Enhancing adaptive random testing through partitioning by edge and centre

Chen, Tsong Yueh; Kuo, Fei Ching; Liu, Huai

Chen, T. Y., Kuo, F. C., & Liu, H. (2007). Enhancing adaptive random testing through partitioning by edge and centre. Proceedings of the 18th Australian Software Engineering Conference (ASWEC'07), 265–273. https://doi.org/10.1109/ASWEC.2007.20
Document Version: Accepted Manuscript

Published Version: https://doi.org/10.1109/ASWEC.2007.20

Enhancing Adaptive Random Testing through Partitioning by Edge and Centre

Tsong Yueh Chen Fei-Ching Kuo Huai Liu*
Faculty of Information and Communication Technologies
Swinburne University of Technology
Hawthorn Victoria 3122, Australia
{*tchen, dkuo, hliu*}*@ict.swin.edu.au*

Abstract

Random Testing (RT) is a simple but widely used software testing method. Recently, an approach namely Adaptive Random Testing (ART) was proposed to enhance the fault-detection effectiveness of RT. The basic principle of ART is to enforce random test cases as evenly spread over the input domain as possible. A variety of ART methods have been proposed, and some research has been conducted to compare them. It was found that some ART methods have a preference of selecting test cases from edges of the input domain over from the centre. As a result, these methods may not perform very well under some situations. In this paper, we propose an approach to alleviating the edge preference. We also conducted some simulations and the results confirm that our new approach can improve the effectiveness of these ART methods.

1. Introduction

Random Testing (RT) is a fundamental software testing method, which selects test cases randomly from the *input domain* (the set of all possible inputs of the program under test) [11, 18]. Due to its advantages, such as simplicity, efficiency and randomness, RT has been popularly applied in industry. For example, RT has been used to examine the reliability of some UNIX utility programs [16, 17], and it was found that a great number of programs were crashed by random test data. Moreover, RT was also applied in many automatic testing tools, such as those developed by Microsoft [19], IBM [2] and Bell Lab [10].

It has been observed that *failure-causing inputs* (program inputs that can reveal failures) tend to cluster together [1, 3, 9]. Some researchers [8, 14] found that under such a situation, the performance of RT can be significantly enhanced by evenly spreading generated test cases over the whole input domain. This approach was named as *Adaptive Random Testing* (ART). Based on their work, many ART methods have been proposed, and some typical examples include *Fixed-Sized-Candidate-Set ART* (FSCS-ART) [8, 14], *Restricted Random Testing* (RRT) [4], and *Lattice-based ART* [15]. These methods have been experimentally evaluated and it was confirmed that ART can use fewer test cases to detect the first failure than RT when failure-causing inputs are clustered into contiguous regions (namely *failure regions* [1]).

Recently, some research has been conducted to compare some ART methods [12], and it was pointed out that although all ART methods have the same aim, that is, evenly spreading test cases, their performances are different from one another because they distribute test cases using different approaches. One of the most important observations of the research is that FSCS-ART and RRT, which provide the best performances when failure rate is small, prefer to select test cases from the edge part of the input domain rather than from the central part. This preference, however, deteriorates the performances of FSCS-ART and RRT as the failure rate and dimension increase, and it even can make FSCS-ART and RRT less effective than the original RT under some situations [6, 7, 12].

In this paper, we study the *edge preference* of a particular ART method, namely FSCS-ART. We propose a new approach to alleviating the preference, and thus to enhancing the performance of FSCS-ART. The structure of the paper is introduced as follows. Section 2 gives some background information of FSCS-ART. Section 3 introduces how we measure the edge preference of a testing method, explains our approach to alleviating the edge preference, and compares the fault-detection effectiveness of our approach and the original FSCS-ART. Finally, Section 4 concludes the paper.

*Corresponding author

2. Preliminaries

In *Fixed-Sized-Candidate-Set ART* (FSCS-ART) [8], two sets of test cases are maintained: the *executed set*, which stores all test cases already executed but without revealing any failure, denoted by $E = \{e_1, e_2, \cdots, e_n\}$; and the *candidate set*, which contains k randomly generated inputs, denoted by $C = \{c_1, c_2, \cdots, c_k\}$, where k is fixed throughout the testing process. A candidate will be selected as the next test case if it has the largest distance to its nearest neighbour in E. The algorithm of FSCS-ART is given in Figure 1.

1. Set $n = 0$, $E = \{\}$ and $C = \{\}$.
2. Randomly generate a test case e from the input domain, according to uniform distribution.
3. Test the program with e as the test case.
4. **while** (e does not reveal a failure)
5. Store e into E.
6. Increment n by 1.
7. Randomly generate k candidates from the input domain, according to uniform distribution, and store them into C.
8. **for** each candidate $c_j \in C$, where $j = 1, \cdots, k$
9. Calculate the distance d_j between c_j and its nearest neighbour in E.
10. **end_for**
11. Find $c_b \in C$ such that $d_b \geq d_j$, where $j = 1, \cdots, k$.
12. Set $e = c_b$.
13. Test the program with e as the test case.
14. **end_while**
15. Report the detected failure and exit.

