IET Software

# A Relational Memory Selection: An implementation of adaptive random sequence to enhance the effectiveness and efficiency of Adaptive random testing

SEN-2018-5385.R1 | Research Article

Submitted on: 19-09-2019

Submitted by: MICHAEL OMARI, Jinfu Chen, Patrick Kudjo, Hilary Ackah-Arthur

Keywords: SOFTWARE TESTING, SOFTWARE QUALITY, SOFTWARE ENGINEERING

School of Computer Science and Telecommunication Engineering,
Jiangsu University, Zhenjiang 212013,
Jiangsu, People's Republic of China.
15th September, 2019.

Dear editor,

We are happy to inform you of the resubmission of our previously reviewed manuscript. The idea that was contained in the earlier submission was inadequate to address all the issues raised by the reviewers. As a result, we have totally modified the paper by introducing a new concept which implements the previous idea as a supplementary concept. This has compelled us to also modify the title of the paper accordingly. Thus, the previous manuscript entitled “A speculative and a proactive approach to test case selection: A cost effective adaptive random testing” is now being submitted as “A Relational Memory Selection: An implementation of adaptive random sequence to enhance the effectiveness and efficiency of Adaptive random testing”. In this new method, we adopted AR sequence to improve the efficiency of ART. We have observed that most recent ART overhead reduction strategies have made progress using a subset of unsuccessful test cases to guide the selection of test cases. We recognized the straightforward nature of partitioning based methods and identified MART as a classic example. Thus, we introduced a slightly modified version of MART (entitled RMS for short) which does not rely on mapping functions but exploit larger number of partitions to reduce overhead cost of ART. To handle this large numbers of test cases whiles simultaneously enforcing even-distribution, we incorporate a two-level implementation of AR-sequence which enforces both local and global diversity content of the test case generation within the input domain. Then to further reduce the computational tasks, we implemented the previous method DAC-FSCS as an overhead reduction strategy of RMS. The method maintains the good failure detection ability in high dimension like its predecessors (partitioning based methods) whiles significantly improving upon the efficiency of FSCS and MART when compared. We hope the new updates and makeover will address all the outstanding concerns with the earlier submission. Thank you.

In the submission of the manuscript attached to this letter, I Jinfu Chen acting as the corresponding author wish to state on behalf of the authors that all the materials are original (excluding Reviews), and no part has been submitted for publication elsewhere, besides there is no conflict of interest and all authors have agreed to the editorial policies and have recommended onward submission.

Yours sincerely,

**Jinfu Chen.**

**(Corresponding author)**

Tel.: +86-13615286115
E-mail: *jinfuchen@ujs.edu.cn*

**Original Manuscript ID:** SEN2018-5385

**Original Article Title:** "A speculative and a proactive approach to test case selection: A cost effective adaptive random testing"

**Modified Title:** "A Relational Memory Selection: An implementation of adaptive random sequence to enhance the effectiveness and efficiency of adaptive random testing"

**To:** IET Editor

**Re:** Response to reviewers

Dear Editor,

Thank you for allowing a resubmission of our manuscript, with an opportunity to address the reviewers' comments.

We have provided responses to the comments (below) (response to reviewers) and the updated manuscript. Thank you.

Yours sincerely,

**Jinfu Chen.**

**(Corresponding author)**

Tel.: +86-13615286115

E-mail: *jinfuchen@ujs.edu.cn*

## Responses to Reviewers

### Reviewer #1

**Concern #1** Insufficiency and lack of depth in the proposed method

**Response**: Thank you for your suggestions and comments which has encouraged us to improve upon the concepts, content and general presentation of this manuscript. Though this has taken us a long period to respond, it was worth every bit of the time. To answer all the questions within the framework of the previous manuscript was virtually out of reach. Therefore a new method was necessary which we have accordingly pursued and presented in this new proposal.

**Action**: We have presented a complete makeover of the original paper which has addressed all the concerns raised. In fact, the old concept has been presented as a supplementary idea to the content of this presentation.

The previous concept was proposed to reduce the overhead cost of FSCS which we called DAC-FSCS is now being used as a cost-saving approach to the newly proposed method; a Relational Memory Selection (RMS). The DAC-FSCS appears in this paper as FSCS with overlapping or FSCS-O. These we have done to enable us to handle all the concerns you raised earlier which DAC-FSCS was insufficient to address. This method harnesses a number of concepts from previous works on ART to develop a more improved overhead reduction strategy for ART.

**Concern #2** Unsuitable choice of baseline comparison

**Response**: Our method was proposed as an alternative standalone approach using "overlapping" as opposed to "best candidate" criterion of FSCS for test case selection. We carefully selected the forgetting approaches due to the nature of discarding (forgetting) computations to previously executed test cases. With the newly proposed method, we have selected a more popular and frequently cited overhead reduction strategy for ART as the baseline method of comparison.

**Action:** We have selected one of the most often cited ARTs state of the art methods MART (as you suggested) to assess its general contribution to ART research efforts. The newly proposed method compares favorably with MART with improved failure detection effectiveness.

**Concern #3** Presentation issues

**Response**: The paper was presented in a rush to meet some pressing deadlines. We acknowledge the validity of the presentation issues you have raised and duly apologize for the difficulties you encountered in your review process.

**Action**: We have taken our time to improve on the grammatical expressions and presentation of the previous edition.

Thank you.

## Reviewer #2

**Concern # 1 and 2** Validity and possible confusing figures

**Response**: We do notice some of the seemingly confusing figures and acknowledge it as possible misstatements (commission and omissions) of the results.

**Action**: We have modified the original manuscript which touches on all the points which you have raised. The paper presents a newly proposed method which incorporates the previous idea as an auxiliary concept. The old concept was proposed to reduce the overhead cost of FSCS which we called DAC-FSCS is now being used as a cost-saving approach to the newly proposed method A Relational Memory Selection (RMS). The DAC-FSCS appears in this paper as FSCS with overlapping or FSCS-O. These we have done to enable us to handle all the concerns you raised earlier which DAC-FSCS was insufficient to address. New experiments have been designed with thoroughly verified and validated results.

The quadratic complexity has also been improved to compare favorably with the state of the art methods like MART.

**Concern #3:** Poor presentation:

**Response**: We acknowledge the many grammatical errors and deeply regret any possible headache you may have experienced during your review process. We do recognize the value of a good presentation to IET and its readership.

**Action**: We have improved the constructions, grammar, and formatting of the current manuscript to enhance the quality of the presentation.

Thank you.

# A Relational Memory Selection: An implementation of adaptive random sequence to enhance the effectiveness and efficiency of Adaptive random testing

Michael Omari, Jinfu Chen*, Patrick Kweku Kudjo, Hilary Ackah-Arthur
(School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, 212013, China)
*Corresponding author's e-mail: jinfuchen@ujs.edu.cn*

*Abstract*—**Adaptive random testing (ART) methods of different classifications have been published in the literature, each of which seeks to demonstrate in different ways that, even spreading test cases can significantly improve the effectiveness of random testing (RT). Most recent works on ART attempt to reduce the overhead cost associated with the even spreading objective by tackling the root cause; the continual bloating numbers of unsuccessful test cases. Creating a subset from these test cases runs through almost all the approaches and holds a good promise to current research objectives. To this end, partitioning seems to be the most logical solution, but how can this be achieved effectively and efficiently? We present a *relational memory selection* (RMS); a strategy for implementing adaptive random sequence (ARS). RMS incorporates a two-level implementation of ARS in a nested form. Test cases from all partitions are synchronized and selected in a manner that, its execution order evenly distribute and mimics test case distribution pattern of ART. ART computational overhead is restricted to only a subset of the unsuccessful test cases. Experiments and simulations data shows that RMS is effective and more efficient compared to other ART methods like FSCS and MART.**

Keywords— **Adaptive Random Testing, Adaptive Random Sequence, software testing, mirror adaptive random testing**

## I. INTRODUCTION

Software testing has been accepted as one of the most vital components of software engineering and remains one of the most widely practiced and studied approaches for assessing and improving software quality [1]. With painstaking efforts, through intricate abstractions, software engineers have integrated testing into every level of the software development value-chain. Regardless of all the efforts, "bugs" inadvertently "creep" into software leading to their ultimate failures. Numerous testing techniques have therefore been developed and used to help developers increase their confidence that software has certain qualities that meet the aspiration of various stakeholders. Most of these techniques focus on strategies for selecting test cases[2]. A testing strategy is a mechanism designed for selecting test input (test cases) from the input domain for execution by the software under test. A test case that reveals a failure is referred to as a failure-causing input [3]. In general, failure-causing inputs determine two basic features of a faulty program. One feature is the failure rate (denoted by θ), which refers to the ratio between the number of failure-causing inputs, and the number of all possible inputs. The second feature is the failure pattern which describes the failure region(s) together with their geometric shape [4]. The works of [5-7] have established that failure causing inputs of many programs tends to cluster within their input domains. There are two common approaches to generating test cases for testing software; black-box and white-box. In white-box testing, the structure of the software is used as a basis for selecting test cases. Black-box, on the other hand utilizes no information concerning the structure of the software under test(SUT) [8]. Random testing (RT) is a very popular black box testing technique which is preferred for its simplicity and intuitive appeal. RT selects test cases based on a uniform distribution or according to the operational profile[9]. With random testing, the probability of selecting a failure causing input as test a case solely depends on the size of failure rates [10]. This is true for all program patterns but for block and strip patterns, Chen et al [10] proposed a slightly modified version of RT called Adaptive Random Testing (ART) which improves the chances of selecting a test case from within the failure causing input region. ART combines randomness of test case generation with even distribution within the input domain. The location of failure causing input is unknown before the testing process begins. However, the fact that they crystalize gives rise to the intuition that if test inputs are randomly selected and evenly distributed, it increases the probability of finding the location of these inputs than ordinary random search. It has been observed in experiments based on actual programs that, ART is able to outperform RT by about 25% for programs with a strip failure pattern [10] and in some cases [11] 80% in terms of the number of test cases used to detect the first program programs' failure (F-measure).

