Classification and Improvements of Adaptive Random Testing Methods

Dehao Huang

June 2007
To Hongyan, Alexandra and my parents
Acknowledgements

First and foremost, I wish to thank my supervisor Professor T. Y. Chen for his intellectual guidance and persistent support throughout my PhD study. He has showed me how to find simple solutions for apparently complicated problems, how to present concisely and clearly, and how to collaborate with others. To me, he is not only a supervisor, but also a collaborator, mentor, guide and role model. I am deeply encouraged by his enthusiasm for work, family, friends and students.

I also wish to express my gratitude to all members in Software Testing Group. This thesis benefited a lot from discussions with and comments from them. My research at Swinburne was made more enjoyable because of these great colleagues.

Finally, I am greatly indebted to Hongyan. She is alway my supporter no matter where I am and what I am doing.
Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma, except where due reference is made in the text of the thesis. To the best of my knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Signed:

Dated:
Abstract

Adaptive Random Testing (ART) is an effective enhancement of RT, which is based on the observations that failure-causing inputs are normally clustered in one or a few contiguous regions in the input domain. Hence, it has been proposed that test case generation should refer to the locations of successful test cases (those that have been executed but not revealed a failure) to achieve an even spread of test cases throughout the input domain. Recently, several ART methods (algorithms) have been proposed, which have their own advantages and disadvantages. Since no proper classification of existing ART methods in previous studies, the relationships among ART methods were unclear. These are the main focuses of this thesis.

Firstly, this thesis examines existing ART methods and establishes a taxonomy for them, which is a framework for further studies of ART methods. Existing ART methods are classified into 2 categories based on different interpretations of the basis of ART. They are “far apart” ART methods and Proportional Sampling Strategy (PSS) ART methods. Based on this classification, characteristics for each category of ART methods can be summarized; improvements can be proposed for each category of ART methods rather than individual methods; new ART methods can also be inspired.

Secondly, this thesis focuses on the shortcomings of these ART methods. For “far apart” ART methods, two major shortcomings are extensive computational overhead and boundary effect (test cases prefer to be near the boundary of the input domain). For PSS ART methods, a common problem is that test cases still have chances to be clustered together, which has adverse impact on the fault-detection capability. This thesis investigates the reasons behind and proposes improvements of the original methods.

Finally, with regard to the practicality of ART methods in high dimensional input domains, this thesis proposes a new ART method, ART by balancing, based on a distinctive interpretation of an even spread of test cases, which requires that the centroid of test cases
in a partition should be close to the centroid of that partition. This method has a good fault-detection capability, especially in high dimensional input domains.
## Contents

1 Introduction ........................................... 1  
   1.1 Background of Software Testing ................. 1  
   1.2 Random Testing .................................... 3  
   1.3 Focuses of the thesis ......................... 4  
   1.4 Organization of the Thesis .................... 5  

2 Literature Review ........................................... 7  
   2.1 Terms and Concepts in Software Testing ........... 7  
      2.1.1 Error, Fault and Failure .................. 7  
      2.1.2 Software Testing .......................... 7  
      2.1.3 Problems in Software Testing .............. 8  
      2.1.4 Black Box Testing and White Box Testing . 9  
   2.2 Partition Testing and Random Testing .......... 10  
   2.3 Successful Test Case ......................... 11  
   2.4 Observation-based Testing ........................ 11  
   2.5 Failure Patterns .................................. 13  
   2.6 Adaptive Random Testing (ART) .................. 15  
      2.6.1 Intuition of ART ......................... 15  
      2.6.2 Fault-detection Capability Metrics ......... 16  
      2.6.3 Experimental Sample Size .................. 17  
      2.6.4 Related Works ............................... 18  

3 A Taxonomy of Adaptive Random Testing Methods ...... 20  
   3.1 Introduction ........................................ 20  
   3.2 Taxonomy ............................................ 20  
   3.3 “Far Apart” ART Methods ...................... 22  
   3.4 PSS ART Methods ................................... 24  
   3.5 Summary ............................................. 28
4 Adaptive Random Testing through Iterative Partitioning

4.1 Introduction

4.2 Review and Analysis of Related Works

4.2.1 D-ART and R-ART

4.2.2 Mirror Adaptive Random Testing

4.3 The Basic Algorithm of ART through Iterative Partitioning (IP-ART)

4.3.1 Overview

4.3.2 The Grid Coordinates Used in IP-ART

4.3.3 Categorization of Grid Cells

4.3.4 The Algorithm

4.3.5 Time Complexity of the IP-ART Method

4.3.6 Simulation Studies

4.4 Further Studies on IP-ART

4.4.1 IP-ART with a Finer Initial Grid

4.4.2 Weighting the Grid Cells

4.5 Discussion and Summary

5 Tackling Boundary Effect in “Far Apart” ART Methods

5.1 Introduction

5.2 Boundary Effect

5.3 Virtual Images of Successful Test Cases

5.4 The Enhanced Method

5.4.1 Effective Image in D-ART and R-ART

5.4.2 Enhancement of D-ART and R-ART

5.4.3 Enhancement of IP-ART

5.5 Simulation Studies

5.6 Summary

6 Adaptive Random Testing by Localization

6.1 Introduction

6.2 Unfavorable Scenarios in DP-ART

6.3 Basic Idea of Localization

6.4 RP-ART by Localization

6.4.1 Algorithm Description

6.4.2 Simulation Studies

6.5 B-ART by Localization

6.5.1 Algorithm Description

6.5.2 Simulation Studies

6.6 Summary
7 Enhanced Lattice-based Adaptive Random Testing

7.1 Introduction ............................................. 84
7.2 A Review of L-ART ........................................ 85
  7.2.1 The Original Algorithm .............................. 85
  7.2.2 Test Case Spatial Distribution of L-ART .......... 87
7.3 The Enhanced Method .................................... 90
  7.3.1 Unfavorable Scenario in Original L-ART with Large Shaken Factors 90
  7.3.2 The Enhanced Algorithm .............................. 92
  7.3.3 Time Complexity Analysis ............................ 96
7.4 Simulation Studies ...................................... 97
7.5 Summary ................................................. 100

8 Adaptive Random Testing by Balancing

8.1 Introduction ............................................. 103
8.2 ART by Balancing ....................................... 104
  8.2.1 Preliminary ........................................ 104
  8.2.2 Intuition ........................................... 104
  8.2.3 Algorithm Description ............................... 106
8.3 Simulations .............................................. 107
  8.3.1 Fault-detection Capability ........................... 107
  8.3.2 Spatial Distribution ................................ 111
8.4 Discussion and Summary ............................... 111

9 Conclusion ................................................. 115

9.1 Summary ............................................... 115
9.2 Future Works .......................................... 118

Bibliography ................................................. 120

List of Publications ........................................ 131
## List of Tables

1.1 Recent aerospace losses due to software failures ........................................ 2

4.1 Relationship between the target exclusion ratio and the average actual exclusion ratio in R-ART, adopted from [92] ...................................................... 32

4.2 Relative F-measures of IP-ART in a 2-dimensional input domain under 3 failure patterns ................................................................. 41

4.3 Comparison of relative F-measures of IP-ART and other ART methods on a 2-dimensional input domain (under the block failure pattern) .............. 42

4.4 Recorded CPU time of different ART methods (with $\theta=0.001$ and under the block failure pattern, for a run of 5000 trials each method) ................. 42

4.5 Relative F-measures of IP-ART for different threshold values in a 2-dimensional input domain ................................................................. 49

5.1 F-measures of differently located failure regions for D-ART, R-ART and IP-ART under block failure pattern on a 2-dimensional input domain ($\theta = 0.01$) ......................................................... 53

5.2 F-measures of differently located failure regions for enhanced D-ART, R-ART and IP-ART under block failure pattern on a 2-dimensional input domain ($\theta = 0.01$) ......................................................... 67

5.3 F-measures of original D-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern) ......................................................... 67

5.4 F-measures of enhanced D-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern) ......................................................... 68

5.5 F-measures of original R-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern) ......................................................... 68

5.6 F-measures of enhanced R-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern) ......................................................... 68

5.7 F-measures of original IP-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern) ......................................................... 68

5.8 F-measures of enhanced IP-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern) ......................................................... 69
6.1 Comparison of F-measures of RP-ART by localization and other ART methods on a 2-dimensional input domain ($\theta=0.001$) ........................................ 78

6.2 F-measures of differently located failure regions for RP-ART with D-ART, or with R-ART on a 2-dimensional input domain (under the block failure pattern, $\theta = 0.01$) ........................................ 79

6.3 Comparison of F-measures of B-ART by localization and other ART methods on a 2-dimensional input domain ($\theta=0.001$) ........................................ 82

7.1 Relative F-measures of original L-ART (under the block failure pattern) . . 89

7.2 Comparison of relative F-measures of original and enhanced L-ART (under the strip failure pattern) ........................................ 99

7.3 Comparison of relative F-measures of original and enhanced L-ART (under the point failure pattern) ........................................ 99

8.1 F-measures of ART by balancing in a 2-dimensional input domain under 3 failure patterns ........................................ 109

8.2 F-measures of D-ART in 2-, 3-, and 4-dimensional input domains (under the block failure pattern) ........................................ 110

8.3 F-measures of B-ART in 2-, 3-, and 4-dimensional input domains (under the block failure pattern) ........................................ 110

8.4 F-measures of ART by balancing in 2-, 3-, and 4-dimensional input domains (under the block failure pattern) ........................................ 110

8.5 Width of a hypercube failure pattern in a k-dimensional hypercube input domain with size of 1 ........................................ 114
## List of Figures

2.1 Examples of the three types of failure patterns .................................. 13  
2.2 Code fragments producing examples of the three types of failure patterns .. 14  
2.3 An example that illustrates the intuition of ART ................................. 16  
3.1 An example that illustrates the D-ART method .................................. 23  
3.2 An example that illustrates the R-ART method .................................. 24  
3.3 An example that illustrates the RP-ART method ................................. 25  
3.4 An example that illustrates the B-ART method .................................. 26  
4.1 Partitioning the input domain and classifying grid cells ....................... 35  
4.2 Grid coordinates and the mapping of test cases into grid cells ............... 36  
4.3 An example that illustrates the IP-ART method .................................. 39  
4.4 Differences between the final \( q \) and initial \( p \) partitioning schemes in IP-ART .......................................................... 46  
4.5 Relative F-measures of IP-ART on different initial partitioning schemes .... 46  
4.6 Weighted grid cells in a 2-dimensional input domain ............................ 48  
5.1 Histogram of D-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. ..................................................................................... 54  
5.2 Histogram 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. ..................................................................................... 55  
5.3 Virtual image construction process in a 2-dimensional square input domain  
5.4 Virtual images of a successful test case in a 2-dimensional square input  
domain ..................................................................................................... 59  
5.5 Comparing test case selections between (a) the original and (b) the enhanced  
versions of D-ART .................................................................................. 60  
5.6 Comparing test case selections between (a) the original and (b–d) the en-  
hanced versions of R-ART ........................................................................ 61  
5.7 Virtual adjacent cells in enhanced IP-ART .......................................... 63
5.8 Comparing test case selections between (a) the original and (b) the enhanced versions of IP-ART ................................................. 64
5.9 Histograms of the enhanced D-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. ................................................. 65
5.10 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. ................................................. 66
6.1 An unfavorable scenario in RP-ART ........................................... 71
6.2 An unfavorable scenario in B-ART ........................................... 71
6.3 An example that illustrates RP-ART by localization with D-ART ........ 75
6.4 An example that illustrates RP-ART by localization with R-ART ........ 75
6.5 Comparison of the F-measures of RT and RP-ART by localization with D-ART with different numbers of candidates on a 2-dimensional input domain (under the block failure pattern, θ = 0.001) ................................................. 76
6.6 Test case generation patterns for RP-ART by localization with D-ART under large number of candidates (the number represents the test case generation sequence) ................................................. 77
6.7 Comparison of the F-measures of RT and RP-ART by localization with R-ART with different exclusion radius ratios on a 2-dimensional input domain (under the block failure pattern, θ = 0.001) ................................................. 78
6.8 An example that illustrates B-ART by localization with D-ART ........ 80
6.9 An example that illustrates B-ART by localization with R-ART ........ 81
6.10 Comparison of the F-measures of RT and B-ART by localization with D-ART with different numbers of candidates on a 2-dimensional input domain (under the block failure pattern, θ = 0.001) ................................................. 82
6.11 Comparison of the F-measures of RT and B-ART by localization with R-ART with different exclusion ratios on a 2-dimensional input domain (under the block failure pattern, θ = 0.001) ................................................. 82
7.1 An example that illustrates the original L-ART ................................. 88
7.2 Histogram of original L-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. ................................................. 91
7.3 An unfavorable scenario in L-ART ........................................... 92
7.4 Partitioning the input domain on enhanced L-ART (f = 0.4, m = 4) .... 94
7.5 Comparison of the relative F-measure of original and enhanced L-ART under different failure rates θ and shaken factors f (under the block failure pattern). The x-axis represents f. The y-axis represents the relative F-measure 98
7.6 Comparison of histogram of original L-ART ($f = 0.1$) and enhanced L-ART ($f = 0.6$) 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.

8.1 Basic idea of ART by balancing .................................................. 105

8.2 Basic idea of ART by balancing (continued) .................................. 105

8.3 Histograms of ART by balancing 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. ................................................................. 112

8.4 Histograms of D-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. ................................................................. 113
# List of Algorithms

4.1 Original D-ART algorithm .............................................. 30  
4.2 Original R-ART algorithm .............................................. 31  
4.3 The basic IP-ART algorithm ............................................ 38  
6.1 RP-ART by localization algorithm ................................. 73  
7.1 Original L-ART algorithm .............................................. 86  
7.2 Enhanced L-ART algorithm ............................................. 95  
8.1 ART by balancing algorithm .......................................... 108
Chapter 1

Introduction

1.1 Background of Software Testing

Software has become an essential part of the modern society. It has been deeply embedded in almost every business, from huge public infrastructure such as an energy management system, to a small consumer appliance, such as iPod. As described by a GM engineer, “Many of GM’s products have become reliant on software to the point that they could not be sold, used, or serviced without it [56]. The annual sales of software worldwide have exceeded $200 billion [87]. The world economic becomes highly dependent on the effective operation of reliable software. However, as conceived by Myers [75] 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.

Recently, a report released by the Department of Commerce’s National Institute of Standards and Technology (NIST) pointed out that software quality is still so poor that software failures cost U.S. economy $59.5 billion annually, or about 0.6 percent of the gross domestic product [77]. Table 1.1 lists the losses due to software failures in aerospace industry in recent several years. As illustrated, these catastrophic software failures not only result in loss of money and data, but also loss of lives. In addition, the reputation and future business of the company are lost.

Software development companies have put huge amounts of money to improve software
<table>
<thead>
<tr>
<th>Events</th>
<th>Year</th>
<th>Aggregate cost</th>
<th>Loss of life</th>
<th>Loss of data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airbus 320</td>
<td>1993</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ariane 5 Galileo Poseidon Flight 965</td>
<td>1996</td>
<td>$640 million</td>
<td>160</td>
<td>Yes</td>
</tr>
<tr>
<td>Lewis Pathfinder USAF Step</td>
<td>1997</td>
<td>$116.8 million</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Zenit 2 Delta 3 Near</td>
<td>1998</td>
<td>$255 million</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>DS-1 Orion 3 Galileo Titan 4B</td>
<td>1999</td>
<td>$1.6 billion</td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>


Table 1.1: Recent aerospace losses due to software failures

quality. As pointed out by Hailpern and Santhanam [47], “In a typical commercial development organization, the cost of providing this assurance [that the program will perform satisfactorily in terms of its functional and nonfunctional specifications within the expected deployment environments] via appropriate debugging, testing, and verification activities can easily range from 50 to 75 percent of the total development cost.” However, many bugs are still found by end users. Maintenance team becomes an indispensable department in every software companies to correct errors in delivered products. Hence, software quality has become the key success factor in software industry. This thesis focuses on Software Quality Assurance (SQA).

Generally, the approaches of SQA are divided into two categories: preventive actions and corrective actions [3]. Preventive actions monitor and improve the software constructing process, making sure that any agreed-upon standards and practice guidelines are followed. While corrective actions evaluate the software quality once the software has been constructed. If the software quality is poorer than expected, improvements to the software are proposed. Although preventive actions can significantly reduce the number of defects in software products, they cannot work alone. As indicated by Lauesen and Younessi’s empirical study, 45 percent of defects cannot be detected by preventive actions alone [59]. Moreover, human beings always make mistakes in each transition process of software development life cycle [75]. The statistics also show that a competent programmer still have one to three bugs per hundred statements [94]. Hence, corrective actions are indispensable parts in SQA. This thesis focuses on corrective actions.

According to whether software is being executed, corrective actions can be further classified into two categories: static analysis and dynamic analysis. Static analysis does not
require the execution of software, such as walkthrough and inspection [76], model checking [35] and program proving [54]. Software testing, as a form of verification, is a dynamic method to improve software quality. Software testing has several advantages over static analysis [51], such as relative ease of performing, conformance to the real operation environment, and automation of much the process. Moreover, the past decades have shown that the use of program proving to real life applications has been very limited due to the difficulties of proofs and automation [47]. Software testing, therefore, becomes the primary means adopted by practitioners [3, 47]. This thesis focuses on software testing.

Today software industry suffers inadequate infrastructure of software testing. As the report [77] indicated, more than a third of the costs due to software failures, or an estimated $22.2 billion, could be eliminated by an improved testing infrastructure. Why is software testing so hard and how did so many bugs escape from testing? Whittaker [98] attributed it to insufficient testing and classified the insufficient testing into the following four aspects:

- Some statements were untested.
- Some statements execution sequences were untested.
- Some combination of input values were untested.
- Some user’s operating environments were untested.

Hence, developing efficient test case selection strategies has become the main task in software testing research. This thesis focuses on test case selection strategies.

1.2 Random Testing

Among the test case selection strategies, Random Testing (RT) is regarded as a simple but useful method [49, 62]. RT has several advantages:

- It avoids the overhead of specification-based or program-based analysis of the input domain, which can be very expensive in many situations.
- The test case generation process of RT is cost effective and can be fully automated.
• When the formal specification or the program code is not available, RT can often be a practical solution to generate a large number of inputs to cover cases that are possible to occur in real life but may often be overlooked by human testers [37, 89].

• RT is a core test case generation technique embedded in many testing methods. For example, in path coverage testing, it is very expensive to solve path constraints. An alternative way is to generate test case randomly and check the path coverage.

• RT can also be used as a statistical quantitative estimation of the program reliability.

In fact, RT has been successfully applied in many real life applications [36, 37, 44, 69–71, 78, 89, 101]. In 1990, for example, Miller developed the FUZZ system that generated random data streams to test programs in several versions of the UNIX system [69, 70]. 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 software corporations and implemented in their software testing tools (for example, [4]).

1.3 Focuses of the thesis

Adaptive Random Testing (ART) has been proposed to improve the fault-detection capability of RT [26, 63]. ART is based on the observation that failure-causing inputs are clustered together in one or a few contiguous regions [28]. 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. Simulations and empirical studies of real life programs [26] have shown that ART has significantly enhanced RT in terms of using fewer test cases to detect the first failure.

Based on the same intuition, several ART methods (algorithms) ¹ have been proposed. Due to the nature of the principles adopted, these methods have their own advantages and disadvantages. Some of them require extensive computations to ensure the generation of

¹In this thesis, the terms “ART method” and “ART algorithm” are used interchangeably.
evenly spread test cases, and hence incur high overhead. Some of them have reduced computational overhead, but the fault-detection capabilities are also compromised. Some of them have side-effects, such as test cases preferring to be close to the boundary of the input domain, which have an adverse impact on fault-detection capabilities, especially in high dimensional input domains. Since there was no proper way to classify existing ART methods in previous studies, the relationships among these methods were unclear and it was difficult to compare and improve these ART methods. This thesis establishes a taxonomy to classify the ART methods and proposes improvements for each category of these ART methods.

1.4 Organization of the Thesis

This thesis is organized as follows. Chapter 2 firstly clarifies some basic terms and concepts in software testing literature. Then, the idea of ART and related works are elaborated. Chapter 3 is the core part of this thesis, which establishes the taxonomy of ART methods. From this classification, advantages and disadvantages for each category of the method are summarized. One category of the ART methods is based on the notion of “far apart”. For this category of ART methods, the problems are extensive computational overhead and the tendency to generate test cases near the boundary of the input domain. With regard to these two problems, Chapter 4 and Chapter 5 improve the methods by introducing the innovative notions of “iterative partitioning” and “virtual test cases”, respectively. Another category of ART methods is based on the notion of Proportional Sampling Strategy (PSS). This category of methods has reduced computational overhead. However, test cases generated by this category of methods still have chances to be clustered together, which compromises the fault-detection capability. With regard to the two types of partitioning strategies (dynamic and static) in PSS ART methods, improvements are proposed in Chapter 6 and Chapter 7, respectively. The taxonomy of ART methods also facilitates the development of a new ART method, ART through balancing, which based on a distinctive notion to achieve an even spread of test cases. This new method has an improved fault-detection capability in high dimensional input domains. Chapter 8 presents the idea and the simulation results of this method. Finally, the conclusion is given in Chapter 9.
Note that, the Software Under Test (SUT) mentioned in this thesis is limited in numeric problem domain. The extension to non-numeric problem domain is one of our future works. Furthermore, this thesis assumes that all input parameters of the SUT are bounded.
Chapter 2

Literature Review

This chapter explains some basic terms and concepts in software testing and elaborates the idea of Adaptive Random Testing (ART) and related works.

