Addressing the Performance Challenges of Metamorphic Testing

Author:
Zhihao Ying

Supervisors:
Prof. Dave Towey
Dr. Anthony Bellotti

A thesis submitted to University of Nottingham
for the degree of Doctor of Philosophy

School of Computer Science
Faculty of Science and Engineering

University of Nottingham Ningbo China

May 2024

Author:
Zhihao Ying

Supervisors:
Prof. Dave Towey
Dr. Anthony Bellotti

University:
University of Nottingham Ningbo China

ABSTRACT

Software testing is an important process that should be considered throughout the entire life-cycle of software development: It is used to assess and assure the quality of the System Under Test (SUT). One of the fundamental problems faced in software testing is the oracle problem, which means that it is too expensive or even impossible to implement an oracle, which represents a mechanism to verify the correctness of the output or behavior of the SUT. Metamorphic Testing (MT) is a popular property-based software testing approach that has been proven to be effective in alleviating the oracle problem. As a central component of MT, Metamorphic Relations (MRs) are generally derived from necessary properties of the SUT. To implement MT, some program inputs are first generated as Source Test Cases (STCs), and then an MR can be used to generate new inputs as Follow-up Test Cases (FTCs) based on the STCs. If the actual outputs of STCs and FTCs violate the given MR, then the SUT is referred to as faulty in terms of the property related to the MR. Different from the traditional way of detecting software failures through checking the test result against an oracle, MT detects failures by verifying the MRs among STCs and FTCs as well as their relevant outputs. The STCs and their corresponding FTCs, considered as a whole, are called the Metamorphic Groups (MGs), and the MGs that violate the given MR are called MR-violating MGs.

Despite its increasing popularity, the performance of MT still needs further improvement. This thesis focuses on addressing the performance challenges of MT. Specifically, this thesis attempts to: (1) Improve MT performance (i.e. test effectiveness and efficiency) by enhancing the quality of MRs and MGs; (2) improve MT performance by addressing the problems existing in the design and application of MG-generation algorithms; and (3) improve software testing performance (for testing credit risk models) by employing MT as an additional model testing, validation and selection methodology.

First, the successful implementation of MT as well as its performance are highly dependent on the MRs and MGs, and therefore, this thesis improves the quality of MRs and MGs throughout their entire life-cycles:

- This thesis proposes new MR patterns to guide the identification of concrete MRs.
- This thesis proposes new MG-generation algorithms to generate effective MGs.
- This thesis proposes a new MR-MG pair selection algorithm to automatically and dynamically select effective MR-MG pairs for execution from existing ones.

This thesis evaluates the performance of the proposed methodologies through empirical experiments, and the experimental results indicated that they are capable of improving both the efficiency and effectiveness of MT. In addition, this thesis also introduces the concept of MR-violation regions as an additional evaluation method (for validating experimental results of different MG-generation algorithms).

Second, through the evaluation of MG-generation algorithms, this thesis identifies that previous MG-generation algorithms may encounter certain problems, which may negatively affect their performance (i.e. test effectiveness and efficiency). This thesis summarizes those situations and formally proposes the concepts of the MT-performance evaluation problem and the input-domain difference problem. This thesis also introduces methods to address these problems, with the aim of not only avoiding the existence of the same

problems in the proposed methodologies, but also further improving the performance of previously-published algorithms.

Machine Learning (ML) algorithms have be widely-adopted in financial services for credit risk modelling and show improved predictive performance in comparison with traditional linear models. However, the adoption of ML algorithms may raise serious validation issues, related to the inherent complexity of models. With this consideration, this thesis proposes a new perspective for testing and validating ML-based credit risk models that uses properties of the model, or properties hypothesized by users based on business rationale, to allow testers to predict how a particular change in the input should affect the output. Specifically, this thesis proposes MT as a model testing, validation and selection step, complementary to traditional model fit measures, when ML is used for credit risk modelling.

In summary, this thesis aims to improve the effectiveness and efficiency of MT by focusing on its core parts: The MRs and MGs. The main limitation of the work is that this thesis has only proposed some methodologies specifically for each part of MT. In this context, the future work will include investigating the relationships and the connections between the proposed methodologies and propose an overall MT framework that consists of all these methodologies.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

STATEMENT OF AUTHORSHIP

I, Zhihao Ying, born November 1st, 1996 in Zhejiang China, declare that this thesis titled *Addressing the Performance Challenges of Metamorphic Testing* and the work presented in it are my own. I confirm that this work was done while in candidature for a research degree at University of Nottingham Ningbo China.

Except where reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis accepted for the award of any other degree or diploma. No other person's work has been used without due acknowledgment in the main text of the thesis. This thesis has not been submitted for the award of any degree or diploma in any other tertiary institution.