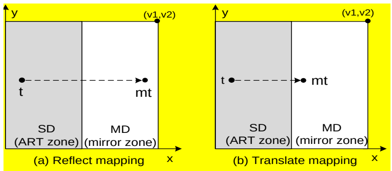
Almost all ART algorithms consist of two independent processes. The first process referred to as candidate generation process involves random generation of program inputs as test case candidates or briefly candidates. The second process (known as test case identification process) applies some criteria to identify test cases among these candidates to ensure an even spread of test cases across the input domain[4]. This can pose a serious computational problem for test case generation. Practicality, therefore, remains a major concern about the application of ART. This concern has been expressed by Acuri [12] and has prompted several research works [13-17] into how best to achieve the even distribution objective with a lower overhead cost. Consequently, most research focus on ART has shifted to bridging the inefficiency gap between ART and RT.

## II. BACKGROUND

The problem of selecting the most cost-effective test suite has been a major research subject in the software testing area since the 1970s [18]. Although cost-effectiveness of ART has been the major focus for most researchers since its early inception, recently there have been renewed concerns that ARTs excessive test case generation time could limit its use and application in testing tools [20]. Nonetheless, most ART methods [10] [9, 19] still remains $O(n^2)$ in time complexity.
Intuitively speaking, the more information about the testing process that can be utilized, the more effective the testing process can be. In other words improving upon effectiveness inadvertently leads to inefficiency. The use of previously executed test cases which has failed to reveal fault (unsuccessful test cases) to randomly filter the input domain to favor the most sparsely populated regions is an indispensable trait in all the propositions of ART. But, using executed test cases to guide test case selection process can be very expensive computationally and accounts for the quadratic time of ART methods. Besides, there is no bound on the number of executed test cases so long as testing condition(s) has not been met unless of course it (test suit size) is part of the conditions. There are several ART methods which exploit partitioning (both dynamic and static) as a basis for fractionalization (or creating a subset of unsuccessful test cases) to meet this objective. The dynamic approaches [13, 15, 16] adjust the partitioning incrementally (iteratively) to prevent the number of executed test cases per partition from exceeding a predefined threshold. Divide and conquer[13] is a classic example. Dynamic mirror adaptive random testing (DMART) [16] also employs a similar approach but additionally, it incorporates mirroring within a section (exactly half) of the input domain to increase efficiency. Filtering has also been used to accomplish the same purpose. A distance-aware forgetting through the exploitation of the spatial information of the already generated test cases also employs grid partitioning dynamically to lesson computational cost of ART [15]. MART [20] adopts a divide and conquer method and heuristic approach but in a rather static manner. Elimination by linear association (EMART) [17] has also been proposed to enhance MART which also combines static partitioning with filtering based on forgetting strategy [15]. Given the same number of already executed test cases, larger numbers of partitions are desirable for lessening computational tasks especially as the size of unsuccessful test cases grows. For example, it has been established by Chen et al [20] that theoretically, the computational order of distance-based ART approach FSCS is $O(n^2)$ but when MART is used in combination of FSCS, the computations reduces to $O(n^2/m^2)$; m being the number of mirrors partitions. It is worthy of note that in each of the methods, the effectiveness of ART was significantly improved especially in high dimensions. The reason behind this is attributable to the border selection preferences of most ART methods (especially distance-based approaches) which is by itself inextricably linked to the difficulty in generating truly random values in high dimensions [19]. Hence, partitioning helps in minimizing the border selection problems of ART and is equally good for reducing the cost of generating test cases. Our proposed method uses static partitioning as a means to reduce the overhead cost of ART. MART has been selected as a baseline method for comparison because it shares some basic concepts in common with our proposed method. In the next subsection, we introduce MART and later, show how it relates to our proposed strategy.

### A. *MIRROR ADAPTIVE RANDOM TESTING (MART)*

In MART, one partition is designated as ART test case generation zone (SD) whiles the rest of the partitions are assigned as mirror zones. Test cases in the mirror zones or mirror domains (MD) are calculated using mapping functions (translate and reflect). Before testing, the tester must select a partitioning scheme. MART scheme is defined as MART $\varepsilon_1 \times \varepsilon_2 \times \varepsilon_3 \ldots \times \varepsilon_d$, where $\varepsilon_i$ is the number of partitions on dimension $i$. An example is given in fig.1 using a 2x1 scheme with the input domain D defined on $[0,v_1]$ and $[0,v_2]$ respectively. Chen et al [20] recommended that $\varepsilon_i$ should be as small as possible in a range $1 \leq \varepsilon_i \leq 2$ in order to maintain effectiveness. Given a $d$-dimensional input domain, there are a total of $\gamma = 2^d - 1$ number of possible schemes.



(a) Reflect mapping (b) Translate mapping

**Fig. 1** Two simple mirror mapping functions

### B. *Adaptive random sequence (ARS)*

Given an input domain (D) partitioned into n subdomains $SD = (D_1, D_2, D_3, \ldots D_n)$, of equal sizes, select test cases $t_1, t_2, t_3, \ldots t_n$ from $D_1, D_2, D_3, \ldots D_n$ respectively i.e. for all $D_i, i = 1,2,3, \ldots, n$ there's only one test case in each subdomain such that the order of selection maximizes the chances of failure detection. Given two sets of strategies S and S', S is said to be more effective if it has a higher probability of detecting program failure with fewer test cases than S'. A selection sequence, therefore, seeks to find the ideal ordering of test cases for testing, so that the tester obtains maximum benefit, even if the testing is prematurely halted at some arbitrary point [21]. ART can be used not only to generate its own sequence of test cases but also to order a given test suite to improve its chance of detecting failure detection [22]; a re-ordered sequence is called an adaptive random sequence (ARS). ARS has been applied to regression testing [22] to improve failure detection effectiveness and can equally be useful for partition testing.

## III. A RELATIONAL MEMORY SELECTION

The simplest and most logical way to reduce the burden of over bloating unsuccessful test cases is to partition the input domain by either static or dynamic means, assigning these test cases to their respective partitions and focusing on one partition at a time. In the case of static, the number of partitions remains the same throughout testing. The dynamic approaches start out with a small number of partitions and iteratively adjust (increase) the partitions as the number of executed test cases grow. While its necessity may depend on the size of partitions employed in the case of static, there will at some point in the testing process (when using dynamic partitioning) where the order of selection becomes very crucial for effective testing. Partitioning testing (with proportional sampling) to some extent, share similar intuition with ART[23] but given a large number of partitions, ARS is necessary to enhance its failure detection abilities. It is at the backdrop of this that we propose a new approach that seeks to leverage on both approaches to enhance effectiveness and efficiency of ART. Our proposed method; a relational memory selection (RMS) and MART shares two basic characteristics in common; a partitioning scheme selection and an active test case generation zone (source domain/partition). However, unlike MART in which test cases are generated before they are ordered, RMS first selects the partitions before the test cases are generated. But that is not all; MART relies on mapping functions to boost the efficiency of generating test cases. RMS, on the other hand, relies on creating a relatively larger number of partitions to reduce the subset of executed test cases used for test case generation. By this, we try to avoid the use of mapping functions due to their problematic nature as has been reported in [16, 17] whiles making up for the extra computations by increasing the number of partitions.

