our findings regarding to the effectiveness and test case distribution of FSCS-ART-FE. These findings lead us to conclude that FSCS-ART-FE is an enhancement of FSCS-ART with the presence of high dimensional input domains and failure-unrelated parameters. Article conclusion is given in Sect. 6.

2 Background

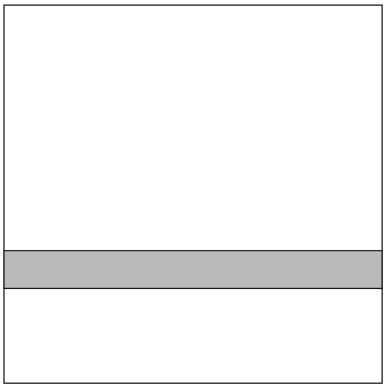
Any faulty program has at least two attributes: *failure rate* (the ratio of the number of failure-causing inputs to the number of all possible inputs) and *failure pattern* (the geometric shapes of failure regions and the distribution of these regions within the input domain). Both attributes are fixed upon completion of coding but unknown to testers before testing. Program 1 gives a sample program fault that causes a *strip* failure pattern as illustrated in Fig. 1. Other sample faults related to failure patterns can be found in (Chen et al. 2005).

Program 1 A sample program fault that causes a strip failure pattern

```
INPUT X, Y
IF (Y< = 0)     /* ERROR: Should be if(Y< = 1) */
{ Z = X − 2Y }
ELSE
{ Z = X + 2Y }
OUTPUT Z
```

Since the introduction of ART, great attention has been paid to how well ART can outperform RT. There are three commonly used metrics to measure the effectiveness of a testing method: *E-measure* (the expected number of detected failures), *P-measure* (the probability of detecting at least one failure) and *F-measure* (the expected number of test cases for revealing the first failure). There are two fundamental differences between

Fig. 1 Failure pattern for program 1



F-measure and the other two measures, that we wish to point out. First, P-measure and E-measure are computed based on (i) the estimated failure rate and (ii) the amount of tests that testers plan to conduct; however, F-measure can be obtained without pre-knowledge of the second parameter. Second, given a set of test cases, P-measure and E-measure do not depend on the test sequence, but F-measure does depend on the test sequence. ART is an *adaptive testing strategy*, "in which the results of previous testing influence subsequent test selection" (Chen and Merkel 2008). In ART, the key issue is how to sequence tests among all possible inputs to effectively detect failures, and hence the test sequence should be considered to reflect the effectiveness of ART. Therefore, F-measure is considered the most appropriate metric in the study of ART (also see Chen et al. 2006). A theoretical evaluation of ART is known to be extremely difficult as the effectiveness of ART depends on many factors (Chen et al. 2005) As a result, almost all studies of ART were carried out by experiments and using F-measure. Like all other ART studies (Mak 1997; Chen et al. 2001, 2004, 2005; Mayer 2005; Chan et al. 2006), we evaluate ART using F-measure and assume that test cases are selected with replacement according to the uniform distribution.

When testing is carried out on a real life faulty program, a failure is said to be found if an incorrect output is observed. When testing is conducted using simulations, in order to simulate the testing process, failure rates and failure patterns must be predefined, and failure regions are randomly placed inside the input domain. A failure is said to be found if a point inside one of the failure regions is picked by a testing method.

Like all other ART studies, we collect F-measures of ART using the following procedure. Given a faulty program (or given a predefined failure rate and failure pattern), conduct testing by an ART algorithm. Collect the F-measure of ART (F_{ART}) in each testing. Repeat testing s times until a significantly reliable mean of F_{ART} ($\pm 5\%$ accuracy range and 95% confidence level are used in our experiments) has been obtained. The value of s is determined dynamically using the same formula as used in (Chen et al. 2004).

As mentioned before, ART is often compared with RT in terms of F-measure. We will use the *ART F-ratio* ($= F_{ART}/F_{RT}$) metric given in (Chen et al. 2005) to show the improvement of ART over RT, where F_{ART} and F_{RT} denote the F-measures of ART and RT, respectively. A smaller ART F-ratio means a greater saving of test cases by ART to detect the first failure, and hence indicates a greater improvement of ART over RT. Since test cases are selected with replacement according to the uniform distribution in this article, F_{RT} is expected to be $1/\theta$ in theory, where θ denotes the failure rate of a faulty program.

ART aims to both randomly select and evenly spread test cases. Several researches have been conducted to investigate the test case distribution of ART algorithms inside the input domain. As explained by Chen et al. (2007), a good even-spread of test cases should possess at least two properties—low dispersion and low discrepancy. Dispersion measures the largest empty space between points, while discrepancy measures the greatest density difference of points across the space. Formal definitions of dispersion and discrepancy would be discussed in Sect. 5.4.

FSCS-ART is known to have an edge preference (that is, generating test cases more frequently in the edge than in the central part of the input domain) (Chen et al. 2005). FSCS-ART (Mak 1997; Chen et al. 2001) maintains two sets of test cases, namely, the *executed set* (E) and the *candidate set* (C), where E stores all executed test cases that do not reveal failures, and C stores *k* random inputs, from which the next test case will be selected. The candidate with the longest Euclidean distance to its nearest neighbour in E is chosen as the next test case. The algorithm of FSCS-ART is given in Fig. 2. In this article, *k* is set to 10 as suggested by previous studies (Mak 1997; Chen it et al. 2001).

Chen et al. (2005) carried out a detailed study on the effectiveness of FSCS-ART. They designed a series of experiments and observed that FSCS-ART performs best when the failure pattern is a single square failure region. As the number of failure regions increases or the compactness of failure regions decreases, the improvement of FSCS-ART over RT decreases. In this article, we will conduct a similar experimental study as Chen et al. (2005) to compare FSCS-ART with FSCS-ART-FE (our proposed solution to high dimension problems of FSCS-ART).

For ease of discussion, we will use 1D, 2D,... and *N*D to denote one-dimensional, two-dimensional,... and *N*-dimensional, respectively.

3 High dimension problems of FSCS-ART

In this article, we aim to study high dimension problems of FSCS-ART. Two major problems are discussed in the following sections.

```
1. $n := 0$ and E := { }.
2. Randomly select a test case, t, from the input domain (according to
   the uniform distribution).
3. $n := n + 1$.
4. IF t reveals a failure THEN GOTO Step 9; ELSE store t in E.
5. Randomly generate $k$ inputs to construct C (according to the
   uniform distribution).
6. $d := 0$
7. FOR $i = 1$ to $k$, DO BEGIN
     calculate the Euclidean distance $d_i$ between $c_i$ and its nearest
     neighbour in E, where $c_i \in C$
     IF $d_i > d$ THEN BEGIN
       t := $c_i$
       $d := d_i$
     END
   END
8. GOTO Step 3.
9. RETURN $n$ and t, and EXIT.
```

Fig. 2 The algorithm of FSCS-ART

3.1 Problem 1

In the first experiment of Chen et al. (2005), it has been observed that when the failure pattern consists of a single square failure region, FSCS-ART could perform even worse than RT under high failure rates (θ). The range of θ in which FSCS-ART is worse than RT grows as dimensionality increases. It is interesting to investigate the cause of this phenomenon.

For ease of discussion, M is used to denote an N dimensional input domain. M_{centre} and M_{edge} are two disjoint subregions of M, and $\mathrm{M}_{centre} \cup \mathrm{M}_{edge}$ = M. M_{centre} resides at the centre of M, and M_{edge} encloses M_{centre}. The shapes of M and M_{centre} are identical, and $|M_{centre}| = a|M|$, where $0 < a < 1$, and $|M_{centre}|$ and $|M|$ denote the sizes of M_{centre} and M, respectively.

Theorem 3.1 *Assume that there exists one and only one rectangular failure region* (F) *inside M. Further assume that* $|M_{centre}| = a \cdot |M|$ *and the shapes of M and* M_{centre} *are identical.* L_i, $\hat{L}_i$ *and* l_i *denote the lengths of M,* M_{centre} *and* F *in the ith dimension, respectively. For any* $0 < a < 1$, *if* $\forall i, l_i > \frac{L_i}{2}$ *(so* $\theta > \frac{1}{2^N}$*),*

(i) *the chance* (p) *of picking an element of* F *from* M_{centre} *is greater than* θ*, and*
(ii) *the chance* (q) *of picking an element of* F *from* M_{edge} *is smaller than* θ*.*

Proof Suppose that for every i, $1 \leq i \leq N$, we have $l_i > \frac{L_i}{2}$. In other words, $l_i = \frac{x_i + L_i}{2}$, where $0 < x_i \leq L_i$. Let $|\mathrm{F}_{centre}|$ and $|\mathrm{F}_{edge}|$ denote the size of F inside M_{centre} and M_{edge}, respectively. w_i is used to denote the length of F inside M_{centre} in the ith dimension. Clearly, $|\mathrm{F}_{centre}| = \prod_{i=1}^{N} w_i$ and $|\mathrm{F}_{edge}| = |\mathrm{F}| - \prod_{i=1}^{N} w_i$. Since M and M_{centre} are identical in shape, we have $\hat{L}_i = \sqrt[N]{a} \cdot L_i$. When F attaches to a corner of M, $w_i = l_i - \frac{L_i - \hat{L}_i}{2} = \frac{x_i + \hat{L}_i}{2}$. However, F can be any place of M, hence we have $w_i \geq \frac{x_i + \hat{L}_i}{2}$.