2.1 Terms and Concepts in Software Testing

2.1.1 Error, Fault and Failure

According to the definition in [90], error refers to human being’s mental mistake, which produces an incorrect result; fault is an incorrect definition or process in the software; failure is an observable violation against the specifications. Here, error emphasizes human mistake, either programmer’s mistake or designer’s mistake. Fault refers to the cause of a malfunction. Failure focuses on the unexpected software behavior, which can be observed after the software is executed. In summary, “When developing software, people make errors, these become faults in the software which then manifest themselves as failures when the software is run.” [85]

2.1.2 Software Testing

Myers defined software testing in his seminal textbook on software testing [74] as: “Testing is the process of executing a program with the intention of finding errors.” The definition emphasizes that testing can only show the presence of errors, but not the
absence of errors. If testing is defined as “the process of confirming that a program is correct”, it will become an impossible mission, because of the reliable test set problem, which will be explained later.

Software testing includes following activities: [75] (1) establishing test objectives; (2) designing test cases; (3) generating test cases; (4) executing test cases; (5) examining results. Among these software testing activities, there are two fundamental problems in designing test cases and examining the results, namely the oracle problem [95] and the reliable test set problem [55].

2.1.3 Problems in Software Testing

In software testing, there is an oracle assumption [95]. Let \( p(x) \) be a program implementing function \( f(x) \) on domain \( D \). To test this program, the tester runs \( p \) on \( T \), a set of test cases: \( T = \{t_1, t_2, \ldots, t_n\} \subset D \), where \( n \geq 1 \). The outputs \( p(t_1), p(t_2), \ldots, p(t_n) \) are then checked against the expected results \( f(t_1), f(t_2), \ldots, f(t_n) \), respectively. If \( p(t_i) = f(t_i) \), then \( t_i \) is called a successful test case; if \( p(t_i) \neq f(t_i) \), then \( t_i \) is called a failure-causing test case. The mechanism by which the tester can decide whether \( p(t_i) = f(t_i) \) for \( i = \{1, 2, \ldots, n\} \) is known as the oracle. In the literature of software testing, it is usually assumed that the oracle is available and, hence, the mainstream of the researches has concentrated on the development of test case selection strategies, the approaches for selecting \( t_i \)'s that have a higher chance of revealing a failure [58, 79, 80, 91, 97]. In some situations, however, the oracle is not available or too expensive to apply. This is known as the oracle problem [95]. For example, the outputs of programs conducting complicated computations, such as numerical integrations, are difficult to verify. In multiple precision arithmetic, the operands involved are very large numbers and, hence, the results are very expensive to check. When testing a compiler, it is not easy to verify whether the generated object code is equivalent to the source code. When testing object-oriented programs, it is very difficult to decide whether two objects are equivalent. Other examples include testing programs performing simulations, conducting combinatorial calculations, drawing graphs to the monitor, etc. On the other hand, when the oracle is available, if it is a human tester,
the manual predictions and comparisons of the test results are often time consuming and
error prone [48, 65]. As a matter of fact, the oracle problem has been “one of the most
difficult tasks in software testing” [65].

To alleviate the oracle problem, the *metamorphic testing* (MT) approach was proposed
by Chen et al. [13] and further developed [20, 22, 29, 30, 45, 46, 99]. MT is both a technique
for automated test case generation and a mechanism for automated output verification. This
technique is based on existing successful test cases (those that have been executed but *not*
revealed a failure), and expected program properties. These properties are known as meta-
morphic relations (MRs), which are necessary relations among the inputs and outputs of
multiple executions of the target program. Theoretically speaking, MRs can be identified in
all application areas [93]. Hence, it has a wide range of applications [102]. Normally more
than one MR can be identified for a given problem. It would be ideal if all MRs could be
used. However, since resources are always limited, testers need practical guidance to know
which MRs should be given priority for use in testing. Hence, selection of efficient MRs
that have higher chances of detecting failures is a key focus in MT research [20, 22].

Another problem in software testing is the reliable test set problem [55]. “A set of
test data $T$ for a program $P$ is *reliable* if it reveals that $P$ contains an error whenever $P$ is
incorrect.” It has been proved that a testing strategy generating such a reliable test set for
all programs cannot be constructed except for exhaustive testing. In other words, the use of
test cases cannot guarantee program correctness on untested inputs [3, 55]. Since exhaustive
testing is impossible even for trivial programs, the key issue of testing becomes “what subset
of all possible test cases has the highest probability of detecting the most errors?” [74].

### 2.1.4 Black Box Testing and White Box Testing

From the perspective of whether considering the software’s internal structure, test case se-
lection techniques are classified into two types: black box testing and white box testing tech-
niques [3]. Black box testing is also known as functional testing, which selects test cases
from specifications, without taking the program’s internal structure into account. Equiva-
lence partitioning and boundary value analysis are two typical examples of black box testing
methods [3]. White box testing is also known as structural testing, which selects test cases by examining program’s internal structure. Typical white box testing techniques include control flow testing [53] and data flow testing [83].

2.2 Partition Testing and Random Testing

One of the main techniques towards selection of test cases is *partition testing*. Essentially, partition testing has two steps. Firstly, the input domain, $D$, is divided into disjoint subdomains, $(D_1, D_2, \ldots, D_k)$ ($k \geq 1$), according to a particular *partitioning scheme*. Then, one or more test cases are selected from each subdomain according to a particular *allocation scheme*.

The goal of partitioning scheme is to divide input domain into subdomains, such that elements in a subdomain are somehow “the same”, namely if one causes a failure, others do the same. Such a subdomain, either all its members cause the program to succeed or all cause it to fail, is called homogeneous [50]. If the subdomains are homogeneous, random selection of an element from each subdomain can determine the presence or absence of program fault. However, in practice, it is extremely difficult for the subdomains to be truly homogeneous. The only partitioning strategy which satisfies the homogeneous subdomain criteria for all programs is the division of the input domain into single element subdomains, which essentially degrades to the exhaustive testing.

An extreme case of partition testing is *random testing*. The partition contains only one subdomain, namely the whole input domain. Random testing simply selects test cases from the whole input domain randomly and independently [49, 62].

In recent two decades, it has been a controversial topic on the effectiveness of random testing and partition testing. Intuitively, partition testing should be effective in revealing errors in programs as it uses the information of the program specification or the program structure. However, simulations and experiments conducted by Duran and Ntafos [41] found that the effectiveness of partition testing is not, or just slightly superior to random testing. Their surprising results were confirmed by more extensive simulations and experiments [50, 62]. The first analytical investigation towards partition testing and random testing is conducted
by Weyuker and Jeng [96], in which it is found that partition testing can be a good testing strategy or a poor one depending on how well that partitioning strategy groups the failure-causing inputs together. In the subsequent analytical analysis, Chen and Yu [31–34] proved that Proportional Sampling Strategy (PSS), under certain conditions, is the only form of partition testing that always outperforms random testing, in terms of the probability of detecting at least one failure. In PSS, the number of test cases selected from a partition should be proportional to size of that partition.

From the view of overhead, random testing is obviously superior to partition testing, because the initial partitioning of the input domain may cause significant overhead. For example, in path-coverage partition, solving the path predicates and identifying each sub-domain may cause significant overhead, even in moderate size programs.

2.3 Successful Test Case

If a test case does not reveal any failure, it is called a successful test case. Most test cases are successful test cases if the program is written by a competent programmer [38]. In conventional testing, successful test cases are usually considered to be useless [74] and will be discarded or retained only for regression testing later. However, in recent studies, successful test cases have been found to be informative and should be exploited further. Fault-based testing [73] is an example of the utilization of successful test cases to prove the absence of prescribed faults. Another example is Metamorphic Testing (MT) [13], which is proposed to alleviate the oracle problem by employing successful test cases and expected program properties.

2.4 Observation-based Testing

Observation-based testing is a test case selection strategy based on the difference of execution profiles [39, 60, 61, 81]. An execution profile records some aspects of a program’s execution. For example, path profile [84] records the execution frequency of each loop-free intra-procedural path executed in a run. Many aspects of program execution can be profiled,
such as the control flow, data flow, variable values and event sequences.

Basically, observation testing involves following steps:

- taking a large number of test cases as initial test suite;
- executing all test cases and recording the their execution profiles at the same time, _but do not verify the outputs_;
- conducting cluster analysis on the execution profiles, where test cases with similar execution profiles are placed in the same cluster, and test cases with dissimilar execution profiles are placed in different clusters.
- selecting a subset of all executed test cases to verify the outputs.

The method is usually applied in beta testing. The rationale behind is that the unusual behavior of the failure-causing inputs may be reflected in their execution profiles. Most likely, the execution profile of failure-causing inputs will be isolated in clusters of small size. If a test case can detect a failure, then other test cases in the same cluster are also likely to detect a failure.

In the software engineering literature, the difference of execution profiles has been widely used in regression testing [52, 86, 100], software maintenance [84], and software reliability estimation [82]. In the area of regression testing, Rothermel et al. [86] proposed and evaluated several test case prioritizing techniques based on execution profiles. Harrold et al. [52] compared program spectra from a program $P$ and its faulty version $P'$ on the same inputs in regression testing. They found that spectra differences can always be expected if the inputs cause $P'$ to fail. However, not all inputs that produce spectra differences can cause $P'$ to fail. Xie and Notkin [100] enriched the existing program spectra family by proposing a new program spectra, value spectra. For software maintenance, Reps et al. [84] use the path spectra (execution profile) difference to find the point of divergence in computations, and consequently to locate the date-dependent computations to fix the “year 2000 problem”. On software reliability estimation, Podgurski and his colleagues conducted a series of researches [39, 60, 61, 81] based on cluster analysis the execution profiles.
2.5 Failure Patterns

A test case that can detect a failure is regarded as a failure-causing input. The set of failure-causing inputs are called failure domain [2]. In several studies [2, 5, 10, 43], it is found that failure-causing inputs tend to be clustered together. Ammann and Knight studied the geometry of failure domain on experiments of several hypothetical missile launch decision programs [2]. Bishop conducted experiments on two reactor trip programs specified by the UK Atomic Energy Authority [5]. NASA also conducted two sets of experiments to characterize software failure processes [43]. All of them get similar observations: failure-causing inputs are clustered in contiguous regions. Hence, a term, failure region, is proposed to describe failure domain and its geometry [2]. Furthermore, Bishop [5] identified several factors affecting the geometry of failure region, which includes “incorrect positioning of the boundary between two types of calculation, missing boundaries, incorrect table values, inadequate calculation precision and algorithm approximations.”

Chan et al. [10] classified the typical failure regions into three types and named them as failure patterns. The common failure patterns are the block, strip, and point patterns. Examples of these failure patterns for a program with a 2-dimensional input domain are given in the schematic diagrams in Figure 2.1, where the outer square represents the input domain and the shaded areas represent failure-causing inputs. The characteristic of block failure pattern is that failure-causing inputs are contiguous and restricted in one or a few failure regions. For strip failure pattern, the failure-causing inputs form a narrow area. A typical example for strip pattern is domain errors [97]. For point failure pattern, the failure-causing inputs are separated or form failure regions of a very small size. In practice, non-point failure patterns are more common than point failure pattern. Figure 2.2(a-c) show fragments of pseudo-code producing each of these failure patterns.
INTEGER X, Y
INPUT X, Y
IF (X > 7 AND X < 9)
    AND (Y > 8 AND Y < 12)
THEN
    Z = X + Y
    // should be Z = X * Y
ELSE
    Z = X / Y
OUTPUT Z

(a) Block pattern

INTEGER X, Y
INPUT X, Y
IF (X + Y < 10)
    // should be IF (X + Y < 12)
THEN
    Z = X * Y
ELSE
    Z = X / Y
OUTPUT Z

(b) Strip pattern

INTEGER X, Y
INPUT X, Y
IF (X mod 4 = 0)
    AND (Y mod 4 = 0)
THEN
    Z = X + Y
    // should be Z = X * Y
ELSE
    Z = X / Y
OUTPUT Z

(c) Point pattern

Figure 2.2: Code fragments producing examples of the three types of failure patterns
2.6 Adaptive Random Testing (ART)

2.6.1 Intuition of ART

Adaptive Random Testing (ART) is an improvement of random testing in terms of detecting the first failure by fewer test cases [28, 63]. ART is based on the observation about the types of failure patterns.

Figure 2.3 is a schematic diagram to illustrate the intuition. In this figure, the 2-dimensional input domain is represented by outer square and the failure region is represented by shaded areas. The size and position of failure region is fixed but unknown. Suppose the geometry of failure region is a circle with radius $r$, located on the top left of the input domain as shown in Figure 2.3. Test case $t_1$ is randomly generated. Unfortunately, $t_1$ does not hit the failure region, hence it is a successful test case. Now we generate the second test case. If the failure-causing inputs are scattered over the input domain, any points throughout the input domain have the same chance to be a failure-causing input. As shown in failure pattern studies (Section 2.5), non-point failure patterns are more common. Therefore, it is better to generate the next test case outside the dashed circle, namely the distance between the next test case and $t_1$ should be greater than $r$. However, the size and position of the failure region is unknown, hence, the subsequent test cases are required to be evenly spread in the input domain.

From this simple diagram, we can get the following conclusion:

- A set of even spread test cases are more effective than clustered test cases in terms of fault-detection capability.

- The successful test case is informative in the sense of guiding the selection of subsequent test cases.

Hence, Chen et al. suggested that if test cases are more evenly spread, fewer test cases would be required to detect the first failure, and the locations of successful test cases can be exploited to guide the selection of subsequent test cases. In short, the fundamental principle or the basis of ART is to achieve an even spread of test cases throughout the input domain.
2.6.2 Fault-detection Capability Metrics

There are three common metrics for evaluating the fault-detection capability of a testing technique: P-measure, E-measure and F-measure [28, 72]. P-measure is the probability for a given set of test cases to detect at least one failure. E-measure is the expected number of failures detected for a given set of test cases. F-measure is the expected number of test cases to detect the first failure. ART has an assumption that once a failure is detected, the fault will be corrected immediately. Hence, Chen et al. proposed to use F-measure as the fault-detection capability metric for ART [28]. Other commonly used metrics like P-measure and E-measure require the knowledge of the size of the test suite prior to testing. However, ART is an iterative test case selection method, in which no predefined set of test cases is needed.

Since ART is an enhancement of RT and the fault-detection capability of these two methods are frequently compared, relative F-measure is used to indicate the fault-detection capability of an ART method. Relative F-measure is defined 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. The lower relative F-measure is, the more improvement an ART method achieves. As mathematically analyzed in [23, 68], the F-measure for RT with replacement is \( 1/\theta \), where \( \theta \) is the failure rate.

However, it is extremely difficult to derive a general result for the F-measure of an ART
method. In previous practices [57, 63, 92], the F-measure of an ART method is obtained through simulations. It is much easy to control the critical experiment parameters in simulation studies, such as the failure rate, dimensionality of the input domain, geometry of the failure pattern and the location of failures. The focus of this thesis is to establish taxonomy and propose enhancement for existing ART algorithms. Therefore, we also use simulation studies as the primary means to compare the fault-detection capability.

With regard to empirical studies on real programs, 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. Furthermore, the fault-detection capability depends on the geometry of the failure pattern. ART has better fault-detection capability in block and strip patterns. The results of empirical studies are consistent with the simulation studies. Similar to other projects that involve a chosen set of programs, our results are obviously applicable only to that particular set of programs. However, we believe the choice is reasonably general to cover general situations, and the results are hence generally applicable.

2.6.3 Experimental Sample Size

In previous studies [57, 63, 92], the sample size in the simulations was evaluated according to the Central Limit Theorem [42]. To estimate the mean of F-measure with an accuracy range of \( \pm r\% \) 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, \( \mu \) is the population mean and \( \sigma \) is the population standard deviation. The confidence level was set to 95% and accuracy range was set to \( \pm 5\% \). From statistical table one can look up \( z = 1.96 \) for confidence level 95%. Since \( \mu \) and \( \sigma \) were unknown, the mean \( (\bar{x}) \) and the standard deviation \( (s) \) of the
collected sample data were used instead, respectively. Hence, sample size was evaluated as

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

The simulations estimated the sample size \( S \) as data was being generated. Suppose \( V \) is the number of sample data we have collected. Then, simulations were repeated until \( V \) is greater than \( S \). At this stage, the mean F-measure was reported that is accurate within ±5% of its value at 95% confidence level. For all simulations in this thesis unless otherwise specified, we will follow the previous practices to estimate the simulation sample size according to the Central Limit Theorem and set the confidence level to be 95% and the accuracy range to be ±5%.

### 2.6.4 Related Works

Adaptive software testing was proposed by Cai and his colleagues [6–8], which is essentially different from ART. Their approach regards software testing as a control problem. Software under test is mathematically modeled as a controlled Markov chain, and the theory of controlled Markov chains is used to adjust the testing strategy during the testing process.

Antirandom testing [64] is another attempt to improve the fault-detection capability of pure random testing. The rationale behind is to generate new test case to cover “some part of the functionality not yet covered by tests already generated” [64]. The first test case is generated randomly, and the subsequent test cases are selected such that it maximizes the total distance from all previously generated test cases. The main character of this method is that the selection of the first test case will completely determine the sequence of the subsequent test cases. If a deterministic method is used to deal with the tie-breaking equally distance test cases, only the first test case is randomly selected. ART is different from antirandom testing in the sense that all test cases are randomly generated. Generally speaking, ART is a failure-based testing strategy, whose basis is on the observation of failure patterns.

Previous studies on ART focus on the following aspects:

- Various methods to implement ART [12, 14, 15, 21, 24, 92].
• Theoretical analysis on the upper bound of software testing effectiveness. Merkel [68] conducted a mathematical analysis for the case of a single failure region, whose the size and shape (but not location) are known. He deduced that the “optimal” ART method will reduce the expected F-measure by no more than 50% over RT, if the size of the failure region is small comparing with the size of the input domain. The result was then generalized to any number of distinct failure regions by Chen and Merkel [27].

• Distribution of failure-causing input and the effectiveness of ART. Chen et al. [25, 57] conducted a series of simulations to identify the factors affecting the effectiveness of ART. They found that the effectiveness of ART depends on the dimensionality of the input domain, failure rate, failure region(s)’s attributes (such as amount, compactness, proximity to the boundary of input domain).

• Statistical properties of ART. The studies [23] enhanced the understanding of testing case spatial distribution and the sampling distribution of P-, E-, F-measures.

• The application of ART to non-numeric programs. The researches mainly focus on two problems: random generation and dissimilarity scheme of non-numeric test cases [57, 68].

Designing ART methods with good fault-detection capability and low overhead is the major part of researches on ART. However, existing ART methods have their own advantages and disadvantages. This thesis focuses on classifying existing ART methods and proposing improvements for each category of ART methods.
Chapter 3

A Taxonomy of Adaptive Random Testing Methods

3.1 Introduction

Several ART methods have recently been developed based on their own particular understanding of the basis of ART. These ART methods have their own advantages and disadvantages. Since no proper classification of existing ART methods in previous studies, the relationships among ART methods were unclear and it was difficult to compare and improve these ART methods. This chapter establishes a taxonomy of ART methods and summarizes the characteristics for each category of ART methods. This taxonomy is the framework for further studies on ART methods.

3.2 Taxonomy

The basis of ART is to achieve an even spread of test cases throughout the input domain. However, even spread is an abstract and qualitative concept and does not indicate a specific method to implement it. Different ART methods have their own specific and quantitative interpretations of an even spread of test cases. Therefore, a fundamental way to classify the ART methods is from different interpretations of the basis of ART.
In this thesis, ART methods are classified into 3 categories\(^1\) based on their different interpretations of the basis of ART.

- **“Far apart” ART methods:** This category interprets an even spread of test cases as all test cases are far apart from one another. The major characteristic is to select the candidate that is farthest away or sufficiently far away from the nearest successful test case (that is, test cases that have been executed but *not* revealed a failure) as the next test case by exploring the locations of successful test cases. Distance-based ART (D-ART) [26, 63] and Restriction-based ART (R-ART) [11, 12] are two typical methods in this category with different test case selection processes.

