

Enhancing Adaptive Random Testing in High Dimensional Input Domains

F. -C. Kuo^{*}
Faculty of Informatics,
University of Wollongong,
Wollongong, NSW, Australia
dkuo@uow.edu.au

T. Y. Chen H. Liu
Faculty of Information and
Communication Technologies,
Swinburne University of
Technology,
Hawthorn, VIC, Australia
{tchen,hliu}@ict.swin.edu.au

W. K. Chan
Department of Computer
Science
City University of Hong Kong
Tat Chee Avenue, Hong Kong
wkchan@cs.cityu.edu.hk

ABSTRACT

Adaptive random testing (ART) is an enhancement of random testing (RT). It can detect failures more effectively than RT when failure-causing inputs are clustered. Having test cases both randomly selected and evenly spread is the key to the success of ART. Recently, it has been found that the dimensionality of the input domain could have an impact on the effectiveness of ART. The effectiveness of some ART methods may deteriorate when the dimension is high. In this paper, we work on one particular ART method, namely Fixed-Sized-Candidate-Set ART (FSCS-ART) and show how it can be enhanced for high dimensional domains. Since the cause of the problems for FSCS-ART may also be valid for some other ART methods, our solutions to the high dimension problems of FSCS-ART may be applicable for improving other ART methods.

Categories and Subject Descriptors

D.2.5 [Software Engineering]: Testing and Debugging

Keywords

Software testing, Random testing, Adaptive random testing, Fix-Sized-Candidate-Set ART, high dimension problem

1. INTRODUCTION

A major challenge of software development is to establish the correctness of the software. *Software testing* is a major activity of software quality assurance. It assures software quality by actively and systematically detecting faults in order to reduce occurrences of software failures.

There exist many testing methods to guide the selection of inputs for test (known as the *test cases*). One simple method is called *Random Testing* (RT), in which test cases

^{*}Corresponding author

are selected in a random manner from the *input domain* (that is, the set of all possible inputs) [11, 18]. 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 does not need 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. Because of these merits, RT has been widely used in industry for detecting failures [7, 8, 10, 15, 16, 17, 19, 20, 21]. In [15, 16], for example, Miller *et al.* used RT techniques to investigate the reliability of standard UNIX utilities. In 1990 [15], they reported that 25% to 30% of these utilities could be crashed. They repeated and extended the study five years later [16], and found that a large number of utility programs were still crashed. RT has also been applied on different software, such as websites [17] and SQL database systems [20]. In [20], a system so-called Random Generation of SQL was developed and it was indicated that the system could “generate valid SQL statements 1 million times faster than a human and execute them”.

Despite of the popularity, there are still some criticism against RT. Since it does not make use of any information to generate test cases, it has been regarded by some researchers as ineffective. However, as reported in [1, 2, 9], failure-causing inputs tend to cluster together, and it has been proposed that in such a situation, the *fault-detection effectiveness* of RT can be enhanced by enforcing an even spread of random test cases [13]. This approach was named as *Adaptive Random Testing* (ART). There are various methods to evenly spread test cases, and all studies conducted [3, 5, 6, 14] so far confirmed that ART has used fewer test cases to reveal the first failure than RT when failure-causing inputs do cluster into contiguous regions (known as the *failure regions* [1]). Since both RT and ART use randomly generated inputs as test cases, it is intuitive to consider ART as an alternative to RT.

Recently, Chen *et al.* [6] have found that the effectiveness of *some* ART methods may deteriorate with the *dimension of the input domain* (that is, the number of input parameters). This problem, known as the high dimension problem, gets worse, when an ART method has a preference for its test cases being generated more frequently in the edges and corners than in the centre of the input domain. This preference is a by-product of some procedures that achieve an

even spread of test cases by enforcing them far apart from each other. With such a preference, Chen *et al.* showed that ART could perform even worse than RT when the failure rate is high. Their explanation to such phenomenon is summarized as below. On one hand, when the failure rate is high, the higher the dimension is, the more likely an input in the centre is failure-causing (A detailed justification can be found in [12]). On the other hand, the increase of corners and edges is exponential to the increase of dimensions. Briefly speaking, when the failure rate is high, ART may not be able to select test cases around the centre until most of the corners and edges are filled, and hence ART may take more test cases to detect the first failure than RT.