Zhihao Ying May 16, 2024

ACKNOWLEDGMENTS

I would like to express my deep gratitude to the exceptional support and guidance of my supervisor, Dave Towey, professor in computer science, associate dean for education and student experience for the Faculty of Science and Engineering, Deputy Head of School, Deputy Director of IDIC, for providing invaluable guidance, continued support, and encouragement throughout this research and guiding me to the completion of this thesis. His vision and sincerity inspired me deeply. He taught me the methodology of research and the way to present the research results clearly. I am very grateful to him for everything he has taught me. In the meanwhile, I also want to thank Anthony Bellotti, Associate Professor, my second supervisor, for his kindly help and exceptional guidance throughout this project. It is a great honor to conduct research and study under their supervision.

I would like to give my deep thanks to the teachers who have offered me help during my four years of PhD study. Thank my university, the University of Nottingham Ningbo China, for giving me the abundant educational and research resources, and support, which make it possible for me to go further. I wish my university to achieve its grand goal of becoming a world-class university as soon as possible.

At last, I would like to give my deep thanks to my family, especially my parents, for their love, caring, kindly support, and help. I would also like to give my deep thanks to friends for their kindly help and support.

PUBLICATIONS

The work presented in this thesis has generated the following publications:

- **Zhihao Ying**, Dave Towey, Anthony Bellotti, Zhi Quan Zhou and T. Y. Chen. Preparing SQA professionals: Metamorphic relation patterns, exploration, and testing for big data. In *Proceedings of the 2021 International Conference on Open and Innovative Education (ICOIE'21)*, 2021, pp. 22–30.
- Zhirui Zhang, Dave Towey, **Zhihao Ying**, Yifan Zhang, and Zhi Quan Zhou. MT4NS: Metamorphic testing for network scanning. In *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*, IEEE, 2021, pp. 17–23.
- **Zhihao Ying**, Anthony Bellotti, Dave Towey, T. Y. Chen and Zhi Quan Zhou. Using metamorphic relation violation regions to support a simulation framework for the process of metamorphic testing. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC'22)*, IEEE, 2022, pp. 1722–1727.
- **Zhihao Ying**, Anthony Bellotti, Joe Breeden and Dave Towey. Metamorphic Exploration for Machine Learning Validation and Model Selection. 1. Abstract from Credit Scoring and Credit Control Conference, Edinburgh, United Kingdom, 2023.
- **Zhihao Ying**, Dave Towey, T. Y. Chen and Zhi Quan Zhou. MT-PART: Metamorphic-Testing-Based Adaptive Random Testing Through Partitioning. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC'24)*, 2024, IEEE, pp. 1184–1193.
- **Zhihao Ying**, Dave Towey, Anthony Bellotti, Tsong Yueh Chen, and Zhi Quan Zhou. SFIDMT-ART: A Metamorphic Group Generation Method Based on Adaptive Random Testing Applied to Source and Follow-up Input Domains. *Information and Software Technology*, 2024, pp. 107528.

Currently under review:

- **Zhihao Ying**, Dave Towey, Anthony Bellotti, Caslon Chua, and Zhi Quan Zhou. Metamorphic Relation Patterns for Metamorphic Testing, Exploration, and Robustness. Submitted to *Software Testing, Verification and Reliability*, 2023.
- **Zhihao Ying**, Dave Towey, Anthony Bellotti, and Zhi Quan Zhou. MRGS-ART: Metamorphic Relation and Group Selection based on Adaptive Random Testing. Submitted to *Software Testing, Verification and Reliability*, 2023.

- **Zhihao Ying**, Anthony Bellotti, Joseph Lynn Breeden, and Dave Towey. Metamorphic Testing and Exploration for Machine Learning Credit Score Models. Submitted to *Applied Soft Computing*, 2024.

In Preparation:

- **Zhihao Ying**, Dave Towey, Anthony Bellotti, Tsong Yueh Chen, and Zhi Quan Zhou. Theoretical and Empirical Analyses of the Application of Metamorphic Group Generation Algorithms with Different Numbers of Metamorphic Relations. To be submitted to *International Conference on Software Engineering*.

