Adaptive Random Testing Through Iterative Partitioning

Conference Paper *in* Lecture Notes in Computer Science · January 1970

DOI: 10.1007/11767077_13 · Source: DBLP

3 authors, including:

Zhi Quan Zhou
University of Wollongong
55 PUBLICATIONS **920** CITATIONS

SEE PROFILE

JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 27, 1449-1472 (2011)

On Adaptive Random Testing Through Iterative Partitioning*

TSONG YUEH CHEN[1], DE HAO HUANG[2] AND ZHI QUAN ZHOU[3,+]
[1]*Faculty of Information and Communication Technologies*
[2]*Information Technology Services*
Swinburne University of Technology
Hawthorn, Victoria 3122, Australia
[3]*School of Computer Science and Software Engineering*
University of Wollongong
Wollongong, NSW 2522, Australia
[+]*E-mail: zhiquan@uow.edu.au*

Random Testing (RT) is an important and fundamental approach to testing computer software. Adaptive Random Testing (ART) has been proposed to improve the fault-detection capability of RT. ART employs the location information of successful test cases (those that have been executed but not revealed a failure) to enforce an even spread of random test cases across the input domain. Distance-based ART (D-ART) and Restriction-based ART (R-ART) are the first two ART methods, which have considerably improved the fault-detection capability of RT. Both these methods, however, require additional computation to ensure the generation of evenly spread test cases. To reduce the overhead in test case generation, we present in this paper a new ART method using the notion of iterative partitioning. The input domain is divided into equally sized cells by a grid. The grid cells are categorized into three different groups according to their relative locations to successful test cases. In this way, our method can easily identify those grid cells that are far apart from all successful test cases for test case generation. Our method significantly reduces the time complexity, while keeping the high fault-detection capability.

Keywords: software testing, random testing, adaptive random testing, simulation, algorithm analysis

1. INTRODUCTION

The world becomes highly dependent on the effective operation of reliable software. However, as conceived by Myers [30] about 30 years ago, two basic problems in the software industry, namely the high development cost and the poor qualities of software, have had no significant improvement.

Testing plays a critical role in software quality improvement [19]. Among the various testing methods, *Random Testing* (RT) is a fundamental and straightforward approach [21, 23]. It has been successfully applied in many real-world applications [14, 15, 18, 27-29, 31, 32, 34]. In 1990, for example, Miller developed the fuzz system that generated random data streams to test programs in several versions of the UNIX system [27, 28]. It has been reported that 24% to 33% of the programs tested failed on valid inputs that are randomly generated. Apart from academia, RT techniques have been adopted by large soft-

Received July 31, 2009; revised November 4, 2009; accepted January 5, 2010.
Communicated by Chih-Ping Chu.
* The preliminary version of this paper was presented at the 11th Ada-Europe International Conference on Reliable Software Technologies (Ada-Europe 2006). This project was partially supported by an Australian Research Council Discovery Grant (project No. DP0880295) and a Small Grant of the University of Wollongong.
+ Corresponding author.

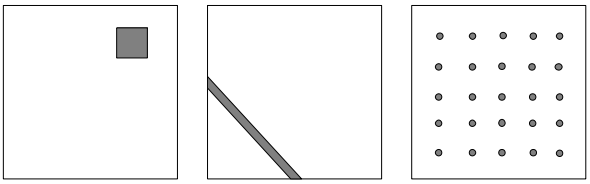


Fig. 1. Examples of the three types of failure patterns.

ware corporations and implemented in their software testing tools (for example, [2]).

In random testing, each test case is selected randomly regardless of the previously executed test cases. Malaiya [25] introduced a concept of *antirandom testing*, which proposes that the first test case be selected randomly, and each subsequent test case be selected in such a way that its total distance to all the previously executed test cases is maximum. In this way it may become quicker to detect a failure.

An *Adaptive Random Testing* (ART) strategy has been proposed to improve the performance of RT in terms of using fewer test cases to detect the first failure [8, 24]. ART is based on the observation that failure-causing inputs form different failure patterns, which can be coarsely classified into three types, namely block, strip and point patterns [5]. In the block and strip patterns, failure-causing inputs are clustered in one or a few regions. In other words, the failure-causing inputs are "denser" in some areas of the input domain. Examples of these failure patterns for a program with a 2-dimensional input domain are given in the schematic diagrams in Fig. 1, where the outer square represents the input domain and the shaded areas represent the failure-causing inputs. Readers interested in the types of faults that yield these failure patterns can refer to [7]. Intuitively, when failure-causing inputs are clustered, selecting a test case close to the previously executed ones that have not revealed a failure is less likely to reveal a failure. ART, therefore, proposes to have test cases evenly spread throughout the input domain.

ART is different from antirandom testing. In antirandom testing, only the first test case is selected randomly. Once the first test case is selected, the sequence of all test cases is deterministic. On the other hand, ART well preserves the randomness since all test cases in ART are randomly selected. Furthermore, the number of test cases in antirandom testing must be decided in the first place. However, ART does not have such a limitation. ART also differs from the *adaptive testing* method developed by Cai *et al.* [3] in both objectives and approaches. The former is developed as an enhancement to RT with an aim of detecting the first failure quickly; whereas the latter involves dynamic selection of more than one testing strategy in one testing process, with an aim of minimizing the total cost of detecting and removing multiple defects.

It has been reported that ART significantly improves the fault-detection capability of RT. In [8], 12 programs were studied to investigate the fault-detection capability of ART. These programs are faulty versions of some numerical computation programs published in [1] with about 30 to 200 statements each. Typical errors were seeded into the programs. It was found that the fault-detection capability of ART (in terms of average number of test cases required to reveal the first failure) had generally achieved an improvement in the order of 30%, and occasionally up to 50%, over that of RT. Following previous practices [5, 11], F-measure (that is, the number of test cases needed to be run to detect the

first failure [5]), is adopted as a metric to compare the failure-detection capability of different testing methods.

The basis of ART is to evenly spread test cases. As the concept of "even spread" can be implemented in different ways, several ART methods (algorithms) have been proposed. Distance-based ART (D-ART) [8, 24] and Restriction-based ART (R-ART) [4] are the first two attempts, which have significantly improved the fault-detection capability of RT. These methods, however, require additional computational overhead in generating test cases.

To reduce the overhead while retaining the high fault-detection capability, we propose a new ART method, namely *Adaptive Random Testing through Iterative Partitioning* (IP-ART). This paper is organized as follows. In section 2 we review and analyze some related work in the area of ART. Section 3 introduces our method, IP-ART. Its time complexity and fault-detection capability are studied. Section 4 conducts further studies into IP-ART. Discussions and conclusion are given in section 5.

2. REVIEW AND ANALYSIS OF SOME RELATED WORK ON ART

2.1 D-ART and R-ART

In this section, D-ART and R-ART methods are briefly reviewed with a focus on their time complexity.

D-ART [8, 10, 24] selects the next test case from a fixed-size set of candidates (this set of candidates is randomly generated every time), based on the criterion of maximizing the minimum distance between the next test case and all the successful test cases (that is, test cases that have not revealed a failure). In this process, the computation and comparison of the distances between test case candidates and all the successful test cases are the main source of computational cost. Suppose the size of the candidate set is e, where $e > 1$, and the ith test case is to be generated, where $i > 0$. The distance computation and comparison for generating the ith test case is therefore in the order of $e \times (i - 1)$. Note that when $i = 1$, only one random input will be generated and, hence, there is no distance computation or comparison. Let n be the total number of test cases finally generated. Let e be a constant. The order of the time complexity of D-ART is calculated as follows,

$$\sum_{i=1}^{n} e \times (i-1) = e \times \sum_{i=1}^{n-1} i \in O(n^2).$$

The time complexity of D-ART is therefore quadratic. As recommended in [8, 24], we set e to 10 in the experiments.

R-ART [4] does not use the candidate set. Instead, it sets an exclusion zone around each successful test case. Candidates are randomly generated one by one until a candidate falls outside of all exclusion zones. This candidate is then selected as the next test case. All exclusion zones have an equal size, which is determined by the number of successful test cases n and the target exclusion area A_T. The radius of these exclusion zones

can be calculated using the formula $\sqrt{A_T / (n \times \pi)}$. The target exclusion area A_T relates to both the size (D) of the input domain and the target exclusion ratio R_T. Their relation is $R_T = A_T/D$. Because exclusion zones may overlap and may extend out over the border of the input domain, the actual exclusion area A_A is less than the target exclusion area A_T. Let R_A be the actual exclusion ratio, defined as A_A/D. R_A is therefore also less than R_T. In [33], the relationship between R_A and R_T is studied empirically. As shown in Table 1, the difference between R_A and R_T relates to the value of R_T and the number of exclusion zones (that is, the number n of test cases generated). The gap between R_A and R_T is more significant when R_T is relatively large. Even when R_T becomes 150%, R_A still ranges between 86.82% and 91.67%. Generally, when n is small and increases, R_A will slightly increase. When n is large, R_A becomes steady. In our experiments, we followed the practice in [4] to set the target exclusion ratio to 150%.