In this paper, we work on one particular ART method, *Fixed-Sized-Candidate-Set ART* (FSCS-ART) [13] which has a preference of generating test cases in edges and corners, and investigate how FSCS-ART can be enhanced in the high dimensional case. The paper 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 why the performance of FSCS-ART deteriorates with dimensions of the input domain. Section 4 details our approach to enhancing FSCS-ART, and reports our experiment and findings. Section 5 concludes the paper.

2. BACKGROUND

FSCS-ART [13] 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 distance to its nearest neighbour in E is chosen as the next test case. The algorithm of FSCS-ART is given in Figure 1.

1. Set $n = 0$ and $E = \{ \}$.
2. Randomly select a test case, t , from the input domain (according to uniform distribution).
3. Increment n by 1.
4. If t reveals a failure, go to Step 9; otherwise, store t in E.
5. Randomly generate k inputs to construct C (according to uniform distribution).
6. For each $c_i \in C$, calculate the distance d_i between c_i and its nearest neighbour in E.
7. Find $c_b \in C$ such that its $d_b \geq d_i$ where $k \geq i \geq 1$.
8. Let $t = c_b$ and go to Step 3.
9. Return n and t , and EXIT.

Figure 1: FSCS-ART

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 the regions formed by the failure-causing inputs and the distribution of these regions within the input domain). Both attributes are fixed upon completion of the coding but unknown to testers before testing.

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 number of test cases required to reveal the first failure). Chen *et al.* [4] have conducted a study on the statistical properties of these metrics for ART. Among these three metrics, F-measure is not only the most commonly used one but also the most appropriate one for comparing the fault-detection effectiveness of ART and RT. In this paper, we will use F-

measure as the effectiveness metrics in order to facilitate the comparison with previous studies. Most of the experimental studies on ART were carried out through simulations. For each simulation study, failure rates and patterns are pre-defined, and the failure regions are randomly placed in the input domain. A failure is said to be found if a point inside the failure regions is picked by ART. ART is repeatedly applied to such a setting until the simulation has been run a sufficient number of times (S) to achieve a significantly reliable average F-measure (denoted as F_{ART}) with an accuracy range of $\pm 5\%$ and a confidence level of 95%. The detailed method of obtaining S can be found in [5]. Like all the previous studies, we assume that all inputs have an equal chance of being selected with replacement, and hence, the expected F-measure of RT (denoted as F_{RT}) is theoretically known as $1/\theta$ where θ denotes the failure rate. In this paper, F_{ART} refers to the F-measure of FSCS-ART, unless otherwise specified, and the *ART F-ratio* (that is, F_{ART}/F_{RT})[6] is used as a measurement of the improvement of ART over RT.

3. HIGH DIMENSION PROBLEM OF FSCS ART

In this paper, we aim to investigate the high dimension problem of FSCS-ART. We have identified two causes that affect the performance of FSCS-ART. One is related to the preference of test case selection. The other is about the metric (Euclidean distance) used for evenly spreading test cases.

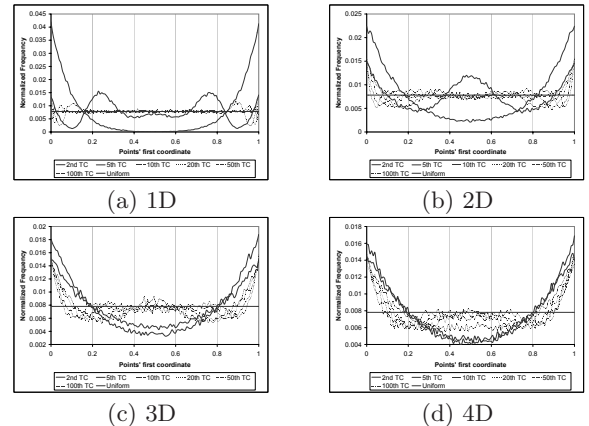


Figure 2: Frequency distribution of the first coordinate of points generated by FSCS-ART