Figure 1. The algorithm of FSCS-ART

It has been pointed out in [6, 7, 12] that FSCS-ART has a preference of selecting test cases from the edge part of the input domain over from the central part. In [12], some ART methods were compared in terms of the test cases distribution. The edge preference of these methods was measured by a metric, namely $M_{Edge:Centre}$, defined as the ratio of the number of test cases inside the subdomain D_{Edge} to the number of test cases inside the subdomain D_{Centre}, where D_{Centre} is located in the centre of the input domain, D_{Edge} is right outside D_{Centre}, and the sizes of D_{Centre} and D_{Edge} are equal. These two subdomains in 2-dimensional (abbreviated as *2D*) space are illustrated in Figure 2. It was found that $M_{Edge:Centre}$ for FSCS-ART is always greater than 1, which confirms the edge preference of FSCS-ART.

Many studies have been conducted to compare FSCS-ART with RT [8, 14] and other ART methods [12] in terms of their F-measure (the expected number of test cases required to detect the first failure). It is often assumed that all inputs can be repeatedly selected (known as the *selection with replacement* [13]) and have an equal chance of being selected (known as the *uniform distribution* [11]). Therefore, the expected F-measure of RT (denoted by F_{RT}) is equal to $1/\theta$, where θ denotes failure rate, the ratio of the number of failure-causing inputs to the number of all possible inputs. Since F-measure of ART (denoted by F_{ART}) depends on many factors, theoretical study of F_{ART} is extremely difficult. Hence, the majority of the previous research studied F_{ART} through simulations. In each simulation, the failure rate θ and the failure pattern (the shapes of failure regions together with their distribution over the input domain) were predefined [5, 6, 7]. When a point is generated inside a failure region by ART, it is said that a failure has been revealed. The simulation was repeated for a sufficient number (S) of times to ensure that F_{ART} is accurate within a certain confidence level and a certain accuracy range. The details of calculating S can be found in [5].

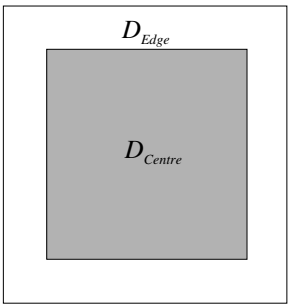


Figure 2. D_{Edge} and D_{Centre} in a 2D input domain

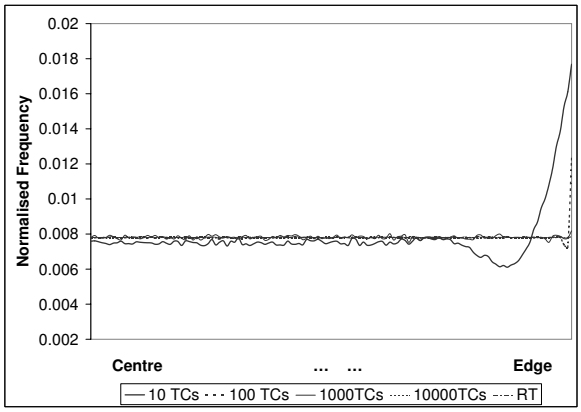
Since ART is to enhance the fault-detection effectiveness of RT, the effectiveness of ART is often measured in terms of ART F-ratio (defined as F_{ART}/F_{RT}). Like all these studies, we will adopt ART F-ratio in this paper for measuring the enhancement of ART over RT.

3. Enhancing FSCS-ART by alleviating the Edge Preference

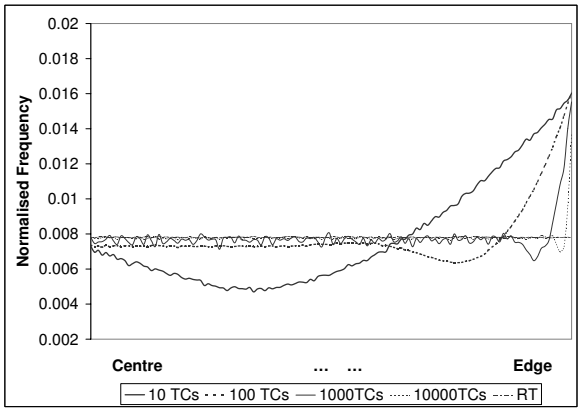
3.1. Measuring the edge preference of a testing method

In this paper, we will measure the edge preference of a testing method in a slightly different way from $M_{Edge:Centre}$ in [12] in order to describe the distribution of test cases in a more precise way. The following illustrates how we measure the edge preference of FSCS-ART.