Table 1. Relationship between the target exclusion ratio and the average actual exclusion ratio in R-ART, adopted from [33].

R_T	R_A					
	number (n) of exclusion zones					
	10	100	500	1000	2000	10000
1%	0.99%	0.99%	1.00%	1.00%	1.00%	1.00%
10%	9.43%	9.73%	9.80%	9.83%	9.84%	9.86%
30%	26.41%	27.98%	28.40%	28.50%	28.58%	28.67%
50%	41.39%	44.57%	45.43%	45.64%	45.79%	46.00%
100%	70.16%	76.76%	78.52%	78.95%	79.24%	79.64%
150%	86.82%	91.33%	91.67%	91.66%	91.63%	91.55%

The analysis of the time complexity of R-ART differs from that of D-ART in that the number of candidates needed to generate a test case in R-ART is not fixed, and is related to R_A as follows. The probability that the first candidate will be generated from outside of all exclusion zones is $1 - R_A$, denoted as $P(X = 1) = 1 - R_A$, where X denotes the number of candidates needed to generate a test case outside of all exclusion zones. If the first candidate falls within one of the exclusion zones (the probability is R_A), a second candidate needs to be generated. Suppose the selection is with replacement. The probability that the second candidate falls outside of all exclusion zones is still $1 - R_A$. Hence, $P(\mathrm{X} = 2) = (1 - R_A) \times R_A$. We can conclude that $P(X = n) = (1 - R_A) \times R_A^{n-1}$. This is an instance of the geometric distribution. The expected number of trials to obtain a candidate outside of all exclusion zones is $\frac{1}{1-R_A}$.

Since R_A becomes steady when n is large, we can regard it as a constant. Hence, for a target exclusion ratio R_T, the number of candidates needed to generate the next test case can also be regarded as a constant c. Consequently, the order of the time complexity of R-ART is:

$$\sum_{i=1}^{n} c \times (i-1) = c \times \sum_{i=1}^{n-1} i \in O(n^2).$$

The time complexity of R-ART is therefore also quadratic.

2.2 Mirror ART

Mirror Adaptive Random Testing (M-ART) is an attempt to reduce the overhead of D-ART and R-ART [7]. In M-ART, the input domain is first partitioned into equally sized disjoint subdomains, one of which is regarded as the *original subdomain*, and the others are *mirror subdomains*. D-ART or R-ART are only applied in the original subdomain, and simple mirror functions are used to map the test cases into other subdomains. Suppose the input domain is partitioned into w subdomains, the distance computation will be reduced to about $1/w^2$ of the computation originally required by D-ART or R-ART. Empirical study shows that the fault-detection capability of M-ART is similar to that of D-ART and R-ART [7]. Nevertheless, the time complexity of M-ART is still quadratic, and the number of subdomains (w) should be kept small to ensure randomness.

2.3 ART through Dynamic Partitioning

ART through Dynamic Partitioning (DP-ART) is another attempt to reduce the overhead of computations [6]. DP-ART is inspired by partitioning testing, which incrementally divides the input domain to identify the sparsely populated partitions to serve as test case generation region. This approach has two partitioning schemes, namely *ART by Random Partitioning* (RP-ART) and *ART by Bisection* (B-ART). RP-ART partitions the input domain using the successful test cases themselves (that is, dividing a region by drawing straight lines perpendicular to each other crossing at the most recently executed successful test case), and then chooses the subdomain having the largest size to generate the next test case. B-ART divides the input domain into subdomains of equal size, and then randomly chooses a subdomain that does not contain any successful test case as the region to generate the next test case. If all subdomains contain successful test cases, then each subdomain will be subdivided into halves and the testing process is repeated until a failure is detected or the testing resources are exhausted. Because DP-ART does not involve distance computation and comparison, its time cost is very low compared with that of D-ART and R-ART, but at the cost that its fault-detection capability is lower than that of the other ART methods.

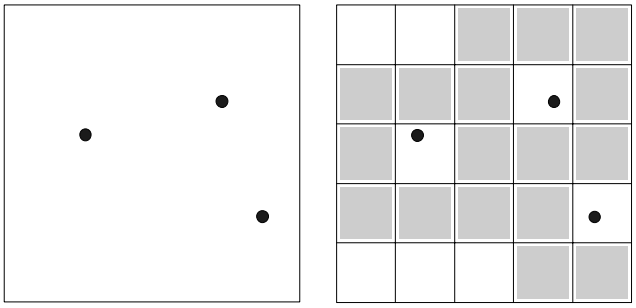
3. THE BASIC ALGORITHM OF ART THROUGH ITERATIVE PARTITIONING (IP-ART)

To further reduce the overhead of ART while keeping a high fault-detection capability, we propose in this paper a new ART method, namely ART through Iterative Partitioning (IP-ART). The basis of ART is to evenly spread test cases by exploiting the location information of successful test cases. Conventionally, partitioning is a strategy to group elements having similar behaviors in some sense into the same subdomain [20]. IP-ART uses partitioning to identify a test case generation region, where inputs have higher chance of being far apart from all successful test cases. If such a test case generation region cannot be identified under current partitioning scheme, the input domain will be repartitioned using a finer partitioning scheme. Since IP-ART does not require the generation of extra candidates and avoids the distance computations and comparisons, it

has much lower computational overhead while keeping a high fault-detection capability comparable to that of D-ART and R-ART.

3.1 Overview

To illustrate the basic idea of IP-ART, let us consider Fig. 2. Fig. 2 (a) shows a square input domain. Suppose we have already run three test cases, represented by the black points, but no failure has been revealed so far. According to the basis of ART, now we want to generate a new test case far apart from all the existing ones. The input domain can be partitioned, for example, using a 5 × 5 grid as shown in Fig. 2 (b). We call the cells that contain the successful test cases *occupied cells*. Obviously, the next test case should not be generated from occupied cells. Furthermore, we call the cells that are surrounding neighbours of the occupied cells *adjacent cells*. These cells themselves do not contain any successful test case but share at least a common side or a common vertex with an occupied cell, as shaded in Fig. 2 (b). If the next test case is generated from an adjacent cell, it will still have a chance of being close to previous test cases. Hence, adjacent cells are also not desirable for test case generation. IP-ART therefore requires that the next test case be generated from the regions that are neither occupied nor adjacent cells, known as *candidate cells*. The blank areas in Fig. 2 (b) represent five candidate cells. Obviously, a test case generated from a candidate cell will have a higher chance of being far apart from all existing test cases. In short, the notion of exclusion is applied in IP-ART, which excludes the regions surrounding successful test cases as test case generation regions. After a new test case is generated, the lists of occupied, adjacent, and candidate cells need to be updated and, sooner or later, all candidate cells will be used up. Then IP-ART will discard the current partitioning scheme (the $p \times p$ grid) and generate a finer $(p + 1) \times (p + 1)$ grid to partition the input domain all over again.



(a) (b)

Fig. 2. Partitioning the input domain and classifying grid cells.

3.2 The Grid Coordinates Used in IP-ART

Grid is a concept widely used in many areas. For instance, to make a map more manageable, it can be partitioned by a set of vertical and horizontal lines into regularly sized grid cells. The size of grid cells is selected according to the resolution of the grid

requested by the user. A grid with resolution of p means that each dimension of the input domain is partitioned into p segments. In a k-dimensional input domain, there are all together p^k grid cells. Whenever a region is required, it can be referred to by the coordinates of the grid directly rather than search from the whole map. Similarly, in IP-ART, the whole input domain is divided into equally sized grid cells under a certain resolution, and any grid cell can be referred to using coordinates.

Fig. 3 depicts the partitioning of a 2-dimensional input domain. Suppose we have a program *prog*(a, b) where a and b are real numbers and $0 \le a, b < M$, and suppose the input domain is partitioned by a $p \times p$ grid, where p is a positive integer. Let C be the side length of each grid cell, then $C = M/p$. Each grid cell is labeled with a pair of integers. The grid cell (u, v) refers to the cell whose lower left vertex has coordinates ($u \times C$, $v \times C$). To conform to rounding conventions, a point on a vertical border belongs to the cell on its right, and a point on a horizontal border belongs to the cell above it. It is straightforward to map any point in the input domain to a grid cell. For any valid test case (x, y), it is mapped into grid cell ($\lfloor x/C \rfloor$, $\lfloor y/C \rfloor$). In Fig. 3, the input domain is partitioned by a 10×10 grid, and M is set to 100. As an example, the test case 28.8, 12.6) is mapped into grid cell (2, 1).