ACRONYMS

ART Adaptive Random Testing
BART ART by Bisection
DMR Deterministic Metamorphic Relation
FTC Follow-up Test Case
FSCS-ART Fixed-Size-Candidate-Set ART
HMR Hypothesized Metamorphic Relation
ME Metamorphic Exploration
MG Metamorphic Group
ML Machine Learning
MR Metamorphic Relation
MRGSART Metamorphic Relation and Group Selection based on ART
MRGSART+ An Enhanced Version of MRGS-ART
MRGS-IPART MRGSART Through Iterative Partitioning
MRGS-RPART MRGSART by Rrandom Partitioning
MRP Metamorphic Relation Pattern
MRIP Metamorphic Relation Input Pattern
MROP Metamorphic Relation Output Pattern
MRR Metamorphic Relation for Robustness
MRT Metamorphic Robustness Testing
MRVR Metamorphic Relation Violation Region
MRVR-S STC-only component of the MRVR
MRVR-F FTC-only component of the MRVR
MT Metamorphic Testing
MT-ART MT-based ART
MT-BART MT-based ART through by Bisection
MT-IPART MT-based ART through Iterative Partitioning
MT-RPART MT-based ART by Rrandom Partitioning
IPART ART by Iterative Partitioning
RPART ART by Random Partitioning
RQ Research Question
RT Random Testing
SFIDMT-ART MT-based ART applied to Source and Follow-up Input Domains
SQA Software Quality Assurance
STC Source Test Case
SUT System Under Test

1 INTRODUCTION

The challenges in Software Quality Assurance (SQA) are growing rapidly due to the increasing complexity and sophistication within computing systems, as well as the lagging pace of SQA tools in adapting to these advanced computing systems [134]. As the core of SQA, software testing [216] should be comprehensively considered by testers throughout the entire software development life-cycle. The typical process of software testing consists of the following steps: (1) Generating test inputs based on a test-case generation algorithm; (2) executing these inputs against the SUT to generate test outputs; and (3) verifying whether or not the SUT behaves as anticipated and provides accurate outputs. The mechanism employed to verify the correctness of SUT behavior/output is commonly referred to as a (test) oracle [17, 293]. Any difference between the actual SUT behavior/output and the oracle is considered as a software failure. The test cases that trigger such differences are referred to as failure-causing test cases [17]. When the oracle is proven to be too expensive or impossible to employ, or when no oracle exists, this is a scenario referred to as the (test) oracle problem. The system that encounters this problem is referred to as an "untestable" system [17]. This constitutes a major obstacle that testers may encounter in software testing: Traditional software testing methodologies often struggle to address the oracle problem [64, 246].

Metamorphic Testing (MT) is a well-developed property-based software testing methodology [52]: It is able to not only address the oracle problem but also provide users with a new approach for generating test cases and verifying test outputs. Rather than verifying the correctness of individual test outputs through an oracle, MT identifies software failures by examining the Metamorphic Relations (MRs) among Source Test Cases (STCs), Follow-up Test Cases (FTCs), and their respective outputs [52, 64, 246]. Some basic concepts in MT are introduced as follows:

- Metamorphic Relations (MRs): Typically, testers identify a set of critical properties of an SUT based on software specifications and formally define them as MRs [64, 246]. The performance of MT is highly dependent on the MRs, but the identification of MRs is often a manual task requiring some knowledge of Metamorphic Relation Patterns (MRPs), creative thinking, and a good understanding of the SUT.

- Metamorphic Relation Patterns (MRPs): The abstractions or templates of various specific MRs and can serve as guidelines for identifying effective MRs [292]. In general, testers identify the most important factors of a set of MRs and formally define them as MRPs [292].

- Source Test Cases (STCs): The test cases generated through a specific algorithm [64, 246].
- Follow-up Test Cases (FTCs): The test cases generated through the STCs and an MR [64, 246].
- Metamorphic Groups (MGs): The combination of associated STCs and FTCs [64, 246].
- MR-Violating MGs: The MGs that violate an MR [64, 246, 284].
- Non-MR-Violating MGs: The MGs that satisfy an MR [64, 246, 284].
- Recent additions to MT literature include the following two concepts:
    1. Metamorphic Exploration (ME): Enables testers to identify MRs from the viewpoint of software users, in order to provide testers with an approach to better understand, test, and implement the SUT [292]. In addition, ME can be employed to assess whether or not the design of a function might negatively affect the user experience [292].
    2. Metamorphic Robustness Testing (MRT): Assess SUT robustness even in the presence of the oracle problem [295]. Robustness is a crucial aspect of SQA, which relates to the capability of the SUT to function correctly and produce accurate outputs even when given invalid inputs, or operating in an unforeseen environment [272].

The application of MT typically involve the following steps [52, 64, 246]:

1. Identify MRs for the SUT from scratch or select one from existing ones.
2. Generate STCs through a specific algorithm.
3. Derive FTCs according to the STCs and the MRs.
4. Execute both the STCs and the FTCs against the SUT, and examine for MR violations. A MR violation typically indicates the presence of a software failure.