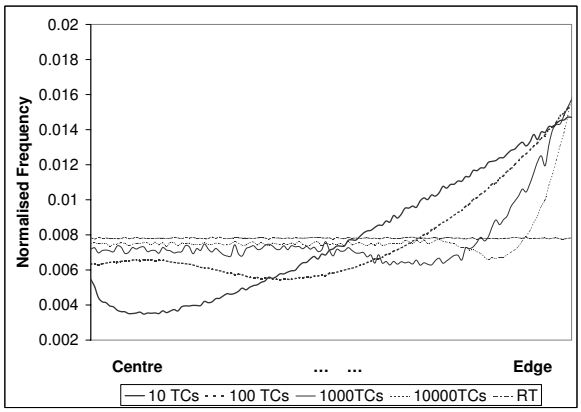
The input domain was first partitioned into 128 equal-sized and disjoint subdomains (instead of 2 subdomains in [12]) from the edge to the centre of the input domain. Then, test cases were generated by FSCS-ART in 1D, 2D, 3D and 4D space, respectively. We separately obtained the first n test cases, where $n = \{10, 100, 1000, 10000\}$, and recorded the related number of points in each subdomain. Such processes were repeated for a sufficient number of times to obtain a reliable average frequency with a confidence level of 95% and $\pm 5\%$ accuracy range, and finally we got four groups of values. For ease of comparison, we also conducted the above simulations using pure RT. It was found that no matter what n is, the test cases of RT are always uniformly distributed, as theoretically expected. The empirical frequency distributions for test cases of FSCS-ART with various n are shown in Figure 3, in which, x- and y-axes denote the subdomain's location and the normalised frequency of test cases inside each subdomain, respectively.The figure confirms the edge preference of FSCS-ART, and it is clearly shown that the preference becomes more and more serious with the increase of the dimension of the input domain.



3.a Test cases distribution for 1D FSCS-ART



3.b Test cases distribution for 2D FSCS-ART



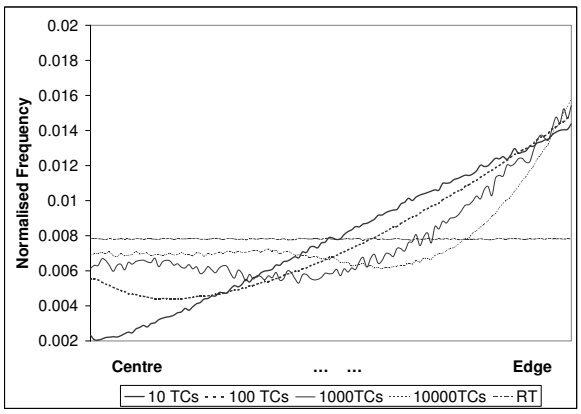
3.c Test cases distribution for 3D FSCS-ART

3.2. Our Approach

The new approach alleviates the edge preference of FSCS-ART by introducing a new partitioning method, that is, partitioning the input domain into some equal-sized partitions from the edge to the centre of the input domain. Test cases are evenly allocated among these partitions. Without loss of generality, we will illustrate this approach using a two-dimensional input domain, as shown in Figure 4, where the input domain is partitioned into four equal-sized partitions D_1, D_2, D_3 and D_4, from the edge to the centre, respectively.

Suppose that the first test case t_1 is randomly generated from the whole input domain and happens to be located inside D_2. For the second test case to be selected, the original FSCS-ART generates six candidates, c_1, c_2, c_3, c_4, c_5 and c_6, as shown in Figure 4. However, since c_2 is inside the same partition as t_1, our approach requires a replacement for c_2, say c_2' which is not located inside the same partition as t_1. Then, the second test case t_2 is selected from c_1, c_2' (instead of c_2), c_3, c_4, c_5 and c_6 according to the original FSCS-ART, and finally we got $t_2 = c_2'$. We term the new approach as *FSCS-ART with Partitioning by Edge and Centre* (ECP-FSCS-ART).

There are two methods of partitioning the input domain. One method is to dynamically partition the input domain, that is, the number of partitions is varying with the number of executed test cases. The other method



3.d Test cases distribution for 4D FSCS-ART

Figure 3. Frequency distribution of test cases for FSCS-ART

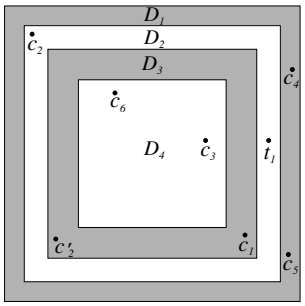


Figure 4. Illustration of the new approach

is static partitioning, that is, the number of partitions is fixed throughout the testing process. These two methods, namely *Dynamic ECP-FSCS-ART* and *Static ECP-FSCS-ART*, are shown in Figures 5 and 6, respectively.