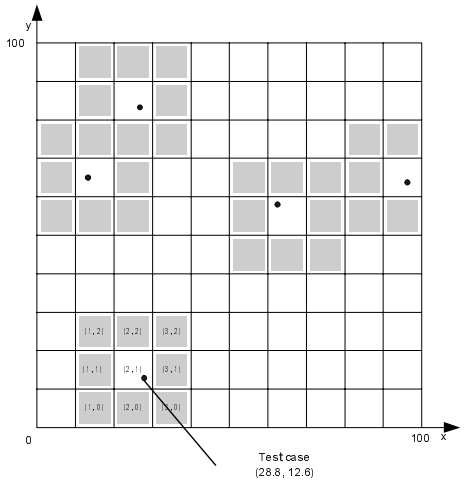


Fig. 3. Grid coordinates and the mapping of test cases into grid cells.

3.3 Categorization of Grid Cells

As introduced earlier, in IP-ART there are three types of grid cells, namely occupied cells that contain successful test cases, adjacent cells that do not contain any test case but are surrounding neighbours of some occupied cells, and candidate cells (those that are neither occupied nor adjacent cells). In a k-dimensional input space, let (o_1, o_2, …, o_k) and (a_1, a_2, …, a_k) be the coordinates of an occupied cell and one of its adjacent cells, respectively, where $1 \le k$, then $|a_i - o_i| \le 1$, for $i = 1, 2, \ldots, k$. Obviously, an occupied cell has $3^k - 1$ adjacent cells around it.

3.4 The Algorithm

In IP-ART, we need to first decide the resolution of the grid for partitioning the input domain. Initially, a coarse grid is appropriate, because the number of successful test cases is small at the early stage of testing and if the grid is too fine at the beginning, many candidate cells will not be sufficiently far away from occupied cells. Further analysis on the resolution of initial grid will be conducted in section 4.1. If no failure is revealed and no candidate cell is available, then the current $p \times p$ partitioning scheme will be discarded and a finer partitioning scheme using a $(p + 1) \times (p + 1)$ grid will be applied to partition the input domain all over again.

Algorithm 1 is the IP-ART algorithm for a 2-dimensional square input domain with a size of $M \times M$. Extension to input domains of higher dimensionalities is straightforward. A Boolean matrix *GridCells* is used to represent the grid cells. Each entry of the matrix corresponds to a grid cell. If a matrix entry corresponds to an occupied or adjacent cell, it will be assigned a value of *F*; otherwise it corresponds to a candidate cell and will be assigned a value of *T*. In the algorithm, *p* indicates the resolution of the grid.

Algorithm 1 The basic IP-ART algorithm

It is assumed that the program under test is program (parameter1, parameter2), where parameter1 and parameter2 are real numbers and $0 \le parameter1, parameter2 < M$.

1. Initialize the grid by setting $p = 1$; set the successful test case set, *S*, to be empty.
2. Construct a $p \times p$ Boolean matrix, *GridCells*, and assign *t* to all its entries. Use *CntCandidateCell* to count the number of candidate cells, and set $CntCandidateCell = p \times p$.
3. Map each successful test case ($parameter1 = x$, $parameter2 = y$) in *S* into a grid cell by assigning *f* to the corresponding occupied cell $GridCells(\lfloor x \times p/M \rfloor, \lfloor y \times p/M \rfloor)$. Update *CntCandidateCell*.
4. For each occupied cell $GridCells(u, v)$, assign *f* to all its adjacent cells, namely $GridCells(u - 1, v - 1)$, $GridCells(u - 1, v)$, $GridCells(u - 1, v + 1)$, $GridCells(u, v - 1)$, $GridCells(u, v + 1)$, $GridCells(u + 1, v - 1)$, $GridCells(u + 1, v)$, $GridCells(u + 1, v + 1)$, as necessary. Update *CntCandidateCell*.
5. While ($CntCandidateCell > 0$)
   (a) Randomly select an available candidate cell, *R*, as the test case generation region.
   (b) Randomly generate a test case *tc* from within *R*.
   (c) If *tc* is a failure-causing input, report the failure and terminate. Otherwise add *tc* to *S*, assign *f* to entries of *GridCells* that correspond to *R* and all its adjacent cells; update *CntCandidateCell*.
6. Discard (release) the Boolean matrix *GridCells*. Set $p = p + 1$.
7. If testing resources are not exhausted go to step 2.

An example that illustrates Algorithm 1 is given in Fig. 4. Initially, no test case is generated. The only candidate cell is the whole input domain. A test case t_1 is randomly generated and the cell becomes occupied. Suppose this is not a failure-causing input.

Then, since there is no more candidate cells, the input domain is partitioned by a 2×2 grid. The successful test case is mapped into the new partition and cell (1, 1) is identified as the occupied cell, and cells (0, 0), (0, 1) and (1, 0) are adjacent cells. This partitioning scheme cannot provide us with any candidate cell. Therefore, this partitioning scheme is discarded as shown in Fig. 4 (c). The input domain is then partitioned again using a 3×3 grid as shown in Fig. 4 (d), and the occupied and adjacent cells are accordingly identified. Now cells (0, 2), (0, 1), (0, 0), (1, 0) and (2, 0) are candidate cells and one of them, say, cell (0, 1), is randomly selected as the region for generating the next test case. A test case t_2 is randomly generated from within this region. Then the adjacent cells surrounding t_2 are marked as shown in Fig. 4 (e). Fig. 4 (e) also shows that the cell (2, 0) is now the only remaining candidate cell and, therefore, the third test case will be generated from this region. This process will be repeated until a failure is detected or the testing resources are exhausted.

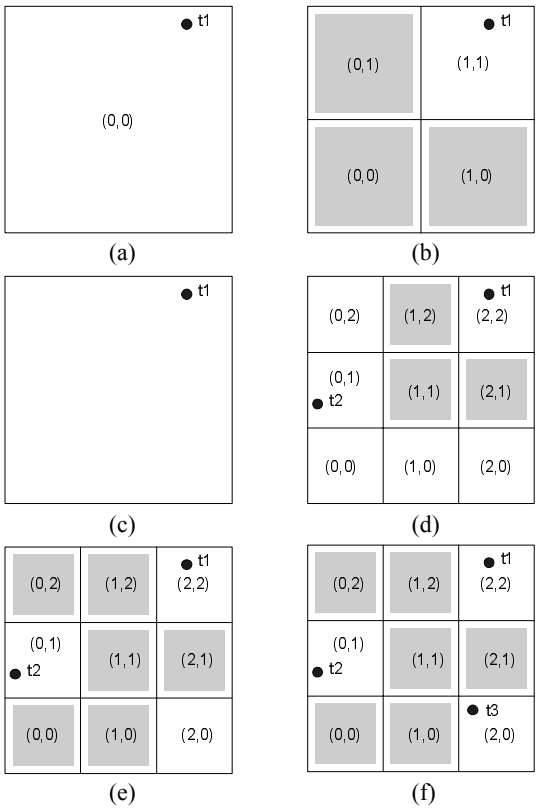


Fig. 4. An example that illustrates the IP-ART method.

3.5 Complexity of the IP-ART Method

The time complexity of Algorithm 1 is analyzed as follows. Since each test case is directly generated from within a candidate cell, this step costs only constant running time.

The main cost of Algorithm 1 is to map all the successful test cases (in E) to new grid cells when the input domain is repartitioned.

We first prove that, at any time, each cell contains at most one successful test case. Firstly, new test cases cannot be selected from within occupied cells in test case generation process according to the method. Secondly, two test cases cannot be mapped into the same cell in the mapping process because of the following reason. Suppose the side length of the input domain is 1 in each dimension. Suppose the old partitioning scheme is p, which means each dimension of the input domain is partitioned into p equally sized segments and, hence, the side length of each grid cell is $1/p$. Suppose we are now going to repartition the input domain using the new partitioning scheme $p + 1$, where the side length of each grid cell is $1/(p + 1)$. Since a test case cannot be generated from adjacent cells, in the old partition the minimum distance between two test cases in each dimension is $1/p$. After repartitioning, the side length of each grid cell is reduced to $1/(p + 1)$. Since $1/(p + 1) < 1/p$ when $p \geq 1$, a grid cell cannot contain more than one test case.