Despite the increasing popularity of software testing and MT, their performance continues to require enhancement [64, 246]. This thesis aims to enhancing the performance (i.e., test effectiveness and efficiency) of MT. Specifically, it makes the following main contributions:

1. Enhances the effectiveness and efficiency of MT by identifying problems in the design, application, and evaluation of MG-generation algorithms, and proposing solutions to address them.
2. Enhances the effectiveness and efficiency of MT from the perspectives of MRs and MGs through the following measures:

a) Introduces new MRPs to facilitate the identification of specific MRs from scratch, as well as a new MT framework to assist in the identification and implementation of MRPs.

b) Introduces new MG-generation algorithms aimed at generating effective MGs from scratch.

c) Introduces a new MR-MG pair selection algorithm for the automatic and dynamic selection of effective MR-MG pairs from existing ones for execution.

3. Enhances the performance of software testing (particularly for credit risk models) through the incorporation of MT and ME as supplementary model testing, validation, and selection methodologies.

The first primary contribution of this thesis involves identifying issues in MG-generation and proposing solutions to mitigate them. In particular, while designing, applying, and evaluating MG-generation algorithms, prior published MG-generation algorithms may encounter certain challenges that could negatively affect their performance, including test effectiveness and efficiency. This thesis outlines these scenarios and formally introduces the concepts of the MT-performance evaluation problem (in Chapter 3) and the input-domain difference problem (in Chapter 4). Furthermore, this thesis presents methodologies to tackle these challenges (see Chapters 3 and 4 for details), with the aim of not only circumventing the existence of the same problems in the algorithms proposed in this thesis, but also enhancing the performance of previously-published algorithms.

Given the fact that MRs and MGs play crucial roles in the successful implementation and performance of MT, the second contribution of this thesis focuses on producing effective MRs and MGs. Specifically, this thesis presents the following methodologies:

- Firstly, this thesis concentrates on guiding the identification of MRs from scratch. Specifically, this thesis proposes a series of new MRPs targeted at facilitating the identification of effective MRs across diverse systems, and a new MT framework to facilitate the identification and application of MRPs. In addition, it introduces the concept of MRP trees (as well as two instances of MRP trees) for the classification of MRPs, with the aim of providing users with a more efficient means of searching for their desired MRPs for reuse, reference, or inference. This thesis subsequently presents findings (such as the successful detection of multiple MR violations) from three case studies that apply MRPs and the MT framework to identify effective MRs for big data systems.

- Secondly, this thesis focuses on the generation of effective MGs from scratch. Specifically, this thesis introduces three new MG-generation algorithms. This rationale behind these algorithms is to enhance MT performance by ensuring an even distribution of STCs and FTC throughout their respective input domains. Two case studies are reported evaluating and comparing the performance of these algorithms with previously-published MG-generation algorithms in the MT literature. The experimental results demonstrate that, compared with previously-published MG-generation

algorithms, the proposed algorithms exhibit better performance in terms of both efficiency and effectiveness.

- Lastly, this thesis introduces a new algorithm for the selection of effective MR-MG pairs from existing ones. More specifically, this thesis presents a new metric from a black-box perspective to define the criteria for an effective MR: Given an MG, an effective MR should aim to maximize the distance between this MG and non-MR-Violating MGs. This metric was designed based on the same rationale used for the design and development of the proposed MG-generation algorithms. Building upon this metric, this thesis introduces a novel MR-MG pair selection algorithm known as Metamorphic Relation and Group Selection based on Adaptive Random Testing (MRGS-ART). This algorithm is capable of automatically and dynamically selecting effective MR-MG pairs for execution from existing ones. A case study is reported validating and comparing the performance of MRGS-ART with the other algorithm. The experimental results demonstrate that, compared with a previously-published MR and MG selection algorithm, MRGS-ART exhibits better performance in terms of both efficiency and effectiveness.

The final contribution lies in the implementation of MT and ME as supplementary model testing, validation and selection methodologies for credit risk models. With advancements in Machine Learning (ML) technology and data accessibility, ML algorithms have gained widespread adoption in credit risk modeling within the financial industry [112]. These algorithms often exhibit superior predictive capabilities compared to traditional linear models [27, 171]. Nevertheless, the adoption of ML algorithms has introduced significant validation challenges due to the inherent complexity of these models. Consequently, this necessitates testing methodologies that may differ from those employed for linear models or are more readily applicable to linear models. Specifically, the following conditions must be satisfied:

- The decisions made using ML-based models need to be explainable.
- The ML algorithms need to be robust over time and different data segments.
- The models need to be fair and unbiased.
- The models need to match business intuition.

Extensive research efforts have been devoted to tackling the first three challenges [31, 38, 118, 171, 277, 288]; however, this thesis concentrates on the fourth challenge, which has received relatively less attention. In particular, this thesis presents a new approach for empirically assessing and validating the quality and performance of credit risk models on the basis of MT and ME. The general process of the proposed approach includes:

- Based on the business expectations hypothesized by users, predict the possible impact that certain modifications to the input may have on the output.
- Formally define them as MRs, which are so-called Hypothesized MRs (HMRs).