First, the input domain is partitioned using a scheme defined at two levels. The first level called mirror frames (or frames for short) identified as $\omega$ subdivides the input domain into major subdomains similar to MART schemes. The second level called mirror partitions (identified by $\varphi$ defines another partitioning which is restricted to the individual subdomains (or frames). One frame is designated as the source frame (SF) in which test case generation zones within all the mirror frames are identified. RMS then defines a two-level AR sequences for selecting the partitions for generating test cases. In the example demonstrated in fig.2, the number of frames is four (4) whiles the number of partitions in each frame (mirror partitions) is sixteen (16).When a test case is selected in the source frame (SF), its partition must be identified, then all related partitions in each frame are selected for test case generation. This creates a large pool of test cases that must be executed whiles simultaneously enforcing even distribution. The first sequence called level-one (L1) sequence is used as a global selection sequence or global diversification whiles the second (L2) enforces local diversity content of partitions within the major subdomains. Unlike L1 sequence, the L2 generates itself as test cases in the source partition are generated. This is used to enforce randomness and reduce the overhead burden of generating the L2-sequence.

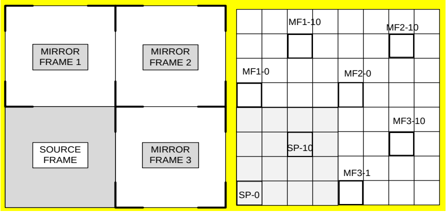
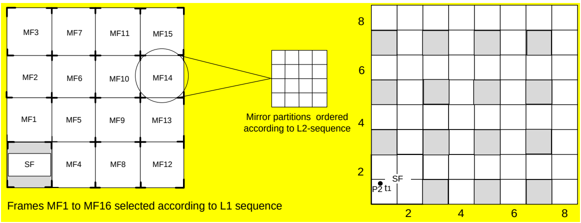


**Fig.2** Related mirror partitions in a 2x2 RMS scheme


MF3
MF7
MF11
MF15
MF2
MF6
MF10
MF14
MF1
MF5
MF9
MF13
SF
MF4
MF8
MF12
Mirror partitions ordered according to L2-sequence
Frames MF1 to MF16 selected according to L1 sequence
8
6
4
2
SF
P2
t1
2
4
6
8

(a) (b)

**Fig. 3** ARS selection framework for test case diversification and execution in RMS

Fig.3 gives an illustration of related partitions within the global (L1-sequence) and local selection (L2-sequence) framework for RMS. Here, a single test case has been located in the first compartment of the source frame, cascading a selection of related partitions within all the major subdomains. The selection follows a round robbing order where the global sequence is iterated for every local sequence.

Suppose that partitions in subdomain D1 (or source frame) are ordered according to AR sequence, then if those of subdomain D2 has a one to one relation with D1, then even though test case distribution in D1 is different from D2, their sequence of selection makes D2 possess similar quality of even distribution as D1. There could be multiple ways to establish a one to one association among these partitions. The simplest and most straightforward approach is to maintain the order of partitions based on the sequence of generation within the source and mirror frames partitions. We provide the details of the test case generation process in the next section.

Given the same number of partitions, all things being equal, MART is expected to be more efficient than RMS since test cases are regenerated and independent of those within the source domain. But beyond a certain number of partitions, the computational gap can be significantly reduced to surpass MART's efficiency. The reason is that, since both RMS and MART employs static partitioning schemes, the number of executed test cases in the source domain of MART will grow at a faster rate than those within individual frames of RMS.

Besides its efficiency appeal, the proposed strategy has some technical merits with regards to generating effective test cases. The arguments supporting this claim are twofold; firstly, generating test cases based on targeted partitions distribution will directly reduce the boarder selection preference of FSCS as was the case of [24]. Secondly, discrepancy measure which determines whether different regions within the input space have the same point densities is one of the key metric [25] for measuring even spread of test cases. Therefore a sufficiently large number of partitions with an equal number of test cases per this metric is desirable and enhances the even spread objective of ARTs. But we quickly want to point out that excessive number of partitions can adversely affect randomness and to some extent diversity. As the number of partitions increases it may lead to poorly diversified test cases based on the observations of Chen et al [19]. For example, in fig.3 (b) even though sixteen partitions have been selected for test case generation, none of the even partitions on either dimension have been selected.

### A. Test case generation using RMS

In the test case generation process, FSCS has been selected because it seems to be the most suitable option for the proposed method. Additionally, for the purpose of comparison we also observe that historically [16, 17, 20] FSCS has been jointly applied with MART to assess its effectiveness. Before generating test cases, the L1 sequences must be generated. Two approaches can be used to do this; using ART to generate at least one test case in each partition and determining the sequence in which the partitions were selected. Alternatively, the midpoint of each partition (representing the center of gravity) can be used as representing a point in the partition. FSCS can then be applied to order these points to obtain an AR sequence for the partitions. We follow the former to generate the L2-sequence whiles the latter is used for generating the L1 sequence. In generating L2, two sets of candidates are generated; $k_1$ number of partitions and $k_2$ number of localized candidate test cases within the selected partitions. Thus, the total candidate used in generating test case from the source frame is $k_1 \times k_2$. This is done to introduce randomness in the partitions selection process.

For convenience and ease of presentation, we demonstrate test case generation process of RMS in a one dimensional input domain. In fig 5, the input space is divided into two (2) frames i.e. $\omega = 2$ and the number of mirror partitions (MP) within each frame is set as 6 i.e. $\varphi = 6$ given by $P = \{P_1, P_2, P_3, P_4, P_5, P_6\}$. For the purposes of our illustration, we use 3 partition candidates ($k_1 = 3$) and 3 candidates within each of $k_1$ partitions (i.e. $k_2 = 3$ ).

The first test case is selected within the source frame from a random partition ($P_2$). The second test case is generated from its corresponding mirror partition MP1-2 with a random value in the ranges of MP2. The probability of selecting $P_2$ is 0, thus $P_2$ is eliminated from the available test case generation zone whiles the chances of selecting any of the remaining partitions increases. In the generation of the next test case, three (3) candidates partitions ($P_1$, $P_4$ and $P_5$) are randomly selected. Three (3) candidates are generated within each of the partitions using a uniform distribution yielding a total of nine (9) candidate test cases. Using FSCS with the subset of executed test cases (E) within SP and candidates selected within $P_1$ ($c_1, c_2, c_3$), $P_4$ ($c_1, c_2, c_3$), and $P_5(c_1, c_2, c_3)$, the candidate with the maxmin distance ($P_5 c_2$) is selected for execution. Its related mirror partition is selected within the MP (i.e. MP1-5) 3 candidates are locally generated and used for distance computation with a subset of E within the mirror frame. After $c_3$ of MP5 is selected for execution, the process is repeated until all empty partitions within the SF have been selected. The partitions selection history is kept and used as the L2 sequence. Supposed each subdomain SD is partitioned into $\varepsilon = \{\varphi_1, \varphi_2, \varphi_3, \ldots \varphi_d\}$ where $\varphi_i$ represents the number of partitions on dimension $i$ of mirror frames, then $\lambda = \prod_{i=1}^{n} \varphi_i$ represents the threshold test cases before the level _2 sequence is triggered. Therefore higher values of $\varphi$ places a greater cost burden on generating the L2-sequence which requires the use of a larger number of candidates.

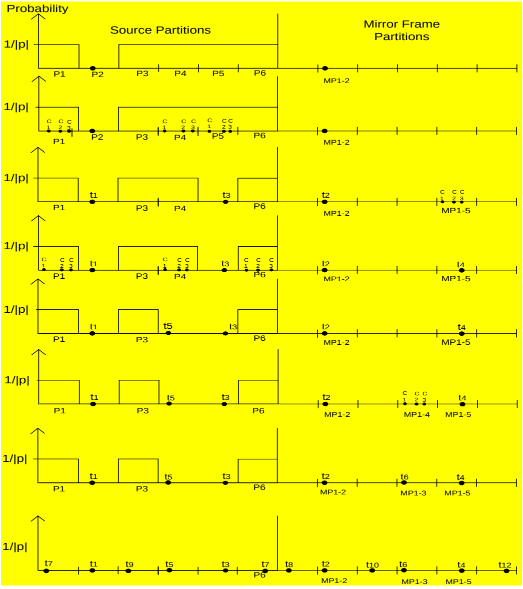


**Fig. 4** Test case generation process for RMS ($\omega_i = 2$, $\varphi_i = 6$, after |P|=0, L2_sequence= (P2, P5, P4, P1, P3, P6).

## IV. FSCS-O: AN APPROACH TO MINIMIZE THE COST OF TEST CASE GENERATION IN RMS

Compared to MART, a larger number of candidates ($k_1 \times k_2$) may be required for generating test cases within the source frame using distance-based algorithm FSCS. To make up for this shortfall, we introduce a new form of distance-based approach to FSCS called FSCS with overlapping (FSCS-O) whose computational task is far less expensive compared to the original approach but maintains the same failure detection effectiveness.