- **Proportional Sampling Strategy (PSS) ART methods:** This category interprets an even spread of test cases as the number of test cases selected from a partition should be proportional to the size of that partition. The major characteristic is to select subsequent test case from the sparsely populated partitions to assure a similar test case density in all partitions of the input domain. ART through Dynamic Partitioning (DP-ART) [14] and Lattice-based ART (L-ART) [67] are two typical methods with different partitioning strategies.

- **Balancing ART methods:** This category interprets an even spread of test cases as the centroid of test cases in a partition should be close to the centroid of that partition. This is an innovative interpretation of the basis of ART proposed in this thesis. The major characteristic is to partition the input domain incrementally and apply the balancing strategy in individual partitions. The balancing strategy requires selecting those candidates that make resultant centroid of test cases close to the centroid of the partition.

In the following sections, we shall review “far apart” and PSS ART methods. Details of balancing ART methods will be given in Chapter 8.

\(^1\)including the new ART method proposed in Chapter 8
3.3 “Far Apart” ART Methods

Distance-based ART (D-ART) and Restriction-based ART (R-ART) are the first two ART methods. In previous studies [57, 92], they were regarded as different ART methods based on distinctive notions. D-ART was interpreted as “selecting the best from a pool of candidates” and R-ART was interpreted as “picking up the first meeting the criterion as the next test case”. This section briefly reviews these two methods and proposes that they are essentially based on the same interpretation of an even spread of test cases. Further analysis of D-ART and R-ART will be given in Section 4.2.1 and Section 5.2.

D-ART [26, 63] maintains a set of candidates and a set of successful test cases. The candidate set consists of a fixed number of test case candidates, from which new test cases will be selected. The successful set records the locations of all successful test cases, which are used to guide the selection of the next test case. D-ART selects the next test case from the candidate set, based on the criterion of maximizing the minimum distance between the next test case and all the successful test cases.

Figure 3.1 (a-f) illustrate an example of the test case selection process of D-ART in a 2-dimensional input domain. The solid circles denote the successful test cases, while the solid triangles denote the candidates. Suppose there are 3 successful test cases, $s_1$, $s_2$ and $s_3$, in the successful set. To select the 4th test case, 3 candidates ($c_1$, $c_2$ and $c_3$) are randomly generated to form the candidate set. After calculating the distances between $c_1$ and all elements in the successful set, the minimum distance between $c_1$ and the successful set is identified, say, $Min_1$, as shown in Figure 3.1 (d). Similarly, the minimum distances between $c_2$, $c_3$ and the successful set are also identified. As shown in Figure 3.1 (e), $c_3$ has the largest minimum distance, it is therefore selected as the next test case $s_4$.

R-ART [11, 12] 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.

Figure 3.2 (a-f) illustrate an example of the test case selection process of R-ART in a 2-dimensional input domain. The dashed circles denote the exclusion zones. Others are the same as Figure 3.1. At first, exclusion zones are set for successful test cases, $s_1$, $s_2$ and $s_3$. 22
Candidate $c_1$, $c_2$ and $c_3$ are randomly generated in the input domain one by one, as shown in Figure 3.2 (c-e), respectively. Since $c_1$ and $c_2$ reside in the exclusion zones of $s_1$ and $s_2$, respectively, they are discarded. Candidate $c_3$ lies outside all exclusion zones of successful test cases and is qualified as next test case $s_4$.

The notion of D-ART is to find the best among a set of potential test cases as the next test case. “Best” is based on the criterion that the candidate has farthest distance from its nearest successful test case. For R-ART, a qualification is set prior to test case generation process and the first qualified candidate is deemed as the next test case. The “qualification” is that the next test case cannot locate within the exclusion zones of any successful test cases. It should be noted that if a candidate is outside the exclusion zone of its nearest successful test case, it should be outside all other exclusion zones. At first glance, these two methods are based on distinctive notions. D-ART is to select the best among all candidates, while R-ART is to find the first satisfying the qualification. However, the selection criterion in D-ART and the qualification in R-ART are essentially the same. Both try to assure next test case is far apart from nearest successful test cases, so that all test cases can be far apart from one another. Keeping test cases far apart from one another is their common interpretation of
Experimental studies have showed that these two methods have greatly outperformed RT. However, both methods have two common disadvantages. One is extensive computations, which is quadratic to number of test case generated. Another is boundary effect, which means test cases have a general preference to be close to the boundary of the input domain. Chapter 4 and Chapter 5 will analyze these two problems in detail and propose improvements.

### 3.4 PSS ART Methods

ART through Dynamic Partitioning (DP-ART) [14] and Lattice-based ART (L-ART) [67] are two PSS ART methods. Both directly generate subsequence test cases in the sparsely populated regions. The difference is their partitioning strategy: DP-ART partitions the input domain dynamically, while L-ART partitions the input domain statically.

The following briefly reviews DP-ART. Further analysis will be given in Section 6.2. DP-ART [14] was inspired by partitioning testing, which divides the input domain to iden-
RP-ART divides the input domain by 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.

An example of the test case generation process of RP-ART is illustrated in Figure 3.3 (a-f). At first, since the whole input domain is the only subdomain, it is obviously the test case generation region. Within the test case generation region, a test case ($s_1$) is randomly generated. Suppose $s_1$ is a successful test case. Then, the test case generation region is further divided by $s_1$ into four subdomains. The largest subdomain denoted with thick borders is selected as the next test case generation region, as shown in Figure 3.3 (b). A test case ($s_2$) is randomly generated within this test case generation region. Suppose $s_2$ is also a successful test case and the subdomain containing $s_2$ is further partitioned. Similar process for the following test case $s_3$ is illustrated in Figure 3.3 (e-f).
Figure 3.4: An example that illustrates the B-ART method

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 further subdivided into halves. The testing process is repeated until a failure is detected or the testing resources are exhausted.

Figure 3.4 (a-f) illustrate an example of the test case generation process of B-ART. The first test case ($s_1$) is randomly selected from the whole input domain. Suppose $s_1$ is a successful test case. Since the only subdomain has already contained a successful test case, the input domain is bisected. The “empty” subdomain, denoted by thick borders, becomes the test case generation region, as shown in Figure 3.4 (b). Again, a test case ($s_2$) is randomly generated within the test case generation region. Suppose it does not detect a failure. Then, subdomains are further subdivided into halves, since they all contain a successful test case. At this stage, two “empty” subdomains are available, as shown in Figure 3.4 (d). One is randomly selected as the test case generation region and $s_3$ is randomly generated within this region. Similar process for the following test case $s_4$ is illustrated in Figure 3.4 (e-f).

L-ART [67] exploits a static way to partition the input domain. Basically, it has two
steps. Firstly, a lattice structure is formed by a set of vertical and horizontal lines that equally partition the input domain. Lattice nodes are systematically placed on the intersections of these vertical and horizontal lines. Secondly, test cases are formed by randomly selecting and “shaking” these lattice nodes. “Shaking” means the lattice node is shifted by a random vector. Initially, a coarse lattice structure is used. If failures cannot be detected under current lattice structure, the lattice structure will be refined. Detailed analysis of the algorithm and the example of test case generation process will be elaborated in Section 7.2.

In D-ART and R-ART, all the elements of the entire input domain have equal chances of being selected as candidates. However, after the executions of several successful test cases, the input domain becomes uneven, where some regions have a higher density of successful test cases than other regions. Hence, in order to achieve an even spread of test cases, the next test case should be generated from the sparsely populated regions. DP-ART was proposed based on this observation. This method applies a partitioning scheme on the input domain to differentiate regions of varying densities of successful test cases. Instead of generating candidates from the entire input domain and then conducting distance computations to decide the most appropriate one as the next test case, DP-ART partitions the input domain first and then directly generate the next test case (rather than the “candidates”) from the sparsely populated regions. Effectively, generating subsequent test cases in the sparsely populated partitions leads all partitions have similar densities of test cases, which is exactly the principle of PSS. For L-ART, test cases are generated by randomly shaking the predefined lattice nodes. In fact, the principle of predefining the lattice nodes is to assure that the number of test cases in a partition is proportional to the size of that partition.

PSS ART methods run much faster than “far apart” methods, since distance computations and comparisons between each candidate and all successful test cases are totally avoided. However, we find that in PSS methods test cases still have chances to be clustered together, which has adverse impact on the fault-detection capability. Chapter 6 and Chapter 7 will further investigate this problem and propose improvements.
3.5 Summary

This taxonomy of ART methods is the foundation of this thesis. With the taxonomy, the thesis can summarize the characteristics for each category of ART methods rather than individual methods. With the knowledge of the characteristics of a category of ART methods, improvements can be proposed for a category of methods rather than individual ones. Furthermore, the taxonomy facilitates the development of innovative notions to achieve an even spread of test cases.

This taxonomy is based on different interpretations of the basis of ART, which gives rise to the fundamental difference of ART methods. Other differences, like test case selection process, are secondary. This taxonomy can never be complete, simply because new ART methods may arise. Nevertheless, we believe that our proposed taxonomy is sufficiently precise for the current situation.
Chapter 4

Adaptive Random Testing through Iterative Partitioning

4.1 Introduction

“Far apart” ART methods have two major shortcomings. One is extensive computations of selecting test cases. Another shortcoming is a phenomenon in D-ART and R-ART, we call it “boundary effect”\(^1\), which has adverse impact on the fault-detection capability. This chapter and the following chapter will examine these two shortcomings respectively. In this chapter, a new ART method, namely *Adaptive Random Testing through Iterative Partitioning* (IP-ART) [19, 21], is proposed to reduce the overhead while retaining the high fault-detection capability. This method is also based on the notion of “far apart” to achieve an even spread of test cases.

4.2 Review and Analysis of Related Works

4.2.1 D-ART and R-ART

In this section, the time complexity of D-ART and R-ART methods are analyzed. D-ART [26, 63] selects the next test case from a fixed-size set of randomly generated can-

\(^1\)details will be explained in Chapter 5
didates, based on the criterion of maximizing the minimum distance between the next test case and all the successful test cases. This procedure is elaborated as follows. Let $C = \{C_1, C_2, \ldots, C_e\}$ and $S = \{S_1, S_2, \ldots, S_l\}$ be the candidate set and the successful set, respectively. Let us denote the Cartesian distance in a $k$-dimensional input space between a candidate $C_i = \{C_{i_1}, C_{i_2}, \ldots, C_{i_k}\}$ and a successful test case $S_j = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}\}$, where $k \geq 1$, by $\text{dist}(C_i, S_j) = \sqrt{\sum_{p=1}^{k}(c_{ip} - s_{jp})^2}$. Let $\text{Min}_i$ be the minimum Cartesian distance between candidate $C_i$ and all the members in $S$, that is $\text{Min}_i = \min\{\text{dist}(C_i, S_j) | 1 \leq j \leq l\}$.

Then D-ART will select a candidate $C_q$, where $\text{Min}_q$ is the largest among $(\text{Min}_1, \text{Min}_2, \ldots, \text{Min}_e)$, to be the next test case and discard all the other candidates. The original D-ART algorithm for testing a program with $k$-dimensional input domain is shown in Algorithm 4.1.

**Algorithm 4.1 Original D-ART algorithm**

Set $S$ to be empty and $l$ to be 0

\[
\text{do}
\{
\begin{align*}
\text{Randomly generate } e \text{ test cases to form the candidate set } C \\
\text{For each candidate } C_i \text{ in } C \\
\{ \\
\text{For each successful test case } S_j \text{ in } S \\
\{ \\
\text{dist}(C_i, S_j) = \sqrt{\sum_{p=1}^{k}(c_{ip} - s_{jp})^2} \\
\text{Min}_i = \min\{\text{dist}(C_i, S_j) | 1 \leq j \leq l\} \quad \text{// Min}_i \text{ is the minimum Cartesian distance between candidate } C_i \text{ and all successful test cases in } S \}$
\}
\text{Take } C_q \text{ as the test case such that } \\
\text{Min}_q = \max\{\text{Min}_i | 1 \leq i \leq e\} \\
\text{Add } C_q \text{ to } S \\
\text{l = l + 1}
\\}
\text{while (C_q does not reveal a failure and resource limit has not been reached)}
\]

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 $i^{th}$ test case is to be generated, where $i > 0$. The distance computation and comparison for generating the $i^{th}$ 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 [26, 63], we set $l$ to 10 in the experiments.

R-ART [12] only maintains the successful set $S = \{S_1, S_2, \ldots, S_l\}$ without any candidate set. Instead, R-ART specifies exclusion zones around every successful test cases. Candidates are generated one by one from the whole input domain randomly and independently. The first one outside all exclusion zones is selected as the next test case. The original R-ART algorithm for testing a program with k-dimensional input domain is shown in Algorithm 4.2.

**Algorithm 4.2 Original R-ART algorithm**

Set $S$ to be empty, $l$ to be 0

\[
\begin{align*}
\text{do} & \quad \text{do} \\
\quad \text{Randomly generate a candidate } c & \\
\quad \text{For each successful test case } S_j \text{ in } S & \\
\quad \quad \text{dist}(c, S_j) = \sqrt{\sum_{p=1}^{k}(c_p - s_{j_p})^2} & \\
\quad \quad \text{if dist}(c, S_j) > \text{exclusion zone radius of } S_j & \\
\quad \quad \quad c \text{ is outside the exclusion zone} & \\
\quad \quad \quad \text{else} & \\
\quad \quad \quad \quad c \text{ is inside the exclusion zone} & \\
\quad \text{while } (c \text{ is not outside all the exclusion zones}) & \\
\quad c \text{ is the next test case} & \\
\quad \text{Add } c \text{ to } S & \\
\quad l = l + 1 & \\
\text{while } (c \text{ does not reveal a failure and the resource limit has not been reached}) & 
\end{align*}
\]

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 $R_T = A_T/D$. 

31
Table 4.1: Relationship between the target exclusion ratio and the average actual exclusion ratio in R-ART, adopted from [92]

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 [92], the relation between $R_A$ and $R_T$ is studied empirically. As shown in Table 4.1, 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 [12] to set the target exclusion ratio to 150%.

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, $P(X = 2) = (1 - R_A) \times R_A$. 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.

### 4.2.2 Mirror Adaptive Random Testing

Mirror Adaptive Random Testing (M-ART) is an attempt to reduce the overhead of D-ART and R-ART [24]. 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 [24]. Nevertheless, the time complexity of M-ART is still quadratic, and the number of subdomains ($w$) should be kept small to ensure randomness.

### 4.3 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 chapter 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 [50]. 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.

4.3.1 Overview

To illustrate the basic idea of IP-ART, let us consider Figure 4.1. Figure 4.1(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 $5 \times 5$ grid as shown in Figure 4.1(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 Figure 4.1(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 Figure 4.1(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.
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.

Figure 4.2 depicts the partitioning of a 2-dimensional input domain. Suppose we have a program $\text{prog}(a, b)$ where $a$ and $b$ are real numbers and $0 \leq a, b < M$, 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 Figure 4.2, the input domain is partitioned by a $10 \times 10$ grid, and $M$ is set to 100. As an example, the test case $(28.8, 12.6)$ is mapped into grid cell $(2, 1)$. 

4.3.2 The Grid Coordinates Used in IP-ART
An algorithm is described for partitioning the input domain into grid cells. The algorithm starts with a coarse grid and refines it based on the resolution of the initial grid. The categorization of grid cells includes occupied cells with successful test cases, adjacent cells without any test case but surrounding neighbors of some occupied cells, and candidate cells that are neither occupied nor adjacent cells. The algorithm decides on the resolution of the initial grid and refines it if necessary. If no failure is revealed and no candidate cell is available, the current $p \times p$ partitioning scheme is discarded and a finer partitioning scheme using a $(p+1) \times (p+1)$ grid is applied to partition the input domain.
Algorithm 4.3 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 $\text{GridCells}$ 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.

An example that illustrates Algorithm 4.3 is given in Figure 4.3. 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 $2 \times 2$ 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 Figure 4.3(c). The input domain is then partitioned again using a $3 \times 3$ grid as shown in Figure 4.3(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 Figure 4.3(e). Figure 4.3(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.

### 4.3.5 Time Complexity of the IP-ART Method

The time complexity of Algorithm 4.3 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 4.3 is to map all the successful test cases (in $E$) to new grid cells when the input domain is repartitioned.
Algorithm 4.3 The basic IP-ART algorithm

It is assumed that the program under test is program\( (\text{parameter1}, \text{parameter2}) \), where parameter1 and parameter2 are real numbers and \( 0 \leq \text{parameter1}, \text{parameter2} < M \).

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, \( \text{GridCells} \), and assign \( T \) to all its entries. Use \( \text{CntCandidateCell} \) to count the number of candidate cells, and set \( \text{CntCandidateCell} = p \times p \).

3. Map each successful test case \((\text{parameter1} = x, \text{parameter2} = y)\) in \( S \) into a grid cell by assigning \( F \) to the corresponding occupied cell \( \text{GridCells}(\lfloor x \times p / M \rfloor, \lfloor y \times p / M \rfloor) \). Update \( \text{CntCandidateCell} \).

4. For each occupied cell \( \text{GridCells}(u, v) \), assign \( F \) to all its adjacent cells, namely \( \text{GridCells}(u - 1, v - 1), \text{GridCells}(u - 1, v), \text{GridCells}(u - 1, v + 1), \text{GridCells}(u, v - 1), \text{GridCells}(u, v + 1), \text{GridCells}(u + 1, v - 1), \text{GridCells}(u + 1, v), \text{GridCells}(u + 1, v + 1) \), as necessary. Update \( \text{CntCandidateCell} \).

5. While \( (\text{CntCandidateCell} > 0) \)

   (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 \( \text{GridCells} \) that correspond to \( R \) and all its adjacent cells; update \( \text{CntCandidateCell} \).

   EndWhile

6. Discard (release) the Boolean matrix \( \text{GridCells} \). Set \( p = p + 1 \).

7. If testing resources are not exhausted go to step 2.
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 scheme starts from \( 1 \times 1 \), the input domain is repartitioned \( m - 1 \) times when \( m \times m \) 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 [40]. 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.

### 4.3.6 Simulation Studies

To compare the fault-detection capability of IP-ART with that of other methods, a series of simulations have been conducted. For each test run, a failure region with a specified failure rate \( \theta \) 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.

Table 4.2 presents the results of the simulations conducted in a 2-dimensional input domain with failure rate \( \theta \) varying from 0.01 to 0.001 with respect to the three types of
Table 4.2: Relative F-measures of IP-ART in a 2-dimensional input domain under 3 failure patterns

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 4.3 compares the fault-detection capability of IP-ART and four major ART methods with $\theta$ 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.

Since time complexity hides constants that may be important in practices, 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 4.4 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.

4.4 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.

<table>
<thead>
<tr>
<th>Failure Rate $\theta$</th>
<th>F-measure of RT ($F_{RT}$)</th>
<th>Block Pattern</th>
<th>Strip Pattern</th>
<th>Point Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>100</td>
<td>63 63%</td>
<td>78 78%</td>
<td>96 96%</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>122 61%</td>
<td>156 78%</td>
<td>188 94%</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>301 60%</td>
<td>387 77%</td>
<td>476 95%</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>605 61%</td>
<td>781 78%</td>
<td>925 93%</td>
</tr>
</tbody>
</table>
### Table 4.3: Comparison of relative F-measures of IP-ART and other ART methods on a 2-dimensional input domain (under the block failure pattern)

<table>
<thead>
<tr>
<th>Failure rate $\theta$</th>
<th>D-ART</th>
<th>R-ART</th>
<th>RP-ART</th>
<th>B-ART</th>
<th>IP-ART</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>67%</td>
<td>65%</td>
<td>76%</td>
<td>75%</td>
<td>63%</td>
</tr>
<tr>
<td>0.005</td>
<td>66%</td>
<td>63%</td>
<td>77%</td>
<td>74%</td>
<td>61%</td>
</tr>
<tr>
<td>0.002</td>
<td>65%</td>
<td>62%</td>
<td>79%</td>
<td>74%</td>
<td>60%</td>
</tr>
<tr>
<td>0.001</td>
<td>65%</td>
<td>62%</td>
<td>80%</td>
<td>75%</td>
<td>61%</td>
</tr>
</tbody>
</table>

### Table 4.4: Recorded CPU time of different ART methods (with $\theta=0.001$ and under the block failure pattern, for a run of 5000 trials each method)

<table>
<thead>
<tr>
<th>ART Methods</th>
<th>D-ART</th>
<th>R-ART</th>
<th>RP-ART</th>
<th>B-ART</th>
<th>IP-ART</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time (in seconds)</td>
<td>3645.8</td>
<td>1562.1</td>
<td>1141.1</td>
<td>6.4</td>
<td>12.0</td>
</tr>
</tbody>
</table>

1. Instead of initializing the input domain to a $1 \times 1$ grid, can a finer grid be used to partition the input domain at the very beginning?

2. Instead of broadly classifying the cells into three different types, can different grid cells be assigned different weights according to their locations relative to the occupied cells?