- Train and test credit risk models to examine for HMR violations.

That is, as an adjunct to model fit and calibration performance measures, the proposed approach verifies whether or not the output changes as anticipated according to prior application knowledge. Using the credit score as a predictor example: It would be expected that a model would predict a decrease in credit risk as the credit score increases. A case study is reported investigating the integration of conventional evaluation metrics with MT and ME for the testing and validation of credit risk models. Specifically, this thesis examines the performance of models chosen based on traditional evaluation metrics when validated using ME. Experimental results indicate that multiple HMR violations have been identified and the number of HMR violations grows as the complexity of the model increases. In this context, when ML algorithms are employed for constructing credit risk models, this thesis advocates for the use of MT and ME as supplementary tools for the testing, validation, and selection of models, in addition to traditional model fit evaluation metrics.

The remaining chapters of this thesis, as well as the connections among them, are structured as follows.

Chapter 2 provides background information relevant to this thesis.

Chapter 3 presents a novel MT simulation framework to tackle the MT-performance evaluation challenge: It is capable of swiftly and efficiently assessing MT performance from a new perspective. This chapter also introduces the concept of MR-Violation Region (MRVR) to facilitates the MT simulation framework, as well as three kinds of MRVRs that may exist in MT. A case study is reported exploring the presence of potential MRVRs in the systems taking numerical inputs.

Chapter 4 uses MRVRs to investigate the performance of previously-published MG-generation algorithms, and successfully identifies a challenge encountered by the algorithms, referred to as the input-domain difference challenge. This challenge may affect the performance (i.e., effectiveness and efficiency) of MG-generation algorithms. Subsequently, this chapter introduces a potential solution to address this challenge. By tackling the challenge within an existing MG-generation algorithm, this chapter proposes an enhanced algorithm, called MT-based ART applied to Source and Follow-up Input Domains (SFIDMT-ART), which can demonstrate superior performance in terms of efficiency and effectiveness. A case study is reported on that validates and compares the performance of this algorithm with existing MG-generation algorithms in the MT literature.

Chapter 5 introduces two new MT partitioning methods (MT by bisection and MT by iterative partitioning) to address the computational overhead problem of SFIDMT-ART (the algorithm presented in Chapter 4). This chapter then introduces two novel

MG-generation algorithms using the solution presented in Chapter 4 (in order to avoid the impact of the input-domain difference problem presented in Chapter 4). A case study is reported that shows, compared with the algorithms proposed in Chapter 4, these two algorithms are capable of achieving a better balance between efficiency and effectiveness.

Chapter 6 introduces a series of new MRPs to guide the identification of MRs. This chapter further introduces a new MT framework to facilitate the identification and application of MRPs, and the concept of MRP trees, as well as two instances of MRP trees, to facilitate the selection of MRPs. A case study is reported applying MRPs, MRP trees, and the MT framework to identify MRs for big data systems.

Chapter 7 further improves the solution presented in Chapter 4, and uses it, as well as the MT partitioning method (MT by bisection) introduced in Chapter 5, to designs a novel metric (which can guide the design and development of MR-MG selection algorithm) and an MR-MG pair selection algorithm, called Metamorphic Relation and Group Selection based on Adaptive Random Testing (MRGS-ART). A case study is reported on that evaluates the performance of MRGS-ART and compares it with an existing algorithm in the MT literature.

Chapter 8 introduces the implementation of MT and ME as a supplementary methodology for testing, validating and selecting credit risk models. A case study is reported applying the proposed approach, in conjunction with a traditional model evaluation method, to validate popular credit risk models.

Chapter 9 concludes the thesis, summarizing the contributions of this thesis, and discussing potential limitations and future research directions.

2

LITERATURE REVIEW

2.1 CONVENTIONAL SOFTWARE TESTING

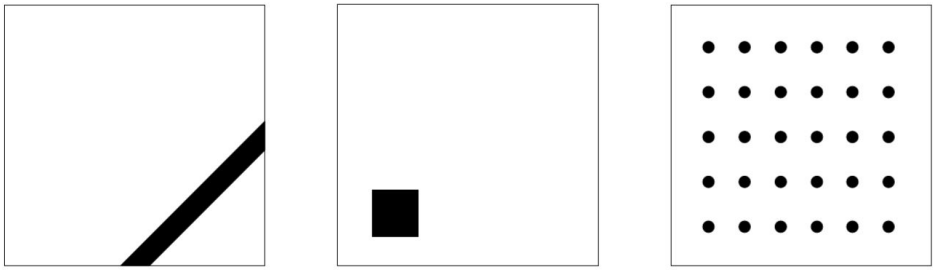
2.1.1 *Overview*

In conventional software testing, errors committed by program developers can lead to faults within the program; and a failure may be detected if a fault is encountered during the execution of the system [146]. In particular, following the execution of a test case, a failure is identified if the output or behavior of the SUT differs from the expected outcome. The executed test cases that result in software failures are termed failure-causing test cases, while those that pass are referred to as non-failure-causing [64, 140]. The ratio of failure-causing test cases to the total number of possible test cases within the input domain determines the failure rate [64, 140].