### A. FSCS test case distribution

Test case selection by FSCS involves random selection (candidates) as well as even distribution by selecting the farthest candidates from already executed test cases. The selection criterion does not always generate test cases with consistently declining distances (maxmins) between executed test cases due to the random nature of selecting candidates.

In order to get a good picture of the behavior of the maxmin values for selected test cases, we analyze a plot of a mined data (of test suit). We assumed a two dimension input domain with the range of [0, 1] on each dimension. Fig 5 provides a snapshot of the first 200 test cases and their distribution. The vertical axis represents the distances validated for selecting test cases. The points on the graph represent differences between consecutive maximum-minimum distances for the nth test case generation. It can be noticed that there are almost equal instances where $MaxMin_n > MaxMin_{n-1}$ (n=the nth test case generation). We observed a continual overlapping of the consecutive maxmin distances indicated by the undulating nature of the points around the mean value 0.

### B. Predicting the maxmin values and minimizing wasted computational efforts.

We introduce a new form of forgetting strategy which we have termed FSCS with overlapping (FSCS-O) to reduce the number of distance computations of FSCS that is based on the behavior of the maxmin distances. In FSCS, distance computation must be effected between each candidate in C and all executed test cases in E irrespective of whether that distance was determined ‘earlier’ or not. This we consider as a logical trap because it requires the evaluation of all candidates before the valid (best) candidate is selected. In the overlapping rule, we discard (forget) further distance computations between candidates set and executed whenever an overlapping situation occurs.

As an illustration, let’s consider a one dimensional input domain. Supposed the maxmin for the 6th test case generated was 8, and already executed test case set is E={11,81,55,29,70,85} then to generate the 7th test case, 10 candidates C={16,4,20,7,50,23,6,18,12,33} were randomly selected. The maxmin for the currently selected candidates is 9 given by candidate 3 (i.e. $c_3$= 20). FSCS must go through computing the minimum distance between each of the ten candidates before finally settling on candidate 3. We propose that after the third candidate was evaluated giving a value of $9 > 8$ (the maxmin for the 6th test case), the rest of the computations for the 4th to the 10th candidate should be discarded. Suppose for the purpose of argument, we assume the candidates selected was {16,4, 20,7,50,23,6,18,45,33}, it can be observed that candidate 9 (value of 45) has the highest minimum distance of 10, however we argue that since 9 is greater than the previous maxmin value (7), it suffices to settle on the 3rd candidate because it is “far enough. The results from the mined data placed the number of overlapping instances at 50.02% ≈50% of total computed maxmin. Fig.6 presents the plot of the number of candidates selected for every test case

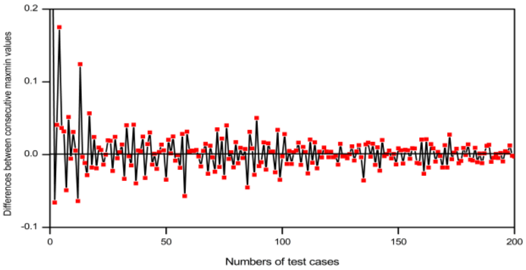


**Fig. 5** Differences between consecutive maxmin distances

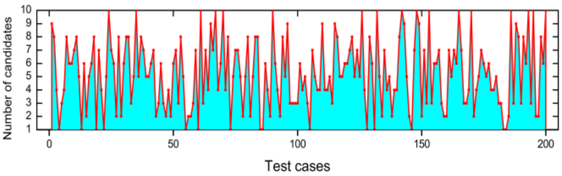


**Fig. 6** Savings in computations using FSCS-O

$(1 < n \leq 200)$ generation in a 2D simulation with range [0, 1] on each dimension. The number of candidate selection by FSCS is always 10 (i.e. k=10) whiles the number of candidate selection using overlapping condition ranges from 1 to 10. The area above the graph represents the savings in the number of computations.

We conducted a simulation study to compare the use of overlapping criterion (FSCS-O) and ordinary FSCS in terms of effectiveness and efficiency. The results showed that statistically, FSCS-O and FSCS are not different in terms of effectiveness. On the other hand, we observed a significant reduction of about 25% in the fm-time values. This shows that using the overlapping condition does not lead to deterioration in the level of effectiveness of FSCS.

### C. Determination of efficiency ratio

Supposed in the determination of a suitable candidate for the nth test case, an overlapping condition occurs on candidate $C_i$, the number of discarded computations for the nth test case generated is $n \times (k - C_i)$ where k is the number of candidates. Supposed again that after the test suit size reaches γ the number of overlapping conditions is $\alpha$ , then the total number of discarded computations is given by;

$$N = \sum_{i=1}^{\gamma} \{(k - C_i).|E|_j\} \quad . . . \quad (2)$$

Where $|E|_j$ represents the size of E when $\alpha = i$ .

In comparison with FSCS, the number of computations with overlapping rule when test suit reaches γ is given by:

$$N_{\boldsymbol{FSCS-O}} = k \sum_{i=1}^{\gamma} i \bigg|_{\boldsymbol{FSCS}} - \sum_{i=1}^{\alpha} \{(k - C_i).|E|_j\} \bigg|_{\boldsymbol{overlap-rule}}$$

$C_i$ is the candidates upon which the overlapping condition occurred.

The ratio of the number of discarded computations to the number of test computations by FSCS is; $\sigma = \frac{N}{k \sum_{i=1}^{\gamma} i|_{FSCS}}$ .

### RMS-Algorithm 1

```
1: Use FSCS to generate the first two test cases t_1 and t_2
2: Set E= {t_1, t_2} and Set Previous_Farthest=dist (t_1, t_2)
   // maxmin distance computed in step 1
3: Set maxmin =0, min=0;
4: for  candidate 1 to k //k=number of candidates set
5:     Randomly generate test case from input domain
        (according to uniform distribution)
6.      Add test case to C;
7: end for
8: for each Candidate c in C
9:          Compute Euclidean distance between c and its
            nearest neighbor in E and assign to min
10:      if min>maxmin
11:              maxmin=min;
12:              Farthest_Candidate=c
13:              if maxmin >Previous_Farthest
14:                    Previous_Farthest=maxmin
15:                    goto step 20
16:              end if
17:         end if
18: end for
19: Assign maxmin to Previous_Farthest
20: execute Farthest_candidate
21: if failure detected goto step 24
22: Add Farthest_candidate to E
23: goto step 3
24: return |E| and exit
```

The number of computations of FSCS is k $\sum_{i=1}^{n} i = \frac{kn(n+1)}{2}$. After the introduction of the overlapping-rule, the computations reduce to;

$$\approx k \times (1 - \sigma) \sum_{i=1}^{n} i = \frac{kn(n+1)}{2} \times (1 - \sigma) = 0(n^2).$$

In order to determine the value of σ we performed multiple simulations using various dimensions (the number of dimensions influences the values of N) and generating 10,000 test cases in each simulation. By taking the averages across the various dimensions, the value of σ is estimated to be approximately 0.2464301 which is approximately 25%.

### *D. Time complexity analysis*

First, the input domain must be partitioned according to a selected scheme $\varepsilon = RMM - \tau$ where $\tau$ is the number of partitions on each dimension. This has a complexity of O $(\tau^d)$; where $d$ = number of dimensions; a constant. Since distance computations are restricted to individual mirror frames, the number of test cases for distance computation is $|E|/\omega^d$. Since $\omega$ is constant whiles |E| grows, its complexity is thus; $O(n^2 /\omega^d)$. That of MART is estimated to be $O(n^2/m^2)$. It is not hard to see that when $\omega^d > m^2$ overtime RMS can be more efficient than MART. The efficiency ratio σ of FSCS-O is approximately ¼th of the original FSCS, therefore combined with RMS; the complexity is expected to be O$(3n^2/4\ \omega^d)$ which is still quadratic.