As shown in Figure 5, Dynamic ECP-FSCS-ART partitions the input domain more and more precisely with the increase of the number of test cases (Statement 7). Since the number of partitions is always larger than the number of executed test cases ($n+1 > n$), at least one partition will not contain any executed test case. Therefore, the candidate set can be constructed in such a way that all candidates are from "blank" partitions (Statement 8), and thus the next test case will be from a partition where there is no executed test case. This will prevent test cases from being allocated more frequently in certain subdomains, and hence alleviate the edge preference of FSCS-ART.

Static ECP-FSCS-ART partitions the input domain at the beginning of testing (Statement 2 of Figure 6), and this partitioning strategy is reserved throughout the testing process. For the candidate set to be constructed, the partitions which contain the fewest test cases are first found, and then all candidates are generated only from these partitions(Statement 8). Briefly speaking,

1. Set $n = 0$, $E = \{\}$ and $C = \{\}$.
2. Randomly generate a test case e from the input domain, according to uniform distribution.
3. Test the program with e as the test case.
4. **while** (e does not reveal a failure)
5. Store e into E.
6. Increment n by 1.
7. Divide the input domain into $n+1$ disjoint partitions D_i, from the edge to the centre of the input domain, where all D_i have the same size, and $i = 1, \cdots, n+1$.
8. Randomly generate k candidates from partitions where there does not exist any test case, and store them into C.
9. **for** each candidate $c_j \in C$, where $j = 1, \cdots, k$
10. Calculate the distance d_j between c_j and its nearest neighbour in E.
11. **end_for**
12. Find $c_b \in C$ such that $d_b \geq d_j$, where $j = 1, \cdots, k$.
13. Set $e = c_b$.
14. Test the program with e as the test case.
15. **end_while**
16. Report the detected failure and exit.

Figure 5. The algorithm of Dynamic ECP-FSCS-ART

1. Set $n = 0$, $E = \{\}$ and $C = \{\}$.
2. Divide the input domain into m disjoint partitions D_i, from the edge to the centre of the input domain, where all D_i have the same size, and $i = 1, \cdots, m$.
3. Randomly generate a test case e from the input domain, according to uniform distribution.
4. Test the program with e as the test case.
5. **while** (e does not reveal a failure)
6. Store e into E.
7. Increment n by 1.
8. Randomly generate k candidates from partitions where the number of associated tests is the fewest, and store them into C.
9. **for** each candidate $c_j \in C$, where $j = 1, \cdots, k$
10. Calculate the distance d_j between c_j and its nearest neighbour in E.
11. **end_for**
12. Find $c_b \in C$ such that $d_b \geq d_j$, where $j = 1, \cdots, k$.
13. Set $e = c_b$.
14. Test the program with e as the test case.
15. **end_while**
16. Report the detected failure and exit.

Figure 6. The algorithm of Static ECP-FSCS-ART

Static ECP-FSCS-ART makes all partitions contain almost the same number of test cases, and hence ensures that test cases will not be allocated more frequently in certain subdomains; as a result, the edge preference of FSCS-ART can be alleviated.

3.3. Experiment 1

Let us first check to what extent our approach alleviates the edge preference of FSCS-ART. We repeated the simulations in Section 3.1 using Dynamic ECP-FSCS-ART, and Static ECP-FSCS-ART with 100 and 1000 predefined partitions. The size of candidate set (k) is set to 10 and the shape of the input domain is set to square. The simulations results are shown in Figures 7, 8 and 9. For ease of discussion, we will use *Static-m ECP-FSCS-ART* to denote the method of Static ECP-FSCS-ART with m partitions.

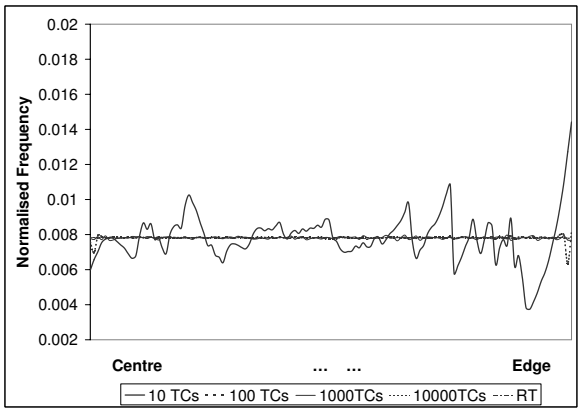
Based on these figures, we have the following two observations.

- Dynamic ECP-FSCS-ART does distribute test cases more evenly than the original FSCS-ART, but there is still a small edge preference.
- When the number of test cases is small, neither Static-100 nor Static-1000 ECP-FSCS-ART methods can alleviate the edge preference very well.