Broadly speaking, conventional software testing algorithms can be categorized into two types: Black-box and white-box testing [128, 185]. Black-box testing, referred to as functional testing, generates new test cases according to software specifications [185]. In black-box testing, testers lack access to the source code of the SUT. White-box testing, also known as structural testing, generates new test cases according to the source code of the SUT [185]. In other words, testers have visibility into the source code of the SUT in white-box testing.

2.1.2 *Random Testing*

Random Testing (RT) (also known as monkey testing) stands as one of the most popular black-box software testing technique [123]. It automatically generates random and independent test cases for the SUT without reference to any system design or source code [25, 222]. The concept of RT was first proposed by Melvin Breuer in 1971 [5], who employed it for hardware examination. Subsequently, Prathima et al. [5] conducted the first study to validate the effectiveness of RT in 1975. Hamlet [123] reported that due to RT's lack of systematic selection of new inputs, there exists no correlation between the inputs generated by RT. In the IEEE Guide to the Software Engineering Body of Knowledge [25], RT is listed as one of four popular input-domain based testing algorithms (as well as equivalence partitioning, pairwise testing and boundary value analysis). Orso et al. [222] categorized RT as one of four most popular test-case generation algorithms, alongside symbolic execution algorithm, search algorithm and hybrid algorithm. Various RT algorithms have been



(a) Strip pattern (b) Block pattern (c) Point pattern

Figure 1: Examples of failure regions in 2D input domains [140]

developed for better identifying failures, with two notable approaches being uniform distribution and non-uniform distribution. Uniform distribution [83] means that all test cases in the input domain have an equal probability of selection, while non-uniform distribution [63] means that the test cases in the input domain do not have an equal probability of selection. It should be noted that sometimes a test-case distribution approach may be too expensive or even impossible to be applied. For instance, adaptive random testing [77, 140] attempts to improve the performance of software testing by achieving an even distribution of test cases over the input domain. However, this distribution approach may be too expensive to be used when the input domain has an irregular polygon shape, or is unavailable where the similarity/distance between test cases cannot be calculated (e.g., non-numeric test cases). At this point, additional guards are required to ensure the running of the testing. For example, common guards include expanding the distribution approach, switching to other approaches, or involving human participants.

Studies have been carried out to validate and compare the efficiency and the effectiveness of RT with several state-of-the-art software testing algorithms, revealing RT as an efficient test-case generation algorithm [251, 275]. For instance, Sharma et al. [251] reported on a case study validating and comparing the performance of RT and shape abstraction (an approach for generating test containers) [275]. They found that RT achieved comparable effectiveness to shape abstraction, but with significantly lower computational overheads. One of the main advantages of RT is its ease of understanding and use. RT has been proved to be a valuable algorithm in numerous fields, such as secure software systems [178]; embedded software systems [235]; Java and .NET libraries [224]; database systems [20]; windows NT systems [113]; SQL database systems [20]; Android systems [213]; and MT [246].

2.1.3 *Adaptive Random Testing*

2.1.3.1 *Overview*

It has been argued that RT may sometimes be not efficient since it does not make any use of the features of the SUT [246]. Chen et al. [77] introduced Adaptive Random Testing (ART) to address this limitation by utilizing the generic information on common fault patterns within the SUT. It has been reported that failure-causing test cases often cluster

into contiguous regions known as failure regions [9, 24, 111, 240, 279]. Chan et al. [39] identified and classified three types of failure regions: Block, point, and strip. Fig. 1 provides typical examples of these regions within a 2D input domain, accompanied by the following explanations [39, 140]:

- Strip region: Failure-causing test cases typically cluster in a narrow and contiguous area.
- Block region: Failure-causing test cases typically cluster in a single contiguous area.
- Point region: Failure-causing test cases dispersed throughout the input domain, either individually or in small groups.

According to Chan et al. [39], the block region is the most common failure region. In this context, the neighboring test cases of a failure-causing test case are likely to also be failure-causing, while the neighboring test cases of a non-failure-causing test case are likely to remain non-failure-causing [39, 140]. That is to say, the test cases chosen from the regions distant from non-failure-causing test cases are more likely to be failure-causing. Building on this insight, Adaptive Random Testing (ART) was proposed: It aims to improve fault-detection capability by ensuring an even distribution of test cases throughout the entire input domain [77, 83, 140].