Clearly, $(x_i + \sqrt[N]{a}L_i) > (\sqrt[N]{a}x_i + \sqrt[N]{a}L_i)$ (because $0 < a < 1$ and $0 < x_i$)

$$\Rightarrow \prod_{i=1}^{N} \frac{x_i + \hat{L}_i}{2} > a \cdot \prod_{i=1}^{N} \frac{x_i + L_i}{2}$$

$$\Rightarrow \prod_{i=1}^{N} \frac{x_i + \hat{L}_i}{2} > a \cdot \prod_{i=1}^{N} l_i$$

$$\Rightarrow \prod_{i=1}^{N} \frac{x_i + \hat{L}_i}{2} > a \cdot |\mathrm{F}|$$

$$\Rightarrow \frac{1}{a|M|} \prod_{i=1}^{N} w_i > \frac{|\mathrm{F}|}{|M|} \left(\text{because } w_i \geq \frac{x_i + \hat{L}_i}{2} \right)$$

$$\Rightarrow \frac{|\mathrm{F}_{\mathrm{centre}}|}{|\mathrm{M}_{centre}|} > \frac{|\mathrm{F}|}{|M|}$$

$$\Rightarrow p > \frac{|\mathrm{F}|}{|M|} = \theta$$

As proved above, $\prod_{i=1}^{N} \frac{x_i + \hat{L}_i}{2} > a \cdot |\mathrm{F}|$

$$\Rightarrow |\mathrm{F}| - \prod_{i=1}^{N} \frac{x_i + \hat{L}_i}{2} < |\mathrm{F}| - a|\mathrm{F}|$$

$$\Rightarrow \frac{1}{|M|}\left(|\mathrm{F}| - \prod_{i=1}^{N} \frac{x_i + \hat{L}_i}{2}\right) < \frac{1-a}{|M|}|\mathrm{F}|$$

$$\Rightarrow \frac{1}{(1-a)|M|}\left(|\mathrm{F}| - \prod_{i=1}^{N} \frac{x_i + \hat{L}_i}{2}\right) < \frac{|\mathrm{F}|}{|M|} \quad \text{(remark: } (1-a) > 0)$$

$$\Rightarrow \frac{1}{(1-a)|M|}\left(|\mathrm{F}| - \prod_{i=1}^{N} w_i\right) < \frac{|\mathrm{F}|}{|M|} \quad \left(\text{because } w_i \geq \frac{x_i + \hat{L}_i}{2}\right)$$

$$\Rightarrow \frac{|\mathrm{F}_{\mathrm{edge}}|}{|\mathrm{M}_{edge}|} < \frac{|\mathrm{F}|}{|M|}$$

$$\Rightarrow q < \frac{|\mathrm{F}|}{|M|} = \theta$$

□

Normally, if we select test cases from M, the chance of detecting failures is θ. If there exists one rectangular failure region, Theorem 3.1 shows that when $\theta > \frac{1}{2^N}$, the chance of detecting failures is higher for test cases selected from M_{centre} than those selected from M_{edge}. This theorem is valid irrespective of the size of M_{centre}.

Since $\frac{1}{2^N}$ decreases exponentially as N increases, a small increase in N will significantly increase the likelihood of satisfying $(\theta > \frac{1}{2^N})$ which gives test cases from M_{centre} a higher chance of detecting failures than those from the whole M. On the other hand, an increase in N will increase the edge preference of FSCS-ART (Chen et al. 2005, 2007). More details about the test distribution of FSCS-ART can be found in Sect. 5.4. These two facts explain why Chen et al. (2005) have the following two observations. First, FSCS-ART performs worse than RT for large values of θ in high dimensional space. Second, the larger N is, the larger the ART F-ratio of FSCS-ART is, and the wider the range of θ where the ART F-ratio of FSCS-ART being greater than 1 is.

3.2 Problem 2

In essence, FSCS-ART tries to keep test cases apart from each other. Its test case is selected from a candidate set, C, according to the selection criterion based on the Euclidean distance between a candidate c and its nearest neighbour in E. The candidate with the maximum distance to its nearest neighbour in E is selected for testing. This way of selecting test cases does not take dimensionality into consideration. Next, we will explain the problems of this simple selection criterion in high dimensional space.

When the input domain is 1 dimensional (1D), no matter where points (inputs) are located, they will all appear on one line. Therefore, merely keeping test cases apart in distance is sufficient to achieve an even-spread of test cases. However, when the input domain is N dimensional (where $N > 1$), the *spatial* distribution of points is more complicated. If FSCS-ART only aims at keeping test cases apart, it cannot fully ensure an even-spread of test cases all over the input domain. Consider two sets of test case distribution in 2D space (Fig. 3). the test cases in Fig. 3a are farther apart from one another than those in Fig. 3b. Intuitively speaking, the dispersion metric measures the largest empty space between points (its formal definition is given in Sect. 5.4). The smaller the dispersion is, the more evenly spread the test cases are. Therefore, Fig. 3a is considered less even-spread than Fig. 3b. However, FSCS-ART tends to produce test case distribution like Fig. 3a rather than Fig. 3b because its test selection criterion does not take dimensionality into consideration, but rather picks the farthest candidate for testing.

As shown in Sect. 2, software failures of Program 1 are only sensitive to Y parameter, not X. Hereafter, we will call these two types of parameters (those related to failures, and those unrelated to failures) "failure-related" and "failure-unrelated" parameters, respectively. Obviously, the larger dimensionality is, the less likely *all* input parameters are failure-related (or equivalently, the more likely *some* parameters are failure-unrelated). FSCS-ART should take this feature into consideration when selecting the best candidates for testing.

Consider testing Program 1 using FSCS-ART, where C consists of two candidates $c_1(c_1^X, c_1^Y)$ and $c_2(c_2^X, c_2^Y)$. Assume that both c_1 and c_2 have an identical distance from their corresponding nearest neighbour in E. In other words, they both are entitled to be the next test case according to the currently used selection criterion in FSCS-ART. Further assume there exists an element e_i of E such that either $e_i^X = c_1^X$ or $e_i^Y = c_1^Y$, while no such a relationship exists between c_2 and any element in E (in other words, c_2 is different from every element of E in all parameters (dimensions)). Even c_1 and c_2 have such different characteristics, FSCS-ART will treat these two candidates equally, and would randomly select any one of them for testing. We, however, argue that c_2 should be preferable to c_1 as the next test case, because of the following reasons.

- Besides keeping test cases apart, intuitively speaking, having test cases different in all dimensions should *cover* larger parts of the input domain than allowing test cases to be similar in some dimensions. Thus, from a *spatial coverage* point of view, c_2 should be preferable to c_1.
- Since failures of Program 1 are only sensitive to Y parameter, if we are unable to detect a failure by a test case t, we know that failure-causing inputs must be different from t with respect to Y parameter. Since it is normally unknown in advance which input parameter is failure-related, in order to effectively detect failures, the next test case is better to be different from E (its elements are unable to reveal failures) as much as possible in each dimension, not just the Euclidean distance. Therefore, c_2 should be preferable to c_1.

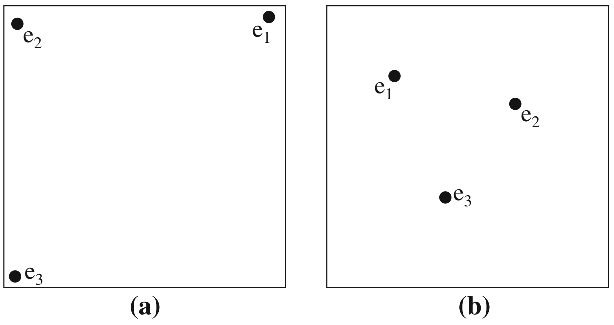


Fig. 3 Different distributions of test cases

In summary, when dimensionality is high, using "Euclidean distance" as the sole selection criterion may generate test cases which are neither evenly spread nor effective in detecting failures. Problem 2 suggests that we should enforce test cases different from each other in all dimensions, while keeping them apart in distance. Our solution to high dimension problems of FSCS-ART will be presented in Sect. 4. Experimental results (Sect. 5) show that our solution can both alleviate Problems 1 and 2.