With respect to the first cause, Chen *et al.* [6] have pointed out that FSCS-ART prefers to select test cases from the edges and corners instead of the centre. We conducted a simulation to verify their hypothesis. In this simulation, the input domain was square and the value range of each dimension was (0, 1). Since points will be relatively close to the edge or corner if any of its coordinates is relatively close to 0 or 1, we decided to check only one coordinate of points in this simulation, say the 1st coordinate, to judge their location. Sequences of test cases were generated by FSCS-ART in one-, two-, three- and four-dimensional space. The position of the n th points with respect to the 1st coordinate was recorded, where $n = \{2, 5, 10, 20, 50, 100\}$. We repeated the simulation 100,000 times and then got six groups of data. Figure 2 shows the frequency distribution the 1st coordinate of points generated by FSCS-ART. Points

generated randomly according to uniform distribution were also plotted for comparison. Clearly, all sets of data related to FSCS-ART in Figure 2 show that there is a higher normalised frequency at two ends of x-axis than at the centre. This implies that points generated by FSCS-ART are more likely to appear in the edges or corners with respect to the first coordinate, and without loss of generality, with respect to any coordinate. With the increase of the dimensionality, this phenomenon becomes increasingly obvious. On the other hand, the randomly generated points appear everywhere in the input domain with almost equal frequency.

The second cause is explained as follows. When the input domain is 1 dimensional, no matter where points (inputs) are located, they will all appear on one line as such, so 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, the spatial distribution of points is more complicated. If FSCS-ART only aims at keeping test cases apart, it cannot fully assure an even spread of test cases all over the input domain. An example is illustrated in Figure 3. Obviously, the test cases in Figure 3(a) are farther apart from one another than those in Figure 3(b). However, it may be controversial to argue that the former is more evenly spread than

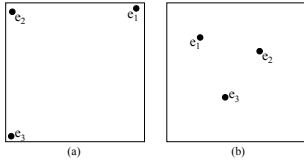


Figure 3: Different locations of test cases

Consider Figure 4 where $C = \{c_1, c_2\}$ (so $k = 2$) and $N = 2$. Assume that c_1 and c_2 have identical distance from their nearest neighbour in E . In other words, they both are entitled to be the next test case according to the existing selection criterion used in FSCS-ART. Further assume there exists an element e_i of E such that e_i and c_1 have the same value on y -axis, while no such a relationship exists between c_2 and any element in E . We argue that c_2 should be preferable to c_1 as the next test case. Two reasons are given as below.

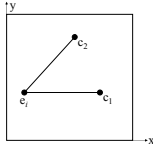


Figure 4: Different candidates with identical distance to e_i

- Besides keeping test cases apart, intuitively speaking, getting test cases dissimilar in all dimensions should *cover* larger parts of the input space than allowing test cases to be similar in some dimensions. From a *testing coverage* point of view, c_2 should be preferable to c_1 .
- Consider the case where the program failures depend on only part of program input parameters (like Program 1). Since failures are only sensitive to Y in Program 1, if we have failed to detect a failure by a test case t , we know that it is better for the follow up test cases different from t with respect to Y . Since it is normally unknown in advance which input parameter (dimension) program failures depend on, to effectively detect failures in all kinds of programs, it is better for

the next test case to be different from E (its elements are the inputs unable to reveal failures) as much as possible, not just with respect to the “Euclidean distance” but also with respect to the “value in each input parameter”. From this perspective, c_2 should be preferable to c_1 .

In summary, it has been shown that when $N > 1$, using only “distance” as the selection criterion may generate test cases which are neither evenly spread nor effective in detecting failures. In next section, we will present our solutions to address this problem.