As mentioned previously, the dominant cost is mapping all the successful test cases when repartitioning occurs. So we need to identify the maximum number of times the input domain may be repartitioned for a given F-measure n. Let k be the dimensionality of the input domain, m be the final resolution of the grid (there are all together m^k cells). Since the partitioning starts from an initial resolution 1, the input domain is repartitioned $m - 1$ times when the final resolution m is reached. Since each successful test case can create at most 3^{k-1} adjacent cells (hence 3^k cells are excluded for test case generation including itself), and repartitioning only occurs when the current partitioning schemes are full (that is, when there is no more candidate cells), we can conclude that $(m - 1)^k < 3^k \times n$, that is, $m < 3n^{1/k} + 1$.

Since each cell can contain at most one successful test case, there are no more than p^k successful test cases in transition from partitioning scheme p to $p + 1$. Therefore, the number of mapping successful test cases should be less than

$$\sum_{p=1}^{m-1} p^k.$$

This is exactly the classic Bernoulli's power sum problem [16]. The result of this summation is in $O((m - 1)^{k+1})$. Since $m < 3n^{1/k} + 1$, the time complexity of IP-ART is in $O(3^{(k+1)} \cdot n^{(1+1/k)})$. Therefore, for the 2-dimensional input domain, the time complexity is in $O(n^{1.5})$. For 3- and 4-dimensional input domains, the time complexities are in $O(n^{1.33})$ and $O(n^{1.25})$, respectively. We would like to point out that IP-ART is not suitable for programs with input domains that have a very high dimensionality. When k, which is the dimensionality of the input domains, is large, the coefficient $3^{(k+1)}$ will become even larger and, hence, will significantly increase the running time of the algorithm. Furthermore, in this situation a large amount of memory space is also required. Since the previous Boolean matrix *GridCells* will be released when repartitioning the input domain, the space complexity analysis can just focus on the final partitioning scheme with resolution m. Since $m < 3n^{1/k} + 1$, for a given F-measure n, the maximum number of grid cells required is

$$m^k < (3n^{1/k} + 1)^k \in O(3^k \cdot n).$$

In very high dimensional input domains, the coefficient 3^k will be a large number and the algorithm will need a large amount of space.

3.6 Simulation Studies

To compare the fault-detection capability of IP-ART with that of other methods, a series of simulations have been conducted under the assumption of uniform distribution of inputs. The (simulated) input domain was square. For each test run, a (simulated) failure region with a specified failure rate θ ($\theta = area_2 \div area_1$ where $area_1$ is the area of the input domain and $area_2$ is the area of the failure region) and a specified failure pattern was randomly placed in the input domain. For the block failure pattern, it was a square; for the strip failure pattern, two points on the adjacent borders of the input domain were randomly chosen and connected to form a strip with specified size; for the point failure pattern, 10 equally sized circular regions were randomly placed in the input domain without overlapping so that their total area will equal the specified failure rate. Each simulation (test run) was conducted as follows: (1) randomly place the failure region in the input domain; (2) set *counter* to 0; (3) generate a test case using the IP-ART algorithm and increase *counter* by 1; (4) if the generated test case is not located in the failure region then goto step 3; (5) print *counter* (that is, the F-measure of the present test run) and stop.

In our simulations, the Central Limit Theorem [17] is applied to decide the sample size. To estimate the mean of F-measure with an accuracy range of ± r% and a confidence level of $(1 - \alpha) \times 100\%$, the sample size S required should be at least

$$S = \left(\frac{100 \cdot z \cdot \sigma}{r \cdot \mu} \right)^2,$$

where z is the normal variate of the desired confidence level, μ is the population mean and σ is the population standard deviation. In our simulations, the confidence level was set to 95% and accuracy range was set to ± 5%. For confidence level 95%, statistical table gives 1.96 for z. Since μ and σ were unknown, the mean ($\bar{x}$) and the standard deviation (s) of the collected sample data were used instead respectively. Therefore, we have

$$S = \left(\frac{100 \times 1.96 \times s}{5 \times \bar{x}} \right)^2.$$

We use the above equation to decide when the process of collecting samples should be stopped.

We use the *relative F-measure* to compare the fault-detection capability of ART and RT, which is calculated as $F_{ART}/F_{RT} \times 100\%$, where F_{ART} is the F-measure for an ART method and F_{RT} is the F-measure for RT. Theoretically, $F_{RT} = 1/\theta$ if the test cases are randomly selected with replacement. Obviously, the lower the relative F-measure is, the more improvement the ART method achieves.

Table 2 presents the results of the simulations conducted in a 2-dimensional input domain with failure rate θ varying from 0.01 to 0.001 with respect to the three types of failure patterns. For the block pattern, the relative F-measure of IP-ART is 60-63%. For the strip pattern, the relative F-measure of IP-ART is 77-78%. For the point pattern, the improvement is not significant.

Table 3 compares the fault-detection capability of IP-ART and four major ART methods with θ ranging from 0.01 to 0.001. We can see that the fault-detection capability of IP-ART is comparable to that of D-ART and R-ART, and obviously better than that of RP-ART and B-ART.

Table 2. Relative F-measures of IP-ART in a 2-dimensional input domain under 3 failure patterns.

Failure Rate θ	F-measure of RT (F_{RT})	Block Pattern		Strip Pattern		Point Pattern	
		F_{ART}	F_{ART}/F_{RT}	F_{ART}	F_{ART}/F_{RT}	F_{ART}	F_{ART}/F_{RT}
0.01	100	63	63%	78	78%	96	96%
0.005	200	122	61%	156	78%	188	94%
0.002	500	301	60%	387	77%	476	95%
0.001	1000	605	61%	781	78%	925	93%

Table 3. Comparison of relative F-measures of IP-ART and other ART methods on a 2-dimensional input domain (under the block failure pattern).

Failure rate θ	F_{ART}/F_{RT}				
	D-ART	R-ART	RP-ART	B-ART	IP-ART
0.01	67%	65%	76%	75%	63%
0.005	66%	63%	77%	74%	61%
0.002	65%	62%	79%	74%	60%
0.001	65%	62%	80%	75%	61%

Table 4. Recorded CPU time of different ART methods (with θ = 0.001 and under the block failure pattern, for a run of 5000 trials each method).

ART Methods	D-ART	R-ART	RP-ART	B-ART	IP-ART
CPU time (in seconds)	3645.8	1562.1	1141.1	6.4	12.0

Since time complexity hides constants that may be important in practice, we also investigate the actual running time of these ART methods. All experiments were conducted in the same hardware and software environment: an HP Compaq PC with a 2.6 GHz Intel Pentium IV processor equipped with 256M RAM, running Microsoft Windows XP SP1. Table 4 lists the CPU time we recorded for running 5000 trials of each method. It shows that the running time of IP-ART is negligible compared with D-ART, R-ART and RP-ART, and less than twice that of B-ART. The table shows that the running time of RP-ART is also high. This is because, although it avoids distance computations, RP-ART has to search in the entire input domain for the largest subdomain to generate the next test case, and this contributes greatly to the overhead.

4. FURTHER STUDIES ON IP-ART

The basic IP-ART method has been introduced and examined in the previous section. There are, however, the following factors in the IP-ART method, which are worth further investigation.

1. Instead of initializing the input domain to a 1 × 1 grid, can a finer grid be used to partition the input domain at the very beginning?
2. Are the test cases generated by IP-ART uniformly distributed?

We shall investigate these two questions in the following subsections.

4.1 IP-ART with a Finer Initial Grid

In the basic IP-ART method, the input domain is initially partitioned by a coarse grid. A finer grid will be used to repartition the input domain if no candidate cells are available. As analyzed in section 3.5, the major cost of IP-ART is to map all the successful test cases when the input domain is repartitioned by a finer grid. In this section, we would like to investigate whether we should use a finer grid to partition the input domain from the beginning, so that the frequency of repartitioning can be reduced. The study is conducted through analysis and simulations.

Firstly, let us investigate the case where a failure can be detected without repartitioning. For ease of presentation, the analysis is conducted in a 2-dimensional input domain with a single block failure region. Let θ be the failure rate and m be the resolution of partitioning scheme. Then the input domain is partitioned into $m \times m$ equally sized grid cells. According to the IP-ART test case selection strategy, each successful test case results in up to 3 × 3 cells being excluded for test case generation. Only when the failure region occupies at least 3 × 3 grid cells, can the current partitioning scheme guarantee to detect the failure. This is because, if a failure region has a size of 1 × 1 (that is, it contains only one grid cell), and when it is an adjacent cell (that is, when it is located on the border of an excluded region), it cannot be detected using the current partitioning scheme; if a failure region has a size of 2 × 2, and when half of the failure region is located on the border of an excluded region and the other half is located on the border of another excluded region, it cannot be detected using the current partitioning scheme; if a failure region has a size of 3 × 3, however, it will be impossible for *all* these 3 × 3 = 9 grid cells to be adjacent cells (that is, they cannot be all located on the borders of certain excluded regions). This is because any adjacent cell must have a neighbour that is an occupied cell. Therefore, if the center of the failure region is an adjacent cell, then one of the remaining eight grid cells that surround it must be an occupied cell. Hence, it is impossible for all the nine grid cells in the failure region to be excluded for test case generation.