4 The proposed solution: FSCS-ART with filtering by eligibility (FSCS-ART-FE)

In this section, we provide one solution to high dimension problems of FSCS-ART. The following notations and concepts are required to facilitate our discussion. In N dimensional input domains (I_i denotes each of its dimensions), the coordinates of two points A and B are denoted as (a_1, a_2,…,a_N) and (b_1, b_2,…,b_N), respectively. dist(A, B) is used to denote the Euclidean distance between point A and point B, and dist_i(A, B) is used to denote $|a_i - b_i|$ with respect to I_i. Among all dist_i(A, B), the shortest and the longest distance are denoted as minDist(A, B) and maxDist(A, B), respectively. At last, we define DistRatio(A, B) as the ratio of minDist(A, B) to maxDist(A, B). Obviously, the range value of DistRatio(A, B) is [0, 1].

Consider the same example as discussed in Sect. 3.2. There are two candidates c_1 and c_2 that have the same shortest distance from E; but unlike c_1, the candidate c_2 differs from E with respect to all coordinates. In that example, we have argued that c_2 is more preferable than c_1. Following the same argument, we will choose candidates that have as large DistRatio as possible from all elements of E, as test cases.

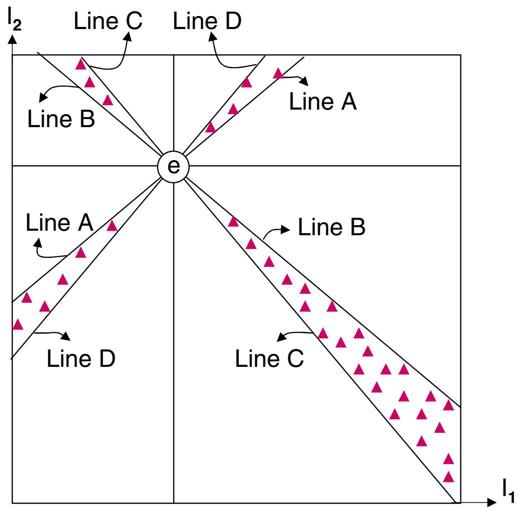
Our enhanced FSCS-ART is basically the same as the original FSCS-ART, but with one additional feature, that is, an eligibility filtering process to ensure that the candidates are far apart from previously executed test cases in terms of "input parameters". An input c is *eligible* if for every e_i of E, DistRatio(c, e_i) is greater than v where v is a value chosen from the range of [0, 1]. For ease of discussion, the condition that determines the eligibility of a candidate is referred as the *eligibility criterion*. In the sequel, we will elaborate the details of our algorithm (namely, FSCS-ART-FE). Without loss of generality, we will illustrate this algorithm using 2D space.

For the sake of explaining the notion of eligible inputs, consider Fig. 4 where e is the only element in E, which is intersected by Lines A, B, C and D having the slope of v, $-v$, $\frac{-1}{v}$ and $\frac{1}{v}$, respectively. In such a scenario, the eligible inputs occupy the dotted regions, and are separated from the ineligible inputs by Lines A, B, C and D.

Next, the impact of v and the size of E (|E|) on the number of eligible inputs is investigated. Suppose the input domain consists of 49 elements and |E| = 1, as shown in Fig. 5. There are 0, 20 and 36 elements out of 49 elements, which are eligible when v = tan(45°), tan(30°) and tan(15°), respectively. Obviously, the number of eligible inputs increases as v decreases. On the other hand, for a fixed v, the growth of E will "exclude" more and more elements from being eligible. As an example of illustration, refer to Fig. 6 where v remains unchanged but the number of elements in E is different (|E| = 1 or 2 in Fig. 6a or b, respectively). As can be seen, the number of eligible inputs will decrease with the increase of |E| if v remains unchanged.

The algorithm of FSCS-ART-FE is given in Fig. 7 where Steps 6–10 are introduced to replace Step 5 of Fig. 2 (algorithm of FSCS-ART). The basic difference is that we need to construct a candidate set C such that all its elements are eligible.

Fig. 4 Eligible inputs (forming the dotted regions)



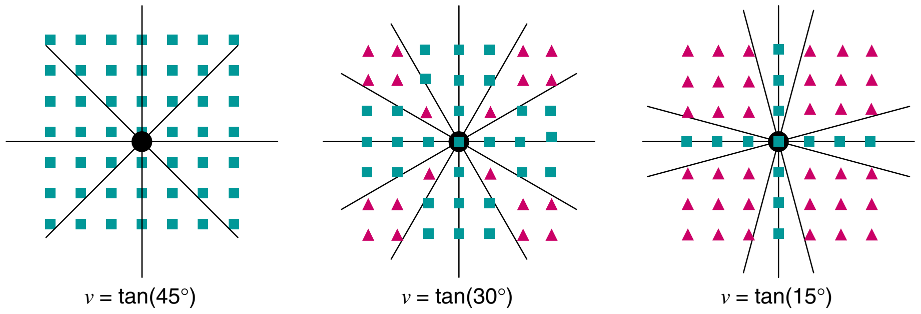
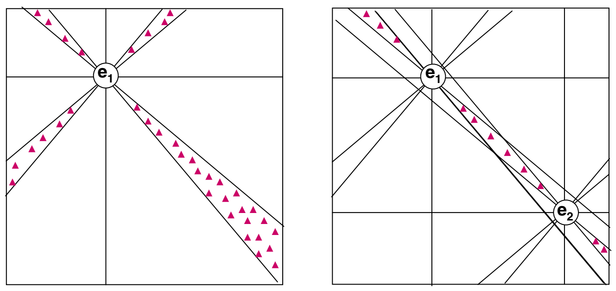


Fig. 5 The relationship between *v* and the number of eligible inputs (triangles and squares represent eligible and illegible inputs, respectively)



(a) $|E| = 1$ **(b)** $|E| = 2$

Fig. 6 The relationship between $|E|$ and the number of eligible inputs (forming the dotted regions)

To use FSCS-ART-FE, the tester needs to set 2 parameters *v* and *r*. The role of *v* has been explained above, and the role of *r* is explained as follows. Since E grows along with the testing, we will eventually reach a situation where it is impossible or too expensive to

```
1.  INPUT $v$ and $r$, where $1 > r > 0$ and $1 \geq v \geq 0$.
2.  $n := 0$, E := { }, C := { }.
3.  Randomly select a test case, t, from the input domain (according
    to the uniform distribution).
4.  $n := n + 1$.
5.  IF t reveals a failure THEN GOTO Step 14; ELSE store t in E.
6.  Randomly generate $k$ inputs to construct C (according to the
    uniform distribution).
7.  FOR $i = 1$ to $k$, DO BEGIN
      examine the eligibility of $c_i$ and mark $c_i$ 'eligible' or 'ineligible'
      accordingly, where $c_i \in C$.
    END
8.  IF all elements of C are eligible THEN GOTO Step 11.
9.  nTrial := 0.
10. REPEAT
      Replace each ineligible $c_i$ by another random input.
      Examine the eligibility of all replacements, and mark them
      'eligible' or 'ineligible' according to $v$.
      nTrial := nTrial + 1.
      IF nTrial = 4 THEN BEGIN
        IF fewer than 70% of candidates are eligible THEN BEGIN
          nTrial := 0
          $v := v \cdot r$.
        END
      END
    UNTIL all $c_i$ of C are eligible.
11. $d := 0$
12. FOR $i = 1$ to $k$, DO BEGIN
      calculate the Euclidean distance $d_i$ between $c_i$ and its nearest
      neighbour in E, where $c_i \in C$.
      IF $d_i > d$ THEN BEGIN
        t := $c_i$
        $d := d_i$.
      END
    END
13. GOTO Step 4.
14. RETURN $n$ and t, and EXIT.
```

Fig. 7 The algorithm of FSCS-ART-FE

construct C. To resolve this problem, we propose to dynamically relax the eligibility criterion during the testing process when an insufficient number of eligible candidates has been generated after g attempts. The role of r, which is within the range (0, 1), is to reduce the value of v (by resetting v to be $v \cdot r$) so that the eligibility criterion will be relaxed.

Since the filtering effect will disappear when v becomes 0, v should be adjusted gradually and only when necessary. Clearly, the larger g is, the less frequently v is to be adjusted. After g attempts to incrementally construct C, if fewer than $p\%$ of elements inside C are eligible, we consider the current eligibility criterion too strict and thus there is a need to reduce v. Note that in this study, g and p were arbitrarily set to 4 and 70, respectively.

The filtering process in FSCS-ART-FE checks the eligibility of candidates according to their DistRatios. Since minDist and maxDist for all 1D inputs are identical, any candidate selected at random will satisfy the eligibility criterion. As a result, FSCS-ART-FE and FSCS-ART are equivalent in 1D input domains.