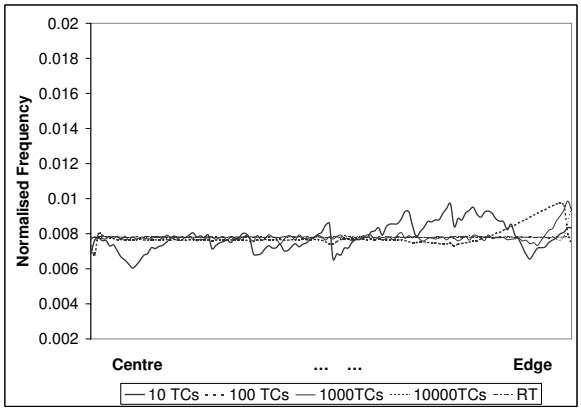
The first observation is understandable, because although all candidates are from "blank" partitions, the selected candidate will still have a higher probability of being close to the edge of the input domain than other candidates due to the nature of FSCS-ART. As far as the second observation is concerned, when the number of executed test cases is much smaller than the number of partitions, there are too many partitions with the fewest test cases (no test case). As a result, the candidate set is effectively constructed from the whole input domain, that is, at the beginning of Static ECP-FSCS-ART with a large number of partitions, the testing is almost the same as the original FSCS-ART. On the other hand, when the number of executed test cases exceeds the number of partitions, the edge preference should have been alleviated. As a justification, refer to Figures 8 and 9, where it can be observed that once the number of executed test cases exceeds 100 and 1000, respectively, the edge preference is significantly alleviated.

3.4. Experiment 2

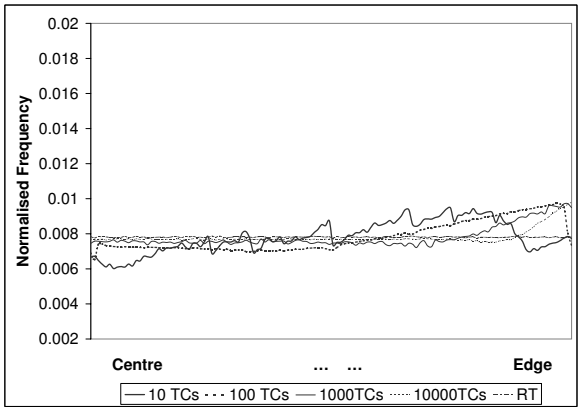
We have demonstrated that our approach can alleviate the edge preference of FSCS-ART. It is also im-



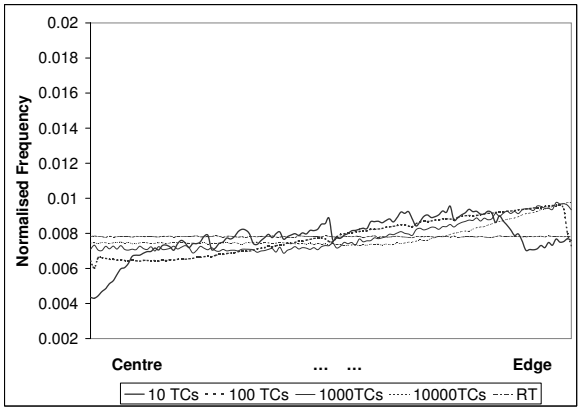
7.a Test cases distribution for 1D Dynamic ECP-FSCS-ART