As a result, we have proved that only when the failure region occupies at least 3 × 3 grid cells, can the current partitioning scheme guarantee to detect the failure. In this situation, $\theta \geq 9/m^2$. This means that the initial partitioning scheme m should be equal to or greater than $\sqrt{9/\theta}$.

Then, the second question arises: to reduce the frequency of repartitioning, can we

just choose a very large m? Note that if it is guaranteed that a failure can be detected under current partitioning scheme (that is $m \geq \sqrt{9/\theta}$, then test case selection process in IP-ART is simply equivalent to random sampling *without* replacement, because test cases cannot be generated from the excluded cells.

For ease of presentation, let us suppose an idealized situation. Suppose the failure region occupies exactly $3r \times 3r$ (r is an integer and $r \geq 1$) grid cells in a 2-dimensional input domain. And the excluded cells (including occupied cells and adjacent cells) have no overlapping and all reside in the input domain, so that each successful test case can exclude 3×3 grid cells. To satisfy this assumption, $m \times m$, the number of grid cells in the input domain, should be a multiple of 9. The failure rate can be denoted as $\theta = 9r^2/m^2$. The F-measure for random testing with replacement is therefore $F_{RT} = 1/\theta = m^2/9r^2$. As noted above, in this idealized situation, the test case selection process in IP-ART is equivalent to random sampling without replacement from a test set of j elements of which k elements will reveal failures. As studied in Lemma 8 of [11], the average number of trials to detect the first failure (F-measure) can be calculated as

$$F(j,\ k) = \frac{j+1}{k+1}.$$

Here, the size of the test set j is $m^2/9$, since it is supposed the excluded cells have no overlapping and all reside in the input domain and a successful test case can exclude 3×3 grid cells. The number of elements that can reveal failure is r^2, since the failure region occupies exactly $3r \times 3r$ grid cells. Therefore,

$$F(m^2/9, r^2) = \frac{m^2/9\ +\ 1}{r^2+1} = \frac{r^2 \cdot F_{RT} + 1}{r^2 + 1}$$
$$\approx \frac{r^2 \cdot F_{RT}}{r^2 + 1} = \frac{F_{RT}}{1/r^2 + 1}.$$

When $r = 1$, $F(m^2/9, r^2)$ has the minimal value: $\frac{1}{2}F_{RT}$. When r increases, $F(m^2/9, r^2)$ will approach to F_{RT}.

Now two conclusions can be drawn. Firstly, if the failure region occupies exactly 3×3 grid cells ($r = 1$), then the failure can be detected without repartitioning and a low F-measure can be achieved. Since r is proportional to m, m needs to be small in order to have a low F-measure. Secondly, the F-measure in this idealized situation is *nearly* the optimal F-measure of any testing strategy where the size, shape, and orientation (but not the location) of failure regions are known [11, 26].

Although such an idealized situation rarely occurs, the above analysis gives us an insight into the relationship among the resolution of the initial partitioning scheme, failure rate, and F-measure. The resolution (m) of the initial partitioning scheme should be set against the failure rate (θ): the smaller the failure rate is, the finer the initial partitioning scheme should be. In reality, however, it is difficult to precisely estimate the failure rate θ. We suggest, therefore, overestimating θ (that is, assuming θ is large). Otherwise the initial partitioning scheme will be too fine to achieve a low F-measure. When θ is overestimated, a coarse initial partitioning scheme will be used (that is, m is small).

Consequently, the failure region may not occupy at least 3 × 3 grid cells and hence it cannot not guaranteed that a failure can be detected under initial partitioning scheme. Even if such an initial partitioning scheme may not reveal a failure, we can repeatedly repartition the input domain, and such repartitioning will not affect the F-measure because, as proved earlier, each grid cell always contains at most one test case, and the test case selection sequence is irrelevant. Setting the initial partitioning scheme to m instead of 1 also saves the cost in partitioning the input domain from 1 to m.

This guideline is confirmed by simulation results. The simulations were conducted under a 2-dimensional square input domain and a square failure pattern, with failure rate θ varied from 0.01 to 0.001. The input domain is initially partitioned by a $p \times p$ grid, where p = 1, 5, 10, 15, …, 95, 100. Firstly, let us look at the difference between the initial and the final partitioning schemes. Suppose the final partitioning scheme is a $q \times q$ grid, then the difference is given by $d = q - p$. The results are shown in Fig. 5. For each failure rate, the difference becomes zero after a certain point (which means that a failure can be detected under initial partitioning scheme), and we call this point *steady partitioning scheme*. The steady partitioning scheme depends on the failure rate: the larger the failure rate, the lower the steady partitioning scheme. This result confirms that few repartitions will be required if the initial partitioning scheme is close to an "optimal" value, which depend on the failure rate.

Secondly, we would like to investigate the relative F-measures of IP-ART on different initial partitioning schemes. As shown in Fig. 6, the relative F-measures keep steady when the initial partitioning scheme is smaller than a certain value. After that point, the relative F-measure begins to increase (that is, the fault-detection capability becomes worse). Note that, for each failure rate, that point is very close to the steady partitioning scheme. If the initial partitioning scheme is coarser than the steady initial partitioning scheme, the best performance can still be achieved after several iterations of repartitioning. If initial partitioning scheme is finer than the steady initial partitioning scheme, the performance will be compromised. This result confirms that our guideline of overestimating θ and selecting a smaller m as the resolution of the initial partitioning scheme is appropriate.

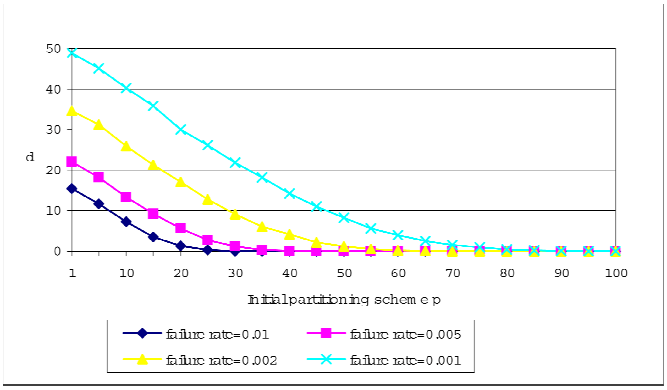


Fig. 5. Differences between the final (q) and initial (p) partitioning schemes in IP-ART, where $d = q - p$.

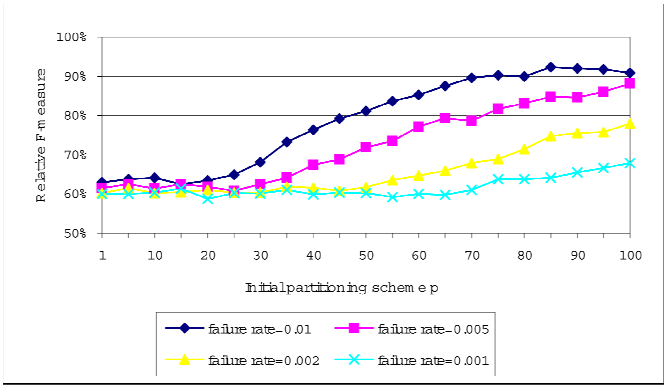


Fig. 6. Relative F-measures of IP-ART on different initial partitioning schemes.

4.2 Tackling the Boundary Effect

Like D-ART and R-ART, the basic IP-ART algorithm attempts to generate evenly spread test cases by making them far apart from each other. We note, however, that being "far apart from each other" does not necessarily imply being "evenly spread" over the input domain. In fact, in IP-ART, cells near the boundary of the input domain have a higher chance to be selected as test case generation regions. This is because no successful test cases exist outside of the boundary of the input domain. Therefore, this method has a preference for regions close to the boundary of the input domain when generating test cases. We refer to this phenomenon as the *boundary effect*. Consequently, the fault-detection capability depends on the location of the failure region.

To investigate into the boundary effect, we studied the spatial distributions of the test cases generated by IP-ART without considering the failure regions.