We shall investigate these questions in the following subsections.

#### 4.4.1 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 4.3.5, 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 $\theta$ 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 \( 3 \times 3 \) cells being excluded for test case generation. Hence, only when the failure region occupies at least \( 3 \times 3 \) 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 \( 3 \times 3 \) 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 [27], 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 \( 3 \times 3 \) 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 $3 \times 3$ 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 [27, 68].

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 ($\theta$): 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 $\theta$. We suggest, therefore, overestimating $\theta$ (that is, assuming $\theta$ is large). Otherwise the initial partitioning scheme will be too fine to achieve a low $F$-measure. When $\theta$ is overestimated, a coarse initial partitioning scheme will be used (that is, $m$ is small). Consequently, the failure region may not occupy at least $3 \times 3$ 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 $p = 1, 5, 10, 15, \ldots, 95, 100$. 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 $q - p$. The results are shown in Figure 4.4. 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 Figure 4.5, 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.

### 4.4.2 Weighting the Grid Cells

In the basic IP-ART method, all cells surrounding an occupied cell are defined as adjacent cells. Since adjacent cells are close to successful test cases, they are not used for test case generation. The chances for inputs in different adjacent cells to be close to successful test cases, however, are in fact different. Consider a 2-dimensional input domain, for example, an adjacent cell may share a common side with an occupied cell; while another may share only a common vertex with the occupied cell. Obviously, a point in the former adjacent cell
Figure 4.4: Differences between the final ($q$) and initial ($p$) partitioning schemes in IP-ART

Figure 4.5: Relative F-measures of IP-ART on different initial partitioning schemes
will have a higher chance of getting close to the successful test case in the occupied cell. Furthermore, an adjacent cell may be a neighbour of more than one occupied cell, and this will increase the chance of its points getting close to successful test cases. It is therefore interesting to investigate whether it is worthwhile to treat different adjacent cells differently.

In the investigation, we assign different weights to different types of cells. The principle is that the more likely a cell is close to successful test cases, the higher weight this cell will receive. Every time a test case is to be generated, a cell with the minimum weight will be selected. A threshold is set as the maximum acceptable weight for a cell to be selected to generate a test case. The weight of occupied cells is set to be a large integer. The weight of all candidate cells is set to be zero, since these cells should be selected first. Note that, we do not further differentiate candidate cells, since IP-ART is based on the notion of exclusion, which excludes the regions surrounding the successful test cases and requires subsequent test cases to be selected from other regions. For adjacent cells, they are weighted according their relative locations to occupied cells. Initially, all adjacent cells are assigned a weight of 1. Then for each adjacent cell, its coordinate in each dimension is compared with that of its occupied cells. If it has an identical coordinate in a dimension, the weight of this adjacent cell will increase by 1. If the adjacent cell is a neighbour of more than one occupied cell, its weight is accumulated. In an $n$-dimensional input domain, if an adjacent cell is the neighbour of only one occupied cell, its maximum weight is $n$.

Figure 4.6 shows an example of weighting grid cells. Occupied cells are assigned a very large weight, say 1000. The blank cells are candidate cells and are assigned a weight of zero. Let us look at adjacent cells. Cell $(2, 2)$ is a neighbour of occupied cell $(1, 2)$ and they have one identical coordinate. Hence, cell $(2, 2)$ has got a weight of 2. Furthermore, cell $(2, 2)$ is also an neighbour of occupied cell $(3, 3)$ but they have no identical coordinate. The weight contribution from cell $(3, 3)$ is therefore 1. As a result, the final weight of cell $(2, 2)$ is 3.

Let us also use Figure 4.6 to illustrate how the threshold value is used to control test case generation. If the threshold is set to zero, only candidate cells can be used. This is the basic IP-ART method. If the threshold is set to 1, adjacent cells $(3, 0), (0, 1), (2, 1), (0, 3), (2, 4)$,
Figure 4.6: Weighted grid cells in a 2-dimensional input domain

and (4,4) can also be used as test case generation regions when there is no more candidate cells. In this way the repartitioning of the input domain will become less frequent. Note that each of these adjacent cells is a neighbour of only one occupied cell, and shares only a vertex with it. In a 2-dimensional input domain, the maximum weight of an adjacent cell is 12, when it is surrounded by eight occupied cells. Hence, if the threshold has a value from 12 to 999, all adjacent cells can be used for test case generation.

The fault-detection capability of this version of IP-ART is investigated through simulations. The simulations were conducted in a 2-dimensional input domain with failure rate $\theta$ varied from 0.01 to 0.001, and under the block failure pattern. The threshold, $h$, was varied from 0 to 12. Table 4.5 shows the relative F-measure for different threshold values. The table shows that $h = 0$ gives the highest fault-detection capability. When $h$ increases, the relative F-measure also increases and reaches a plateau at $h = 7$. After that the relative F-measure decreases and become around 71% when the maximum threshold is reached. This result suggests that allowing test cases to be generated from some adjacent cells will have an adverse impact on the fault-detection capability of IP-ART.
Table 4.5: Relative F-measures of IP-ART for different threshold values in a 2-dimensional input domain

<table>
<thead>
<tr>
<th>Threshold h</th>
<th>θ = 0.01</th>
<th>θ = 0.005</th>
<th>θ = 0.002</th>
<th>θ = 0.001</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>63%</td>
<td>61%</td>
<td>60%</td>
<td>61%</td>
</tr>
<tr>
<td>1</td>
<td>67%</td>
<td>67%</td>
<td>64%</td>
<td>64%</td>
</tr>
<tr>
<td>2</td>
<td>72%</td>
<td>70%</td>
<td>67%</td>
<td>67%</td>
</tr>
<tr>
<td>3</td>
<td>74%</td>
<td>73%</td>
<td>70%</td>
<td>70%</td>
</tr>
<tr>
<td>4</td>
<td>76%</td>
<td>75%</td>
<td>73%</td>
<td>72%</td>
</tr>
<tr>
<td>5</td>
<td>79%</td>
<td>77%</td>
<td>74%</td>
<td>74%</td>
</tr>
<tr>
<td>6</td>
<td>80%</td>
<td>78%</td>
<td>75%</td>
<td>74%</td>
</tr>
<tr>
<td>7</td>
<td>81%</td>
<td>78%</td>
<td>76%</td>
<td>76%</td>
</tr>
<tr>
<td>8</td>
<td>78%</td>
<td>78%</td>
<td>76%</td>
<td>76%</td>
</tr>
<tr>
<td>9</td>
<td>78%</td>
<td>76%</td>
<td>74%</td>
<td>75%</td>
</tr>
<tr>
<td>10</td>
<td>78%</td>
<td>75%</td>
<td>74%</td>
<td>73%</td>
</tr>
<tr>
<td>11</td>
<td>76%</td>
<td>75%</td>
<td>73%</td>
<td>73%</td>
</tr>
<tr>
<td>12</td>
<td>72%</td>
<td>71%</td>
<td>70%</td>
<td>70%</td>
</tr>
</tbody>
</table>

4.5 Discussion and Summary

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.

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 chapter has proposed 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; (2) which cells can be used for test case
generation. 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. In the simulations of weighting grid cells, it is found that selecting some adjacent cells as test case generation regions has an adverse impact on the fault-detection capability.

We would like to point out that IP-ART is not suitable for programs with input domains that have a very high dimensionality. As analyzed in Section 4.3.5, the time complexity of IP-ART is in $O(3^{(k+1)} \cdot n^{(1+1/k)})$. 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.

How to apply IP-ART to programs with a very high dimensional input domain is one of our future research areas. In addition, we shall study programs with complex data structures as inputs, for which dividing the input domain into equally sized partitions is not straightforward or even impossible.
Chapter 5

Tackling Boundary Effect in “Far Apart” ART Methods

5.1 Introduction

In ART, test cases generated by some methods are not uniformly distributed in the input domain. For “far apart” ART methods, test cases prefer to be near the boundary of the input domain. This phenomenon adversely influences the fault-detection capability, and this impact grows with the increase of dimensionality of the input domain. In addition, fault-detection capability of “far apart” ART methods will depend on the locations of the failure regions. This chapter analyzes the cause of this phenomenon and proposes an approach to tackle it. Our approach is based on an innovative concept of virtual images of successful test cases [18]. Simulation results have clearly indicated that test cases generated by our enhanced methods are more evenly spread throughout the input domain. As a result, the fault-detection capability has also been improved. This improvement is particularly significant for high dimensional input domains.

5.2 Boundary Effect

“Far Apart” ART methods include three methods. They are Distance-based ART (D-ART), Restriction-based ART (R-ART) and ART through Iterative Partitioning (IP-ART). All of
them interpret an even spread of test cases as test cases being far apart from one another. D-ART and R-ART explicitly use distance as a gauge to measure whether a chosen candidate is the farthest apart or sufficiently far apart from all successful test cases. While IP-ART partitions the input domain to directly generate test cases in those regions that are far apart from successful test cases. However, test cases generated by these attempts are far apart but not necessarily evenly spread. Since no successful test cases can be outside the boundary of the input domain, for D-ART and R-ART the candidates near the boundary have a higher chance to be selected as test cases; for IP-ART the cells next to the boundary have a higher chance to be test case generation regions. Hence, these methods have a general preference in generating test cases close to the boundary of the input domain. This phenomenon is regarded as boundary effect in this thesis.

Two series of simulations were conducted to demonstrate such effect. In the first simulation, we investigated the spatial distributions of the test cases generated by D-ART, R-ART and IP-ART without considering the failure-causing inputs. In each trial of test case generation, the locations of the first $n$ test cases, where $n = 1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 40, 50, 100, 500, \text{ or } 1000$, 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 $n$.

To clearly demonstrate the spatial distribution, the positions of the test cases were projected onto one dimension. We analyzed the distribution in one dimension without loss of generality because ART methods treat every dimensions independently. The simulation was conducted in a 2-dimensional input domain in the shape of a unit square. 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. Figure 5.1 illustrates the histograms for D-ART. The histograms of R-ART are not listed, as they are similar to D-ART. It can be seen that test cases always prefer to be close to the boundary of the input domain, but the preferred region becomes narrower with
### Table 5.1: F-measures of differently located failure regions for D-ART, R-ART and IP-ART under block failure pattern on a 2-dimensional input domain ($\theta = 0.01$)

<table>
<thead>
<tr>
<th>Location of failure-causing input</th>
<th>D-ART</th>
<th>R-ART</th>
<th>IP-ART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>53</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Centre</td>
<td>67</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>Anywhere</td>
<td>66</td>
<td>62</td>
<td>63</td>
</tr>
</tbody>
</table>

the increase of test cases.

Histograms of IP-ART are shown in Figure 5.2. 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 with much less test cases residing in the areas adjacent to them.

The second simulation investigates the fault-detection capability if the locations of failure regions were purposely controlled to be close to the boundary or centre. The locations of failure regions are classified as the centre area or edge area as follows: the centre area (“Centre”) is defined as the central 80% of the whole input domain and the other area is defined as the edge area (“Edge”). In the simulations, a square failure-causing region with failure rate 0.01 was randomly assigned anywhere in the input domain (“Anywhere”) or confined to specified areas (namely “Centre” or “Edge”). The simulations were conducted in a 2-dimensional input domain. As shown in Table 5.1, D-ART, R-ART and IP-ART have higher fault-detection capabilities when the failure regions are close to the boundary.

### 5.3 Virtual Images of Successful Test Cases

This chapter proposes an approach to tackling the boundary effect of “far apart” ART methods. As analyzed in the last section, the reason for the boundary effect is that no successful test cases can be outside the input domain. Our approach introduces a new concept of virtual images of successful test cases, which let the successful test cases “step out” the input domain. Whenever the locations of the successful test cases are considered, not only the
Figure 5.1: Histogram of D-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.
Figure 5.2: Histogram 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.
locations of the originals but also locations of the virtual images are taken into account.

Intuitively, the virtual image can be constructed by shifting the input domain. Consider, for example, a 2-dimensional square input domain, as shown in Figures 5.3(a) to 5.3(h). The squares with solid lines represent the original input domain with an input range of \( m \), and the solid dots represent a successful test case \((x, y)\). For example, Figure 5.3(a) shows that the input domain is virtually shifted left horizontally by a distance of \( m \). The squares with dashed lines represent the virtual images of the input domain, and the hollow dot represents the virtual images of the successful test case. By this horizontal left shift, a virtual image \((x - m, y)\) is introduced outside the input domain. The 2-dimensional input domain can be shifted along one dimension or both dimensions. Figures 5.3(a) to 5.3(d) show shifts along one dimension whereas Figures 5.3(e) to 5.3(h) show shifts along both dimensions. There are a total of 9 virtual images of the successful test case \((x, y)\). They are \((x - m, y), (x + m, y), (x, y + m), (x, y - m), (x - m, y + m), (x + m, y + m), (x - m, y - m), (x + m, y - m), and (x, y)\). It should be noted that the original test case can also be regarded as an image of itself.
For a k-dimensional input domain, let \( s = s_1, s_2, \ldots, s_k \) be a successful test case and \((m_1, m_2, \ldots, m_k)\) be the ranges of the input domain. Let \( v = v_1, v_2, \ldots, v_k \) be a virtual image of \( s \) that can be computed from its original coordinates and the offset \( f = f_1, f_2, \ldots, f_k \) as follows:

\[ v_i = s_i + f_i. \]

where \( f_i = -m_i, 0, m_i \) for \( i = 1, 2, \ldots, k \). Obviously, in a k-dimensional input domain, there are \( 3^k \) virtual images of a successful test case. In the following sections, we shall elaborate how to incorporate virtual images into original D-ART, R-ART and IP-ART.

### 5.4 The Enhanced Method

#### 5.4.1 Effective Image in D-ART and R-ART

In original D-ART and R-ART methods, the distance computations only covered actual successful test cases. For example, in a k-dimensional input domain, the distance between a successful test case \( s = s_1, s_2, \ldots, s_k \) and a candidate \( c = c_1, c_2, \ldots, c_k \) is calculated as follows:

\[ dist(s, c) = \sqrt{\sum_{i=1}^{k} (s_i - c_i)^2}. \]

The enhanced methods use the effective image of the successful test case rather than the successful test case itself in distance computations. The effective image \( e = e_1, e_2, \ldots, e_k \) of a successful test case \( s \) with respect to candidate \( c \) is defined as the virtual image of \( s \) that has the minimum distance from \( c \). The same successful test case may have different effective images for different candidates. It should be noted that the identification of effective images does not require the computation of the distance between every virtual images and the candidate. On the contrary, if a virtual image has the minimum offset to the candidate \( c \) in each dimension, then this image will automatically have the minimum distance from \( c \). As mentioned before, \( e_i (i = 1, 2, \ldots, k) \) can only have a value of \( s_i, s_i + m_i \) or \( s_i - m_i \).
With respect to candidate \( c \), the minimum offset in the \( i^{th} \) dimension is

\[
\begin{align*}
  s_i - c_i & \quad \text{if } |s_i - c_i| \leq m_i/2 \\
  s_i + m_i - c_i & \quad \text{if } |s_i - c_i| > m_i/2 \text{ and } s_i < c_i \\
  s_i - m_i - c_i & \quad \text{if } |s_i - c_i| > m_i/2 \text{ and } s_i > c_i
\end{align*}
\]

Therefore, we know that the effective image \( e = e_1, e_2, \ldots, e_k \) has the following property:

\[
e_i = \begin{cases} 
  s_i & \text{if } |s_i - c_i| \leq m_i/2 \\
  s_i + m_i & \text{if } |s_i - c_i| > m_i/2 \text{ and } s_i < c_i \\
  s_i - m_i & \text{if } |s_i - c_i| > m_i/2 \text{ and } s_i > c_i
\end{cases}
\]

Consequently, the distance computation in the enhanced methods is changed to

\[
dist(s, c) = \sqrt{\sum_{i=1}^{k} (s_i - e_i)^2}.
\]

As an example of illustration, consider a 2-dimensional square input domain (Figure 5.4). The notions are the same as Figure 5.3 except that the solid triangles represent candidates. For candidate (1), the effective image is \((x + m, y)\), which is the virtual image closest to it. For candidate (2), the effective image is \((x, y - m)\).

### 5.4.2 Enhancement of D-ART and R-ART

D-ART makes use of distance as a gauge to measure whether test cases are far apart from one another and selects the candidate with maximum distance between itself and the successful set as the next test case. However, since no successful test case can be outside the input domain, candidates closer to the boundary are more likely to have a maximum distance from the successful set than candidates closer to the centre of the input domain. Our enhanced D-ART method introduces virtual images of the successful test case and uses effective images in distance computations. Other parts are the same as original D-ART method.

Figures 5.5(a) and 5.5(b) compare the original and the enhanced versions of D-ART in a 2-dimensional input domain. Each of Figures 5.5(a) and 5.5(b) puts the original input do-
Figure 5.4: Virtual images of a successful test case in a 2-dimensional square input domain

main and its 8 images together. The rectangles with solid lines represent the input domain, the solid dots represent the successful test case, and the solid triangles represent the candidates. The rectangles with dashed lines represent the images of the input domain while the hollow circles represent the virtual images of successful test cases. For each candidate in the original D-ART in Figure 5.5(a), only the distance from the original successful test case is calculated and, hence, candidate (2) is selected as the next test case. For each candidate in the enhanced method in Figure 5.5(b), the effective image is identified first. Each dotted line represents the distance between the candidate and its effective image. Suppose candidate (3) has the maximum distance to the effective image of the successful test case comparing with candidates (1) and (2). Then, candidate (3) will be selected as the next test case. As shown in this example, the preference of selecting test cases close to the boundary no longer exists.

Although R-ART has a different test case selection process, both R-ART and D-ART utilize Cartesian distance to measure how far apart test cases are. Hence, similarly to D-ART, candidates close to the boundary of the input domain have a higher chance to be outside all exclusion regions than those close to the centre.
Figure 5.5: Comparing test case selections between (a) the original and (b) the enhanced versions of D-ART

Similar to the enhanced D-ART method, R-ART can be enhanced by using effective images instead of the original successful test cases in judging whether a candidate is outside the exclusion region. As in D-ART, among the images of a successful test case, an effective image is defined as the one closest to the candidate. It is obvious that if a candidate is outside the exclusion region of the effective image, it will be outside the exclusion regions of all other images. Hence, it is only necessary to check whether the candidate is outside the exclusion region of the effective image.

Figures 5.6(a) and 5.6(b-d) compare the original and enhanced versions of R-ART in a 2-dimensional input domain. The notations are the same as those of Figure 5.5, except that circles with dashed lines are used to denote exclusion regions. In the original R-ART shown in Figure 5.6(a), since candidate (1) is outside the exclusion region of the successful test case, it is selected as the next test case. For each candidate in the enhanced method shown in Figure 5.6(b-d), the effective image is identified first. The images shown as dashed circles are the effective images. If a candidate is outside a dashed circle, it is selected as the next test case. Obviously, candidate (3) will be selected as the next test case. As illustrated in this example, the boundary effect will be reduced, if not totally avoided, by our enhanced version.

Time complexity analyses of enhanced D-ART and R-ART are similar to their original
versions. The only extra work in the enhanced versions is to identify the effective images, the runtime of which is a constant for each distance computation. Hence, the time complexities in our enhanced versions are the same as original methods. Both are in the order of quadratic to the number of test cases generated.

5.4.3 Enhancement of IP-ART

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 incorporates 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 Figure 5.7. The square with solid lines represents the original input domain, which is partitioned by a $4 \times 4$ 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.

Virtual adjacent cells within the original input domain can be identified efficiently. Suppose an $k$-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 $1 \leq k$, $0 \leq a_i$, $o_i \leq p - 1$. 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 $(0, 3)$. Note that, this enhanced process of identification of adjacent cells will not increase the time complexity of original IP-ART.

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 next to the boundary have a higher chance to be selected. In the enhanced IP-ART, virtual images of successful test cases are introduced to be outside the input domain. 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 next to the boundary of the input domain, will have a similar chance to be selected for test case generation.

Figures 5.8(a) and 5.8(b) compare the original and enhanced versions of IP-ART in a 2-dimensional input domain. The notations are the same as those of Figure 5.7. In the original IP-ART shown in Figure 5.8(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 5.8(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 Figure 5.8(a).

### 5.5 Simulation Studies

The main objective of these simulations is to answer the following two questions:

- Are the test cases generated by the enhanced methods more evenly spread throughout
Figure 5.8: Comparing test case selections between (a) the original and (b) the enhanced versions of IP-ART

the input domain?

- Are the fault-detection capabilities of enhanced methods better than those of the original methods?

To answer the first question, we repeated the distribution analysis in Section 5.2 for the enhanced D-ART, R-ART and IP-ART. Figure 5.9 shows the histograms for the enhanced D-ART. The histograms for the enhanced R-ART are again omitted because there is no significant difference from those of the enhanced D-ART. It is obvious from the figure that the test cases generated by our enhanced methods are more evenly spread throughout the input domain in all the test suites under study. For the enhanced IP-ART, the distributions have slight fluctuation when $n$ is small due to the partitioning of the input domain as illustrated in 5.10. With the increase of $n$, the test cases become more uniformly distributed.