### RMS Algorithm 2

```
1.  Select a scheme $\varepsilon = (\omega, \varphi)$
2.  Partition input domain $D$ according to $\varepsilon$ // $D =$
    $(F_1, F_2, F_3, \ldots, F_\eta)$, $F_j = P_{j1}, P_{j2}, P_{j3}, \ldots, P_{\varphi^d})// \eta = \omega^d$
3.  Select source frame $SF$ from $D$
4.  Generate L1_$ARS$ for $D$ // using centers of gravity
5.  Initialize containers Executed $E' = \{E_1, E_2, E_3, \ldots, E_\eta\}$ for
    $F_j$ ,$E_j = \{\ \}$ $j = 1,2,3, \ldots, \eta$; testBuffer={ } ,L2_$ARS$ $= \{\ \}$,
6.  Initialize pointers $tb\_ptr \leftarrow 0$, $L1\_ptr \leftarrow 0$ , $L2\_ptr \leftarrow 0$
7.  do
8.      Select a test case $t$ using FSCS-O (E, C) and add
        selected partition Index $I_p$ to L2_$ARS$ // C is $k_2$
        partitions from $SF$ and $k_1$ candidates in each partition
9.      Add test case to testBuffer
10.     for each $F$ in $D$ // all frames
            Construct C using $F$ ($I_p \rightarrow$ L1_$ARS$) and E using
             E' ($L1\_ptr \rightarrow$ L1_$ARS$)// subset for E'
            Generate test case $t$ = FSCS(C, E)
            Add $t$ to testBuffer
            Increment $L1\_ptr$
11.     end for
12.     do
            Execute $t = b\_ptr \rightarrow$ testBuffer
            Add $t$ to E according to E' ($L1\_ptr \rightarrow$ L1_$ARS$)
            If failure is found goto exit
            $bt\_ptr$:= $bt\_ptr$ +1// Increment buffer pointer
13.     while $tb\_ptr < |testBuffer|$
14.     clear testBuffer and reset $tb\_ptr \leftarrow 0$, $L1\_ptr \leftarrow 0$
15.     store $I_p \rightarrow SF$ in $L2\_ARS$
16.     Isolate $I_p \rightarrow SF$ from $SF$
17. while $|SF| > 0$
18. do
19.     for each $F$ in $D$ // all frames in D
            Select partition p from F according to:
            $D$($L1\_ptr \rightarrow$ L1_$ARS$, $L2\_ptr \rightarrow$ L2_$ARS$ )
            // L1 and L2 being used as global and local
            Select E from E' according to: // diversity resp.
            E' ($L1\_ptr \rightarrow$ L1_$ARS$)
            Generate test case $t$ = FSCS-O(C, E, p)
            Add $t$ to testBuffer
20.     end for
21.     do
            Execute test case $t = tb\_ptr \rightarrow$ testBuffer
            Add $t$ to E according to E' ($L1\_ptr \rightarrow$ L1_$ARS$)
            If failure is found goto exit
             $tb\_ptr := tb\_ptr + 1$
22.     while $tb\_ptr < |testBuffer|$
23.     $tb\_ptr \leftarrow 0$ //Initialize buffer pointer
24.     $L2\_ptr$ : = $L2\_ptr$ +1//point to next p of $F$ in sequence
25.     if $L2\_ptr$ =| L2_$ARS$ |
                $L2\_ptr \leftarrow 0$ // reset local diversity
26.     end if
27. until failure is found
28. exit
```

## V. SIMULATIONS AND EXPERIMENT STUDIES

### A. *Research questions*

RMS is proposed as an alternative to MART which was by itself proposed to enhance the efficiency of ART. Our proposed method is examined from three broad perspectives.

**RQ1**: How does the number of frames and partitions impact on the effectiveness of RMS?

**RQ2**: How effective is RMS in relation to other ART methods?

**RQ3**: How efficient is RMS in relation to other ART methods?

### B. *Evaluation metrics and Data collection method*

F-measure has been historically used as the effectiveness metric for the comparison. For RT, whenever the failure rate (θ) is known, the f-measure can be estimated as 1/θ. However, a theoretical evaluation of ART is known to be extremely difficult (and non-existent at the moment) as the effectiveness of ART depends on many factors [26]. In evaluating our proposed method, F-measure obtained after every experiment is recorded and repeated for a sufficiently large number (N) of times until the mean value obtained is statistically reliable as has been the practice. The details for estimating N with a 95% confidence level and an error margin of 5% can be obtained from Chen et al [10]. The term F-ratio has also been of particular interest as it measures ART effectiveness improvement upon RT. i.e. F-ratio=$F_{ART}/F_{RT.}$ We use these metrics to evaluate the proposed method due to its wide application to ARTs.

### C. *Parameter setting*

The parameters $k_1 = 8,\ k_2 = 3$ .The number of candidates after L2_sequence is triggered $k$ =10. Also in the study, the generation of L1_AR-sequence is reinitiated in every simulated program until the number of mean f-measure samples has been attained.

### D. *System environment Specifications*

The simulation and experiments were carried out on a system with the following characteristics; Processor: Del Inspiron 3847, Intel® Core™ i3-4170 CPU @3.7GHz (4CPUs) ~3.7GHz, Memory: 8GB, OS: Microsoft windows10 Pro 64-bit, Development environment: Dev C++ 4.9 -std= c++ 11.

## VI. RESULTS OF SIMULATION AND EXPERIMENT

### A. *Simulation studies*

Simulations were carried out by defining generic programs using basic parameters defined for a real-world program. The size of the failure causing inputs is generated based on the failure rate and randomly inserted within the input domain.

**Answer to RQ1:** The effect of the number of frames and partitions on the effectiveness of RMS.

To answer this question, we selected a 2-dimensional input domain with a varied number of frames and partitions. Using a square failure pattern and a failure rate θ=0.001 we analyze the behavior of the F-measure values as well as the standard deviations which is a good measure of stability in terms of effectiveness. The results have been presented in Table 1 and 2. It can be observed that increasing the number of frames and partitions increases the F-ratio values at marginal rates. Also, the standard deviation in the effectiveness was higher as the number of frames and partitions becomes larger but the values are highly sensitive to the number of mirror partitions than the number of frames.

Based on the previous discussions and the results of the simulations, we provide two guidelines for selecting frames and partitions.

1) One dimensional input domain doesn't suffer from the curse of dimensionality in relation to test case distribution. Furthermore, there is no pattern formation in RMS even with an increasing number of mirror partitions. Therefore it is pointless to maintain a small number of schemes when higher scheme can enhance efficiency.

2) In the definition of a scheme for programs with more than one dimension, $\varphi$ should be as small as possible whiles $\omega$ should be cautiously selected to balance efficiency with effectiveness.

Accordingly, in all simulations and experiment in two dimensions and above, $\varphi$ is fixed at 2 whiles we examine the value of $\omega$ to determine the effect on effectiveness/efficiency. Three (3) schemes i.e. $\omega = 3,\ 4\ \ and\ 5$ have been selected for evaluation to help answer the research question. In all one dimensional programs, the value of $\omega$ is arbitrary set to 50. We represent RMS scheme as $\varepsilon = RMM - \omega$. The definition is thus strictly based on the number of frames since the mirror partitions are held constant.

**Answer to RQ2:** Comparison of effectiveness

We carried out simulations using different dimensions (d=1, 3, 5, and 7) of input domain and a block failure pattern with failure rate (θ) of 0.0050 and 0.001. We selected three samples of MART schemes to represent MART except for 1-dimension. The explanation is that given a hyper cubic failure region, MART scheme $\varepsilon^i = (\varepsilon_1, \varepsilon_2, \varepsilon_3, \dots, \varepsilon_d)$ where $1 \leq \varepsilon_i \leq 2$ and $i = 1,2,3, \dots d$ is equivalent to $\varepsilon^j = (\varepsilon_{11}$, $\varepsilon_{12}$, $\varepsilon_{13}, \dots, \varepsilon_{1d})$ where $\varepsilon^j$ is a permutation of $(\varepsilon_1, \varepsilon_2, \varepsilon_3, \dots, \varepsilon_d)$. For example in a 3D input domain, MART scheme 2x2x1, and 2x1x2 are equivalent. Hence, even though in a three (3) dimensional input space, the number of

**Table 1** F-ratios for a 2-D input domain with varying number of frames and partitions (θ=0.001).

| $\varphi^d$ | $\omega^d$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 4 | 9 | 16 | 25 | 36 | 49 | 64 |
| 4 | 0.639527 | 0.649276 | 0.645636 | 0.633152 | 0.64774 | 0.63288 | 0.6504 |
| 9 | 0.637784 | 0.644267 | 0.631883 | 0.624085 | 0.62986 | 0.63481 | 0.62120 |
| 16 | 0.645238 | 0.647581 | 0.649071 | 0.657366 | 0.64146 | 0.64534 | 0.64084 |
| 25 | 0.639632 | 0.647688 | 0.638299 | 0.654991 | 0.65152 | 0.63662 | 0.65956 |
| 36 | 0.647982 | 0.666408 | 0.649329 | 0.656514 | 0.65582 | 0.66053 | 0.65247 |
| 49 | 0.653074 | 0.674672 | 0.676174 | 0.679686 | 0.66096 | 0.67136 | 0.67952 |
| 64 | 0.653981 | 0.659410 | 0.668783 | 0.660536 | 0.66916 | 0.67911 | 0.68593 |