7.b Test cases distribution for 2D Dynamic ECP-FSCS-ART

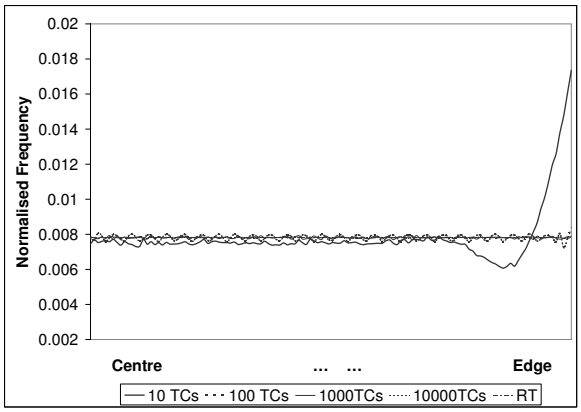


7.c Test cases distribution for 3D Dynamic ECP-FSCS-ART

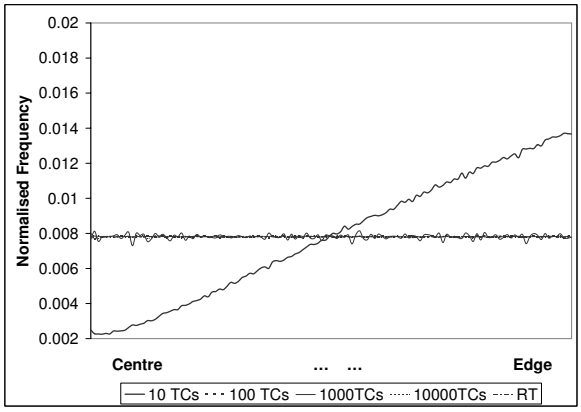


7.d Test cases distribution for 4D Dynamic ECP-FSCS-ART

Figure 7. Frequency distribution of test cases for Dynamic ECP-FSCS-ART

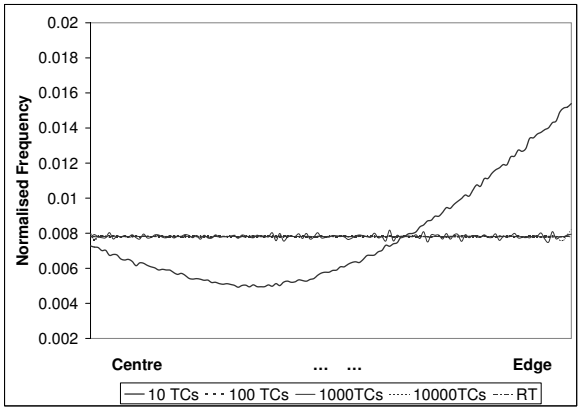


8.a Test cases distribution for 1D Static-100 ECP-FSCS-ART

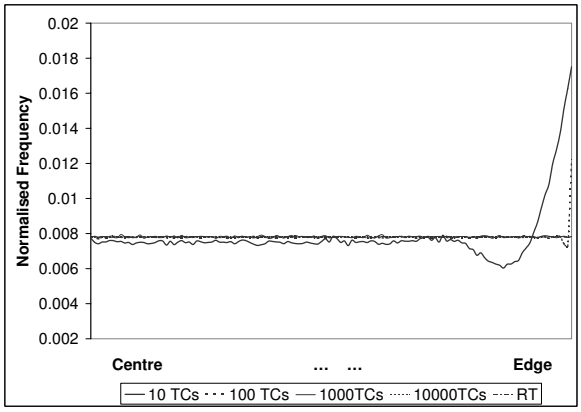


8.d Test cases distribution for 4D Static-100 ECP-FSCS-ART

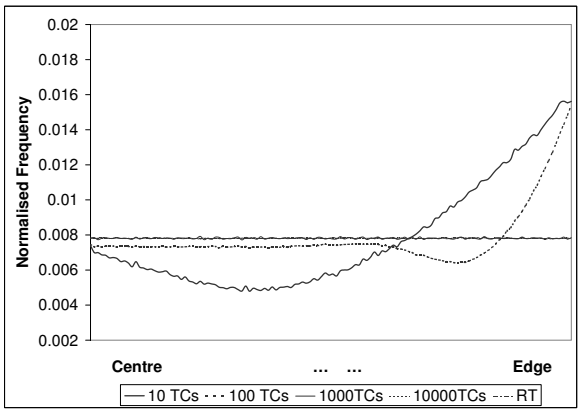
Figure 8. Frequency distribution of test cases for Static-100 ECP-FSCS-ART



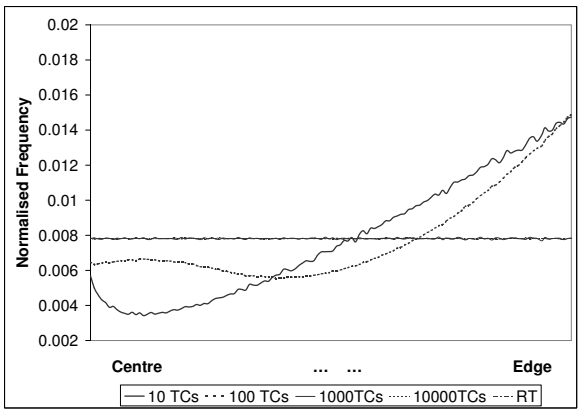
8.b Test cases distribution for 2D Static-100 ECP-FSCS-ART



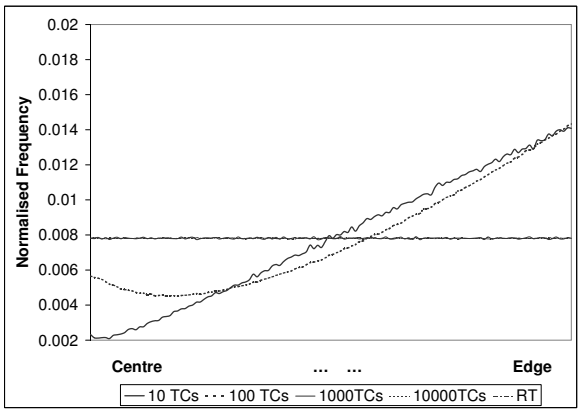
9.a Test cases distribution for 1D Static-1000 ECP-FSCS-ART



9.b Test cases distribution for 2D Static-1000 ECP-FSCS-ART



9.c Test cases distribution for 3D Static-1000 ECP-FSCS-ART



9.d Test cases distribution for 4D Static-1000 ECP-FSCS-ART

Figure 9. Frequency distribution of test cases for Static-1000 ECP-FSCS-ART

portant to examine whether the effectiveness of FSCS-ART can be improved, so we further conducted a series of simulations to investigate the performance of ECP-FSCS-ART. The parameters for these simulations are predefined as follows.