5 Analysis into FSCS-ART-FE

In this section, we investigate how well FSCS-ART-FE can resolve the high dimension problems of FSCS-ART. This study consists of the following. First, we study the ART F-ratio of FSCS-ART-FE using simulations. In addition, we compare the test case distributions of FSCS-ART-FE and FSCS-ART. Unless otherwise specified, the designs of all experiments in this section are the same as those described in Sect. 2.

5.1 Impact of key settings on the effectiveness of FSCS-ART-FE

We conducted simulations to investigate the impact of v and r on the effectiveness of FSCS-ART-FE. First, we set both v and r to 0.5 (so $v \approx \tan(26.57°)$) and applied FSCS-ART-FE to the first simulation settings reported by Chen et al. (2005), where the failure pattern consisted of a single square (or cubic) failure region, the failure rate (θ) varied from 1 to 0.00005, and dimensionality (N) varied from 2 to 4. For comparison purpose, the ART F-ratios of FSCS-ART previously reported by Chen et al. (2005) are also reproduced in this section. Note that it is unnecessary to study FSCS-ART-FE in 1D input domains because it is equivalent to FSCS-ART in 1D space.

The results of this study are summarized in Fig. 8, from which we have the following observations.

- Like FSCS-ART, the ART F-ratio of FSCS-ART-FE depends on N and θ.
- When θ is large, the ART F-ratio of FSCS-ART-FE is smaller than the corresponding ART F-ratio of FSCS-ART.
- As θ decreases, the difference between the ART F-ratios of FSCS-ART-FE and FSCS-ART decreases.
- For a larger N, there exists a wider range of θ where the ART F-ratio of FSCS-ART-FE is smaller than that of FSCS-ART.

This study shows that the process of filtering does make FSCS-ART-FE more effective than FSCS-ART. FSCS-ART-FE outperforms FSCS-ART when θ is large, but the improvement decreases as θ decreases. The rationale is explained as follows. It is known that for a smaller θ, more test cases are required to detect the first failure (that is, a larger F-measure), and hence there will be a larger E just prior to detecting the first failure. Since v tends to decrease as E grows, FSCS-ART-FE will become more and more FSCS-ART-like as testing proceeds. As a consequence, the smaller the θ is, the closer the ART F-ratios of FSCS-ART-FE and FSCS-ART are. Having said that, for a large N, FSCS-ART-FE can outperform FSCS-ART across a wider range of θ.

We conducted further experiments with the following settings. In these experiments, the input domain is set to be 4D.

- v is either 0.9 ($\approx \tan(41.99°)$), 0.5 ($\approx \tan(26.57°)$) or 0.1 ($\approx \tan(5.71°)$)
- r is either 0.9, 0.5 or 0.1

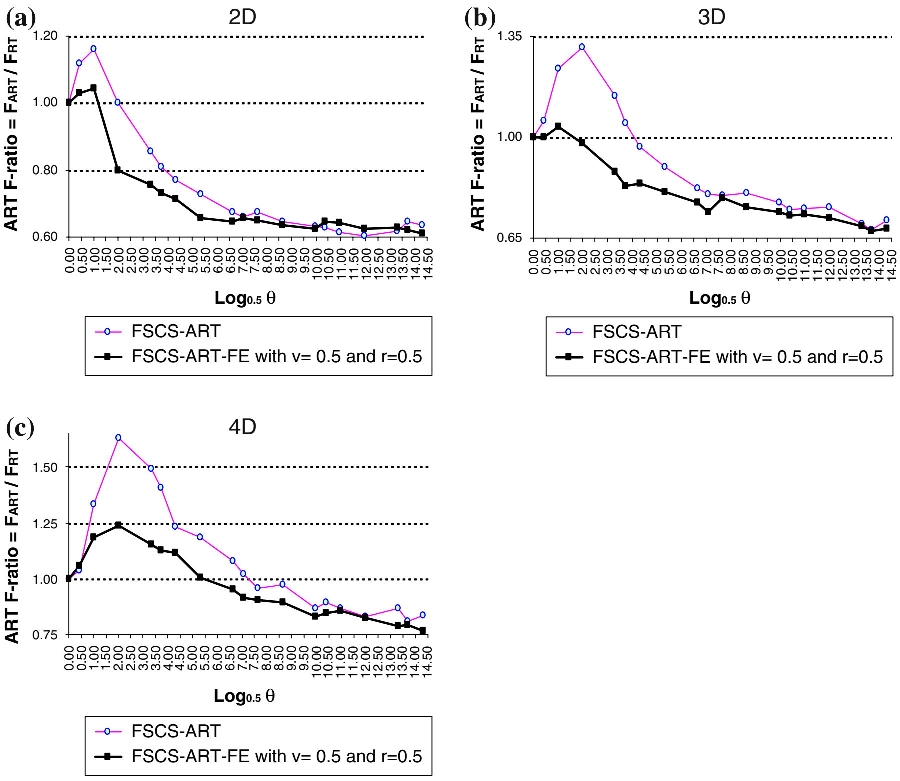


Fig. 8 Comparison of FSCS-ART and FSCS-ART-FE when the failure pattern is a single square failure region

There are nine different scenarios in total. We group the results into Fig. 9. Based on these data, we have the following observations:

- The larger r is, the smaller the ART F-ratio is.
- The impact of v on the ART F-ratio of FSCS-ART-FE decreases with the decrease of r.
- For a given r, when v is larger than a certain value, increasing v will not significantly affect the ART F-ratio of FSCS-ART-FE.

As mentioned before, it is desirable to have test cases different from each other as much as possible in all dimensions, in order to better cover the whole input domain and increase the chance of detecting failures. The eligibility criterion imposed during the filtering process serves this purpose. Note that the eligibility criterion depends on v which in turn depends on r. Since the effect of r is accumulative because of its repeated use to adjust v, it is understandable that r has a more dominating impact than v on the effectiveness of FSCS-ART-FE as seen in the second observation.

Since v affects how much the next test case could differ from E in all dimensions, the initial value of v cannot be too small; otherwise, FSCS-ART-FE will behave just like FSCS-ART. Nevertheless, our last observation shows that a large initial value of v does not imply a small ART F-ratio. We further investigated the impact of v and observed that a large initial v could seldom generate a sufficient number of eligible candidates within a

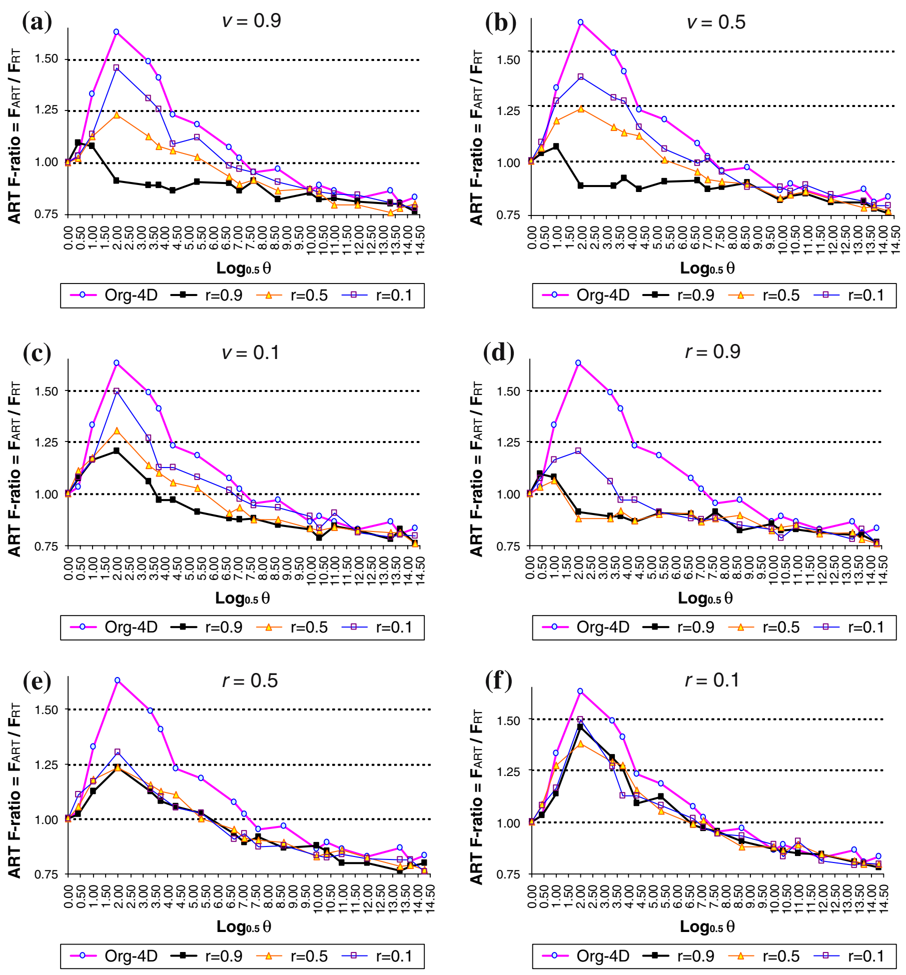


Fig. 9 Impact of key settings on the effectiveness of FSCS-ART-FE

permitted number of trails. As a result, a large initial *v* is almost certain to be adjusted immediately after its first use.