**Table 2** Influence of the number of frames and partitions on the effectiveness stability (standard deviations)

| $\varphi^d$ | $\omega^d$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 4 | 9 | 16 | 25 | 36 | 49 | 64 |
| 4 | 53.54328 | 38.0371 | 49.38803 | 48.97923 | 45.88266 | 49.23499 | 44.24029 |
| 9 | 56.87612 | 56.62838 | 65.77889 | 60.21103 | 60.13433 | 65.64307 | 80.13628 |
| 16 | 64.7077 | 64.98679 | 66.2432 | 60.81001 | 74.93126 | 75.26428 | 87.70915 |
| 25 | 74.84343 | 100.8608 | 96.93917 | 99.07338 | 92.93427 | 104.0159 | 116.2216 |
| 36 | 80.04502 | 77.77716 | 80.18221 | 79.771 | 90.75096 | 85.05056 | 113.7504 |
| 49 | 100.1647 | 102.3775 | 119.2658 | 107.7305 | 105.6704 | 103.0889 | 106.9738 |
| 64 | 106.5546 | 86.68856 | 120.9186 | 114.1945 | 129.574 | 137.7364 | 134.615 |

**Table 3** F-ratios comparison of RMS, MART and FSCS under different failure rates using simulated programs

| d | θ | FSCS | MART | RMS | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Scheme1 | Scheme2 | Scheme3 | RMS-3 | RMS-4 | RMS-5 |
| | | | $2^2$x1 | 2x$1^2$ | $2^3$ | | | |
| 3 | 0.0010 | 0.7561 | 0.9757 | 0.9441 | 0.9566 | 0.81451 | 0.8702 | 0.8716 |
| | 0.0050 | 0.7980 | 0.8171 | 0.8334 | 0.81364 | 0.94232 | 0.9121 | 0.9324 |
| | | | $1^3$x$2^2$ | $1^2$x$2^3$ | $2^5$ | | | |
| 5 | 0.0010 | 1.0931 | 1.0972 | 1.1164 | 1.1109 | 0.983490 | 0.8902 | 1.034 |
| | 0.0050 | 1.2317 | 1.1188 | 1.2065 | 1.0914 | 1.00345 | 1.0352 | 1.123 |
| | | | $1^2$x$2^5$ | $1^2$x$2^5$ | $2^7$ | | | |
| 7 | 0.0010 | 1.6152 | 2.1017 | 1.7108 | 1.3017 | 0.93234 | 1.2326 | 1.3462 |
| | 0.0050 | 1.8827 | 1.5607 | 1.0930 | 0.9724 | 1.2131 | 1.3730 | 1.3259 |

combinations is seven (7) i.e. $2^3$-1, the selected schemes adequately represent MART. Tables 3 provide a summary of the results. In lower dimensions and higher failure rates, FSCS and MART show improvement over RMS in terms of failure finding effectiveness. In high dimensions, however, RMS has better failure detection in each of the schemes for both failure rates under study. RMS's effectiveness also deteriorated slightly as we move to higher schemes. Effectiveness of MART schemes also differ slightly from each other and appears to improve as the number of bisected dimensions increases.

**Answer to RQ3**: Comparison of test case generation cost

To help address research question three, we examined the test case generation cost of each of the selected methods and compares it with RMS. The result of our simulations is provided in Table 4. MART schemes which yield a similar number of mirror partitions have the same computational overhead. Therefore we selected only one scheme to represent all those schemes. For instance, in a 2-dimension input domain, MART has three (3) different schemes (1x2, 2x1 and 2x2) but because 2x1 and 1x2 have the same number (i.e. 1) of mirror domains, only one (1x2) is selected to represent both schemes. As can be observed from table 4, within RMS variants, higher schemes are more efficient than lower schemes. Compared to FSCS, RMS uses a lesser time to generate any size of test suite irrespective of the scheme selected. In relation to MART, some of the schemes are almost similar in terms of efficiency. It is important to note that so long as the number of test cases in the source domain of MART increases at a faster rate than those within the frames of RMS (i.e. when $\omega^d > m^2$), whenever MART starts out to be more efficient, overtime (where generating $m$ number of test cases in RMS becomes less costly than generating a single test case within the source domain of MART), the computational times will converge and eventually diverge in favor of RMS. For example, in the 4D we notice that MART-$2^4$ is slightly better than RMS-5 at the initial stages (Fig.7) but at a certain threshold (N=15000), the

**Table 4** Comparison of execution time for RMS with MART and FSCS

| **Method** | \|D\| | $m^2/\omega^d$ | Number of test cases/Execution time in ms | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2D | | N=1000 | N=2000 | N=5000 | N=10000 | N=12500 | N=15000 | N=17500 | N=20000 |
| FSCS | | | 1.023 | 6.107 | 75.398 | 552.204 | 1061.802 | 1818.096 | 2865.247 | 4250.77 |
| MART-1X2 | | 1 | 0.531 | 3.048 | 37.944 | 281.335 | 540.953 | 925.138 | 1460.75 | 2172.504 |
| MART-2X2 | | 16 | 0.172 | 0.796 | 9.492 | 67.495 | 128.878 | 219.408 | 344.946 | 511.438 |
| RMS-3 | | 9 | 0.317 | 1.599 | 17.632 | 124.119 | 235.767 | 400.324 | 627.965 | 929.01 |
| RMS-4 | | 16 | 0.202 | 0.968 | 10.233 | 69.897 | 132.608 | 224.685 | 351.744 | 519.928 |
| RMS-5 | | 25 | 0.156 | 0.687 | 7.1 | 47.795 | 89.818 | 151.456 | 236.14 | 347.964 |
| | 3D | | | | | | | | | |
| FSCS | | | 1.477 | 8.66 | 109.687 | 813.332 | 1564.665 | 2675.703 | 4217.398 | 6259.981 |
| MART-$1^2$X2 | | 1 | 0.578 | 3.319 | 41.357 | 305.729 | 588.103 | 1005.819 | 1585.922 | 2355.274 |
| MART-1X$2^2$ | | 9 | 0.188 | 1.121 | 14.307 | 90.225 | 166.998 | 281.101 | 438.723 | 647.476 |
| MART-$2^3$ | | 49 | 0.079 | 0.329 | **3.361** | 23.143 | 43.806 | **74.39** | 116.713 | 172.295 |
| RMS-3 | | 27 | 0.1539 | 0.6867 | 6.9228 | 45.8892 | 86.1093 | 144.9459 | 225.8514 | 332.379 |
| RMS-4 | | 64 | 0.108 | 0.4455 | 3.9897 | 24.2523 | 44.2512 | **73.3581** | 113.7978 | 167.1084 |
| RMS-5 | | 125 | 0.1116 | 0.4077 | **3.0204** | 17.1459 | 30.3183 | 48.7926 | 73.4616 | 105.0912 |
| | 4D | | | | | | | | | |
| FSCS | | | 3.1675 | 18.6087 | 235.420 | 1746.07 | 3357.033 | 5735.700 | 9032.595 | 13399.546 |
| MART-$1^3$ x2 | | 1 | 0.735 | 4.425 | 56.728 | 431.055 | 825.558 | 1404.214 | 2203.22 | 3259.764 |
| MART-$2^2$X$1^2$ | | 16 | 0.217 | 1.201 | 14.65 | 106.391 | 205.166 | 353.683 | 559.995 | 832.919 |
| MART-$2^3$X$1^2$ | | 64 | 0.063 | 0.345 | 4.002 | **30.157** | 59.044 | 102.213 | 161.436 | 239.837 |
| MART-$2^4$ | | 256 | 0.016 | 0.094 | 1.203 | 8.501 | 15.814 | **26.455** | 40.861 | 59.439 |
| RMS-3 | | 81 | 0.1404 | 0.5247 | 4.4847 | **26.7768** | 48.4236 | 79.2873 | 120.8619 | 174.9258 |
| RMS-4 | | 256 | 0.0981 | 0.3744 | 2.7774 | 14.2911 | 24.1344 | 37.3959 | 54.7992 | 76.9122 |
| RMS-5 | | 625 | 0.117 | 0.4365 | 2.5497 | 10.6263 | 17.3025 | **25.9245** | 36.7695 | 50.0796 |

generation cost of MART outstrips those within the individual mirror frames of RMS causing the trend to reverse. At the height of this, RMS is more efficient when the test suit size reaches 2x$10^4$. We have highlighted the tipping points of some of these instances in table 5.

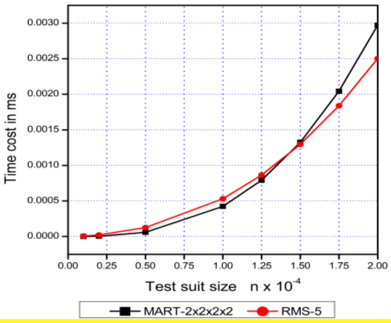


**Fig. 7** RMS outstrips MART in terms of efficiency in 4D

### *B. Experimental studies*

Having completed the simulation studies, we also performed further assessment of our method by experimentation using real world programs. For a contextual assessment and appraisal of our proposed method, we have adopted twelve (12) programs which have been used extensively [9-11, 13, 15, 16, 20, 27, 28] in the past to assess the effectiveness /efficiency of most ART algorithms. These programs were originally written in Fortran Pascal or C; designed for numerical computations and have been converted to C++ programs [14]. The programs have been widely published and can be accessed from ACM's collected algorithms and Numerical Recipe [18, 19]. In table 6 we provide some essential features about the programs including their names, failure rates, number of seeded faults, and the type of fault seeded. The seeded error types include; Arithmetic Operator Replacement (AOR), Relational Operator Replacement (ROR), Constant Replacement (CR) and Scalar variable Replacement (SVR).