Secondly, we repeated the controlled failure-causing region simulations for the enhanced methods. It clearly demonstrates that the fault-detection capabilities for the enhanced versions do not depend on the location of the failure regions, as illustrated in Table 5.2.
Figure 5.9: Histograms of the enhanced D-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.
Figure 5.10: 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 5.2: F-measures of differently located failure regions for enhanced D-ART, R-ART and IP-ART under block failure pattern on a 2-dimensional input domain ($\theta = 0.01$)

<table>
<thead>
<tr>
<th>Location of failure-causing input</th>
<th>D-ART</th>
<th>R-ART</th>
<th>IP-ART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>63</td>
<td>63</td>
<td>60</td>
</tr>
<tr>
<td>Centre</td>
<td>62</td>
<td>63</td>
<td>60</td>
</tr>
<tr>
<td>Anywhere</td>
<td>63</td>
<td>63</td>
<td>60</td>
</tr>
</tbody>
</table>

Table 5.3: F-measures of original D-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern)

<table>
<thead>
<tr>
<th>Failure Rate $\theta$</th>
<th>F-measure of RT($F_{RT}$)</th>
<th>2D</th>
<th>3D</th>
<th>4D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$F_{ART}$</td>
<td>$F_{ART} / F_{RT}$</td>
<td>$F_{ART}$</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>67</td>
<td>67%</td>
<td>85</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>132</td>
<td>66%</td>
<td>159</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>323</td>
<td>65%</td>
<td>382</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>648</td>
<td>65%</td>
<td>754</td>
</tr>
</tbody>
</table>

To compare the fault-detection capabilities between the enhanced methods and the original ones, 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. Obviously, the fault-detection capability of the enhanced D-ART, R-ART and IP-ART outperformed the original versions for every combinations of failure rates and dimensionalities of the input domain, as illustrated in Figure 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8.

There are two known observations about the original D-ART and R-ART [25]: (a) With the increase of dimensionality of the input domain, the fault-detection capability decreases dramatically (that is, the F-measure increases). (b) The fault-detection capabilities at lower failure rates are better than that at higher failure rates. These observations are also applicable to original IP-ART. For the enhanced versions, the fault-detection capability also decreases with the increase of the dimensionality of the input domain, but the rate is much moderated. Furthermore, the fault-detection capability appears to be independent of the failure rates. Obviously, the rectification of the boundary effect has significantly improved on the fault-detection capability for D-ART, R-ART and IP-ART.
<table>
<thead>
<tr>
<th>Failure Rate θ</th>
<th>F-measure of RT((FER))</th>
<th>2D</th>
<th>3D</th>
<th>4D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(F_{ART})</td>
<td>(F_{ART}/F_{RT})</td>
<td>(F_{ART})</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>63</td>
<td>63%</td>
<td>69</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>126</td>
<td>63%</td>
<td>137</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>312</td>
<td>62%</td>
<td>346</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>632</td>
<td>63%</td>
<td>680</td>
</tr>
</tbody>
</table>

Table 5.4: F-measures of enhanced D-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern)

<table>
<thead>
<tr>
<th>Failure Rate θ</th>
<th>F-measure of RT((FER))</th>
<th>2D</th>
<th>3D</th>
<th>4D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(F_{ART})</td>
<td>(F_{ART}/F_{RT})</td>
<td>(F_{ART})</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>66</td>
<td>66%</td>
<td>81</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>130</td>
<td>65%</td>
<td>160</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>328</td>
<td>66%</td>
<td>386</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>644</td>
<td>64%</td>
<td>765</td>
</tr>
</tbody>
</table>

Table 5.5: F-measures of original R-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern)

<table>
<thead>
<tr>
<th>Failure Rate θ</th>
<th>F-measure of RT((FER))</th>
<th>2D</th>
<th>3D</th>
<th>4D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(F_{ART})</td>
<td>(F_{ART}/F_{RT})</td>
<td>(F_{ART})</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>66</td>
<td>66%</td>
<td>81</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>130</td>
<td>65%</td>
<td>160</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>328</td>
<td>66%</td>
<td>386</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>644</td>
<td>64%</td>
<td>765</td>
</tr>
</tbody>
</table>

Table 5.6: F-measures of enhanced R-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern)

<table>
<thead>
<tr>
<th>Failure Rate θ</th>
<th>F-measure of RT((FER))</th>
<th>2D</th>
<th>3D</th>
<th>4D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(F_{ART})</td>
<td>(F_{ART}/F_{RT})</td>
<td>(F_{ART})</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>63</td>
<td>63%</td>
<td>82</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>123</td>
<td>62%</td>
<td>159</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>301</td>
<td>60%</td>
<td>380</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>600</td>
<td>60%</td>
<td>739</td>
</tr>
</tbody>
</table>

Table 5.7: F-measures of original IP-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern)
Table 5.8: F-measures of enhanced IP-ART on 2-, 3- and 4-dimensional input domains (under the block failure pattern)

<table>
<thead>
<tr>
<th>Failure Rate ( \theta )</th>
<th>F-measure of RT((F_{RT}))</th>
<th>2D</th>
<th>3D</th>
<th>4D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(F_{ART})</td>
<td>(F_{ART}/F_{RT})</td>
<td>(F_{ART})</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>60%</td>
<td>60%</td>
<td>67%</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>120%</td>
<td>120%</td>
<td>134</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>292%</td>
<td>292%</td>
<td>325</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>583%</td>
<td>583%</td>
<td>649</td>
</tr>
</tbody>
</table>

5.6 Summary

“Far apart” ART methods try to select test cases far apart from one another to achieve an even spread of test cases. However, test case being far apart are not necessarily evenly spread in the input domain. These methods prefer selecting test cases near the boundary of the input domain (known as the boundary effect). This effect adversely affects the fault-detection capability of ART, and the impact grows with the increase of dimensions of the input domain. Moreover, ART as an enhancement of RT should keep the basic characteristic of RT: treating the whole input domain equally. In other words, the fault-detection capability should be independent of the locations of the failure regions. Otherwise, if the failure region is entirely in the area that has low probability to be selected, the performance would be much worse than the average performance.

In this chapter, we have analyzed the cause of the boundary effect and proposed an approach to tackle it in DART, RRT and IP-ART. It is found that the main cause of the boundary effect is no successful test cases can be outside the input domain. Based on the observation, we introduce an innovative concept of virtual images of successful test cases, which makes test cases “step” out of the input domain. Simulation results have clearly indicated that the test cases generated by our enhanced algorithms are more evenly spread throughout the input domain. As a result, the fault-detection capability has also been significantly improved. This improvement is particularly significant for high dimensional input domains. In addition, the enhancement can be implemented in an efficient way, such that the enhanced methods do not increase the time complexity of original methods.
Chapter 6

Adaptive Random Testing by Localization

6.1 Introduction

In previous two chapters, improvements are proposed with regard to the two major shortcomings in “far apart” ART methods. Another category of ART methods is based on Proportional Sampling Strategy (PSS). Here, an even spread of test cases is interpreted as a uniform sampling rate of test cases from all partitions of the input domain. In other words, the number of test cases selected from a partition should be proportional to its size. ART through Dynamic Partitioning (DP-ART) [14] and Lattice-based ART (L-ART) [67] are two PSS ART methods with different partitioning strategies.

This chapter focuses on DP-ART. One shortcoming of DP-ART is that its fault-detection capability is not as good as that of D-ART and R-ART. In this chapter, we will analyze the reasons and propose an enhancement to improve the fault-detection capability of DP-ART [15].

6.2 Unfavorable Scenarios in DP-ART

ART through Dynamic Partitioning (DP-ART) exploits a dynamic way to partition the input domain with the aim of identifying the sparsely populated partition(s). It has two partition-
RP-ART partitions the input domain by the most recently executed successful test case, and selects the partition with the largest size as the test case generation region, from which the next test case is randomly generated. Since the boundaries of every partitions are set according to the successful test cases, selecting the next test case within the largest “blank” partition is most likely to achieve a similar test case density throughout the input domain.

However, we find that the next test case may be very close to the successful test cases, since the successful test cases lie on vertices of the test case generation region. Figure 6.1 illustrates an unfavorable test case generation scenario, where test cases are clustered together. The square represents the input domain, the points represent the test cases and the rectangle with darker edge represents the test case generation region.

ART by Bisection (B-ART) exploits another dynamic partitioning scheme to identify the sparsely populated partition(s). It repeatedly divides the input domain into partitions of equal size, and selects test cases from a partition that has not contained any successful test cases. In this method, there also has a possibility that the test cases are clustered together. As shown in Figure 6.2, the first 4 test cases are clustered in the centre of the input domain.
6.3 Basic Idea of Localization

Introducing distance computations into DP-ART is an effective way to alleviate this unfavorable phenomenon. However, like D-ART and R-ART, distance computations between each candidate and all successful test cases are time consuming. Therefore, this chapter proposes the notion of localization to reduce the extensive computations. By localization, test case generation would be restricted from part of the input domain instead of the whole input domain, and distance computations would be done for some instead of all successful test cases. The following two sections elaborate the enhanced methods for RP-ART and B-ART, respectively.

6.4 RP-ART by Localization

6.4.1 Algorithm Description

Clustered test cases, as shown in Figure 6.1, will compromise the fault-detection capability. Distance is an effective gauge to measure whether the subsequent test cases are far apart from all successful test cases. However, in selecting the next test case, only those successful test cases, which are close to the candidate, should be explored rather than all successful test cases. The partitioning scheme in RP-ART has effectively provided a way to classify the successful test cases. In this thesis, those successful test cases located on the vertices of the test case generation regions are classified as nearby successful test cases. Other test cases are classified as distant successful test cases. Furthermore, the partitioning scheme in RP-ART also tells that test cases from the largest “blank” region may have a higher chance to be far apart from all successful test cases.

Therefore, localization in this thesis has two aspects: the first one is to restrict the selection of test cases from a part of the whole input domain, where test cases are sparsely populated; the second one is to confine distance computations to those successful test cases which are near the test case generation region. Consequently, our enhanced method has two corresponding steps: firstly, localizing the test case generation region and successful test cases; secondly, generating candidates from the restricted test case generation region and
applying D-ART or R-ART with the confined successful test cases.

We use the same partitioning scheme as RP-ART to divide the input domain and choose the partition with largest size as the test case generation region. The test case generation region not only acts as the region which is more likely to provide test cases far apart from successful test cases, but also divides the successful test cases into two sets: nearby successful test cases and distant successful test cases. In the enhanced version, distance computations are done only for the nearby successful test cases.

The algorithm described in Algorithm 6.1 is for testing a program with two real inputs, $i, j$, where $i_{\text{min}} \leq i \leq i_{\text{max}}$, $j_{\text{min}} \leq j \leq j_{\text{max}}$. It is straightforward to apply it in a high dimensional input domain. Each vertex of a subdomain is denoted as $V(x, y, \text{Flag})$. Flag indicates whether it is a successful test case with $T$ being yes and $F$ being no. A subdomain is denoted by its 4 vertices in clockwise order, say $R(V_1, V_2, V_3, V_4)$.

**Algorithm 6.1** RP-ART by localization algorithm

1. Initiate the subdomain list $L$ with the whole input domain $((i_{\text{min}}, j_{\text{min}}, F), (i_{\text{min}}, j_{\text{max}}, F), (i_{\text{max}}, j_{\text{max}}, F), (i_{\text{max}}, j_{\text{min}}, F))$, as its only element.

2. Select the subdomain with largest size $R$ from $L$ as the test case generation region and remove it from $L$. Set the nearby successful test case set $E$ to be empty.

3. Check the Flag of each vertex of $R$. If Flag is $T$, add it to the nearby successful test cases set $E$. Denote the number of elements in $E$ as $l$.

4. For the case of applying D-ART within $R$, randomly generate a candidate set, $C = \{C_1, C_2, \ldots, C_e\}$. Calculate the Cartesian distance between candidates and nearby successful test cases and denote it by $\text{dist}(C_j, E_i)$. Choose a candidate $C_q$ as the next test case according to the following criterion.

   $$\forall j \in \{1, 2, \ldots, e\}(\min_{i=1}^l \text{dist}(C_q, E_i) \geq \min_{i=1}^l \text{dist}(C_j, E_i))$$

For the case of applying R-ART within $R$, set an exclusion zone around each element in $R$. Randomly generate candidates within $R$ one by one, until a candidate is outside all exclusion zones. Then, this candidate is selected as the next test case, $C_q$.

5. If $C_q$ is a failure-causing input, report the failure and terminate. Otherwise, divide $R$ into four test regions at $C_q$ and add them to $L$.

6. Go to step 2.

If applying D-ART, the test case generation process is illustrated in Figure 6.3. At first,
since the whole input domain is the only element in the subdomain list, it is obviously the test case generation region. Since there are no nearby successful test cases, candidate generations and distance computations are not required. Within the test case generation region, a test case is randomly generated. Suppose this is not a failure-causing input (denoted as $t_1$).

Then, the test case generation region is further divided by $t_1$ into four subdomains (Region 1, 2, 3 and 4). As shown in Figure 6.3, Region 4 is the largest region, therefore it becomes the next test case generation region. Since $t_1$ is located in a vertex of the test case generation region, it is regarded as a nearby successful test case. Then, 4 candidates, $c_1$, $c_2$, $c_3$, $c_4$, are randomly generated within Region 4. In this case, candidate $c_4$ is the farthest from $t_1$, and therefore is chosen as the next test case, $t_2$. Suppose that no failure is detected. Region 4 is further divided into four regions. The algorithm is continued until a failure is detected or resources are exhausted.

If we apply R-ART in the test case generation region (Figure 6.4), an exclusion zone is set around the nearby successful test case $t_1$, rather than generating a set of candidates. Other steps are the same as applying D-ART. Suppose the first candidate, $c_1$, is located within the exclusion zone, then it is discarded. The second test case candidate, $c_2$, is outside the exclusion region and hence designated as the next test case, $t_2$. Since $t_2$ is not a failure-causing input, region 4 is further partitioned.

Comparing with original RP-ART, the overhead of partitioning the input domain and choosing the test case generation region in enhanced method are the same. The extra work is mainly the application of D-ART or R-ART in the test case generation region.

In RP-ART by localization, the possible maximum number of nearby test cases is 2. It is not related to the number of successful test cases or the dimensionality of the input domain, since a successful test case divides every dimensions into 2 segments and the subsequent subdomains are always built on previous ones. If applying D-ART, the number of distance computations in an iteration is at most $2e$, where $e$ is the size of candidate set. Then, the maximum number of distance computations is

$$\sum_{i=1}^{n} 2e = 2e \cdot n,$$
where \( n \) is the number of test cases generated. As discussed in Section 4.2.1, it is reasonable to assume the number of candidates generated in R-ART as a constant. Hence, for RP-ART by localization the total number of distance computations is linear to the F-measure, but it is quadratic for D-ART and R-ART.

### 6.4.2 Simulation Studies

We have conducted a series of simulations using a 2-dimensional square input domain to investigate the fault-detection capability of our enhanced method. In each test run, a failure region of the specified size and pattern was randomly assigned within the input domain. For block failure pattern, a square was used as the failure region. For strip failure pattern, we randomly chose two points on the adjacent borders of the input domain. These two points were connected to form a strip with the specified size. For point failure pattern, 10 equally sized circular regions were randomly located in the input domain without overlapping each other.

The first part of our simulation investigated the fault-detection capability of RP-ART
Figure 6.5: Comparison of the F-measures of RT and RP-ART by localization with D-ART with different numbers of candidates on a 2-dimensional input domain (under the block failure pattern, $\theta=0.001$)

by localization with D-ART under different candidate set sizes ranging from 1 to 30. If
the size of the candidate set is 1, then this algorithm is effectively the original RP-ART. As
shown in Figure 6.5, the lowest F-measure occurs when the size of candidate set is equal
to 3. The lowest F-measure is less than that of RT by about 30%; while original RP-ART
is less than RT by about 20%. When the number of candidates exceeds 3, F-measure will
begin to increase and is greater than that of RT at the size of 30. However, as studied in [63],
with D-ART in the whole input domain, F-measure will decrease with the increase of the
number of candidates and become steady when the number is about 10. This discrepancy
can be explained as follows. When we apply ART by localization with D-ART and when
the candidate set size is large, it is most likely that the next test case will be closer to a
corner of the test case generation region. As a consequence, test cases will be clustered in
narrow strips rather than evenly spread, as shown in Figure 6.6.

The fault-detection capability of RP-ART by localization with R-ART was investigated
in the second part of the simulation. At first, we would like to elaborate on the size of
exclusion zone. Intuitively speaking, the size of exclusion zone should be determined by
the size of the test case generation region. If the test case generation region is larger, then
the radius of exclusion zones should be larger. For convenience, we use $Exclusion\ Ratio$ to describe the ratio of the radius of exclusion zones to the diagonal of the test case
Figure 6.6: Test case generation patterns for RP-ART by localization with D-ART under large number of candidates (the number represents the test case generation sequence)

generation region. As discussed before, for a test case generation region, the maximum number of nearby successful test cases is 2. If there are 2 nearby successful test cases, they must be diagonal to each other. With the increase of Exclusion Radius Ratio, the regions covered by exclusion zones are increasing and the number of attempts to generate next test cases is growing fast. When Exclusion Radius Ratio approaches 50%, the test case generation region will be largely covered by exclusion zones, especially in a narrow test case generation region. To limit the number of trials, we varied Exclusion Radius Ratio between 0% and 40% in our simulations. Again, when the Exclusion Radius Ratio is 0%, our method is effectively the original RP-ART. As can be seen from Figure 6.7, F-measure decreases while Exclusion Radius Ratio increases. At 40% of Exclusion Radius Ratio, the F-measure of our enhanced method is less than that of RT by about 30%, while the F-measure of RP-ART is less than that of RT by about 20%.

In the third part of the simulation, we conducted our simulation using different failure patterns. Table 6.1 shows the results against other ART methods. For block and strip failure patterns, our enhanced algorithm performs better than original RP-ART but worse than D-ART and R-ART in terms of F-measure. For point patterns the differences of all methods are insignificant.

In the last part of our simulations, we investigated whether the locations of failure regions would have any impact on the fault-detection capability of our methods. Following
Figure 6.7: Comparison of the F-measures of RT and RP-ART by localization with R-ART with different exclusion radius ratios on a 2-dimensional input domain (under the block failure pattern, θ=0.001)

<table>
<thead>
<tr>
<th>ART Methods</th>
<th>Failure Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Block</td>
</tr>
<tr>
<td>RT</td>
<td>1000</td>
</tr>
<tr>
<td>D-ART</td>
<td>633</td>
</tr>
<tr>
<td>R-ART</td>
<td>621</td>
</tr>
<tr>
<td>RP-ART</td>
<td>797</td>
</tr>
<tr>
<td>RP-ART by Localization with D-ART</td>
<td>695</td>
</tr>
<tr>
<td>RP-ART by Localization with R-ART</td>
<td>692</td>
</tr>
</tbody>
</table>

Table 6.1: Comparison of F-measures of RP-ART by localization and other ART methods on a 2-dimensional input domain (θ=0.001)

the simulation settings in Section 5.2, the categories of locations of failure regions were defined as follows: the centre area (“Centre”) was defined as the central 80% of the whole input domain and the other area were defined as edge area (“Edge”). In our simulations, the failure region was randomly assigned in anywhere of the input domain (“Anywhere”) or confined to specified area (“Edge” or “Centre”). A block failure region with failure rate of 0.01 was used. The result shows that RP-ART by localization has no preference to the types of locations (see Table 6.2).
Table 6.2: F-measures of differently located failure regions for RP-ART with D-ART, or with R-ART on a 2-dimensional input domain (under the block failure pattern, $\theta = 0.01$)

<table>
<thead>
<tr>
<th>Locations of failure region</th>
<th>RP-ART by Localization with D-ART</th>
<th>with R-ART</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Edge</td>
<td>71</td>
<td>67</td>
</tr>
<tr>
<td>Centre</td>
<td>70</td>
<td>67</td>
</tr>
<tr>
<td>Anywhere</td>
<td>69</td>
<td>68</td>
</tr>
</tbody>
</table>

6.5 B-ART by Localization

6.5.1 Algorithm Description

Inspired by RP-ART by localization [15], Mayer applied the notion of localization to improve the fault-detection capability in B-ART [66]. For the completeness of the thesis, we briefly summarize the method and cite the simulation results here.