Different kinds of ART algorithms have been collected and categorized by Huang et al. [140], summarized as follows:

- Select-Test-From-Candidates [40, 77]: This, in general, selects a set of inputs, named the candidate set, and then selects new inputs from the candidate set according to some specific criteria and executed test cases. It consists of two kinds of algorithms, namely Fixed-Size-Candidate-Set ART Algorithm [77] and Restricted Random Testing (RRT for short, which constructs new inputs according to some predefined restriction criteria) [40].
- Partitioning-based [57, 78, 86]: This combines the partition testing with ART to generate new inputs, which divides the entire input domain into several subdomains based on a predefined criterion, and then selects one within which to generate new inputs.
- Test-Profile-based [183, 184]: It selects new inputs according to a dynamic test profile.
- Quasi-Random ART [80, 180]: This combines the quasi-random sequences (points with low discrepancy and dispersion) with ART based on the findings reported by Chen et al. [60] that, generally, points with lower discrepancy and dispersion have the ability to achieve a wide distribution of inputs throughout the entire input domain.
- Search-based [125, 207]: This combines the search-based software testing techniques with ART to select new inputs that can achieve a wide distribution.
- Hybrid [59, 61]: This combines several different ART algorithms to improve test efficiency or effectiveness.

2.1.3.2 *Select-Test-From-Candidates Algorithms*

The Fixed-Size-Candidate-Set ART (FSCS-ART) [77] stands as the first ART algorithm and falls under the category of select-test-from-candidates algorithms [140]. FSCS-ART aims to improve test effectiveness by utilizing the test cases that have already been executed against the SUT, but without resulting in any failure. To implement FSCS-ART, firstly, a set of test cases is generated as candidates from the input domain at random. Subsequently, FSCS-ART selects one of the candidates based on the distance between each candidate and all non-failure-causing test cases. FSCS-ART stands as the most popular ART algorithm owing to its simplicity and effectiveness in detecting failures [140]. Nonetheless, FSCS-ART may encounter the edge preference problem [140], wherein it exhibits a preference for generating new test cases near the boundary of the input domain. To address this problem, numerous techniques have been proposed, with one of the most effective being FSCS-ART with partitioning by edge and center [59, 73, 143]. This algorithm partitions the input domain from the edge to the center according to the number of executed test cases and selects k $(k > 0)$ candidates from the empty subdomains. It subsequently calculates the distances from each candidate to its nearest executed test case and selects the candidate with the maximum distance as the next test case. Its time complexity is $O(n^2)$ [59, 73].

2.1.3.3 *Partitioning-based Algorithms*

Select-test-from-candidates algorithms, such as FSCS-ART, often encounter a significant challenge: Their time complexities are generally too high. To address the problem of computational and comparison overheads, a series of partition-based ART algorithms have been designed and developed [57, 78]. Partition-based ART algorithms dynamically partition the input domain into subdomains according to specific criteria and then select one for the next test case generation. Chen et al. [57] introduced two partition-based ART algorithms: ART by Random Partitioning (RPART) and ART by Bisection (BART): RPART divides the input domain using executed and non-failure-causing test cases and then generates the next test case randomly in the largest subdomain; and BART dynamically partitions the input domain into equally sized subdomains and selects the next test case from a subdomain containing no executed and non-failure-causing test cases.

Later, Chen et al. [57] introduced a new test-case generation algorithm named ART through Iterative Partitioning (IPART): For example, given n executed and non-failure-causing test cases in a 2D input domain, it firstly divides the input domain into $n * n$ subdomains. Then, it randomly selects a test case in a subdomain devoid of any non-failure-causing test cases and not surrounded by subdomains containing any non-failure-causing test cases. According to Chen et al. [57], IPART may also encounter the edge preference problem.

2.1.3.4 *Advantages and Shortcomings*

Nowadays, ART has been widely used as test-case generation techniques for various areas [140]. Empirical studies together with theoretical discussions have been conducted to

explore and compare test efficiency and effectiveness of ART with other test-case generation algorithms, such as RT [140]. In general, ART has the followings advantages and shortcomings:

1. Enhanced Input Spread: In general, ART achieves a better input distribution throughout the entire input domain than RT [58, 60, 62, 164, 179, 180, 183, 184].

2. Enhanced Test Coverage: Many studies [34, 35, 65, 66, 135, 168] have explored and reported that ART algorithms are able to achieve better test coverage than RT.

3. More Accurate Software Reliability Estimation: It has been reported that the test coverage enhanced method can provide a better software reliability estimation [50, 51]. Therefore, since ART can typically achieve greater code coverage than RT, ART may be a better choice than RT for improving the software reliability estimation of the SUT.