Program 1 A faulty program whose failures depend on only one parameter.

```

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

```

4. ENHANCEMENT OF FSCS-ART

For ease of discussion, we introduce the following notations and concepts. In N dimensional input domains (I_i denotes each of its dimension), the co-ordinates of two points A and B are denoted as $(a_1, a_2 \dots a_N)$ and $(b_1, b_2 \dots b_N)$, respectively. $\text{dist}(A, B)$ denotes the Euclidean distance between point A and point B , and $\text{dist}_i(A, B)$ denotes $|a_i - b_i|$ with respect to I_i . Among all $\text{dist}_i(A, B)$, the shortest and the longest distance are denoted as $\text{minDist}(A, B)$ and $\text{maxDist}(A, B)$, respectively. Figure 5 illustrates the above concepts for a 2 dimensional space, with $\text{minDist}(A, B)$ being $\text{dist}_2(A, B)$ and $\text{maxDist}(A, B)$ being $\text{dist}_1(A, B)$. Finally, we define $\text{DistRatio}(A, B)$ as the ratio of $\text{minDist}(A, B)$ to $\text{maxDist}(A, B)$. Obviously, the range value of $\text{DistRatio}(A, B)$ is $[0, 1]$. Intuitively speaking, a larger DistRatio is preferable if other criteria, such as distance, are the same. As an example of illustration, refer to Figure 4 where c_2 is more preferable than c_1 . Note that $\text{dist}(e_i, c_1) = \text{dist}(e_i, c_2)$, but $0 = \text{DistRatio}(e_i, c_1) < \text{DistRatio}(e_i, c_2) \leq 1$.

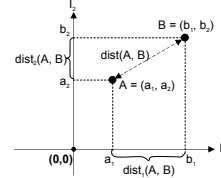


Figure 5: Notations

4.1 Our method

Our enhanced FSCS-ART is basically the same as the original FSCS-ART, but with one additional feature, that is, an eligibility filtering process which is 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 , $\text{DistRatio}(c, e_i)$ 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 called *eligibility criterion*. In the following paragraphs, we will elaborate our method (namely, *FSCS-ART with filtering by eligibility*) in details. Without loss of generality, we will illustrate this method using a 2 dimensional space.

As an example of illustrating the eligible inputs, consider Figure 6 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 $|E| = 1$ (Figure 7). There are 0, 20 and 36 elements out of 49 elements, which are eligible when $v = \tan(45^\circ)$, $\tan(30^\circ)$ and $\tan(15^\circ)$, 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 Figure 8 where v remains unchanged but the number of elements in E is different ($|E| = 1$ or 2 in Figure 8(a) or 8(b), respectively). As shown, the number of eligible inputs will decrease with the increase of $|E|$ if v keeps unchanged.

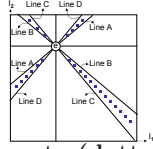


Figure 6: Eligible inputs (dotted regions), v and e (an element of E)

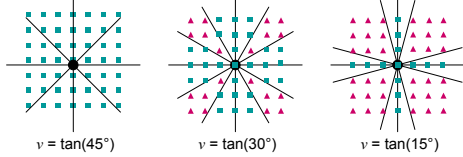


Figure 7: The relationship between v and the number of eligible inputs (represented by triangles)

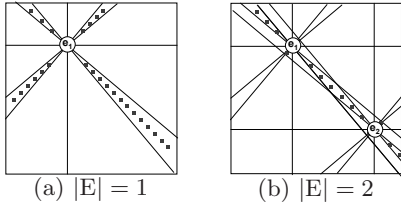


Figure 8: The relationship between $|E|$ and the number of eligible inputs (black spots)

The detailed algorithm of FSCS-ART with filtering by eligibility is given in Figure 9 where Steps 6-14 are introduced to replace Step 5 of Figure 1 (algorithm of FSCS-ART). Basically, we need to construct a candidate set C such that all its elements are eligible. In C , the farthest element away from E is chosen as the next test case.

To use FSCS-ART with filtering by eligibility, 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 construct C . To resolve this problem, we propose to dynamically relax the eligibility criterion during the testing process when 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.

Clearly, the larger the value g is, the longer time it takes to decide whether adjusting v is required or not. In this study, g was arbitrarily set to 4. Since the effect of the filtering disappears when $v = 0$, v is adjusted slowly and only if necessary. Also, we have arbitrarily set that if fewer than 70% candidates are eligible, the current eligibility criterion

is regarded to be too strict and hence there is a need to reduce v .