Like RP-ART, the partitioning scheme in B-ART not only restricts the test case generation region, but also differentiates the successful test cases as nearby successful test cases and distant successful test cases. Nearby successful test cases are those test cases that reside in the neighbour subdomains of the test case generation region. In a 2-dimensional input domain, neighbour subdomains share a common side with the test case generation region. For a test case generation region, there are at most 4 neighbours, say the left, right, upper and lower neighbours. Other successful test cases are regarded as distant successful test cases. To prevent test cases from clustering together, D-ART and R-ART are introduced in the enhanced method, but only applied to the nearby successful test cases.

For the case of applying D-ART, a set of candidates are randomly generated from “blank” partitions. For each candidate, the minimal distance to its nearby successful test cases is computed. The candidate that maximize such minimal distance is selected as the next test case. For the case of applying R-ART, candidates are randomly generated from “blank” partitions one by one. Again, the method only cares for the nearby successful test cases. If a candidate is outside all exclusion zones of its nearby successful test cases, it is selected as the next test case.

Note that in a 2-dimensional input domain, a test case generation region has at most 4
Figure 6.8: An example that illustrates B-ART by localization with D-ART

nearby successful test cases, since it has at most 4 neighbours and a neighbour contains at most 1 successful test case. Hence, the maximum number of distance computations is

\[ \sum_{i=1}^{n} 4e = 4e \cdot n \]

when applying D-ART, where \( e \) denotes the size of candidate set in the candidate set, \( n \) denotes the number of test cases generated. Again, the number of candidates generated in R-ART is assumed to be a constant. Then the number of distance computations is \textit{linear} to the F-measure for B-ART by localization.

An example of test case generation process is illustrated in Figure 6.8 and Figure 6.9. Successful test cases are denoted as solid circle, while candidates are denoted as solid triangle. Subdomains with darken border denote the untested regions. When applying D-ART, the dashed lines denotes distance computations. When applying R-ART, the dashed circles denotes the exclusion regions.

6.5.2 Simulation Studies

Following the simulations for RP-ART by localization, a series simulations were conducted in a 2-dimension square input domain [66]. The first part of the simulation is to investigate
the fault-detection capability of B-ART by localization with D-ART using different candidate set sizes ranging from 1 to 15. If $k = 1$, this algorithm is effectively original B-ART. As shown in Figure 6.10, the best fault-detection capability is reached when $k = 13$, which is about 10% lower than that of original B-ART and 37% lower than that of random testing.

The second part of the simulation is to investigate the fault-detection capability of B-ART by localization with R-ART using different target exclusion ratio $r$ ranging from 0 to 0.7. If $r = 0$, this algorithm is effectively original B-ART. As shown in Figure 6.11, the F-measure is decreasing with the increase of $r$. When $r = 0.7$, the F-measure is about 7% lower than that of original B-ART and about 34% lower than that of random testing.

The third part of the simulation compares the fault-detection capability of B-ART by localization with other ART methods for various failure patterns (Table 6.3). For block failure pattern, the F-measure of the enhanced algorithm is much lower than that of B-ART but comparable with that of D-ART and R-ART. For strip and point patterns, the fault-detection capability is similar to that of B-ART and worse than that of D-ART and R-ART.
Figure 6.10: Comparison of the F-measures of RT and B-ART by localization with D-ART with different numbers of candidates on a 2-dimensional input domain (under the block failure pattern, θ=0.001)

Figure 6.11: Comparison of the F-measures of RT and B-ART by localization with R-ART with different exclusion ratios on a 2-dimensional input domain (under the block failure pattern, θ=0.001)

Table 6.3: Comparison of F-measures of B-ART by localization and other ART methods on a 2-dimensional input domain (θ=0.001)

<table>
<thead>
<tr>
<th>ART Algorithms</th>
<th>Failure Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Block</td>
</tr>
<tr>
<td>RT</td>
<td>1000</td>
</tr>
<tr>
<td>D-ART</td>
<td>633</td>
</tr>
<tr>
<td>R-ART</td>
<td>621</td>
</tr>
<tr>
<td>B-ART</td>
<td>743</td>
</tr>
<tr>
<td>B-ART by Localization with D-ART</td>
<td>636</td>
</tr>
<tr>
<td>B-ART by Localization with R-ART</td>
<td>663</td>
</tr>
</tbody>
</table>
6.6 Summary

PSS ART methods interpret an even spread of test cases as the number of test cases in a partition should be proportional to the size of that partition. Based on this interpretation, DP-ART and L-ART are proposed with different partitioning strategies. This chapter has focused on DP-ART. Compared with D-ART and R-ART methods, the overhead of DP-ART is very low, but the fault-detection capability is also compromised. As analyzed, the reason behind is that test cases still have chances to be clustered together. In this chapter, we propose to incorporate distance computations into the original method to alleviate this unfavorable phenomenon. However, a straightforward application of distance computations is time consuming. Therefore, the notion of localization is employed to reduce the overhead in distance computations.

Compared with original methods, the fault-detection capability has obviously been improved. The major extra computations in enhanced methods are to apply D-ART or R-ART in a restricted region with reference to a selected set of successful test cases. Since the number of nearby successful test cases is a constant, the number of distance computations of the enhanced methods is linear to the number of test cases generated, while it is quadratic for D-ART and R-ART. Compared with D-ART and R-ART, the number of distance computations and comparisons is considerably reduced while fault-detection capability is only moderately affected in terms of F-measure. In our method, distance computations are only associated with the nearby successful test cases, but in D-ART and R-ART, distance computations are associated with all successful test cases.

The shortcoming of our enhanced method is that the identification of nearby successful test cases is not complete. Some successful test cases, which are close to the test case generation region, are still left unidentified. This explains why the fault-detection capability of our methods is slightly worse than that of D-ART or R-ART.
Chapter 7

Enhanced Lattice-based Adaptive Random Testing

7.1 Introduction

Lattice-based Adaptive Random Testing (L-ART) [67] is another PSS ART method with good fault-detection capability and low overhead (linear to the number of test cases generated). From the test cases spatial distribution of L-ART, this chapter finds that to achieve a good fault-detection capability this method has a strong preference in selecting test cases in some locations. Because of this preference, the fault-detection capability of L-ART is sensitive to the locations of failure-causing region. For a faulty Software Under Test (SUT), the size and position of the failure region is fixed but unknown. If the failure region of a SUT happens to be entirely located in a region that has a low density of test cases, the F-measure of L-ART for this SUT will be much higher than others. The fault-detection capability of L-ART becomes dependent on the locations of the failure regions. In practice, it is crucial to employ a testing method with predictable fault-detection capability. Therefore, it is necessary that the test cases have a uniform spatial distribution. In this chapter, we firstly analyze the cause of such an uneven spread of test cases and then propose an enhancement of the L-ART method [17].
7.2 A Review of L-ART

7.2.1 The Original Algorithm

Basically, original L-ART has two steps. Firstly, a lattice structure is formed in the input domain and lattice nodes are systematically placed in the input domain. Secondly, test cases are formed by randomly selecting and “shaking” the lattice nodes.

The original algorithm is illustrated in Algorithm 7.1. The lattice structures are formed by a set of vertical and horizontal lines. Let $m$ be the resolution of the lattice structure, which indicates that each dimension of the input domain is partitioned into $m$ equally sized segments by these vertical and horizontal lines. Lattice nodes are systematically placed on the intersections of these vertical and horizontal lines. For ease of presentation, an index is used to refer to an intersection. For example (Figure 7.1 (b)), in a 2-dimensional input domain, where $m = 4$, any intersections can be indicated by an index $(i, j)$, where $i, j \in 1, 2, \ldots, m - 1$.

Test cases are formed by randomly selecting and “shaking” a lattice node. “Shaking” means the lattice node is shifted by a random shaken vector $(rs_x, rs_y)$. The range of the shaken vector can be represented as $[-f \cdot s, f \cdot s]$, where $s$ is the current lattice spacing, which is defined as $M/m$ and $f \in [0, 1]$ is the shaken factor. Therefore, for each lattice node, there is a corresponding shaken region, from which test case is generated. Let the notation $\{(a, b)(c, d)\}$ to describe a rectangle with vertices at $(a, b)$, $(a, d)$, $(c, d)$ and $(c, b)$ in clockwise order. Then, the shaken region corresponding to a lattice node $(i, j)$ can be represented as $\{(-f \cdot s + i \cdot s, -f \cdot s + j \cdot s)(f \cdot s + i \cdot s, f \cdot s + j \cdot s)\}$.

Initially, the resolution of the lattice structure $m$ is set to be 2. If failures cannot be detected under current lattice structure, a new lattice structure will be formed with the doubled resolution $m$.

An example of the test case generation process of L-ART is illustrated in Figure 7.1, where the outer square represents the input domain; the intersections of the dashed lines are the possible points to place lattice nodes; the shaded areas represent the shaken regions; the solid circles represent the test cases. At the first iteration, $m$ is set to be 2, and there is
Algorithm 7.1 Original L-ART algorithm

It is assumed that the program under test is \( \text{prog}(x, y) \), where \( x, y \) are real numbers and \( 0 \leq x, y < M \).

1. Initialize the resolution of lattice structure \( m \) to be 2, and the current lattice spacing \( s = \frac{M}{m} \).

2. Let \( O = \{1, 3, \ldots, m - 1\} \) and \( E = \{2, 4, \ldots, m - 2\} \).
   Generate a list, \( \text{OddOnly} \), to store the index \( (i, j) \), where \( (i, j) \in O^2 \).
   Generate a list, \( \text{OddEven} \), to store the index \( (i, j) \), where \( (i, j) \in (O \cup E)^2 \setminus O^2 \setminus E^2 \).

3. while (list \( \text{OddOnly} \) or \( \text{OddEven} \) is not empty)
   
   {  
   if(\( \text{OddOnly} \) is not empty)  
   Randomly select an index \( (i, j) \) from \( \text{OddOnly} \),  
   Remove it from the list.  
   else  
   Randomly select an index \( (i, j) \) from \( \text{OddEven} \),  
   Remove it from the list.  
   Randomly choose a vector \( (rsx, rsy) \), where \( -f \cdot s < rsx, rsy < f \cdot s \)  
   Take \( t = (i\cdot s + rsx, j\cdot s + rsy) \) as the next test case.  
   If \( t \) reveal a failure, terminate.  
   }

4. Set \( m = 2m \) and \( s = \frac{M}{m} \).

5. Proceed with Step 2 until testing resources are exhausted.
only one lattice node with index of (1, 1) at the centre of the input domain (Figure 7.1 (a)). The lattice node is shifted by a random vector within the shaken region to be the first test case $t_1$. For the second iteration, $m$ is doubled to 4, there are 9 intersections of the vertical and horizontal lines. Among them, 8 intersections are placed with lattice nodes. The lattice node was not placed on intersection (2, 2) \(^1\), because this intersection was used in the first iteration. To make test cases far apart from one another, the selection of lattice nodes is processed in two steps. The lattice nodes with odd indices only, namely (1, 1), (1, 3), (3, 1) and (3, 3), are selected one by one in a random order in the first step (Figure 7.1 (b)). Each of them is randomly shaken within the shaken region as a test case, that is, $t_2$, $t_3$, $t_4$, $t_5$. After that, lattice nodes with one odd index and one even index, namely (1, 2), (2, 1), (2, 3) and (3, 2), are selected randomly in the second step (Figure 7.1 (c)). Test cases, $t_6$, $t_7$, $t_8$, $t_9$, are formed by shaking these lattice nodes. Figure 7.1 (d) illustrated the first step of the third iteration. The process will be continued until failure is detected or resources are exhausted.

Simulations were conducted to investigate the fault-detection capability of L-ART on different shaken factors $f$ [67]. Table 7.1 lists relative F-measure for block failure pattern on a 2-dimensional input domain. As observed, increasing the randomness (adopting larger $f$) has an adverse impact on the fault-detection capability.

### 7.2.2 Test Case Spatial Distribution of L-ART

Simulation results have showed that L-ART with a smaller shaken factor $f$ has a better fault-detection capability (smaller relative F-measure). However, with a smaller shaken factor $f$, original L-ART forces test cases to concentrate on some parts of the input domain. This phenomenon can be demonstrated by the test case spatial distribution. The simulation settings are the same as those in Section 5.2. The distributions of the first $n$ test cases were recorded, $n = 10, 50, 100, 500$. Since L-ART treats every dimension independently, the positions of the test cases are projected onto one dimension. We conducted the simulation in a 2-dimensional square input domain with side length of 1. The test case distributions in one dimension are illustrated as histograms with equal bins of size 0.01, consisting of

\(^1\)the index was (1, 1) in the first iteration
Figure 7.1: An example that illustrates the original L-ART
Table 7.1: Relative F-measures of original L-ART (under the block failure pattern)

<table>
<thead>
<tr>
<th>f</th>
<th>0.01</th>
<th>0.005</th>
<th>0.002</th>
<th>0.001</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>55%</td>
<td>50%</td>
<td>59%</td>
<td>48%</td>
</tr>
<tr>
<td>0.05</td>
<td>56%</td>
<td>50%</td>
<td>59%</td>
<td>50%</td>
</tr>
<tr>
<td>0.1</td>
<td>55%</td>
<td>51%</td>
<td>60%</td>
<td>54%</td>
</tr>
<tr>
<td>0.15</td>
<td>55%</td>
<td>54%</td>
<td>61%</td>
<td>58%</td>
</tr>
<tr>
<td>0.2</td>
<td>56%</td>
<td>56%</td>
<td>61%</td>
<td>59%</td>
</tr>
<tr>
<td>0.25</td>
<td>57%</td>
<td>59%</td>
<td>64%</td>
<td>64%</td>
</tr>
<tr>
<td>0.3</td>
<td>60%</td>
<td>63%</td>
<td>66%</td>
<td>65%</td>
</tr>
<tr>
<td>0.35</td>
<td>63%</td>
<td>65%</td>
<td>68%</td>
<td>70%</td>
</tr>
<tr>
<td>0.4</td>
<td>66%</td>
<td>68%</td>
<td>69%</td>
<td>72%</td>
</tr>
<tr>
<td>0.45</td>
<td>68%</td>
<td>69%</td>
<td>73%</td>
<td>75%</td>
</tr>
<tr>
<td>0.5</td>
<td>71%</td>
<td>71%</td>
<td>75%</td>
<td>76%</td>
</tr>
<tr>
<td>0.55</td>
<td>70%</td>
<td>74%</td>
<td>77%</td>
<td>78%</td>
</tr>
<tr>
<td>0.6</td>
<td>72%</td>
<td>75%</td>
<td>80%</td>
<td>80%</td>
</tr>
<tr>
<td>0.65</td>
<td>75%</td>
<td>77%</td>
<td>80%</td>
<td>80%</td>
</tr>
<tr>
<td>0.7</td>
<td>75%</td>
<td>78%</td>
<td>83%</td>
<td>81%</td>
</tr>
<tr>
<td>0.75</td>
<td>76%</td>
<td>80%</td>
<td>80%</td>
<td>83%</td>
</tr>
<tr>
<td>0.8</td>
<td>79%</td>
<td>80%</td>
<td>82%</td>
<td>83%</td>
</tr>
<tr>
<td>0.85</td>
<td>82%</td>
<td>81%</td>
<td>85%</td>
<td>86%</td>
</tr>
<tr>
<td>0.9</td>
<td>83%</td>
<td>82%</td>
<td>82%</td>
<td>87%</td>
</tr>
<tr>
<td>0.95</td>
<td>82%</td>
<td>86%</td>
<td>87%</td>
<td>87%</td>
</tr>
<tr>
<td>1.00</td>
<td>83%</td>
<td>84%</td>
<td>87%</td>
<td>88%</td>
</tr>
</tbody>
</table>

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. A million independent trials are conducted. Figure 7.2 compares histograms of test case spatial distribution with various $f$. Here, we selected $f = 0.05$ as a representative for small shaken factor and $f = 0.5$ for large shaken factor.

Figure 7.2 shows that the test case distribution of $f = 0.05$ is highly skewed. Some parts of the input domain become test case favorable regions, where test cases are highly concentrated. Some parts become test case unfavorable regions, where just a few or even no test cases are generated. Although in later test case generation stages (larger $n$), the gap of test case densities between favorable and unfavorable regions is reduced, there are still some regions that contain no test cases. For example, as shown in Figure 7.2, when $n = 1000$, the region from 0.08 to 0.09 contains no test cases. On the contrary, the region from 0.09 to 0.10 contains more than 30,000 test cases. Clearly, in this case, the F-measure for a SUT with
failure regions located in favorable regions is much less than that with failure regions located in unfavorable regions. Since the locations of failure regions are unknown, professional testers would prefer that fault-detection capability of a testing method is predicable, and independent of the locations of failure regions, rather than a testing method, whose fault-detection capability is dependent on the locations of failure regions.

When $f$ is increased to 0.5, the distribution is more even. Although favorable and unfavorable regions also exist, the gaps of test case density between favorable and unfavorable regions are smaller.

7.3 The Enhanced Method

From the review of original L-ART, we found that it faces a dilemma in choosing a suitable value of shaken factor $f$. If choosing a smaller $f$, the fault-detection capability will increase but the spatial distribution of test cases is highly skewed. If choosing a larger $f$, the highly skewed distribution is alleviated but fault-detection capability will be compromised. The objective of our improvement is to alleviate the bias of test case distribution while keeping the good fault-detection capability. Therefore, we aim to improve the fault-detection capability when the method adopts a large shaken factor $f$. At first, the reason of compromised fault-detection capability with large shaken factors is analyzed.

7.3.1 Unfavorable Scenario in Original L-ART with Large Shaken Factors

Essentially, the principle of placing the lattice nodes in original L-ART can be interpreted as Proportional Sampling Strategy (PSS) [28], which requires the number of test cases in a partition should be proportional to its size. In a test case generation iteration, the lattice nodes are evenly placed in the input domain. Although the actual test cases are formed by shaking the lattice nodes, test cases should have similar density all over the input domain.

However, with the increase of shaken factor, shaken regions will be frequently overlapped, and the probability of test cases being clustered together is increasing. Figure 7.3 illustrates an unfavorable scenario of original L-ART, where shaded regions represent shaken regions, and the shaken factor $f$ is set to be 0.4. In the first iteration, where resolution of the
Figure 7.2: Histogram of original L-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.
lattice structure \( m \) is set to be 2, a test case \( t_1 \) is randomly selected within the shaken region, as shown (Figure 7.3 (a)). Since a large \( f \) is adopted, the shaken regions of the second iteration \( (m = 4) \) are overlapped with the shaken region of the first iteration. Hence, the chances of test cases being clustered together are increasing. As illustrated in Figure 7.3 (b), test cases \( t_1 \) and \( t_2 \) are clustered together. This unfavorable scenario explains the reason why fault-detection capability of original L-ART is decreasing with the increase of the shaken factor \( f \), as illustrated in Table 7.1.

### 7.3.2 The Enhanced Algorithm

To reduce the probability of test cases being clustered together, we propose to add a checking process before generating a test case in the selected shaken region. If there are successful test cases located within the selected shaken region, subsequent test case cannot be selected from this shaken region since new test case has a higher chance to be close to the successful test cases. Otherwise, a test case is randomly generated within the shaken region.

A straightforward application of the checking process will lead the runtime in the order of quadratic to number of test cases generated, since the locations of all successful test cases need to be explored for each new test case. In this chapter, we adopt a flexible partitioning scheme similar to that in IP-ART [21] to facilitate the checking process. The input domain is partitioned into equally sized grid cells, such that the shaken region occupies exactly \( w \)
grid cells, where \( w \) is an integer. We name those grid cells consisting a shaken region as \textit{shaken grid cells}. Each successful test case is mapped into a grid cell. Now, the checking process has transformed to checking whether the shaken grid cells contain successful test cases. Since the number of shaken grid cells for a shaken region is fixed, the runtime for this checking process becomes a constant.

In the following, we shall analyze how to set the size of grid cells, such that the shaken region can occupy exactly \( w \) grid cells. For ease of presentation, the analysis is conducted in a 2-dimensional square input domain with side length of \( M \) in each dimension. Let \( p \) be the resolution of the grid cells, which means the input domain is equally partitioned into \( p \times p \) grid cells and \( m \) be the resolution of lattice structure. Each grid cell can be referred to by using its coordinates \((u, v)\), where \( u, v \in \{0, 1, 2, \ldots, p-1\} \). Suppose shaken factor \( f \) can be represented as a fraction in lowest terms, \( \frac{h}{g} \), where \( h \) and \( g \) are integers. If \( p \) is set to be \( m \times g \) and updated with the refining of the lattice structure, a shaken region will reside in exactly \( w = 2h \times 2h \) grid cells.

The reason is explained as follows. Let \( c \) be the side length of each grid cells, then \( c = M/p \). Let \( s \) be the lattice spacing, then \( s = M/m \). Since \( p = m \cdot g \), then \( c = M/(m \cdot g) = \frac{1}{g} \cdot \frac{M}{m} = \frac{s}{g} \), which means the side length of each grid cells \( c \) is \( 1/g \) of the lattice spacing \( s \). As mentioned in Section 7.2, the range of shaken vector \((\text{rsx}, \text{rsy})\) can be represented as \([-f \cdot s, f \cdot s]\). Since \( f \) can be represented as \( \frac{h}{g} \) and \( s = c \cdot g \), the range can be transformed as \([-h \cdot c, h \cdot c]\). Hence, the range of shaken vector is exactly \( w = 2h \times 2h \) grid cells.