In our experimental studies, we examined the effectiveness of our proposed method in two (2) separate ways;

(1) Compare the effectiveness and efficiency of RMS to MART and FSCS using homogeneous partitioning schemes.

(2) For programs whose dimensions differ in magnitudes we scale the number of partitions of RMS along each dimension according to their magnitudes and examine the impact on effectiveness and efficiency.

Apart from one dimension where MART defines only one scheme, different MART schemes have different level of effectiveness especially where program dimension is not related to one or more of the input parameters. To be impartial, we selected the best scheme (in terms of effectiveness) to represent MART in our analysis.

**Table 5** Experimental programs and their basic features

| Program | D | **Input Domain** | | **Seeded fault type** | | | | **Total** | **Failure rate** |
|---|---|---|---|---|---|---|---|---|---|
| | | FROM | TO | AOR | ROR | CR | SVR | | |
| Airy | 1 | -5000 | 5000 | | | | 1 | 1 | 0.000716 |
| Bessj | 2 | (-2.0,-1000) | (300.0,15000.0) | 2 | 1 | | 1 | 4 | 0.001298 |
| Bessj0 | 1 | -300000.0 | 300000.0 | 2 | 1 | 1 | 1 | 5 | 0.001373 |
| cel | 4 | (0.001,0.001,0.001,0.001) | (1.0,300,10000,1000) | 1 | 1 | | 1 | 3 | 0.000332 |
| el2 | 4 | (0.0,0.0,0.0,0.0) | (250,250,250,250) | 1 | 3 | 2 | 3 | 9 | 0.000332 |
| erfcc | 1 | (-30000.0) | (30000.0) | 1 | 1 | 1 | 1 | 4 | 0.000690 |
| gammq | 2 | (0.0, 0.0) | (1700.0, 40.0) | | 3 | | 1 | 4 | 0.000574 |
| golden | 3 | (−100.0,-100.0,100.0) | (60,60,60) | | 3 | 1 | 1 | 5 | 0.000830 |
| plgndr | 3 | (10.0,0.0, 0.0) | (500.0, 11.0, 1.0) | 1 | 2 | | 2 | 5 | 0.000368 |
| probks | 1 | (-50000.0) | (50000.0) | 1 | 1 | 1 | 1 | 4 | 0.000387 |
| sncndn | 2 | (-5000.0,-5000.0) | (5000.0,5000.0) | | | 4 | 1 | 5 | 0.001623 |
| tanh | 1 | (-500) | (500) | 1 | 1 | 1 | 1 | 4 | 0.001817 |

**Table 5** The most effective schemes representing MART

| 2D | scheme | 3D | scheme | 4D | scheme |
|---|---|---|---|---|---|
| bessj | [1,2] | golden | [1,1,2] | cel | [1,2,1,1] |
| gammq | [1,2] | plgndr | [2,1,1] | el2 | [2,1,1,1] |
| sncndn | [1,2] | | | | |

The most effective schemes of MART in programs with two or more dimensions based on our experiment programs have been presented in table 5.

### (1) Answer to RQ2: Effectiveness comparison

In the one-dimensional programs, RMS recorded a remarkable improvement over the effectiveness of FSCS in each of the five (5) programs (presented in fig 8). In terms of percentages, RMS outperformed FSCS by 44%, 43%, 35%, 37%, and 49% respectively in the programs airy, bessj0, erfcc, probks, and tanh. Also, there was a marginal rate of improvement over the effectiveness of MART in all the one dimension programs. In the 2-dimension programs and above, the effectiveness of RMS is generally similar to FSCS and the best case of MART with the exception of cel and bessj in which MART and FSCS had a slight edge. Also, in the cel program, there were some slight variations in the effectiveness in the schemes of RMS. That said, true to the objective of ART, our proposed method is effective than RT in all the programs under consideration irrespective of the scheme selected. Fig 9 gives a summary of the results.

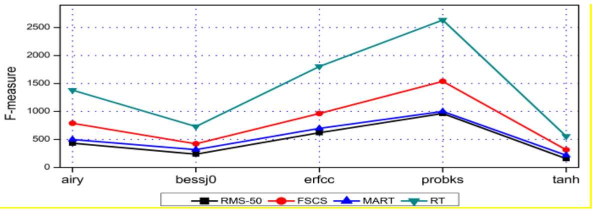


**Fig. 8** Comparison of average F-measures of RMS, FSCS and MART in one-dimension programs.

**Table 7** Proportional partitioning scheme for programs with different dimension magnitudes

| # | Program | RMS-3 | RMS-4 | RMS-5 |
|---|---|---|---|---|
| 1 | Gammq | [1,9] | [1,16] | [1,25] |
| 2 | Bessj | [9,1] | [16,1] | [25,1] |
| 3 | Plgndr | [9,3,1] | [16,4,1] | [25,5,1] |
| 4 | cel | [1,1,27,3] | [1,1,64,4] | [1,1,125,5] |

Four (4) out of the twelve programs including bessj, gammq, plgndr and cel had dimensions which varied significantly in magnitudes. In the experiment to evaluate proportionally partitioned schemes (where the numbers of partitions are scaled according to dimension magnitudes) we have presented the new partitioning scheme in Table 7.

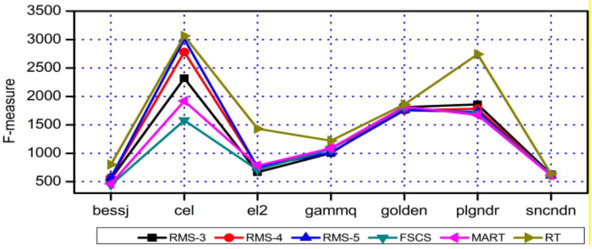


**Fig. 9** Comparison of average F-measures of RMS, FSCS MART (best case) and RT in programs with more than one dimension

The values were obtained by taking the ratios of the dimension magnitudes and accordingly awarding multiple values of $\omega$ (i.e. based on the specific RMS scheme). Fig 10 provides the results of that experiment. The effectiveness of plgndr and gammq remain unchanged whiles bessj and cel recorded an average of 30% and 48% improvement over the uniform partitioning schemes. Also, it was observed that unlike the previous experiment, the effectiveness of RMS is better than FSCS and MART in the cel program while the effectiveness gap was eliminated in bessj. The programs; plgndr and gammq had no significant variations.

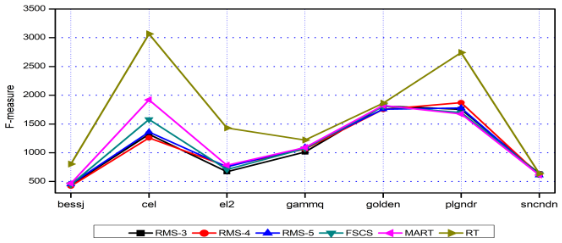


**Fig.10** Comparison of average F-measures of RMS (scaling), FSCS, MART (best case) and RT.

### (1) Efficiency comparison

In software testing practice, the testing process is terminated once a testing criterion or a stopping condition(s) has been met. One such criterion is when a failure is detected. At this point, if we are interested in comparing the relative efficiency of a method in meeting the testing goal, then per the setup of our experiment, we compare the time taken from the beginning of the testing process until the first program failure is found (also known as the fm-time). Since this is also variable for every experiment, its value is determined simultaneously as the f-measure is been computed. In an analysis of the experimental results, it turned out that, effectiveness to some extent influences efficiency. Clearly, comparing the execution times based on a given size of test suit may not be a practical way to assess efficiency. This aspect of our study takes both possibilities into account.

**Answer to RQ3:** The results of the efficiency of the one dimension programs presented in fig.11 correlates with the effectiveness results, RMS significantly outperformed FSCS and MART in terms of efficiency of the testing process for most of the selected programs. In two dimensional and above, programs, the results (fig. 12) indicate that, RMS has a superior overhead advantage over FSCS even with a smaller number of frames. It also recorded moderate improvements over most of the schemes of MART. A higher number of frames was consistent with the simulation results and reflected in lower overhead cost as compared to lower schemes. Also, for the programs whose dimensions were non-homogeneous, we observed that effectiveness translated into an improvement in the level of efficiency. There was a 48% and 30% improvement in the effectiveness of the cel and bessj programs respectively within the schemes of RMS which culminated in a 46% and 21% drop in fm-time respectively; which is very significant. That of plgndr and gammq recorded negligible differences in all the schemes. Also, for the programs whose dimensions were non-homogeneous, we observed that effectiveness translated into an improvement in the level of efficiency. The improvement in the effectiveness of the cel and bessj programs respectively within the schemes of RMS culminated in a 46% and 21% drop in fm-time respectively; which is very significant. That of plgndr and gammq recorded negligible differences in all the schemes. Fig. 11 summarizes this result.