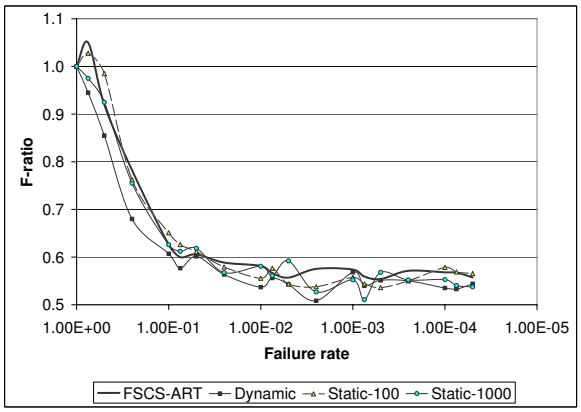
- Testing methods: Dynamic ECP-FSCS-ART, Static-100 ECP-FSCS-ART and Static-1000 ECP-FSCS-ART.
- Shape of input domain: square.
- Dimension of input domain: 1, 2, 3 and 4.
- Failure region: a single square failure region is randomly placed inside the input domain.
- Failure rate: 1, 0.75, 0.5, 0.25, 0.1, 0.075, 0.05, 0.025, 0.01, 0.0075, 0.005, 0.0025, 0.001, 0.00075, 0.0005, 0.00025, 0.0001, 0.000075, and 0.00005.
- The size of candidate set (k) $= 10$.
- Confidence Level: 95%.
- Accuracy range: $\pm 5\%$

It should be noted that the setting of this experiment is exactly the same as Experiment 1 in [6, 7]. The results of these simulations are reported in Figure 10. The previous simulations results on the performance of FSCS-ART are also plotted for ease of comparison.

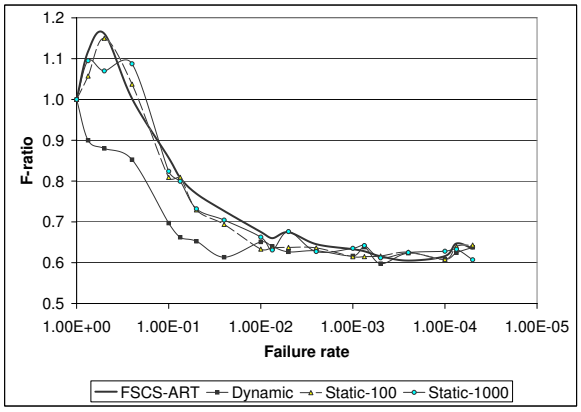
We have the following findings based on the data.

- ECP-FSCS-ART outperforms the original FSCS-ART under the following situations.
  - $\star$ When the dimension of the input domain is high or the failure rate is high, the performance of Dynamic ECP-FSCS-ART is better than that of the original FSCS-ART.
  - $\star$ When the dimension of the input domain is high and failure rates are smaller than 0.01 and 0.001, respectively, the performances of Static-100 and Static-1000 ECP-FSCS-ART methods are better than that of the original FSCS-ART.
- ECP-FSCS-ART is comparable to the original FSCS-ART method under other situations.

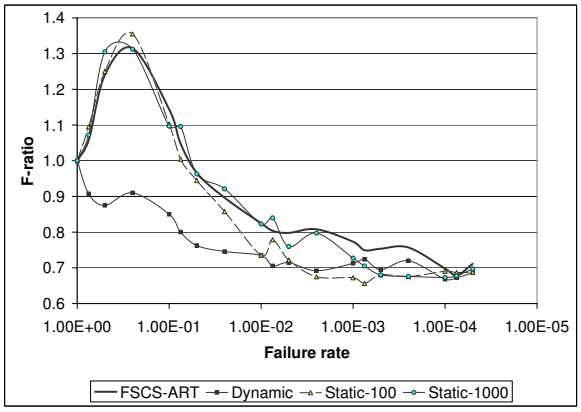
The finding about Dynamic method is intuitively expected by this paper, that is, FSCS-ART can be enhanced by alleviating the edge preference. What was found about Static method is consistent with the observations in Section 3.3. The edge preference is not alleviated very well when the number of test cases is much