This study shows that the most dominating factor affecting the effectiveness of FSCS-ART-FE is *r*. In summary, an effective FSCS-ART-FE requires a sufficiently large *v*, and more importantly, a significantly large *r*. Hence, we will set *v* and *r* both to 0.9 in the rest of our experimental study.

5.2 Impact of failure patterns on the effectiveness of FSCS-ART-FE

FSCS-ART-FE is an enhanced version of FSCS-ART. It is interesting to repeat the same investigation of Chen et al. (2005) into the impact of failure patterns on the effectiveness of FSCS-ART-FE.

First, we applied FSCS-ART-FE to the second simulation settings reported by Chen et al. (2005), where θ was either 0.005, 0.001 or 0.0005, and N was either 2 or 3. The failure pattern is set to a strip failure region (a long rectangle or cuboid), and the parameter of α is used to determine the compactness of the failure region. In 2D space, the width and length of a rectangle are in the ratio of 1:α, while in 3D space, the edge lengths of a cuboid are in the ratio of 1:α:α. The smaller α is, the more compact the failure region is. The simulation results are reported in Fig. 10. Our study shows that in general, FSCS-ART-FE behaves similarly as FSCS-ART, whose effectiveness depends on compactness of a failure region.

Next, we applied FSCS-ART-FE to the third simulation settings by Chen et al. (2005), where the number of square failure regions varied from 1 to 100. The simulation results are reported in Fig. 11. Our study shows that FSCS-ART-FE behaves similarly as FSCS-ART, whose effectiveness depends on the number of failure regions.

The findings of this investigation are consistent with those by Chen et al. (2005). Both simulations show that FSCS-ART-FE behaves similarly as FSCS-ART, whose ART F-ratio depends on the failure pattern. Furthermore, both FSCS-ART and FSCS-ART-FE perform best when the failure pattern is a single square failure region. As the number of failure regions increases or the compactness of failure regions decreases, their ART F-ratios increase and approach to a constant.

5.3 Impact of the number of failure-unrelated parameters on the effectiveness of FSCS-ART-FE

As mentioned in Sect. 3.2, the Euclidean distance metric by itself is not a good test case selection criterion for FSCS-ART when there are failure-unrelated parameters. In fact, the higher dimensionality is, the more likely *some* input parameters are failure-unrelated. Hence, we propose to make test cases different in all dimensions while keeping them apart. This triggers the development of FSCS-ART-FE. It is important to examine the effectiveness of FSCS-ART-FE with the presence of failure-unrelated parameters.

We conducted a simulation to investigate the effect of the number (m) of failure-unrelated parameters on the effectiveness of FSCS-ART-FE. As shown in Sect. 3.2, when there exist failure-unrelated parameters, the failure pattern will consist of failure regions that span across *failure-unrelated dimensions*.

As an example of illustration, consider a 3D rectangular input domain, whose edge length is L_i in dimension i, and a rectangular failure region, whose edge length is l_i in dimension i. Suppose θ is 0.01. If program failures are unrelated to the 1st and 2nd parameters (so $m = 2$), then we have l_1:l_2:l_3 = L_1:L_2:0.01L_3. If failures are unrelated to the 1st parameters (so $m = 1$), then we have l_1:l_2:l_3 = L_1:0.1L_2:0.1L_3.

Obviously, for any faulty program, there must exist at least one failure-related parameter, therefore, m must be smaller than dimensionality (N). In our simulation, one single rectangular failure region was assumed to reside in a rectangular N dimensional input domain, where N varied from 2 to 4. The edge length of the failure region in each *failure-related dimension* was $\sqrt[N-m]{\theta}$, where θ was either 0.01, 0.005, 0.001 or 0.0005.

The simulation results for various combinations of N and m are summarized in Table 1, which shows that FSCS-ART-FE outperforms FSCS-ART when there exist failure-unrelated parameters, and FSCS-ART-FE has the most significant improvement over FSCS-ART when $m = N - 1$.

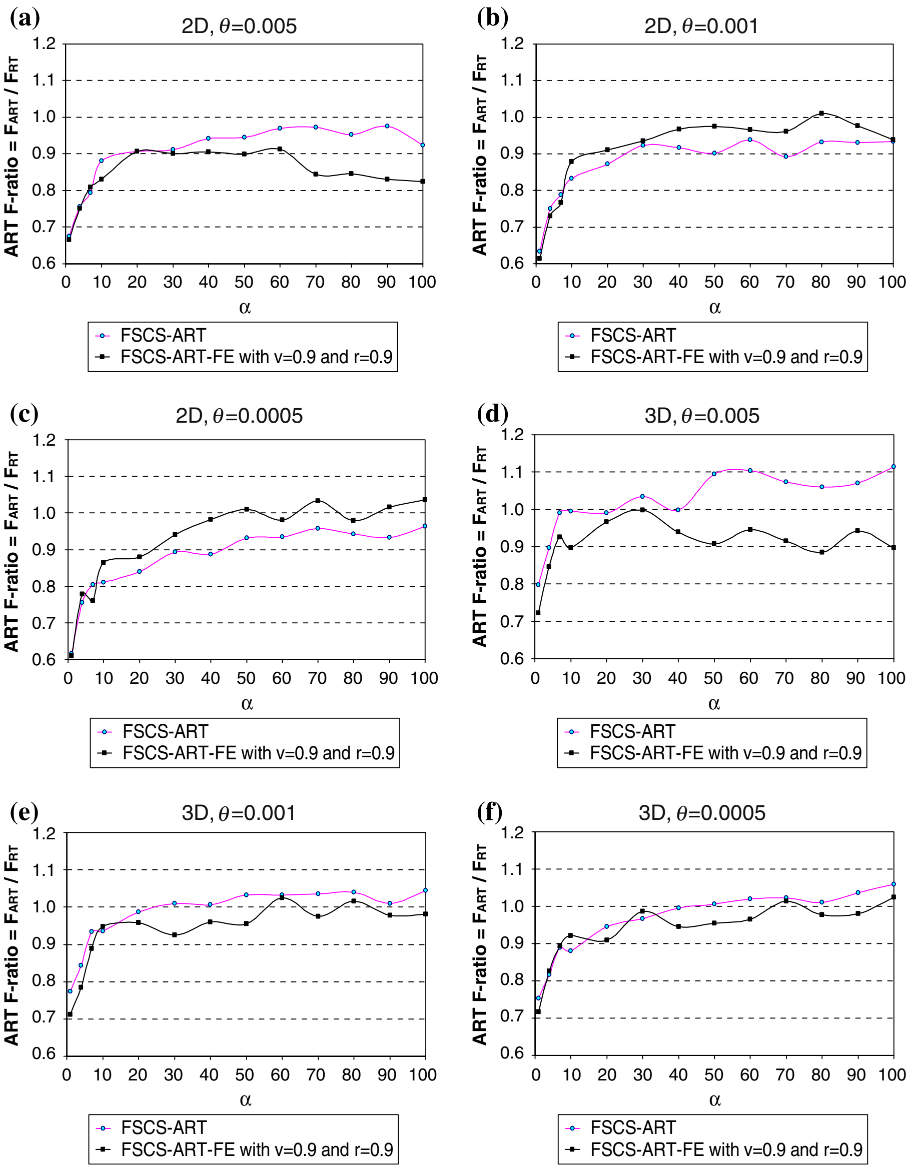


Fig. 10 Comparison of FSCS-ART and FSCS-ART-FE when the failure pattern is a strip failure region with different degrees of compactness

5.4 Test case distribution of FSCS-ART-FE

As explained in Sect. 3.1, FSCS-ART may not ensure a truly even-spread of test cases because it simply enforces test cases far apart from each other in distance. Furthermore, it has been explained in Sect. 3.2 how the edge preference of FSCS-ART affects the effectiveness of FSCS-ART in high dimensional input domains. Therefore, we aim to assess the test case distribution of FSCS-ART-FE in this section.