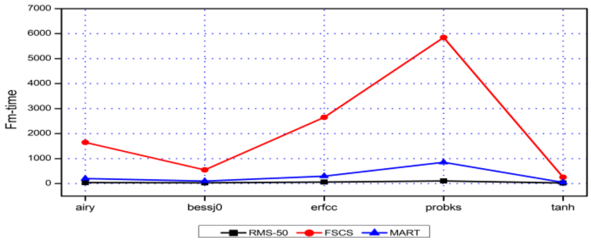


**Fig. 11** Time taken to reveal first fault comparison of RMS, FSCS and MART

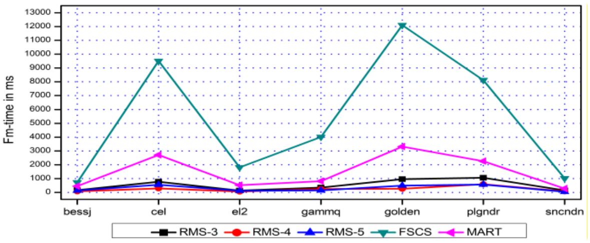


**Fig. 12** Time taken to reveal first fault comparison of RMS, FSCS and MART (best case) in 2-D programs and above

## VII. CONCLUSION

ART has gained notoriety for being unpleasantly inefficient in response to which proponents have intensified research over the years to provide a cure. We have observed that most recent ART overhead reduction strategies have made great progress in using a subset of unsuccessful test cases to guide the selection of successive test cases. These approaches include partitioning methods which use either static or dynamic (grid forms or otherwise) to create these subsets to aid cheaper test cases selection. We recognized the straightforward nature of these approaches and identified MART as a classic example. Unfortunately, there is a technical caveat which places a limitation on the extent to which it can be used to improve efficiency. Thus, we introduced a slightly modified version of MART entitled a relational memory selection (RMS) which does not rely on mapping functions but exploit larger number of partitions to reduce overhead cost of ART. To handle this large numbers of test cases whiles simultaneously enforcing even-distribution, we incorporate a two-level implementation of AR-sequence which incorporates both local and global diversity content of the test case generation within the input domain. Then to further reduce the computational tasks, we proposed a new version of FSCS which proactively select test cases compared to the passive approach of the original approach. RMS maintains the good failure detection ability in high dimension like its predecessors (partitioning based methods) whiles significantly improving upon the efficiency of FSCS and MART when compared. In addition, we verified through simulations that whenever the number of partitions for RMS exceeds $m^2$ (m being the number of mirror partitions in MART), RMS surpasses the test case generation efficiency of MART. The results obtained using the lowest possible schemes from the above is encouraging and provide confidence that RMS is a good overhead reduction strategy for ART and a good alternative to MART.

## ACKNOWLEDGEMENT

This work is partly supported by National Natural Science Foundation of China (NSFC grant numbers: U1836116, 61502205 and 61872167), and the project of Jiangsu provincial Six Talent Peaks (Grant number XYDXXJS-016).

## REFERENCES

[1] A. Orso and G. Rothermel, "Software testing: a research travelogue (2000–2014)," in *Proceedings of the on Future of Software Engineering*, 2014, pp. 117-132.

[2] M. J. Harrold, "Testing: a roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, 2000, pp. 61-72.

[3] T. A. Budd, "Mutation analysis: Ideas, examples, problems and prospects," *Computer Program Testing,* vol. 8, pp. i29-l48, 1981.

[4] T. Y. Chen, F.-C. Kuo, and H. Liu, "Application of a failure driven test profile in random testing," *IEEE Transactions on Reliability,* vol. 58, pp. 179-192, 2009.

[5] P. E. Ammann and J. C. Knight, "Data diversity: An approach to software fault tolerance," *IEEE Transactions on Computers,* pp. 418-425, 1988.

[6] P. G. Bishop, "The variation of software survival time for different operational input profiles (or why you can wait a long time for a big bug to fail)," in *Proceedings of the Twenty-Third International Symposium on Fault-Tolerant Computing, 1993. FTCS-23. Digest of Papers.*, , 1993, pp. 98-107.

[7] G. B. Finelli, "NASA software failure characterization experiments," *Reliability Engineering & System Safety,* vol. 32, pp. 155-169, 1991.

[8] L. J. White, "Software testing and verification," in *Advances in computers*. vol. 26, ed: Elsevier, 1987, pp. 335-391.

[9] K. P. Chan, T. Y. Chen, and D. Towey, "Restricted random testing," in *Software Quality—ECSQ 2002*, ed: Springer, 2002, pp. 321-330.

[10] T. Y. Chen, H. Leung, and I. Mak, "Adaptive random testing," in *Annual Asian Computing Science Conference*, 2004, pp. 320-329.

[11] 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,* vol. 16, pp. 553-584, 2006.

[12] A. Arcuri and L. Briand, "Adaptive random testing: An illusion of effectiveness?," in *Proceedings of International Symposium on Software Testing and Analysis*, 2011, pp. 265-275.

[13] C. Chow, T. Y. Chen, and T. Tse, "The art of divide and conquer: an innovative approach to improving the efficiency of adaptive random testing," in *in Proceedings on 13th International Conference on Quality Software (QSIC)*, 2013, pp. 268-275.

[14] A. Shahbazi, A. F. Tappenden, and J. Miller, "Centroidal voronoi tessellations-a new approach to random testing," *IEEE Transactions on Software Engineering,* vol. 39, pp. 163-183, 2013.

[15] C. Mao, T. Y. Chen, and F.-C. Kuo, "Out of sight, out of mind: a distance-aware forgetting strategy for adaptive random testing," *Science China Information Sciences,* vol. 60, p. 092106, 2017.

[16] R. Huang, H. Liu, X. Xie, and J. Chen, "Enhancing mirror adaptive random testing through dynamic partitioning," *Information and Software Technology,* vol. 67, pp. 13-29, 2015.

[17] M. Omari, J. Chen, H. Ackah-Arthur, and P. K. Kudjo, "Elimination by linear association: An effective and efficient static mirror adaptive random testing," *IEEE Access,* 2019.

[18] P. Borba, A. Cavalcanti, A. Sampaio, and J. Woodcook, *Testing Techniques in Software Engineering: Second Pernambuco Summer School on Software Engineering, PSSE 2007, Recife, Brazil, December 3-7, 2007, Revised Lectures* vol. 6153: Springer, 2010.

[19] F.-C. Kuo, T. Y. Chen, H. Liu, and W. K. Chan, "Enhancing adaptive random testing for programs with high dimensional input domains or failure-unrelated parameters," *Software Quality Journal,* vol. 16, pp. 303-327, 2008.

[20] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and S. P. Ng, "Mirror adaptive random testing," *Information and Software Technology,* vol. 46, pp. 1001-1010, 2004.

[21] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability,* vol. 22, pp. 67-120, 2012.

[22] J. Chen, L. Zhu, T. Y. Chen, D. Towey, F.-C. Kuo, R. Huang, *et al.*, "Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering," *Journal of Systems and Software,* vol. 135, pp. 107-125, 2018.

[23] T. Y. Chen, "Fundamentals of test case selection: Diversity, diversity, diversity," in *in Proceedings on the 2nd International Conference on Software Engineering and Data Mining (SEDM), 2010* 2010, pp. 723-724.

[24] T. Y. Chen, F.-C. Kuo, and H. Liu, "Enhancing adaptive random testing through partitioning by edge and centre," in *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian,* 2007, pp. 265-273.

[25] T. Y. Chen, F.-C. Kuo, and H. Liu, "Distribution metric driven adaptive random testing," in *Seventh International Conference on Quality Software (QSIC 2007)*, 2007, pp. 274-279.

[26] T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou, "On favourable conditions for adaptive random testing," *International Journal of Software Engineering and Knowledge Engineering,* vol. 17, pp. 805-825, 2007.

[27] F.-C. Kuo, T. Y. Chen, H. Liu, and W. Chan, "Enhancing adaptive random testing in high dimensional input domains," in *Proceedings of ACM Symposium on Applied Computing*, pp. 1467-1472.

[28] J. Mayer, "Lattice-based adaptive random testing," in *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, 2005, pp. 333-336.