1. Input v and r , where $0 < r < 1$ and $0 \leq v \leq 1$
2. Set $n = 0$, $E = \{ \}$, $C = \{ \}$.
3. Randomly select a test case, t , from the input domain (according to uniform distribution).
4. Increment n by 1.
5. If t reveals a failure, go to Step 18; otherwise, store t in E .
6. Randomly generate k inputs to construct C (according to uniform distribution).
7. For each $c_i \in C$, examine the eligibility of c_i and mark c_i ‘eligible’ or ‘ineligible’ accordingly.
8. If all elements of C are eligible, go to Step 15.
9. Set $n\text{Trial} = 0$.
10. Do Steps 11-14 until all c_i of C are eligible
11. Replace each ineligible c_i by another random input.
12. Examine the eligibility of all replacements, and mark them ‘eligible’ or ‘ineligible’ according to the updated value of v .
13. Increment $n\text{Trial}$ by 1.
14. After 4 attempts (when $n\text{Trial} = 4$), if fewer than 70% of candidates are eligible, then set $n\text{Trial} = 0$ and $v = v \cdot r$.
15. For each $c_i \in C$, calculate the distance d_i between c_i and its nearest neighbour in E .
16. Find $c_b \in C$ such that its $d \geq d_i$ where $k \geq i \geq 1$.
17. Let $t = c_b$ and go to Step 4.
18. Return n and t . EXIT.

Figure 9: FSCS-ART with filtering by eligibility

4.2 Experiment 1

It is interesting to know how the effectiveness of FSCS-ART with filtering by eligibility is affected by v and r . We investigate the impact by conducting some simulations. In these simulations, θ was varied from 1 to 0.00005, and the failure region was predefined as a single square which is randomly placed within the input domain.

The results for the setting $v = 0.5$ ($\approx \tan(26.57^\circ)$) and $r = 0.5$ for $N = 2, 3$ and 4 are reported in Figure 10. Under this setting, the experimental data show that the process of filtering does make our method more effective than the original FSCS-ART.

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

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

There are 9 different scenarios in total. We group the results into Figure 11. Based on these data, we have the following observations:

- the higher r is, the smaller the ART F-ratio is
- when r is 0.5 or 0.1, v seems to have little or no impact on the F-measure of FSCS-ART with filtering by eligibility, but when $r = 0.9$, a small v like 0.1 will not deliver an effective FSCS-ART with filtering by eligibility.
- For a given r , any value of v higher than 0.5 will yield similar F-measures.

The first observation is intuitively expected, because a higher r means that v will be decreased more slowly, and hence a more strict eligibility criterion has been imposed. As a consequence, test cases will be enforced more evenly spread over the input domain, and hence the ART F-ratio becomes smaller. The second observation is also understandable because when r is small, no matter what v is used, v will become small very soon (as v will be adjusted by r in Step 14

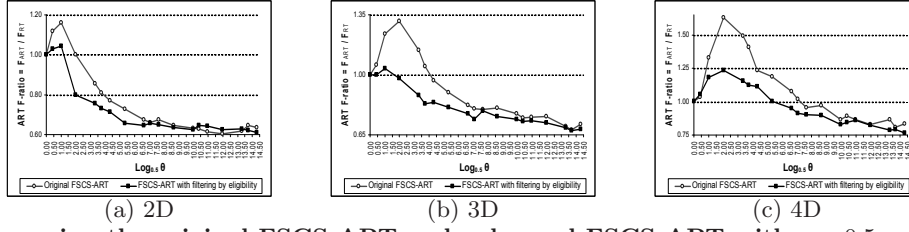


Figure 10: Comparing the original FSCS-ART and enhanced FSCS-ART with $v = 0.5$, $r = 0.5$ and $k = 10$