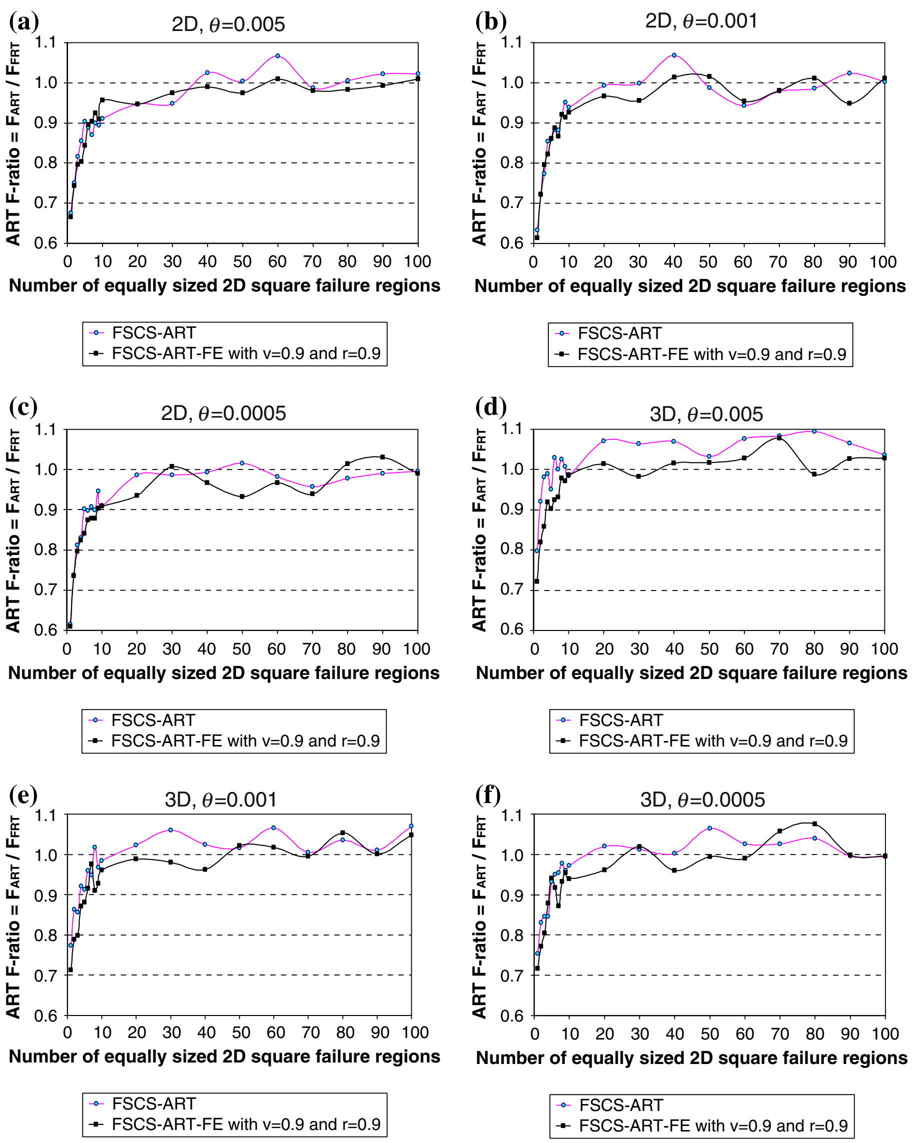


Fig. 11 Comparison of FSCS-ART and FSCS-ART-FE when the failure pattern consists of some failure regions

Chen et al. (2007) used three metrics to measure the test case distributions (the distribution of E inside the input domain (M)) of various ART algorithms. Among these metrics, the discrepancy and dispersion (denoted as $M_{Discrepancy}$ and $M_{Dispersion}$, respectively) are two commonly used metrics for measuring the equidistribution of sample points (Branicky et al. 2001), while the $M_{Edge:Centre}$ metric was particularly introduced to measure the edge preference of some ART algorithms (Chen et al. 2007). These metrics are formally defined as follows.

Table 1 Impact of the number of failure-unrelated parameters on effectiveness of FSCS-ART-FE

N	m	Algorithm	F_{ART}			
			$\theta = 0.01$	$\theta = 0.005$	$\theta = 0.001$	$\theta = 0.0005$
2	1	FSCS-ART	96.08	189.48	996.84	1975.79
		FSCS-ART-FE with $v = 0.9$ and $r = 0.9$	68.00	137.67	704.77	1490.81
3	2	FSCS-ART	103.12	197.64	1020.31	2009.87
		FSCS-ART-FE with $v = 0.9$ and $r = 0.9$	71.89	148.86	745.14	1535.78
	1	FSCS-ART	93.21	192.79	973.18	1987.53
		FSCS-ART-FE with $v = 0.9$ and $r = 0.9$	84.14	184.34	958.56	1801.86
4	3	FSCS-ART	98.37	197.98	970.15	2022.04
		FSCS-ART-FE with $v = 0.9$ and $r = 0.9$	73.86	149.58	809.42	1546.10
	2	FSCS-ART	108.97	214.50	1044.55	2126.33
		FSCS-ART-FE with $v = 0.9$ and $r = 0.9$	101.79	196.92	1012.63	2003.46
	1	FSCS-ART	104.81	208.93	983.61	1927.09
		FSCS-ART-FE with $v = 0.9$ and $r = 0.9$	90.70	185.15	942.24	1932.91

$$M_{Discrepancy} = \max_{i=1\ldots m}\left|\frac{|\mathrm{E}_i|}{|\mathrm{E}|} - \frac{|\mathrm{M}_i|}{|\mathrm{M}|}\right| \tag{1}$$

where M_i denotes a randomly defined subset of M; and E_i denotes a subset of E residing in M_i. Like (Chen et al. 2007), m is set to 1000 in the following simulations.

$$M_{Dispersion} = \max_{i=1\ldots|\mathrm{E}|} dist(e_i, nn(e_i, \mathrm{E})) \tag{2}$$

where $\mathrm{e}_i \in \mathrm{E}$ and $nn(e_i, \mathrm{E})$ denotes the nearest neighbour of e_i in E.

$$M_{Edge:Centre} = \frac{|\mathrm{E}_{\mathrm{edge}}|}{|\mathrm{E}_{centre}|} \tag{3}$$

where E_{edge} and E_{centre} denote two disjoint subsets of E residing in M_{edge} and M_{centre}, respectively; and $|\mathrm{M}_{edge}| = |\mathrm{M}_{centre}| = 0.5\,\mathrm{M}$.

$M_{Discrepancy}$ indicates whether all subregions of M have an equal density of the points. $M_{Dispersion}$ indicates whether any point in E is surrounded by a very large empty spherical region (containing no points other than itself). $M_{Edge:Centre}$ indicates whether there is an equal density of points in M_{centre} and M_{edge}. E is considered reasonably equidistributed if the $M_{Discrepancy}$ is close to 0, $M_{Dispersion}$ is small, and $M_{Edge:Centre}$ is close to 1. An edge preference (or a centre preference) is said to occur when $M_{Edge:Centre} > 1$ (or $M_{Edge:Centre} < 1$). Clearly, in order for $M_{discrepancy}$ to be small, the $M_{Edge:Centre}$ should be close to 1; otherwise, different parts of M have different densities of points.

We repeated the simulations of Chen et al. (2007) on FSCS-ART-FE. The comparisons among RT, FSCS-ART and FSCS-ART-FE using $M_{Edge:Centre}$, $M_{Discrepancy}$ and $M_{Dispersion}$ are summarized in Figs. 12–14, respectively. From these data, we have the following observations:

- In all cases, FSCS-ART-FE has a smaller $M_{Edge:Centre}$ than FSCS-ART. FSCS-ART-FE even has a centre preference in 2D space.

- In 2D space, $M_{Discrepancy}$ of FSCS-ART-FE is larger than that of FSCS-ART, but the relationship is reversed for 3D and 4D space.
- In general, FSCS-ART-FE has a smaller $M_{Dispersion}$ than FSCS-ART.

The first observation is consistent with our expectation, that is, the edge preference of FSCS-ART can be alleviated by the FSCS-ART-FE. The second observation can be explained as follows. As explained above, if $M_{Edge:Centre}$ is far away from 1, $M_{Discrepancy}$ cannot be very small. This explains why FSCS-ART has a larger $M_{Discrepancy}$ than RT in 3D and 4D space. In 2D space, the value of $1/M_{Edge:Centre}$ for FSCS-ART-FE is much larger than the value of $M_{Edge:Centre}$ for FSCS-ART, that is, the centre preference of FSCS-ART-FE is more serious than the edge preference of FSCS-ART. Therefore, it is intuitively expected that FSCS-ART-FE has a larger $M_{Discrepancy}$ than FSCS-ART. FSCS-ART-FE has a lower edge preference than FSCS-ART in 3D and 4D space, so the former has a smaller $M_{Discrepancy}$ than the latter.

Chen et al. have analysed the relationship between the test case distribution and effectiveness of ART algorithms, and concluded that $M_{Dispersion}$ should be more appropriate than $M_{Discrepancy}$ to measure the even-spread of test cases of ART algorithms (Chen et al. 2007). Together with their conclusion, our last observation implies that test cases generated by FSCS-ART-FE are generally more evenly spread than those generated by FSCS-ART.