Because of the relationship between the resolution of grid cell \( p \) and the resolution of lattice structure \( m \), a lattice node \((i, j)\) should lie on the common vertex of grid cells \((i \cdot g - 1, j \cdot g - 1), (i \cdot g - 1, j \cdot g), (i \cdot g, j \cdot g - 1), (i \cdot g, j \cdot g)\). If a test case \( t \) is generated by selecting and shaking lattice node \((i, j)\), it is equivalent to generate \( t \) within the grid cells \((u, v) \in \{(i \cdot g + a, j \cdot g + b) \mid a, b \in \{-h, -h + 1, \ldots, h - 2, h - 1\}\} \).

For example, Figure 7.4 illustrates a 2-dimensional input domain with \( m = 4 \). The darked vertical and horizontal lines denote the \( 4 \times 4 \) lattice structure. Suppose shaken factor \( f \) is 0.4, which can be represented by a fraction in lowest terms, \( 2/5 \), and hence, \( h = 2, g = 5 \). The resolution of the grid cell \( p \) is set to be \( m \times g = 4 \times 5 = 20 \). The input domain is equally
partitioned into $20 \times 20$ grid cells, which are represented by dashed lines. Suppose the next test case is generated by randomly shaking the lattice node $(i, j) = (1, 3)$, illustrated by a solid node. In the enhanced algorithm, this test case is randomly generated within the 16 shaken grid cells $(u, v)$ with $u \in 3, 4, 5, 6$ and $v \in 13, 14, 15, 16$, illustrated by shaded squares.

The enhanced algorithm illustrated in Algorithm 7.2 is for a 2-dimensional input domain. It is straightforward to apply in a high dimensional input domain. The successful test cases are stored in a set of successful test cases $S$. A boolean matrix $GridCells$ is used to indicate whether the grid cells contain successful test cases. If a grid cell contains any test cases, the corresponding entry of $GridCells$ will be assigned a value of $F$; otherwise it will be assigned a value of $T$. For a shaken region, if any of its corresponding entries of $GridCells$ is $F$, then the shaken region contains successful test cases and next test case cannot be generated in this region.
### Algorithm 7.2 Enhanced L-ART algorithm

It is assumed that the program under test is \( \text{prog}(x, y) \), where \( x, y \) are real numbers and \( 0 \leq x, y < M \).

1. Initialize the resolution of the lattice structure \( m \) to be 2. Set the list of successful test cases \( S \) to be empty.

2. Initialize the resolution of the grid cells \( p \) to be \( m \times g \). Construct a \( p \times p \) Boolean matrix, GridCells, and assign \( T \) to all its entries.

3. Map each successful test case \( (x, y) \) in \( S \) into a grid cell by assigning \( F \) to the corresponding cell \( ([x \times p/M], [y \times p/M]) \).

4. Let \( O = \{1, 3, \ldots, m-1\} \) and \( E = \{2, 4, \ldots, m-2\} \).
   - Generate a list, OddOnly, to store the index \((i, j)\), where \((i, j) \in O^2\).
   - Generate a list, OddEven, to store the index \((i, j)\), where \((i, j) \in (O \cup E)^2 \setminus O^2 \setminus E^2\).

5. while (list OddOnly or OddEven is not empty)
   
   \{ 
   
   if(OddOnly is not empty)
   
   Randomly select an index \((i, j)\) from OddOnly,
   
   Remove it from the list.
   
   else
   
   Randomly select an index \((i, j)\) from OddEven,
   
   Remove it from the list.
   
   Set the shaken grid cells of the shaken region.
   
   Let \((u, v)\) be a coordinate of these grid cells.
   
   \((u, v) \in \{(i \cdot g + a, j \cdot g + b) \mid a, b \in \{-h, -h+1, \ldots, h-2, h-1\}\} \).
   
   If any of the shaken grid cells \((u, v)\) is occupied
   
   Proceed with Step 5;
   
   Randomly generate a test case \( t \) within the shaken grid cells as the next test case.
   
   If \( t \) reveal a failure, terminate.
   
   otherwise, add \( t \) to \( S \)
   
   \}

6. Discard (release) the Boolean matrix GridCells.

7. Set \( m = 2m \) and \( p = m \times g \).

8. Proceed with Step 2 until testing resources are exhausted.
7.3.3 Time Complexity Analysis

Since the checking process has transformed to checking whether the shaken grid cells contain successful test cases, and the shaken region resides in exactly $w = 2h \times 2h$ grid cells, the runtime of this checking process becomes a constant for each test case generation. The extra cost of this algorithm is to map the successful test cases into corresponding grid cells after each refining of the lattice structure.

The following analyzes the runtime of mapping the successful test cases. Let $k$ be the dimensionality of the input domain, $q$ be the resolution of current lattice structure (that is, each dimension is partitioned into $q$ equally sized segments). In such a lattice structure, there are $(q - 1)^k$ intersections of the lines partitioning the input domain. For a lattice structure of resolution $q$, lattice nodes are placed on all the intersections except those intersections that are used in previous iterations. Therefore, when refining a lattice structure from resolution $q$ to $q + 1$, there are at most $(q - 1)^k$ successful test cases. Note that, with smaller $f$ the number of successful test cases will be closer to $(q - 1)^k$, since the probability of shaken regions being overlapped together is low with smaller $f$. For example, as shown in Figure 7.1 (b-c), when $q = 4$ in a 2-dimensional input domain, the number of intersections is $3 \times 3 = 9$.

Lattice nodes are placed on all the intersections except $(2, 2)$, which has been used in the first iteration. In this lattice structure, that are at most $(q - 1) \times (q - 1) = 3 \times 3 = 9$ test cases.

Suppose the F-measure is $n$, the final resolution of the lattice structure is $m$. Since $m$ is doubled in each repartitioning, it can be denoted as $2^r$, where $r$ is an integer and $r \geq 1$. Since the number of test cases generated cannot exceed the number of intersections of the lattice structure, and since refining the lattice structure only occurs when all placed lattice nodes are used up, the relation of $m$ and $n$ can be represented as $(m/2 - 1)^k < n < (m - 1)^k$. Hence, for a given $n$, the final resolution of the lattice structure $m$ should be less than $2(\sqrt[n]{n} + 1)$.

Therefore, the number of mapping of successful test cases can be represented as

$$(2^1 - 1)^k + (2^2 - 1)^k + (2^3 - 1)^k + \ldots + (2^r - 1)^k <$$
\[ 2^k + 2^2 \cdot k + 2^3 \cdot k + \ldots + 2^r \cdot k = \]

\[ 2^r k (\frac{1}{2(r-1)k} + \frac{1}{2(r-2)k} + \frac{1}{2(r-3)k} + \ldots + \frac{1}{2(r-r)k}) < 2 \cdot (2^r)^k = 2^{m^k} < 2 \cdot (2(\sqrt{n}+1))^k. \]

From the above equation, we can conclude that the time complexity of the extra cost of mapping successful test cases in the enhanced algorithm is in \( O(n) \). Since the time complexity of original L-ART is also linear to the number of test cases generated [67], the enhanced L-ART does not increase the time complexity of the original L-ART.

A potential problem of Algorithm 7.2 is the space complexity. Since the previous boolean matrix \( \text{GridCells} \) will be discarded when refining the lattice structure, the analysis of space complexity can just focus on the final lattice structure with resolution \( m \). In a \( k \)-dimensional input domain, the number of grid cells with the final lattice structure is \( (m \cdot g)^k \).

Since \( m < 2(\sqrt{n}+1) \), the number of grid cells should be less than \( (2(\sqrt{n}+1))^k \cdot g^k \). In very high dimensional input domains, \( (2g)^k \) could be a large number, and the algorithm may need a large amount of space. In that case, we suggest discarding the grid cells and applying the checking process in the straightforward way by exploring the successful test cases one by one, which has time complexity in \( O(n^2) \).

### 7.4 Simulation Studies

The first part of the simulation was conducted to confirm whether the enhanced method can improve the fault-detection capability of L-ART when choosing a large value for \( f \).

For each type of failure pattern, block, strip and point, simulations were conducted in a 2-dimensional unit input domain with varying failure rate \( \theta \) from 0.01 to 0.001 and varying shaken factor \( f \) from 0 to 1. Block failure patterns were simulated by a square region with size \( \theta \) randomly generated within the input domain. For strip patterns, 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, 50 non-overlapping equally sized disks were randomly placed in the input domain, of which the total size makes up the failure rate \( \theta \).

Figure 7.5 compares the relative F-measure between original L-ART and the enhanced
Figure 7.5: Comparison of the relative F-measure of original and enhanced L-ART under different failure rates \( \theta \) and shaken factors \( f \) (under the block failure pattern). The \( x \)-axis represents \( f \). The \( y \)-axis represents the relative F-measure.

As shown, both enhanced and original methods have similar F-measures when \( f \) is smaller than 0.4. When \( f \geq 0.4 \), the F-measure of enhanced method is lower than that of original method. The improvement becomes more significant with the increase of \( f \), since the F-measure of original method is always increasing with the increase of \( f \) while that of enhanced method is almost steady.

Table 7.2 and Table 7.3 list the results for strip and point failure patterns, respectively. Since the F-measure of enhanced method is similar to that of original method with small \( f \), the F-measure were listed from \( f = 0.4 \) to \( f = 1 \). For the strip failure pattern, the enhanced method gets the best fault-detection capability for \( f = 0.6 \). The relative F-measure is between 92% to 97%, which is an observable improvement over original L-ART. For original L-ART, the result is slightly higher than 1. When \( f = 0.6 \), the enhanced method also achieves the best results for point failure pattern, which is between 96% to 97%. Comparing with original L-ART, the improvement is not statistically significant.

The second part of the simulation is to compare the test case distribution of original
<table>
<thead>
<tr>
<th>Enhanced L-ART</th>
<th>Original L-ART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Rate θ</td>
<td>Failure Rate θ</td>
</tr>
<tr>
<td>0.01 0.005 0.002 0.001</td>
<td>0.01 0.005 0.002 0.001</td>
</tr>
<tr>
<td>f=0.4 98% 97% 98% 98%</td>
<td>103% 106% 105% 100%</td>
</tr>
<tr>
<td>f=0.45 99% 96% 97% 99%</td>
<td>106% 106% 106% 103%</td>
</tr>
<tr>
<td>f=0.50 96% 95% 98% 99%</td>
<td>105% 106% 105% 101%</td>
</tr>
<tr>
<td>f=0.55 97% 95% 99% 99%</td>
<td>107% 105% 105% 104%</td>
</tr>
<tr>
<td>f=0.60 92% 97% 96% 97%</td>
<td>108% 106% 105% 103%</td>
</tr>
<tr>
<td>f=0.65 94% 97% 96% 99%</td>
<td>107% 105% 104% 103%</td>
</tr>
<tr>
<td>f=0.70 93% 95% 98% 96%</td>
<td>108% 107% 104% 104%</td>
</tr>
<tr>
<td>f=0.75 95% 94% 98% 98%</td>
<td>108% 107% 104% 102%</td>
</tr>
<tr>
<td>f=0.80 93% 97% 97% 99%</td>
<td>106% 107% 105% 103%</td>
</tr>
<tr>
<td>f=0.85 94% 96% 94% 95%</td>
<td>106% 105% 104% 104%</td>
</tr>
<tr>
<td>f=0.90 93% 95% 96% 97%</td>
<td>104% 105% 105% 103%</td>
</tr>
<tr>
<td>f=0.95 93% 94% 96% 98%</td>
<td>107% 105% 105% 102%</td>
</tr>
<tr>
<td>f=1.00 93% 95% 96% 99%</td>
<td>108% 106% 106% 103%</td>
</tr>
</tbody>
</table>

Table 7.2: Comparison of relative F-measures of original and enhanced L-ART (under the strip failure pattern)

<table>
<thead>
<tr>
<th>Enhanced L-ART</th>
<th>Original L-ART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Rate θ</td>
<td>Failure Rate θ</td>
</tr>
<tr>
<td>0.01 0.005 0.002 0.001</td>
<td>0.01 0.005 0.002 0.001</td>
</tr>
<tr>
<td>f=0.4 99% 98% 99% 97%</td>
<td>103% 100% 98% 97%</td>
</tr>
<tr>
<td>f=0.45 98% 98% 99% 99%</td>
<td>99% 99% 98% 98%</td>
</tr>
<tr>
<td>f=0.50 99% 97% 98% 99%</td>
<td>101% 101% 99% 99%</td>
</tr>
<tr>
<td>f=0.55 98% 97% 97% 99%</td>
<td>100% 100% 102% 98%</td>
</tr>
<tr>
<td>f=0.60 97% 97% 96% 96%</td>
<td>98% 99% 98% 99%</td>
</tr>
<tr>
<td>f=0.65 99% 99% 99% 99%</td>
<td>100% 99% 102% 99%</td>
</tr>
<tr>
<td>f=0.70 97% 96% 97% 99%</td>
<td>99% 100% 98% 98%</td>
</tr>
<tr>
<td>f=0.75 97% 96% 99% 99%</td>
<td>98% 102% 97% 99%</td>
</tr>
<tr>
<td>f=0.80 97% 98% 97% 98%</td>
<td>100% 99% 100% 98%</td>
</tr>
<tr>
<td>f=0.85 97% 95% 99% 97%</td>
<td>98% 98% 98% 99%</td>
</tr>
<tr>
<td>f=0.90 99% 94% 96% 99%</td>
<td>98% 99% 99% 99%</td>
</tr>
<tr>
<td>f=0.95 99% 99% 97% 98%</td>
<td>99% 98% 102% 98%</td>
</tr>
<tr>
<td>f=1.00 99% 95% 99% 97%</td>
<td>98% 102% 100% 99%</td>
</tr>
</tbody>
</table>

Table 7.3: Comparison of relative F-measures of original and enhanced L-ART (under the point failure pattern)
L-ART and enhanced L-ART. Since \( f = 0.1 \) is recommended for original L-ART [67], and enhanced L-ART gets best fault-detection capability when \( f = 0.6 \), Figure 7.6 compares the test case distributions of these two. The simulation settings are the same as that in Section 7.2.2. It is confirmed that the highly skewed test case distribution of original L-ART is much alleviated in enhanced L-ART.

7.5 Summary

L-ART is distinctive from other ART methods in the sense that it does not use the locations of successful test cases in an explicit way. Instead, the locations of successful test cases are exploited inexplicitly through systematically placing lattice nodes in the input domain. Randomness in original L-ART is introduced by shaking lattice nodes within a restricted region. Since original L-ART does not explore the locations of previous test cases, it runs almost as fast as pure random testing.

Our study on the test case spacial distribution found that to achieve a good fault-detection capability the original L-ART should strictly restrict the locations of test cases. This approach, however, will deviate from the basis of ART, an even spread of test cases. One effective way to alleviate the highly skewed distribution is to randomly shift the lattice nodes in a large shaken region (adopting a large \( f \)). However, simply enlarging the shaken region will result in test cases having a higher chance of being clustered together — a common problem of all PSS methods, which will compromise the fault-detection capability. In order to reduce the probability of test cases being clustered together, a checking process is introduced to make sure test cases are only generated in the shaken regions that have not been occupied by successful test cases.

Histograms of test case distribution have showed that spatial distribution of the enhanced method is much evener than that of the original one. Simulations were also conducted to investigate the fault-detection capability of the enhanced L-ART. For large shaken factors (\( f \geq 0.4 \)), we observed that fault-detection capability of enhanced L-ART outperforms original version, and their fault-detection capability are similar for other shaken factors.
Figure 7.6: Comparison of histogram of original L-ART ($f = 0.1$) and enhanced L-ART ($f = 0.6$) 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.
Comparing with original L-ART, the extra cost for the enhancement is to check whether there are successful test cases in the designated shaken region. If the dimensionality of the input domain is not very large, the checking process can be implemented in a cost efficient way, in which the time complexity is linear to the number of test cases generated.
Chapter 8

Adaptive Random Testing by Balancing

8.1 Introduction

In this thesis, existing ART methods are classified into 2 categories. One is “far apart” ART methods, which interprets an even spread of test cases as test cases are far apart from one another. These methods use distance as the selection criterion. The candidate farthest away from all successful test cases or the first candidate sufficiently far away from all successful test cases are selected as the next test case. Another is PSS ART methods, which interprets it as the number of test cases in a partition should be proportional to its size. Subsequent test cases are directly generated in the most sparsely populated partitions.

Empirical studies [14, 15, 21, 24] have showed that ART greatly outperforms RT if the failure-causing inputs are contiguous. However, it is also observed [25] that the fault-detection capability of some ART methods deteriorates with the increase of the dimensionality of the input domain. To tackle this problem, this chapter proposes an innovative ART method, namely ART by balancing. This method has a good fault-detection capability, especially in high dimensional input domains [16].
8.2 ART by Balancing

8.2.1 Preliminary

In physics, the centroid (also called the centre of mass) of a system of particles is defined as the average of the positions of these particles, weighted by their masses [88]. If we regard the test cases in the input domain as particles of equal mass, then the centroid of a set of test cases can be defined as follows. In a $k$-dimensional input domain, let $S = \{s_1, s_2, \ldots, s_l\}$ be the set of test cases, where $l \geq 1$ and $s_i = s_{i1}, s_{i2}, \ldots, s_{ik}$ ($1 \leq i \leq l$). Then the centroid $R = r_1, r_2, \ldots, r_k$ of $S$ can be calculated by

$$r_j = \frac{1}{l} \sum_{i=1}^{l} s_{ij} \quad 1 \leq j \leq k$$

The input domain is also regarded as a system with uniform density, and its centroid is at the centre of the input domain.

8.2.2 Intuition

The basis of ART is to achieve an even spread of test cases in the input domain. Intuitively speaking, if test cases are evenly spread, the centroid of the test cases should be close to the centroid of the input domain. Hence, it has been suggested that a test case selection method should select candidates as test cases in such a way that the centroid of the resultant set of test cases should be close to the centroid of the input domain [9]. This test case selection strategy is called as balancing strategy in this chapter.

Figure 8.1 is used to illustrate the intuition of ART by balancing. The outer square represents the 2-dimensional input domain. The cross at the centre represents the centroid of the input domain. A test case $s_1$ is randomly generated (denoted by solid circles), which does not detect a failure, hence it is regarded as a successful test case. Now, we try to generate the second test case according to the selection strategy of balancing. Three candidates $c_1$, $c_2$ and $c_3$ are randomly generated, as illustrated in Figure 8.1(b). Let us consider the centroid (denoted by solid 5-pointed star) of test cases in the input domain if one of the candidates becomes a test case. In Figure 8.1(c), $r_1$ denotes the centroid of $s_1$ and $c_1$; $r_2$
denotes the centroid of \( s_1 \) and \( c_2 \); \( r_3 \) denotes the centroid of \( s_1 \) and \( c_3 \). Obviously, \( r_2 \) is closest to the centroid of the input domain, hence \( c_2 \) is selected as the next test case.

However, a straightforward application of the balancing strategy will result in a “black hole” effect [9]: the more test cases selected by balancing strategy, the more test cases will be concentrated around the centre of the input domain. This effect is against the basis of ART: an even spread of test cases throughout the input domain. For example, if we continue generating subsequent test cases in Figure 8.1 (d), most of them will be around the centre of the input domain.

Intuitively, when applying the balancing strategy, the more test cases generated in a partition, the higher chance test cases will be concentrated around the centre of the partition. Therefore, to alleviate this “black hole” effect, we propose to partition the input domain incrementally and apply balancing strategy in each partition. For a partition with no successful test cases, the first test case will be randomly selected. For a partition with 2 or more successful test cases, no more test cases will be selected from that partition. When all partitions contain 2 or more successful test cases, the input domain will be repartitioned with a finer partitioning scheme.
The idea is illustrated in Figure 8.2 (a-d), which continues the test case generation process of Figure 8.1. Since the number of test cases already reaches 2 in Figure 8.1 (d), the input domain is divided into partition $A$ and partition $B$. In such a subdivision, test cases $s_1$ and $s_2$ belong to in partition $B$ and $A$, respectively. Since none of the partitions have 2 or more test cases, the balancing test case selection strategy is applied in every individual partitions. For example, in partition $B$, candidate $c_3$ is the best choice among the candidates $c_1$, $c_2$ and $c_3$, which makes the centroid of test cases in this partition be closer to the centroid of the partition as illustrated in Figure 8.2 (c).