In each trial of test case generation, the locations of the first n test cases, where n = 1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 40, 50, 100, 500, or 1000, were recorded. A million independent trials were conducted. The spatial distributions were studied for the first n test cases of each trial for the respective values of n. Without loss of generality, the positions of the test cases were projected onto one dimension, since IP-ART treats every dimension equally and independently.

The simulation was conducted in a 2-dimensional square input domain having a unit size. The test case distributions in one dimension are illustrated as histograms with equal bins of size 0.01, consisting of 0 to 0.01, 0.01 to 0.02, and so on. The number of test cases that reside within each bin is computed. For a fair comparison of the distributions in different test case generation stages, the numbers of test cases in the histograms were normalized to $1/n$ of the actual numbers.

These histograms are shown in Fig. 7. For small n, the histograms look like two ladders with lower steps in the centre and higher steps towards each boundary respectively, because of the partitioning of the input domain. Then, more test cases are generated in the centre of the input domain, but still less than the test cases near the boundary. Later on, test cases have a much more uniform distribution, except that more test cases reside in narrow boundary areas but much less test cases reside in the areas adjacent to them.

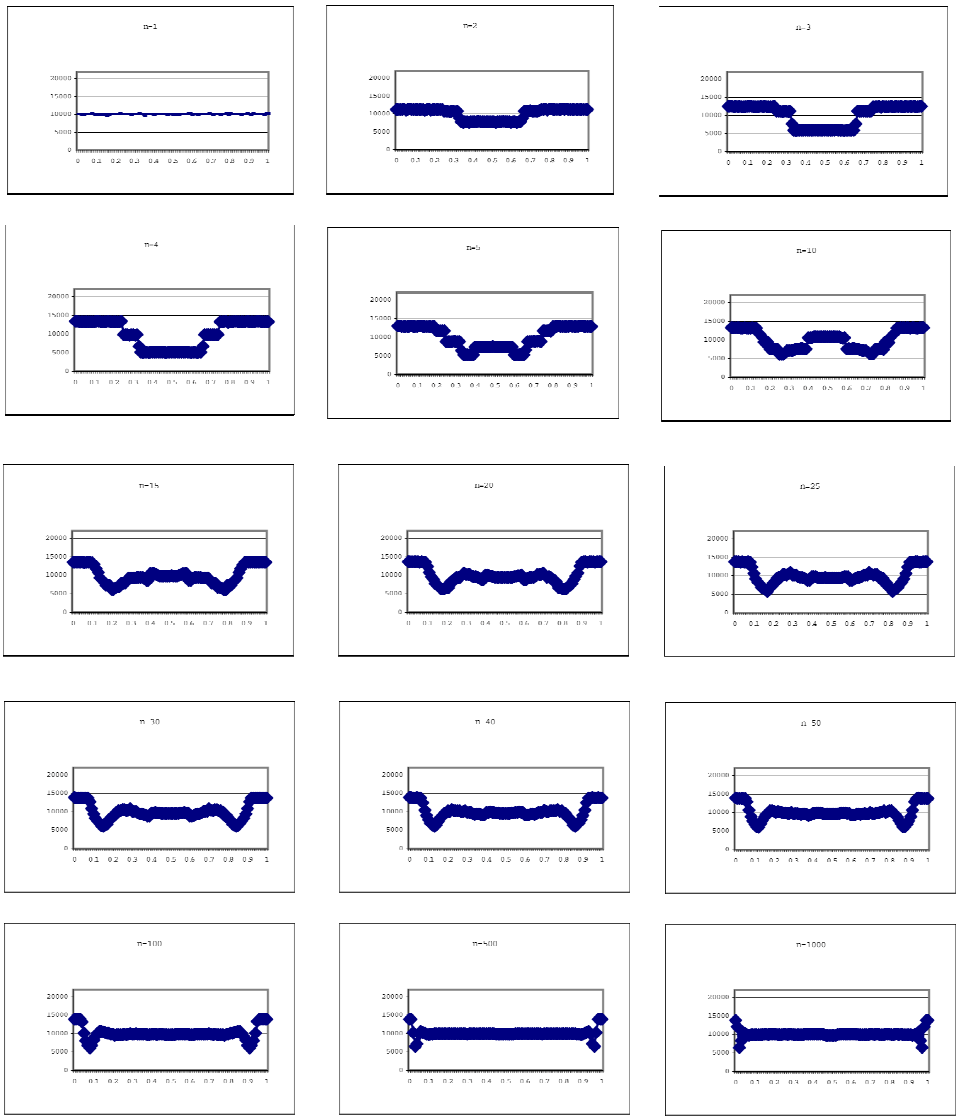


Fig. 7. Histograms of IP-ART test cases in one dimension. The *x*-axis represents the locations of test cases. The *y*-axis represents the number of test cases per bin of size 0.01.

Now we would like to propose an enhancement to the basic IP-ART algorithm. In the basic IP-ART algorithm, cells are classified according to their relative locations to the successful test cases. Cells containing a successful test case are named occupied cell. Cells that do not contain any test case but are surrounding neighbours of some occupied cells are classified as adjacent cells. All the other cells are candidate cells, from which a subsequent test case will be generated. Our enhancement introduces the concept of *virtual images* of successful test cases, which are constructed by shifting the input domain. Whenever classifying the cells in the input domain, we not only consider the locations of

the originals but also locations of their virtual images. In the virtual input domain, a cell containing a virtual image of a successful test case is called a *virtual occupied cell*. Cells around virtual occupied cells are *virtual adjacent cells*. If a virtual occupied cell is next to the original input domain, some of its virtual adjacent cells will locate within the original input domain.

An example of virtual adjacent cells in a 2-dimensional input domain is given in Fig. 8. The square with solid lines represents the original input domain, which is partitioned by a 4 × 4 grid. The square with dashed lines represents a virtual image of the input domain, which is virtually connected to the original input domain. The solid dot and hollow dot represent a successful test case and one of its virtual images, respectively. The cell containing a solid dot, (3, 2), is an occupied cell, and the cells around it are adjacent cells, namely, cell (2, 1), (3, 1), (2, 2), (2, 3) and (3, 3), which are in dark shading. The cell containing a hollow dot is a virtual occupied cell, and the cells around it (in light shading) are virtual adjacent cells. Note that three of the virtual adjacent cells, namely (0, 1), (0, 2), and (0, 3), are in the original input domain.

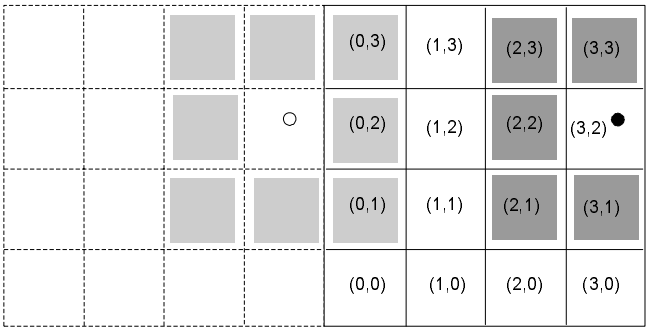


Fig. 8. Virtual adjacent cells in enhanced IP-ART.

Virtual adjacent cells within the original input domain can be identified efficiently. Suppose a k-dimensional input space is partitioned by a $p \times p$ grid. Let $(o_1, o_2, \ldots, o_k)$ be the coordinates of an occupied cell and $(a_1, a_2, \ldots, a_k)$ be the coordinates of one of its adjacent cells, where $1 \le k$, $0 \le a_i, o_i \le p - 1$. In the basic IP-ART algorithm, a_i ($i = 1, 2, \ldots, k$) can only have a value of o_i, $o_i - 1$ and $o_i + 1$. If $o_i - 1 < 0$, or $o_i + 1 > p - 1$, then the adjacent cells is outside the input domain. Here, the virtual adjacent cells within the input domain can be calculated by projecting the outside adjacent cells into the input domain. If $o_i - 1 < 0$, $a_i = o_i + p - 1$; If $o_i + 1 > p - 1$, $a_i = o_i - p + 1$. In the above example, since $p = 4$, $o_1 = 3$, $o_1 + 1 > 4 - 1$, we have $a_1 = o_1 - 4 + 1 = 0$. The adjacent cells outside the input domain are projected to cell (0, 1), (0, 2) and (0, 3).

In the basic IP-ART algorithm, occupied and adjacent cells cannot be used to generate test cases. Since no successful test cases exist outside the input domain, cells close to the boundary have a higher chance to be selected. In the enhanced IP-ART, virtual images of successful test cases outside the input domain are introduced. Some of the virtual adjacent cells locate within the original input domain, and they are also excluded for test cases generation. In this way, all cells, no matter whether they are close to the boundary of the input domain, will have a similar chance to be selected for test case generation.