6 Discussion and conclusion

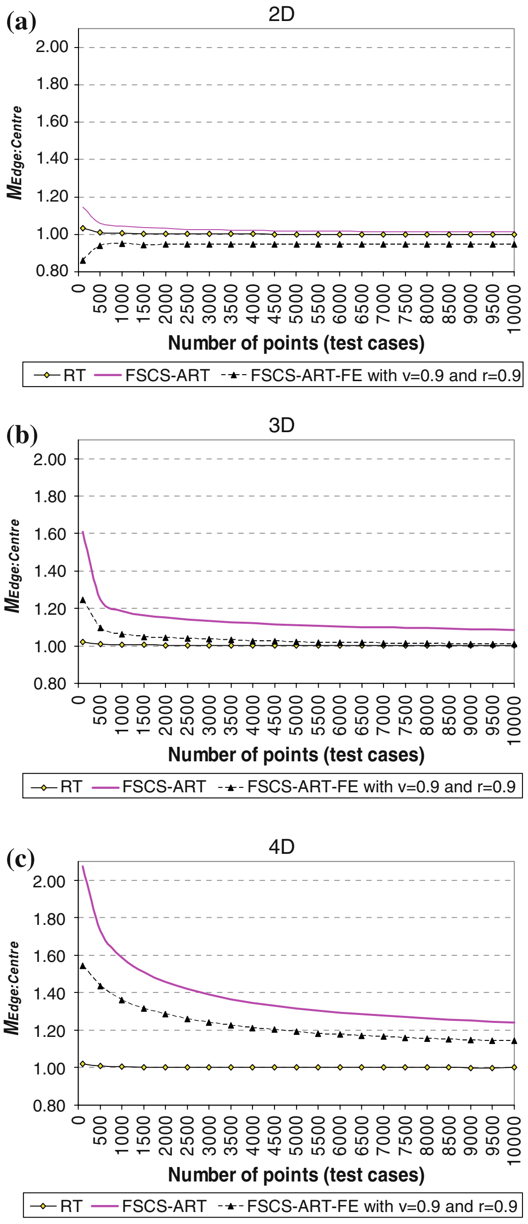
ART was originally proposed to improve the fault-detection effectiveness of RT, especially when failure-causing inputs are clustered together. Recently, it has been observed that the effectiveness of some ART algorithms deteriorates with the increase of dimensionality. In this article, we analysed the high dimension problems of FSCS-ART, and proposed a new algorithm, namely FSCS-ART-FE to address these problems.

FSCS-ART-FE uses a filtering process to enforce test cases far apart from each other in all dimensions. A by-product of this additional filtering process is a lower edge preference (one of the causes deteriorating the effectiveness of FSCS-ART in high dimensional space). Our study shows that FSCS-ART-FE not only has a lower edge preference but also lower dispersion. In other words, test cases generated by FSCS-ART-FE are generally more evenly spread those generated by FSCS-ART.

It has been observed that FSCS-ART-FE behaves similarly as FSCS-ART, but the effectiveness deterioration in higher dimensional space is less significant for FSCS-ART-FE than FSCS-ART. As dimensionality increases, the ART F-ratio of FSCS-ART-FE is smaller than that of FSCS-ART in a wider range of failure rates. In addition, when there exist failure-unrelated parameters (a common situation in high dimensional input domains), FSCS-ART-FE outperforms FSCS-ART.

Our investigation into higher dimensionality confirms that FSCS-ART-FE is an enhancement of FSCS-ART. Its ART F-ratios in 10D and 15D space (summarized in Fig. 15a and b) are smaller than those of FSCS-ART, and the data trends are consistent with the observation given in Sect. 5.1. The test case distribution of FSCS-ART-FE is considered more evenly spread than FSCS-ART with respect to all three metrics (note that distribution data are not presented here due to the page limit).

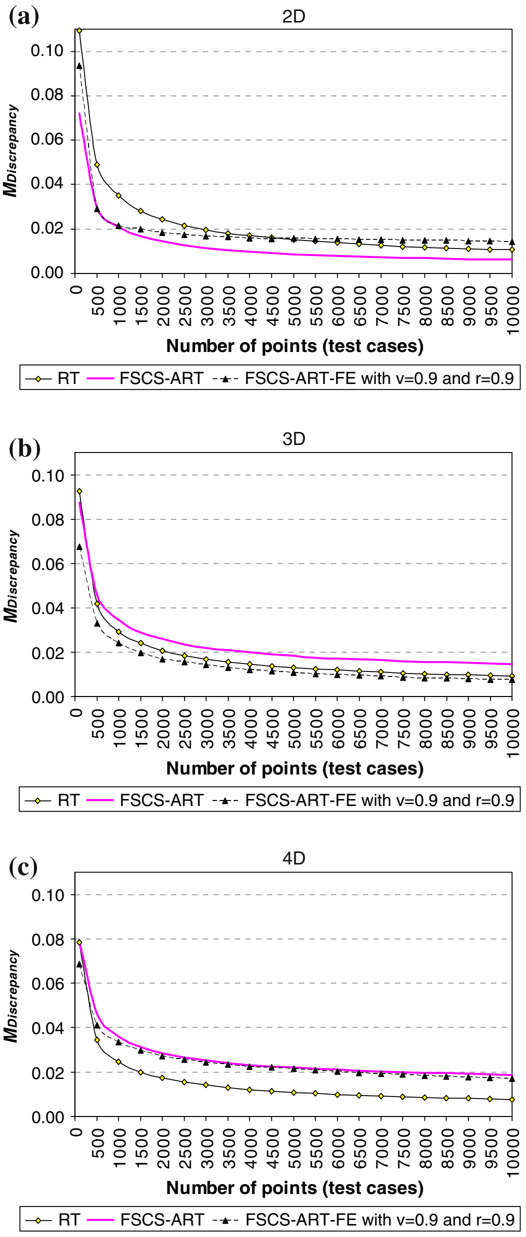
Fig. 12 Comparison of RT, FSCS-ART and FSCS-ART-FE using $M_{Edge:Centre}$



In this article, we only work on the settings of v and r in FSCS-ART-FE. We did not investigate various settings of g, p and the adjustment criteria for v. It is worthwhile to study the impact of these settings and find out how these settings can tune up the test methodology to meet the increase of dimensionality.

Other ART algorithms could face similar problems as FSCS-ART. For example, *restricted random testing* (RRT) (Chan et al. 2006) also has the preference of selecting test cases from the boundary part of the input domain, and uses the Euclidean distance as the

Fig. 13 Comparison of RT, FSCS-ART and FSCS-ART-FE using $M_{Discrepancy}$



metric of selecting the next test case. Therefore, our study brings insight into the improvement of other ART algorithms in high dimensional space. Our future work will be on these relevant algorithms.

In our simulations, we studied the behaviour of FSCS-ART-FE without restricting the location of failure regions because failure regions in real life programs can be in any place within the input domains. When studies are carried out on real life faulty programs, since each faulty program is a special real life case, we have to select a great amount of sample

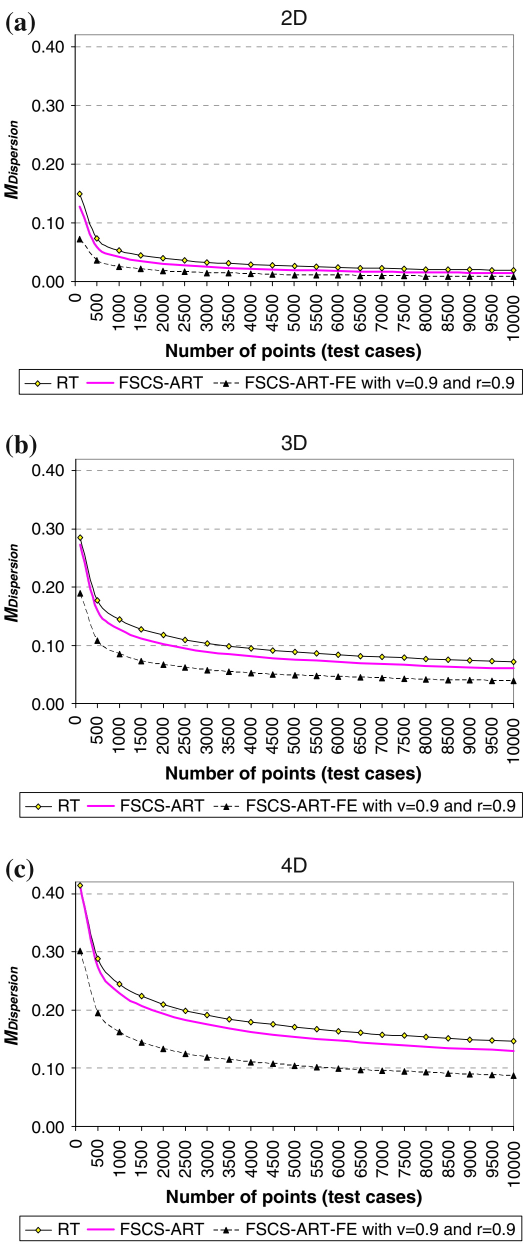


Fig. 14 Comparison of RT, FSCS-ART and FSCS-ART-FE using $M_{Dispersion}$

programs in order to conduct a meaningful study. This empirical study is worthwhile but very labour-intensive, and hence should be part of our future work.