8.2.3 Algorithm Description

The key objective of ART by balancing is to make the centroid of test cases in each partition of the input domain being close to the centroid of the corresponding partition. For ease the presentation, the algorithm described in Algorithm 8.1 is for testing a program with two real inputs. It is straightforward to apply ART by balancing in a high dimensional input domain. Test cases are selected from candidate partitions, where the number of successful test cases is smaller than 2. Each candidate partition $G$ consists of by 4 elements:

- $\{(i_{\text{min}}, j_{\text{min}}), (i_{\text{max}}, j_{\text{max}})\}$, the range of the partition;
- $IC$ is the centroid of the partition, which is also the Ideal Centroid of test cases in the partition;
- $CC$ is the Current Centroid of test cases in the partition;
- $tc$ records the current number of successful test cases in this partitions
- $diff$ records the Cartesian distance between the $IC$ and the $CC$. If there are no test cases inside $G$, $diff$ is set to have a very large value.

Basically, this method selects test cases as follows. The candidate partition with greatest $diff$ are firstly selected as test case generation region. If the test case generation region contains no successful test cases, a test case is randomly generated. If it contains only one successful test case, balancing strategy is applied to selected the second test case — a
set of candidates are randomly generated in this partition and the candidate that makes the resultant centroid closest to the centroid of the partition (IC) is selected as the next test case. If all partitions contain 2 or more successful test cases, this algorithm will discard the current partitioning scheme and repartition the input domain with a finer partitioning scheme. Let \( p \) be the number of partitions along x-axis and \( q \) be the number of partitions along y-axis for the current partitioning scheme. When repartitioning occurs, the smaller one of \( p \) and \( q \) will be increased by 1 and the input domain will be repartitioned by a \((p + 1) \times q\) (supposing \( p < q \)) partitioning scheme. If \( p \) and \( q \) are equal, one of them will be randomly selected to be increased by 1.

8.3 Simulations

The objectives of the simulations are to answer the following 2 questions. (1) Has the fault-detection capability in the high dimensional input domain been improved? (2) What is the test case distribution of ART by balancing?

8.3.1 Fault-detection Capability

We firstly investigate the impact of the size of the candidate set \( e \) on fault-detection capability. Intuitively, the larger size of the candidate set, the better fault-detection capability (lower F-measure). The simulations results have showed that F-measure will decrease with the increase of the number test case candidates and become almost steady when the size reaches 5. And when \( e = 12 \), we got the lowest F-measure. Therefore, in the following simulations, we will fix the size of the candidate set as \( e = 12 \).

Then, fault-detection capability on different failure patterns are investigated in a 2-dimensional square input domain with size of 1. For each test run, a failure region with a specified failure rate \( \theta \) 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 a specified size according to the failure rate \( \theta \); for the point failure pattern, 10 equally sized circular regions were randomly placed in the input domain without overlap-
Algorithm 8.1 ART by balancing algorithm

1. Initiate the candidate partition list $L$ with the whole input domain as the only partition, $(i_{\text{min}}, j_{\text{min}}), (i_{\text{max}}, j_{\text{max}})$, where $IC = \left( \frac{i_{\text{max}} - i_{\text{min}}}{2}, \frac{j_{\text{max}} - j_{\text{min}}}{2} \right)$, $CC = 0$, $tc = 0$, $diff$ is set to be a very large number.

2. Set the number of partitions along x-axis $p$ to be 1, the number of partitions along y-axis $q$ to be 1.

3. Set the successful test case set $S$ to be empty.

4. While $L$ is not empty
   (a) Select the candidate partition $G$ with the greatest $diff$. If there is more than one partition with the same greatest $diff$, choose one randomly.
   (b) If there is no successful test cases in $G$, randomly generate a test case $t$ within $G$.
   (c) Otherwise, randomly generate a candidate set, $C = \{C_1, C_2, \ldots, C_e\}$ within the range of $G$. Select the candidate $C_q$ that makes the new resultant centroid $CC$ closest to the ideal centroid $IC$ of $G$ as the next test case $t$.
   (d) If $t$ is a failure-causing input, report fault detection and terminate.
   (e) Add $t$ to $S$.
   (f) If the number of test cases in the partition $tc$ reaches 2, remove $G$ from $L$, otherwise, update $diff$, $CC$ and $tc$ of the partition $G$.

5. Repartition the input domain.
   (a) If $p$ and $q$ are not equal, the smaller one will be increased by 1, else one of them is randomly selected to be increased by 1.
   (b) Equally partition the input domain, such that there are $p$ partitions along x-axis and $q$ partitions along y-axis.
   (c) For each partition, set $IC$ to be the centre of each partition; $tc = 0$; $CC = 0$; $diff$ to be a very large number.

6. Map each successful test case to a partition, and update $tc$, $CC$ and $diff$ of each partitions accordingly.

7. Assign the candidate partitions (where $tc < 2$) to $L$.

Table 8.1 presents the results. From Table 8.1, we find that the simulation gets the best results in the block failure pattern. The F-measure of ART by balancing is about 30-31% lower than that of random testing. There is 2% to 8% improvements on random testing for the strip pattern and 5% to 6% for the point pattern.

In the third part of the simulation, the fault-detection capability of ART by balancing in high dimensional input domains is investigated. Simulations are 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. Table 8.4 lists the results for ART by balancing. To compare with other ART methods, we list the relative F-measure of Distance-based ART (D-ART) (as the representative of “far apart” ART methods) and ART by bisection (B-ART) (as the representative of PSS ART methods) in Table 8.2 and Table 8.3, respectively.

Let us look at the 2-dimensional input domain first. The relative F-measure of ART by balancing is slightly higher than D-ART, but much lower than B-ART. With regard to the high dimensional input domains, it can be seen for D-ART that with the increase of dimensionality of the input domain, the fault-detection capability decreases dramatically (that is, the relative F-measure increases). For example, the relative F-measure on $\theta = 0.01$ increases from 67% to 108% when dimensionality increases from 2 to 4. In other words, for high dimensional input domains, when the failure rate is large, the F-measure of D-ART is higher than random testing. For B-ART, as illustrated in Table 8.3, the relative F-measure also increases with the increase of dimensionality of the input domain, but not as fast as

<table>
<thead>
<tr>
<th>Failure Rate $\theta$</th>
<th>F-measure of RT ($F_{RT}$)</th>
<th>Block Pattern</th>
<th>Strip Pattern</th>
<th>Point Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean $F_{ART}$ / $F_{RT}$</td>
<td>Mean $F_{ART}$ / $F_{RT}$</td>
<td>Mean $F_{ART}$ / $F_{RT}$</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>69 69%</td>
<td>92 92%</td>
<td>94 94%</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>140 70%</td>
<td>194 97%</td>
<td>189 95%</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>349 70%</td>
<td>490 98%</td>
<td>469 94%</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>689 69%</td>
<td>969 97%</td>
<td>941 94%</td>
</tr>
</tbody>
</table>

Table 8.1: F-measures of ART by balancing in a 2-dimensional input domain under 3 failure patterns.
Table 8.2: F-measures of D-ART in 2-, 3-, and 4-dimensional input domains (under the block failure pattern)

<table>
<thead>
<tr>
<th>Failure Rate $\theta$</th>
<th>F-measure of RT ($F_{RT}$)</th>
<th>2D</th>
<th>3D</th>
<th>4D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$F_{ART}$</td>
<td>$F_{ART} / F_{RT}$</td>
<td>$F_{ART}$</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>67</td>
<td>67%</td>
<td>85</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>132</td>
<td>66%</td>
<td>159</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>323</td>
<td>65%</td>
<td>382</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>648</td>
<td>65%</td>
<td>754</td>
</tr>
</tbody>
</table>

Table 8.3: F-measures of B-ART in 2-, 3-, and 4-dimensional input domains (under the block failure pattern)

<table>
<thead>
<tr>
<th>Failure Rate $\theta$</th>
<th>F-measure of RT ($F_{RT}$)</th>
<th>2D</th>
<th>3D</th>
<th>4D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$F_{ART}$</td>
<td>$F_{ART} / F_{RT}$</td>
<td>$F_{ART}$</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>74</td>
<td>74%</td>
<td>83</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>154</td>
<td>77%</td>
<td>166</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>361</td>
<td>72%</td>
<td>416</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>762</td>
<td>76%</td>
<td>839</td>
</tr>
</tbody>
</table>

Table 8.4: F-measures of ART by balancing in 2-, 3-, and 4-dimensional input domains (under the block failure pattern)

<table>
<thead>
<tr>
<th>Failure Rate $\theta$</th>
<th>F-measure of RT ($F_{RT}$)</th>
<th>2D</th>
<th>3D</th>
<th>4D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$F_{ART}$</td>
<td>$F_{ART} / F_{RT}$</td>
<td>$F_{ART}$</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>69</td>
<td>69%</td>
<td>73</td>
</tr>
<tr>
<td>0.005</td>
<td>200</td>
<td>140</td>
<td>70%</td>
<td>146</td>
</tr>
<tr>
<td>0.002</td>
<td>500</td>
<td>349</td>
<td>70%</td>
<td>370</td>
</tr>
<tr>
<td>0.001</td>
<td>1000</td>
<td>689</td>
<td>69%</td>
<td>736</td>
</tr>
</tbody>
</table>

That in D-ART. For $\theta = 0.01$, the relative F-measure increases from 74% to 90% when dimensionality increases from 2 to 4. However, for ART by balancing, with the increase of the dimensionality of the input domain, the fault-detection capability just slightly increases, as shown in Table 8.4. For example, when $\theta = 0.01$, the relative F-measure increases from 68% to 75% if increasing the dimensionality from 2 to 4. Hence, ART by balancing has a better fault-detection capability in high dimensional input domains compared with D-ART and B-ART.
8.3.2 Spatial Distribution

The spatial distributions of the test cases generated by ART by balancing are investigated without considering the failure-causing inputs. The simulation settings are the same as those in Section 5.2. In each trial of test case generation, the locations of the first $n$ test cases were recorded, where $n = 5, 20, 50, 100, 500, 1000$. Without loss of generality, the positions of the test cases were projected onto one dimension, since ART by balancing treats every dimension independently.

A million independent trials were conducted. The histograms of test case distributions are listed in Figure 8.3. We find that when $n$ is small, there are some test case favorable regions (those that contains a higher density of test cases) and unfavorable regions (those that contains a lower density of test cases) interlaid throughout the input domain. With the increase of $n$, the gap of test cases density in favorable and unfavorable regions is diminishing and the distribution becomes much even. For ease of comparison, the histograms of D-ART are listed in Figure 8.4. Obviously, in D-ART more test cases are generated close to the boundary of the input domain. The histograms of B-ART are not listed, since test cases are uniformly distributed.

8.4 Discussion and Summary

The common problem for some ART methods is the compromised fault-detection capability in high dimensional input domains. The reason can be explained as follows. Suppose the input domain is a hypercube with size of 1 and the failure region is also a hypercube. Let $k$ be the dimensionality of the input domain, $\theta$ be the failure rate. As illustrated in Table 8.5, the width of the failure-causing region, $w = \sqrt{k}\theta$ increases with the increase of dimensionality $k$. Obviously, in high dimensional input domains, the test cases generated from the centre of the input domain have a higher chance to detect failures than those generated from the whole input domain, especially for a large value of failure rate. For D-ART, test cases prefer to be close to the boundary of the input domain. For B-ART, test cases are uniformly distributed. Since both do not prefer to select test cases from the central parts, the F-measure
Figure 8.3: Histograms of ART by balancing 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.
Figure 8.4: Histograms of D-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.
Table 8.5: Width of a hypercube failure pattern in a k-dimensional hypercube input domain with size of 1

<table>
<thead>
<tr>
<th>k</th>
<th>0.01</th>
<th>0.005</th>
<th>0.002</th>
<th>0.001</th>
</tr>
</thead>
<tbody>
<tr>
<td>k=2</td>
<td>0.100</td>
<td>0.070</td>
<td>0.045</td>
<td>0.032</td>
</tr>
<tr>
<td>k=3</td>
<td>0.215</td>
<td>0.171</td>
<td>0.126</td>
<td>0.100</td>
</tr>
<tr>
<td>k=4</td>
<td>0.316</td>
<td>0.266</td>
<td>0.211</td>
<td>0.178</td>
</tr>
<tr>
<td>k=5</td>
<td>0.398</td>
<td>0.347</td>
<td>0.289</td>
<td>0.251</td>
</tr>
<tr>
<td>k=6</td>
<td>0.464</td>
<td>0.414</td>
<td>0.355</td>
<td>0.316</td>
</tr>
<tr>
<td>k=7</td>
<td>0.518</td>
<td>0.469</td>
<td>0.412</td>
<td>0.373</td>
</tr>
</tbody>
</table>

Note that the centroid of the test cases being close to the centroid of the input domain is a necessary property of “an even spread of test cases”. However, having the centroid of a set of test cases close to the centroid of the input domain cannot guarantee an even spread of test cases. For example, if all test cases are concentrated either on the centre or boundary of the input domain, the centroid of test cases can still be close to the centroid of the input domain, but test cases are not evenly spread. In this chapter, we solve this problem by incrementally partitioning the input domain into equally sized partitions and using balancing strategy to select subsequent test case in individual partitions. We believe that new methods could be developed based on the notion of balancing.
Chapter 9

Conclusion

9.1 Summary

Adaptive Random Testing (ART) is a simple but effective improvement of Random Testing (RT) in the sense that using fewer test cases to detect the first failure. The basis of ART is to achieve an even spread of test cases throughout the input domain, which is motivated by the observation that failure-causing inputs tend to be clustered together in one or a few contiguous regions. Several ART methods (algorithms) have recently developed to implement “an even spread of test cases”. Due to the nature of the principles adopted, these ART methods have their own advantages and disadvantages. Since there were no complete studies on the classification of existing ART methods, it was difficult to compare and improve these ART methods. This thesis takes aim at establishing a taxonomy for the ART methods, and proposing improvements of each category of these methods.

After reviewing related works in Chapter 2, Chapter 3 stated the taxonomy of ART methods, which is the framework for further studies on ART methods. The taxonomy is based on the different interpretations of the basis of ART. Existing ART methods were classified into 2 categories. One category is “far apart” ART methods, which interprets an even spread of test cases as test cases being far apart from one another. Another category is Proportional Sampling Strategy (PSS) ART methods, which interprets an even spread of test cases as the number of test cases in a partition should be proportional to the size of that
Based on this taxonomy, this thesis has proposed improvements for each category of ART methods. For “far apart” ART methods, the two major shortcomings are extensive computational overhead and boundary effect, which are the main focuses of Chapter 4 and Chapter 5, respectively. Chapter 4 analyzed the cause of the extensive computational overhead first, in which we found that for both D-ART and R-ART distance computations and comparisons are performed between each candidate and all successful test cases for each round of test case generation. Hence, the time complexities for both D-ART and R-ART are quadratic to the number of test cases generated. Then, a new ART method, namely ART through Iterative Partitioning (IP-ART), was proposed, which totally avoids distance computations and comparisons. This method 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. Compared with D-ART and R-ART, IP-ART has a comparable fault-detection capability, but the time complexity has been significantly reduced. Chapter 4 further analyzed two important parameters in IP-ART. It was found that (1) setting the initial partitioning scheme should refer to the estimation of the failure rate. In practice, an over-estimated failure rate can reduce the cost of repartitioning and achieve a low F-measure; (2) further classifying the adjacent grid cells according to their locations relative to the occupied cells cannot improve the fault-detection capability.

One unfavorable phenomenon in some ART methods is that test cases are not uniformly distributed in the input domain. In “far apart” ART methods, test cases have a common preference to be near the boundary of the input domain, which is regarded as boundary effect. Boundary effect adversely affects the fault-detection capability, especially in high dimensional input domains. In addition, fault-detection capability of “far apart” ART methods will depend on the locations of the failure regions. In Chapter 5, we analyzed the cause of this effect and proposed a new notion of virtual images of successful test cases to tackle this problem. The virtual images are constructed by shifting the input domain, such that success-
ful test cases can “step out” of the input domain. The introduction of virtual images does not significantly increase the overhead of original methods, but make test cases uniformly distributed throughout the input domain. Consequently, the fault-detection capability has been improved.

The key objective of PSS methods is to partition the input domain and generate test cases in the sparsely populated regions. There are two partitioning schemes: dynamic partitioning scheme and static partitioning scheme, which lead to two PSS ART methods, ART through dynamic partitioning (DP-ART) and Lattice-based ART (L-ART). Chapter 6 and Chapter 7 focused on analyzing the shortcomings and proposing improvements for this two methods, respectively. For DP-ART, its fault-detection capability is not as good as “far apart” ART methods. Chapter 6 found that a main reason behind is that test cases still have chances to be clustered together even they are generated from sparsely populated partitions. Introducing distance computations into DP-ART is an effective way to avoid this unfavorable phenomenon. However, like D-ART and R-ART, distance computations between each candidate and all successful test cases are time consuming. In Chapter 6, the concept of localization was introduced to efficiently incorporate distance computations into the enhanced methods. Compared with D-ART and R-ART, the number of distance computation is reduced from quadratic to linear in terms of number of test cases generated. Compared with original methods, the fault-detection capability has been greatly improved.

L-ART has a good fault-detection capability and low overhead. However, in Chapter 7, we found that to achieve a good fault-detection capability L-ART should use a small shaken factor, which result in test cases being highly concentrated on some parts of the input domain. This test case distribution makes the fault-detection capability unpredictable, because the fault-detection capability becomes dependent on the locations the failure regions and the locations of failure regions are unknown. If choosing a large shaken factor, the highly skewed distribution will be alleviated. However, simply enlarging the shaken factor will increase the probability of test cases being clustered together — a common problem of all PSS methods, which will compromise the fault-detection capability. Hence, a checking process was introduced to reduce the probability of test cases being clustered together.
Experimental results have showed that spatial distribution of the enhanced method is much evener than that of the original one and the fault-detection capability is improved. As analyzed, the time complexity of the extra cost in the enhanced version is linear to the number of test cases generated if the dimensionality of the input domain is not very large.

With regard to the practicality of ART methods in high dimensional input domains, Chapter 8 proposes a new ART method, ART by balancing, based on a distinctive interpretation of an even spread of test cases, which requires that the centroid of test cases in a partition should be close to the centroid of that partition. Our experimental results have showed that the fault-detection capability of ART by balancing is just slightly affected by the increase of the dimensionality of the input domain. Hence, it has an improved fault-detection capability in high dimensional input domains.

It is also found that each ART algorithm has its own advantages and disadvantages. For a given Software Under Testing (SUT), the practitioners need to consider several aspects. For example, the dimensionality of the input domain, the data structure of the input parameters and common types of errors of that SUT.

### 9.2 Future Works

This study uses simulations to compare the fault-detection capability of ART methods. More empirical studies on real programs with different failure patterns are necessary. This is because the failure patterns in the real programs may be different from the failure patterns used in the simulations. Moreover, empirical studies on various types of real programs will provide specific guidelines for the users on how to choose ART methods in practice.

In this thesis, the Software Under Test (SUT) is limited to numeric problem domain. Further investigation on applying these ART methods in non-numeric input domain will greatly broaden the applications of ART. There are two major challenges of applying ART methods in non-numeric input domain [68]. The first is how to effectively generate random test cases in the input domain. The second is to establish a metric to judge whether the test case are evenly spread in the input domain.

Furthermore, while reviewing the literature and conducting the experiments for this the-
sis, it is found that combination of ART and other testing techniques is a potential research area. Two interesting research topics are listed below.

One potential area is on the oracle problem of ART. From the view of test case selection overhead, ART is obviously superior to other testing techniques. However, more effort is needed to verify the output of a randomly generated test case. For example, when testing a program \( \text{prog}(x) \) that implements the \( \text{sine} \) function, it is easy to get the output for the special test cases, such as 0, \( \pi/4 \) and \( \pi/2 \). If the test case is a randomly generated number, for example, 1.234, it is hard for the tester to get the expected output, although the range of plausible values of \( \text{prog}(1.234) \) can be narrowed by employing mathematical properties of the \( \text{sine} \) curve. Hence, the advantages of ART could be compromised by the overhead of verifying the output. Metamorphic Testing (MT) (Section 2.1.3) is approach proposed to alleviate the oracle problem. This technique is based on expected program properties, which are necessary relations among the inputs and outputs of multiple executions of the target program. A potential research area would be incorporating MT into ART to reduce the overhead of verifying the output.

Another potential area is on the relationship between ART and observation-based testing. Intuitively, ART requires the subsequent test cases to be “dissimilar” from existing ones, in terms of spatial locations in the input domain. It is of interest that observation-based testing (Section 2.4) also requires subsequent test cases to be “dissimilar” from existing ones, but in the terms of execution profiles. Note that the motivation and intuition behind ART and observation-based testing are substantially different. ART is based on the observation of failure patterns, while observation-based testing is based on execution profiles. Obviously, the cost of selecting “dissimilar” test cases in ART is much lower. One interesting research area would be investigating the relationships between test cases that are spatially “dissimilar” and those that have “dissimilar” execution profiles.
Bibliography


List of Publications