Figs. 9 (a) and (b) compare the original and enhanced versions of IP-ART in a 2-dimensional input domain. The notations are the same as those of Fig. 8. In the original IP-ART shown in Fig. 9 (a), test cases can be generated in candidate cells, namely, cell (0, 0), (1, 0), (2, 0), (3, 0), (0, 1) and (1, 1). Among those candidate cells, only cell (1, 1) is not next to the boundary. Therefore, subsequent test cases have higher chance to be close to the boundary. If we also exclude the virtual adjacent cells as illustrated in Fig. 9 (b), only cell (1, 1) and (3, 0) are candidate cells. Cell (3, 0) is next to the boundary, while cell (1, 1) is not. Hence, the probability of subsequent test cases being close to the boundary is much less than that in Fig. 9 (a).

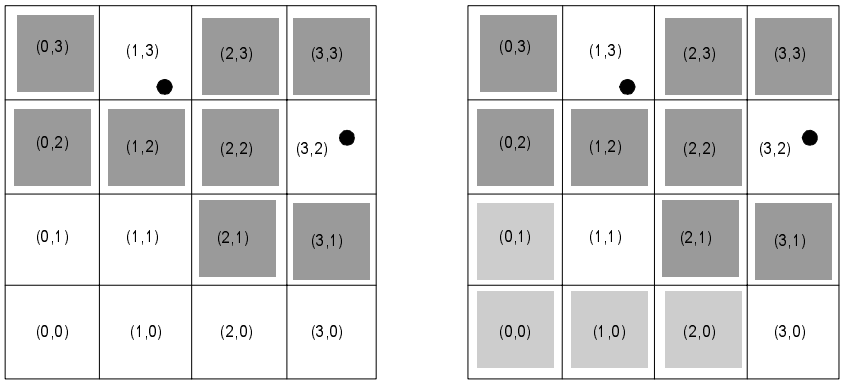


Fig. 9. Comparing test case selections between (a) the original and (b) the enhanced versions of IP-ART.

To confirm the analysis, simulations on spatial distributions were repeated. Fig. 10 shows the histograms for experiments with the enhanced IP-ART. The distributions have slight fluctuation when n is small due to the partitioning of the input domain. With the increase of n, the test cases become more uniformly distributed.

Then, fault-detection capabilities of the enhanced and original IP-ART were compared. Simulations were conducted with failure rates 0.01, 0.005, 0.002 and 0.001 for block failure patterns in 2-, 3-, and 4-dimensional input domains. The fault-detection capability of the enhanced IP-ART outperformed the original version for every combination of failure rate and input domain, as illustrated in Table 5.

5. DISCUSSION AND CONCLUSION

D-ART and R-ART are the first two methods of ART with excellent fault-detection capability. Both these methods attempt to achieve an even spread of random test cases by making them far apart from each other. The time complexity of both these methods is quadratic as much distance computation and comparison is involved.

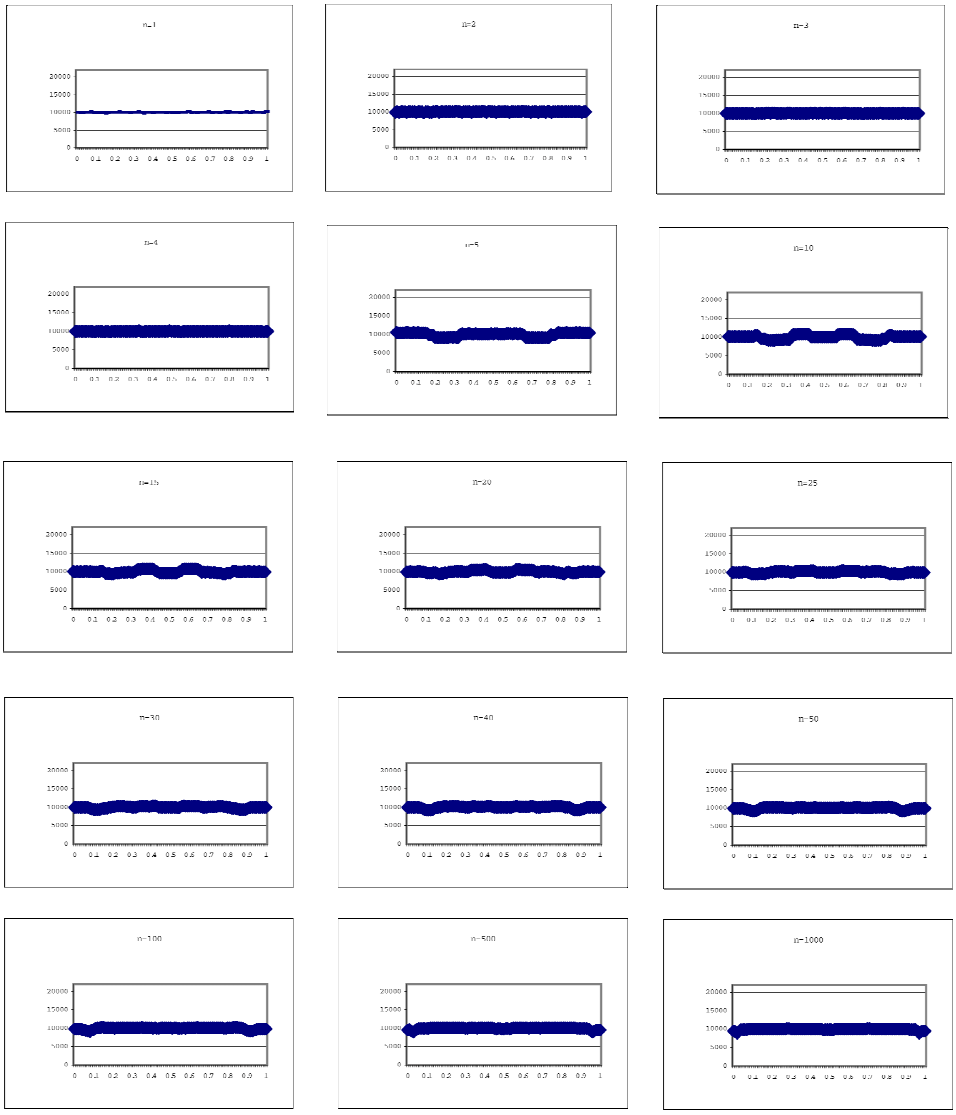


Fig. 10. Histograms of the enhanced IP-ART test cases in one dimension. The *x*-axis represents the location of test cases. The *y*-axis represents the number of test cases per bin of size 0.01.

Table 5. Comparison of relative F-measures of original and enhanced IP-ART in 2-, 3- and 4-dimensional input domains (under the block failure pattern).

Failure rate θ	F_{ART}/F_{RT}					
	Original IP-ART			Enhanced IP-ART		
	2D	3D	4D	2D	3D	4D
0.01	63%	82%	113%	60%	67%	74%
0.005	61%	80%	105%	60%	67%	74%
0.002	60%	76%	99%	58%	65%	74%
0.001	61%	74%	95%	58%	65%	73%

In fact, once the test cases are generated, their locations are fixed. It is not necessary to explore their locations all over again every time a new test case is to be generated. Based on this observation, this paper has presented a new ART method, ART through Iterative Partitioning (IP-ART). The input domain is divided into equally sized cells by a grid. The grid cells are categorized into occupied, adjacent, and candidate cells, according to their relative location to the successful test cases. In this way, IP-ART can easily identify candidate cells for test case generation, as they have a higher chance to make the next test case far away from all the successful test cases. If all candidate cells have been used up, the current partitioning scheme will be discarded, and a finer partitioning scheme will be applied. Compared with D-ART and R-ART, IP-ART has a comparable fault-detection capability, but the time complexity has been significantly reduced.

We have further investigated two key factors of the IP-ART method: (1) how to set the resolution for the initial partitioning scheme; and (2) the test case distribution of IP-ART. The analysis on the initial partitioning scheme has provided a guideline for setting the initial resolution. We suggest that overestimating the failure rate is a safe way in practice. In this way, the cost in repartitioning can be reduced and a lower F-measure can be achieved. With regard to the boundary effect in IP-ART, the notion of virtual adjacent cells is proposed. Simulation results show that enhanced IP-ART has alleviated the boundary effect and improved the fault-detection capability.