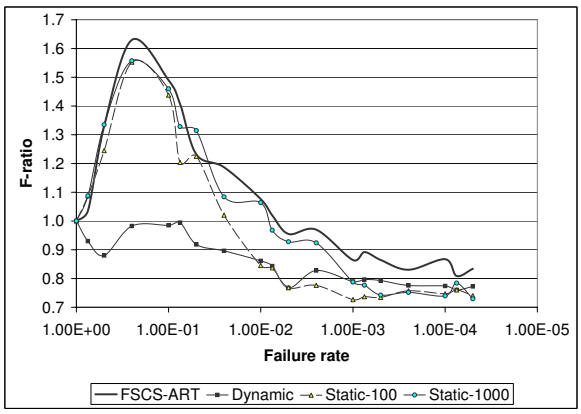
10.a ECP-FSCS-ART's performance in 1D space



10.b ECP-FSCS-ART's performance in 2D space



10.c ECP-FSCS-ART's performance in 3D space



10.d ECP-FSCS-ART's performance in 4D space

Figure 10. Performances of Dynamic ECP-FSCS-ART and Static ECP-FSCS-ART

smaller than the number of partitions. Consequently, only after the number of test cases exceed some specific values ($1/0.01 = 100$ and $1/0.001 = 1000$ for Static-100 and Static-1000 ECP-FSCS-ART methods, respectively), can Static ECP-FSCS-ART improve the performance a lot.

Based on the results in the previous two experiments, we can find a strong correlation between the edge preference and the performance of ECP-FSCS-ART. Since Dynamic ECP-FSCS-ART alleviates the edge preference from the beginning of the testing process, it has a smaller F-measure than the original FSCS-ART for high failure rates and dimensions, and can constantly perform well across various scenarios. It should be noted that although there still exists an edge preference in Dynamic ECP-FSCS-ART, the preference is insignificant and does not have any adverse impact on the performance of Dynamic ECP-FSCS-ART, as shown in Figure 10. On the other hand, Static ECP-FSCS-ART can alleviate the edge preference only when the number of executed test cases exceeds the number of partitions. This is why Static ECP-FSCS-ART has almost the same F-ratio as the original FSCS-ART when the failure rate is adequately high. In summary, both Static and Dynamic ECP-FSCS-ART methods perform better than the original FSCS-ART, but Dynamic method provides a more consistent improvement than Static method on the fault-detection effectiveness of FSCS-ART.

4. Conclusion

ART is an approach to enhancing the fault-detection effectiveness of RT. Recently, it has been pointed out that some ART methods have a preference of selecting test cases from the edge part of the input domain over

from the central part. This edge preference can deteriorate the performance of these ART methods. In this paper, we propose a new approach, namely ECP-FSCS-ART, to alleviating the edge preference, and present two algorithms, Dynamic and Static ECP-FSCS-ART, to implement the approach. Simulations were conducted and the results show that the new approach can improve the effectiveness of FSCS-ART, especially when the failure rate is high or the dimension of the input domain is high. The basic idea of our approach is that for an ART method which prefers to select test cases from certain locations of the input domain, we can alleviate the preference by introducing a corresponding partitioning scheme and fairly allocating test cases among all the partitions. Apparently, this idea is not only applicable and useful to FSCS-ART, but also to all other ART methods whose test case generation is biased toward certain locations of the input domain.

Acknowledgment

This research project is supported by an Australian Research Council Discovery Grant (DP0557246).

References

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

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

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

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

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

[6] T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou. On favourable conditions for adaptive random testing. *Accepted to appear in International Journal of Software Engineering and Knowledge Engineering*.

[7] T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou. 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 2005))*, pages 306–311, Taipei, Taiwan, 2005.

[8] T. Y. Chen, H. Leung, and I. K. Mak. Adaptive random testing. In *Proceedings of the 9th Asian Computing Science Conference*, volume 3321 of *Lecture Notes in Computer Science*, pages 320–329, 2004.

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

[10] P. Godefroid, N. Klarlund, and K. Sen. DART: directed automated random testing. In *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2005)*, pages 213–223, 2005.

[11] R. Hamlet. Random testing. In J. Marciniak, editor, *Encyclopedia of Software Engineering*. John Wiley & Sons, second edition, 2002.

[12] F.-C. Kuo. *On adaptive random testing*. PhD thesis, Faculty of Information and Communications Technologies, Swinburne University of Technology, 2006.

[13] H. Leung, T. H. Tse, F. T. Chan, and T. Y. Chen. Test case selection with and without replacement. *Inf. Sci.*, 129(1-4):81–103, 2000.

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

[15] J. Mayer. Lattice-based adaptive random testing. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005)*, pages 333–336, New York, USA, 2005. ACM.

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

[17] B. P. Miller, D. Koski, C. P. Lee, V. Maganty, R. Murthy, A. Natarajan, and J. Steidl. Fuzz revisited: A reexamination of the reliability of UNIX utilities and services. Technical Report CS-TR-1995-1268, University of Wisconsin, 1995.

[18] G. J. Myers. *The Art of Software Testing*. Wiley, New York, second edition, 1979.

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