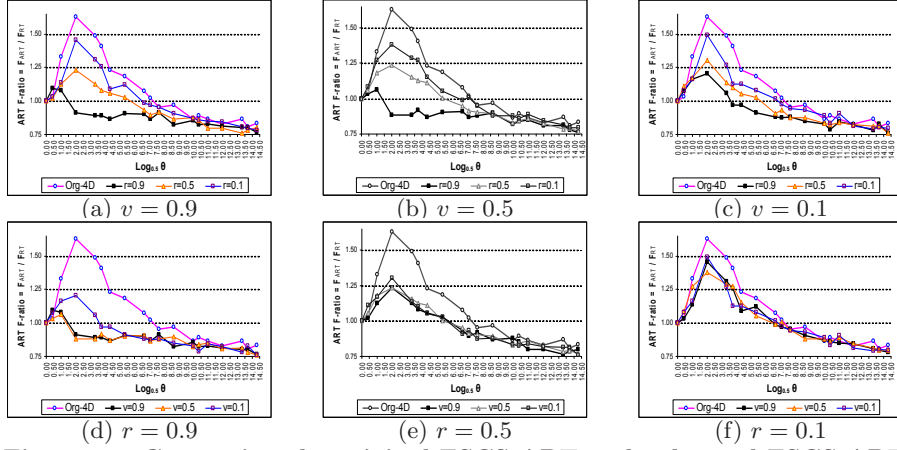


Figure 11: Comparing the original FSCS-ART and enhanced FSCS-ART

of the algorithm given in Figure 9). For the last observation, a higher initial value of v yields a small number of eligible inputs. Therefore, v will be reduced quickly until enough eligible inputs are available, so higher values of v will have similar performance. However, it should be noted that generally speaking, a higher v will have smaller F-measure than a lower v . In summary, the above observations imply that effective FSCS-ART with filtering by eligibility needs a high initial value of v (say 0.9) and a large r (say 0.9) in order to keep v large and its approaching to 0 at a slow rate during its application.

4.3 Experiment 2

We conducted another experiment to investigate the F-measure of FSCS-ART with filtering by eligibility for the situation where failures depend on some but not all input parameters (this is the situation motivating the development of FSCS-ART with filtering by eligibility). One single rectangular failure region is assumed to reside in a square N dimensional input domain whose edge length is L . θ is either 0.01, 0.005, 0.001 or 0.0005, and N is either 2, 3 or 4. For a given N dimensional input domain, the number m of *failure dependent input parameters* (input parameters which cause program failures) is set to be smaller than N and the failure region is assumed to span across $(N - m)$ axes. As an example of illustration, consider a 3 dimensional input domain and let l_i denote the edge length of the failure region in dimension i (or simply say, l_i is the value range of the failure-causing inputs with respect to the i^{th} parameter). Suppose θ is 0.01. If failures are caused by the 3rd parameter (so $m = 1$), then we have $l_1 : l_2 : l_3 = L : L : 0.01L$. If failures are caused by the 2nd and 3rd parameters (so $m = 2$), then we have $l_1 : l_2 : l_3 = L : 0.1L : 0.1L$.

The simulation results are reported in Table 1, which compared FSCS-ART with filtering by eligibility and FSCS-

ART with θ being 0.01, 0.005, 0.001 and 0.0005, respectively. In each table, the performances are compared in terms of different combinations of N and m . It is clearly shown that when $m = 1$, no matter how many dimensions the input domain has, the new method has a significant improvement over the original method. When $m > 1$, the new method only marginally outperforms the original method. In summary, our new method also has a better fault-detection effectiveness than the original FSCS-ART when the program failures only depend on part of the input parameters.

5. DISCUSSION AND CONCLUSION

ART was originally proposed to improve the fault detection effectiveness of RT, especially when the failure-causing input are clustered together. Recently, it was observed that the performance of ART is not so good in higher-dimension situation as that in lower-dimension situation. In this paper, we analysed the high dimension problem of ART, and proposed a new algorithm, FSCS-ART with filtering by eligibility. The results of simulation show that our method improves FSCS-ART not only in high dimensional cases, but also when program failures depend on part of the input parameters. It should be noted that both cases are quite common to occur in real life programs. Therefore, we suggest that FSCS-ART with filtering by eligibility should be used instead of the original FSCS-ART.