The approach used for the empirical evaluation reported in this paper was through simulations, where no actual "program under test" was involved and the simulated inputs only had numerical values. Further, the dimensionality of an input domain has been interpreted as the number of input variables. These simulation settings are, however, not a limitation of the applicability of ART to real-world programs that involve non-numerical inputs or have many input variables. For programs involving non-numerical inputs, a large-scale empirical study based on the approaches proposed by Merkel [26] and Kuo [22] is underway, and encouraging preliminary results have been obtained. Ciupa *et al*. [13] applied ART to the testing of object-oriented software by developing a notion of distance between any two composite objects. Only three dimensions are involved in the computation, namely the values of the objects, the dynamic types of the objects, and (recursively) the primitive values of the attributes or the objects referred to by the attributes. Here the dimensionality (three) is quite low. Readers are also referred to [12] for an in-depth discussion on the applications of ART to real-world programs.

In this paper we focused on continuous input type. When the input type is discrete, treatment should be adjusted. This is because, for discrete input types such as integers, at certain stage it will become impossible to further repartition the input domain using a finer partitioning scheme. In this situation, test cases should be selected from adjacent cells because otherwise certain inputs will never be considered for test case generation. The method presented in this paper can be directly applied to test real-world programs with numerical inputs because the input domains of these programs are similar to the ones used in our simulation studies. Our future research will involve the development of an IP-ART framework that accepts programs with different types of inputs (including both numerical and non-numerical inputs, and both continuous and discrete inputs). The notion of dimensionality will be extended so that only a few dimensions will be involved in the computation regardless of the number of input variables.

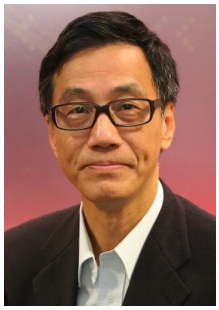
ACKNOWLEDGMENTS

The authors are grateful to Robert Merkel for invaluable comments on the draft of this paper.

REFERENCES

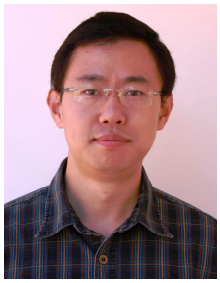
1. ACM, *Collected Algorithms from ACM*, Association for Computing Machinery, 1980.
2. D. L. Bird and C. U. Munoz, "Automatic generation of random self-checking test cases," *IBM Systems Journal*, Vol. 22, 1983, pp. 229-245.
3. K. Y. Cai, B. Gu, H. Hu, and Y. C. Li, "Adaptive software testing with fixed-memory feedback," *Journal of Systems and Software*, Vol. 80, 2007, pp. 1328-1348.
4. K. P. Chan, T. Y. Chen, and D. Towey, "Normalized restricted random testing," in *Proceedings of the 8th Ada-Europe International Conference on Reliable Software Technologies*, LNCS 2655, 2003, pp. 368-381.
5. T. Y. Chen, T. H. Tse, and Y. T. Yu, "Proportional sampling strategy: A compendium and some insights," *Journal of Systems and Software*, Vol. 58, 2001, pp. 65-81.
6. T. Y. Chen, G. Eddy, R. Merkel, and P. K. Wong, "Adaptive random testing through dynamic partitioning," in *Proceedings of the 4th International Conference on Quality Software*, 2004, pp. 79-86.
7. T. Y. Chen, F. C. Kuo, R. Merkel, and S. P. Ng, "Mirror adaptive random testing," *Information and Software Technology*, Vol. 46, 2004, pp. 1001-1010.
8. T. Y. Chen, H. Leung, and I. K. Mak, "Adaptive random testing," in *Proceedings of the 9th Asian Computing Science Conference*, LNCS 3321, 2004, pp. 320-329.
9. T. Y. Chen, D. H. Huang, and Z. Q. Zhou, "Adaptive random testing through iterative partitioning," in *Proceedings of the 11th Ada-Europe International Conference on Reliable Software Technologies*, LNCS 4006, 2006, pp. 155-166.
10. T. Y. Chen, F. C. Kuo, and C. A. Sun, "Impact of the compactness of failure regions on the performance of adaptive random testing," *Journal of Software*, Vol. 17, 2006, pp. 2438-2449.
11. T. Y. Chen and R. Merkel, "An upper bound on software testing effectiveness," *ACM Transactions on Software Engineering and Methodology*, Vol. 17, 2008, pp. 16:1-16:27.
12. T. Y. Chen, F. C. Kuo, R. G. Merkel, and T. H. Tse, "Adaptive random testing: The ART of test case diversity," *Journal of Systems and Software*, Vol. 83, 2010, pp. 60-66.
13. I. Ciupa, A. Leitner, M. Oriol, and B. Meyer, "ARTOO: Adaptive random testing for object-oriented software," in *Proceedings of the 30th International Conference on Software Engineering*, 2008, pp. 71-80.
14. R. Cobb and H. D. Mills, "Engineering software under statistical quality control," *IEEE Software*, Vol. 7, 1990, pp. 45-54.
15. T. Dabóczi, 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, pp. 441-445.
16. H. Dörrie, *100 Great Problems of Elementary Mathematics: Their History and Solu-*

tion, Dover Publications, Inc., New York, 1965.
17. B. S. Everitt, *The Cambridge Dictionary of Statistics*, Cambridge University Press, 1998.
18. 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*, 2000, pp. 59-68.
19. B. Hailpern and P. Santhanam, "Software debugging, testing, and verification," *IBM Systems Journal*, Vol. 41, 2002, pp. 4-12.
20. R. Hamlet and R. Taylor, "Partition testing does not inspire confidence," *IEEE Transactions on Software Engineering*, Vol. 16, 1990, pp. 1402-1411.
21. R. Hamlet, "Random testing," J. Marciniak, ed., *Encyclopedia of Software Engineering*, John Wiley & Sons, 2002, pp. 970-978,
22. F. C. Kuo, "On adaptive random testing," Ph.D. Thesis, Faculty of Information and Communication Technologies, Swinburn University of Technology, Australia, 2006.
23. P. S. Loo and W. K. Tsai, "Random testing revisited," *Information and Software Technology*, Vol. 30, 1988, pp. 402-417.
24. I. K. Mak, "On the effectiveness of random testing," Master's Thesis, Department of Computer Science, The University of Melbourne, Australia, 1997.
25. Y. K. Malaiya, "Antirandom testing: Getting the most out of black-box testing," in *Proceedings of the 6th International Symposium on Software Reliability Engineering*, 1995, pp. 86-95.
26. R. Merkel, "Analysis and enhancements of adaptive random testing," Ph.D. Thesis, Faculty of Information and Communication Technologies, Swinburn University of Technology, Australia, 2005.
27. B. P. Miller, L. Fredriksen, and B. So, "An empirical study of the reliability of Unix utilities," *Communications of the ACM*, Vol. 33, 1990, pp. 32-44.
28. 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 No. CSTR-1995-1268, Computer Sciences Department, University of Wisconsin, 1995.
29. E. Miller, "Website testing," Software Research, Inc., 2005, http://www.soft.com/eValid/Technology/White.Papers/website.testing.html.
30. G. J. Myers, *Software Reliability: Principles and Practices*, Wiley, New York, 1976.
31. N. Nyman, "In defense of monkey testing: Random testing can find bugs, even in well engineered software," Microsoft Corporation, http://www.softtest.org/sigs/material/nnyman2.htm.
32. D. Slutz, "Massive stochastic testing of SQL," in *Proceedings of the 24th International Conference on Very Large Data Bases*, 1998, pp. 618-622.
33. D. Towey, "Studies of different variations of adaptive random testing," Ph.D. Thesis, Department of Computer Science, The University of Hong Kong, Hong Kong, 2006.
34. 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*, 2003, pp. 20-24.



Tsong Yueh Chen (陳宗岳) received his B.S. and M.Phil. degrees from The University of Hong Kong; M.S. degree and DIC from the Imperial College of London University; and Ph.D. degree from The University of Melbourne. He is currently the Chair Professor of Software Engineering and the Director of the Centre for Software Analysis and Testing, Swinburne University of Technology, Australia. His research interests include software testing, debugging, software maintenance, and software design.

De Hao Huang (黃德浩) received his B.Eng. and M.Sc. degrees from East China Normal University, and Ph.D. degree from Swinburne University of Technology. He is currently the Quality Assurance Manger at Information Technology Services, Swinburne University of Technology. His research interest is on software testing processes and automation.



Zhi Quan Zhou (周智泉) received his B.Sc. degree in Computer Science from Peking University, China, and Ph.D. degree in Software Engineering from The University of Hong Kong. He is currently a Senior Lecturer in Software Engineering at the University of Wollongong, Australia. His research interests include software testing, debugging, software maintenance, symbolic execution, and quality assessment of web search engines.