Our study shows that FSCS-ART-FE can improve FSCS-ART not only in the case of high dimensional space, but also in the case of having failure-unrelated parameters. It should be noted that both cases are common in real life programs. Therefore, we recommend that FSCS-ART-FE should be used instead of FSCS-ART whenever possible.

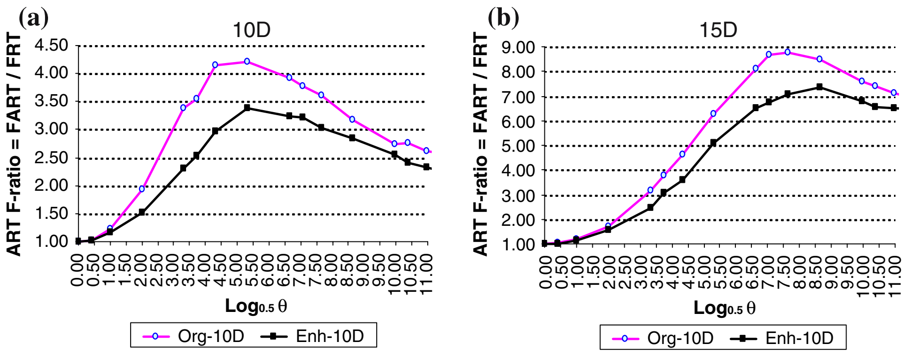


Fig. 15 Comparison between FSCS-ART and FSCS-ART-FE in high dimensional space

Acknowledgment This research project is supported by an Australian Research Council Discovery Grant (DP0880295).

References

Ammann, P. E., & Knight, J. C. (1988). Data diversity: An approach to software fault tolerance. *IEEE Transactions on Computers, 37*(4), 418–425.

Bellman, R. (1957). *Dynamic programming*. New Jersey: Princeton University Press.

Bird, D. L., & Munoz, C. U. (1983). Automatic generation of random self-checking test cases. *IBM Systems Journal, 22*(3), 229–245.

Bishop, P. G. (1993). The variation of software survival times for different operational input profiles. In *Proceedings of the 23rd International Symposium on Fault-Tolerant Computing (FTCS-23)* (pp. 98–107). IEEE Computer Society Press.

Branicky, M. S., LaValle, S. M., Olson, K., & Yang, L. (2001). Quasi-randomized path planning. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation* (pp. 1481–1487).

Chan, K. P., Chen, T. Y., & Towey, D. (2006). Restricted random testing: Adaptive random testing by exclusion. *International Journal of Software Engineering and Knowledge Engineering, 16*(4), 553–584.

Chen, T. Y., Kuo, F.-C., & Liu, H. (2007). On test case distributions of adaptive random testing. In *Proceedings of the 19th International Conference on Software Engineering and Knowledge Engineering (SEKE'07)* (pp. 141–144). Boston.

Chen, T. Y., Kuo, F.-C., & Merkel, R. (2006). On the statistical properties of testing effectiveness measures. *Journal of Systems and Software, 79*(5), 591–601.

Chen, T. Y., Kuo, F.-C., Merkel, R. G., & Ng, S. P. (2004). Mirror adaptive random testing. *Information and Software Technology, 46*(15), 1001–1010.

Chen, T. Y., Kuo, F.-C., & Zhou, Z. Q. (2005). On the Relationships between the distribution of failure-causing inputs and effectiveness of adaptive random testing. In *Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'05)* (pp. 306–311). Taipei, Taiwan.

Chen, T. Y., & Merkel, R. (2008). An upper bound on software testing effectiveness. *ACM Transaction on Software Engineering Methodologies*.

Chen, T. Y., Tse, T. H., & Yu, Y. T. (2001). Proportional sampling strategy: A compendium and some insights. *Journal of Systems and Software, 58*(1), 65–81.

Cobb, R., & Mills, H. D. (1990). Engineering software under statistical quality control. *IEEE Software, 7*(6), 45–54.

Dabóczi, T., Kollár, I., Simon, G., & Megyeri, T. (2003). Automatic testing of graphical user interfaces. In *Proceedings of the 20th IEEE Instrumentation and Measurement Technology Conference 2003 (IMTC'03)* (pp. 441–445). Vail, CO.

Finelli, G. B. (1991). NASA software failure characterization experiments. *Reliability Engineering and System Safety, 32*(1–2), 155–169.

Forrester, J. E., & Miller, B. P. (2000). An empirical study of the robustness of Windows NT applications using random testing. In *Proceedings of the 4th USENIX Windows Systems Symposium* (pp. 59–68). Seattle.

Godefroid, P., Klarlund, N., & Sen, K. (2005). Dart: Directed automated random testing. In *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'05)* (pp. 213–223).

Hamlet, R. (2002). Random testing. In J. Marciniak (Ed.), *Encyclopedia of software engineering* (2nd edn.). Wiley.

Kuo, F.-C., Chen, T. Y., Liu, H., & Chan, W. K. (2007). Enhancing adaptive random testing in high dimensional input domain. In *Proceedings of the 22nd Annual ACM Symposium on Applied Computing (SAC'07)* (pp. 1467–1472). ACM Press.

Mak, I. K. (1997). On the effectiveness of random testing. Master's thesis, Department of Computer Science, University of Melbourne.

Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation, 8*(1), 3–30.

Mayer, J. (2005). Lattice-based adaptive random testing. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE'05)* (pp. 333–336). New York: ACM Press.

Miller, E. (2005). Website testing, http://www.soft.com/eValid/Technology/White.Papers/website.testing.html, Software Research, Inc. Accessed 21 Feb 2008.

Miller, B. P., Fredriksen, L., & So, B. (1990). An empirical study of the reliability of UNIX utilities. *Communications of the ACM, 33*(12), 32–44.

Miller, B. P., Koski, D., Lee, C. P., Maganty, V., Murthy, R., Natarajan, A., & Steidl, J. (1995). Fuzz revisited: A re-examination of the reliability of UNIX utilities and services. Tech. Rep. CS-TR-1995-1268, University of Wisconsin.

Myers, G. J., Sandler, C., Badgett, T., & Thomas, T. M. (2004). *The art of software testing* (2nd edn.). New Jersey: Wiley.

Nyman, N. In defense of monkey testing: Random testing can find bugs, even in well engineered software, http://www.automationjunkies.com/resources/nyman_monkey.rtf, Microsoft Corporation. Accessed 21 Feb 2008.

Regehr, J. (2005). Random testing of interrupt-driven software. In *Proceedings of the 5th ACM International Conference on Embedded software (EMSOFT'05)* (pp. 290–298). New York, NY: ACM Press.

Sen, K., Marinov, D., & Agha, G. (2005). CUTE: A concolic unit testing engine for C. In *Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE-13)* (pp. 263–272). New York, NY: ACM Press.

Slutz, D. (1998). Massive stochastic testing of SQL. In *Proceedings of the 24th International Conference on Very Large Databases (VLDB'98)* (pp. 618–622).

Yoshikawa, T., Shimura, K., & Ozawa, T. (2003). Random program generator for Java JIT compiler test system. In *Proceedings of the 3rd International Conference on Quality Software (QSIC'03)* (pp. 20–24). IEEE Computer Society Press.

Author Biographies

Fei-Ching Kuo is a Lecturer at the Faculty of Information and Communication Technologies in Swinburne University of Technology, Australia. She received her PhD degree in Software Engineering, and BSc (Honours) in Computer Science, both from the Swinburne University of Technology, Australia. Her current research interests include software testing, debugging and project management.

Tsong Yueh Chen is a Chair Professor of Software Engineering at the Faculty of Information and Communication Technologies in Swinburne University of Technology, Australia. He received his PhD degree in Computer Science from the University of Melbourne, MSc and DIC in Computer Science from Imperial College of Science and Technology, and BSc and MPhil from The University of Hong Kong. His current research interests include software testing and debugging, software maintenance and software design.

Huai Liu is a PhD student at the Faculty of Information and Communication Technologies in Swinburne University of Technology. He received his MEng in Communications and Information System in 2003, and BEng in Physioelectronic Technology in 1998, both from Nankai University. His current research interests include software testing and telecommunications.

Wing Kwong Chan is currently a Lecturer at City University of Hong Kong, Hong Kong. He received his PhD degree in Software Engineering from The University of Hong Kong in 2004. Before returning to the academia, he was a software professional in the industry for more than 10 years. He publishes papers in international journals and conferences including CACM, FSE and ICSE. His research interests are software testing, validation and verification, debugging and design techniques in both traditional and pervasive computing settings.