In addition, it should be noted that the causes that affect the performance of FSCS-ART may be also applicable to some other ART methods. *Restricted Random Testing* [3] (RRT), for example, also has the preference of selecting test cases from the boundary part of the input domain, as shown in [12]. And RRT also uses Euclidean distance as the metric of selecting the next test case. Therefore, our solutions to the high dimension problem of FSCS-ART may be applica-

Table 1: Results for Experiment 2

N	m	Method	Average F-measure			
			$\theta=0.01$	$\theta=0.005$	$\theta=0.001$	$\theta=0.0005$
2	1	FSCS-ART with filtering by eligibility	69.76	144.00	726.93	1460.77
		FSCS	96.08	189.48	996.84	1975.79
3	1	FSCS-ART with filtering by eligibility	77.41	154.63	793.34	1605.00
		FSCS	103.12	197.64	1020.31	2009.87
3	2	FSCS-ART with filtering by eligibility	91.59	175.50	954.21	1966.32
		FSCS	93.21	192.79	973.18	1987.53
4	1	FSCS-ART with filtering by eligibility	82.31	160.94	827.06	1661.84
		FSCS	98.37	197.98	970.15	2022.04
4	2	FSCS-ART with filtering by eligibility	97.11	196.27	1020.17	2096.84
		FSCS	108.97	214.50	1044.55	2126.33
4	3	FSCS-ART with filtering by eligibility	98.49	195.59	970.50	1841.60
		FSCS	104.81	208.93	983.61	1927.09

ble for improving other ART methods.

Acknowledgment

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

6. 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] 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.
- [3] K. P. Chan, T. Y. Chen, and D. Towey. Restricted random testing: Adaptive random testing by exclusion. *Accepted to appear in International Journal of Software Engineering and Knowledge Engineering*, 2006.
- [4] T. Y. Chen, F.-C. Kuo, and R. Merkel. On the statistical properties of testing effectiveness measures. *The Journal of Systems and Software*, 79(5):591–601, 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 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.
- [7] R. Cobb and H. D. Mills. Engineering software under statistical quality control. *IEEE Software*, 7(6):45–54, 1990.
- [8] T. Dabóczy, I. Kollár, G. Simon, and T. Megyeri. Automatic testing of graphical user interfaces. In *Proceedings of the 20th IEEE Instrumentation and Measurement Technology Conference 2003 (IMTC '03)*, pages 441–445, Vail, CO, USA, 2003.
- [9] G. B. Finelli. Nasa software failure characterization experiments. *Reliability Engineering and System Safety*, 32(1–2):155–169, 1991.
- [10] J. E. Forrester and B. P. Miller. An empirical study of the robustness of Windows NT applications using random testing. In *Proceedings of the 4th USENIX Windows Systems Symposium*, pages 59–68, Seattle, 2000.
- [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 Communication Technologies, Swinburne University of Technology, 2006.
- [13] I. K. Mak. On the effectiveness of random testing. Master’s thesis, Department of Computer Science, University of Melbourne, 1997.
- [14] 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.
- [15] 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.
- [16] B. P. Miller, D. Koski, C. P. Lee, V. Maganty, R. Murthy, A. Natarajan, and J. Steidl. Fuzz revisited: A re-examination of the reliability of UNIX utilities and services. Technical Report CS-TR-1995-1268, University of Wisconsin, 1995.
- [17] E. Miller. Website testing. http://www.soft.com/eValid/Technology/WhitePapers/website_testing.html, Software Research, Inc., 2005.
- [18] G. J. Myers. *The Art of Software Testing*. Wiley, New York, second edition, 1979.
- [19] K. Sen, D. Marinov, and G. Agha. Cute: a concolic unit testing engine for C. In *ESEC/FSE-13: Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 263–272, New York, NY, USA, 2005. ACM Press.
- [20] D. Slutz. Massive stochastic testing of SQL. In *Proceedings of the 24th International Conference on Very Large Databases (VLDB 98)*, pages 618–622, 1998.
- [21] T. Yoshikawa, K. Shimura, and T. Ozawa. Random program generator for Java JIT compiler test system. In *Proceedings of the 3rd International Conference on Quality Software (QSIC 2003)*, pages 20–24. IEEE Computer Society Press, 2003.