4. Enhanced Effectiveness and Efficiency: Huang et al. [140] stated that, in general, ART effectiveness refers to its fault-detection-capability, while its efficiency consists of test case selection and execution time. F-measure (how many inputs have been used to reveal the first failure) [69], E-measure (how many failures have been detected by a set of inputs) [70] and P-measure (the probability of a group of inputs detecting no less than one failure in the SUT) [70] are three common metrics employed to assess ART effectiveness [140]. F-time [86] is a widely-used metric for assessing ART efficiency: It is a metric of the CPU execution time (including all steps involved in implementing an ART algorithm) until the first failure is detected. Chen et al. [76] explored and identified four factors that may positively influence ART effectiveness: (a) A low fault ratio; (b) a compact failure region; (c) few failure regions; and (d) a predominant area over the entire failure region.

   If at least one of these elements is satisfied, in general, ART is capable of achieving better effectiveness than RT. In general, on the basis of test effectiveness, ART uses fewer test inputs to reveal the first failure (F-measure) than RT. For example, previous studies [53, 72, 78] assessed and compared the effectiveness of ART and RT based on F-measure, and the experimental results indicated that ART was capable of outperforming RT with an F-measure up to about 50% smaller than RT. As for test efficiency, compared with RT, ART naturally requires higher computing costs to construct the same number of inputs, which is known as the computational overhead problem [10]. FSCS-ART and RRT generally cost about $O(n^2)$ and $O(n^2 log(n))$ to generate n inputs, respectively [77, 203]. However, firstly, compared to ART, RT generally requires more time to reveal the first failure (F-time) [10, 138, 175, 186, 248]. Secondly, many hybrid methods have been designed to reduce the computational overheads of ART, such as the improvements on FSCS-ART [59, 61, 71, 72, 88, 138, 162, 192, 199, 220]. Furthermore, other enhancements of ART that combine ART methods with the techniques from other areas have been designed and developed to alleviate this problem [15, 41–43, 49, 79, 138, 223]. For instance, Chan et al. [42, 43] presented an approach that uses a square exclusion area version of RRT to alleviate this problem.

5. Broad Applicability: Huang et al. [140] stated that ART is a widely-used algorithm for generating test cases and has been employed in various fields, such as simulations and models [195, 259], numerical systems [12, 65, 77, 276, 290], embedded software programs [130–132, 145] and object-oriented systems [47, 89, 175, 231]. ART has been employed to enhance the efficiency and effectiveness of various other software testing techniques, such as regression testing [286], reliability testing [215], MT [19, 143, 144] and combinatorial testing [219]. For instance, Barus et al. [19] conducted a comparison between RT and ART in terms of their effectiveness in generating STCs for MT. The experimental results revealed that ART outperformed RT in enhancing the effectiveness of MT.

6. While ART offers numerous advantages over RT, it still faces some shortcomings and challenges, including the edge preference problem [55, 140] and the high dimension problem [60, 75, 140].

   Some ART algorithms, such as FSCS-ART and RRT, may tend to produce new inputs closer to the edges rather than the centre of the entire input domain, particularly under conditions of high fault ratio and dimensionality [55]. Numerous methods have been explored and developed to mitigate the edge preference problem: For example, some studies [59–61, 63, 163, 166] increased the selection probability of inputs near the centre of the input domain rather than those at the edges, while others [55, 57, 116, 198, 202, 204] have addressed the problem by connecting or extending the edges.

   While some ART algorithms, such as partition-based ones [60, 184], remain unaffected by the dimensionality, a higher dimensionality (typically exceeding two) may occasionally result in poorer performance of some ART algorithms (i.e., FSCS-ART [58, 60, 62, 164] and RRT [60, 184]) compared with RT, owing to the curse of dimensionality [22]. This constitutes one of the primary challenges encountered by ART, known as the high dimension problem [60, 75, 140]. Although no technique has been devised to completely address this challenge, several have been introduced to alleviate it. It has been reported that, in high-dimensional input domains, failure regions are prone to clustering at the center [54, 200]. Consequently, the edge preference problem may have an impact on the high dimension problem, and solutions [54, 55, 57, 59, 61, 63, 116, 141, 163, 164, 166, 198, 200, 202, 204] employed to tackle the edge preference problem might also help alleviate the high dimension problem. Nevertheless, the effectiveness of these solutions is still influenced by the input dimensions. Additionally, Schneckenburger et al. [241] integrated the Hill-Climbing algorithm with continuous distance [203], and presented a search-based ART algorithm capable of alleviating the high dimension problem.

In summary, RT is a widely-adopted software testing algorithm for generating test cases, while ART is designed and developed as an improvement upon RT. The rationale behind ART is that it attempts to improve RT performance by achieving a more uniform distribution of inputs throughout the entire input domain. In comparison to RT, ART typically exhibits four primary strengths: Better test-input spread, better test coverage, more accu-

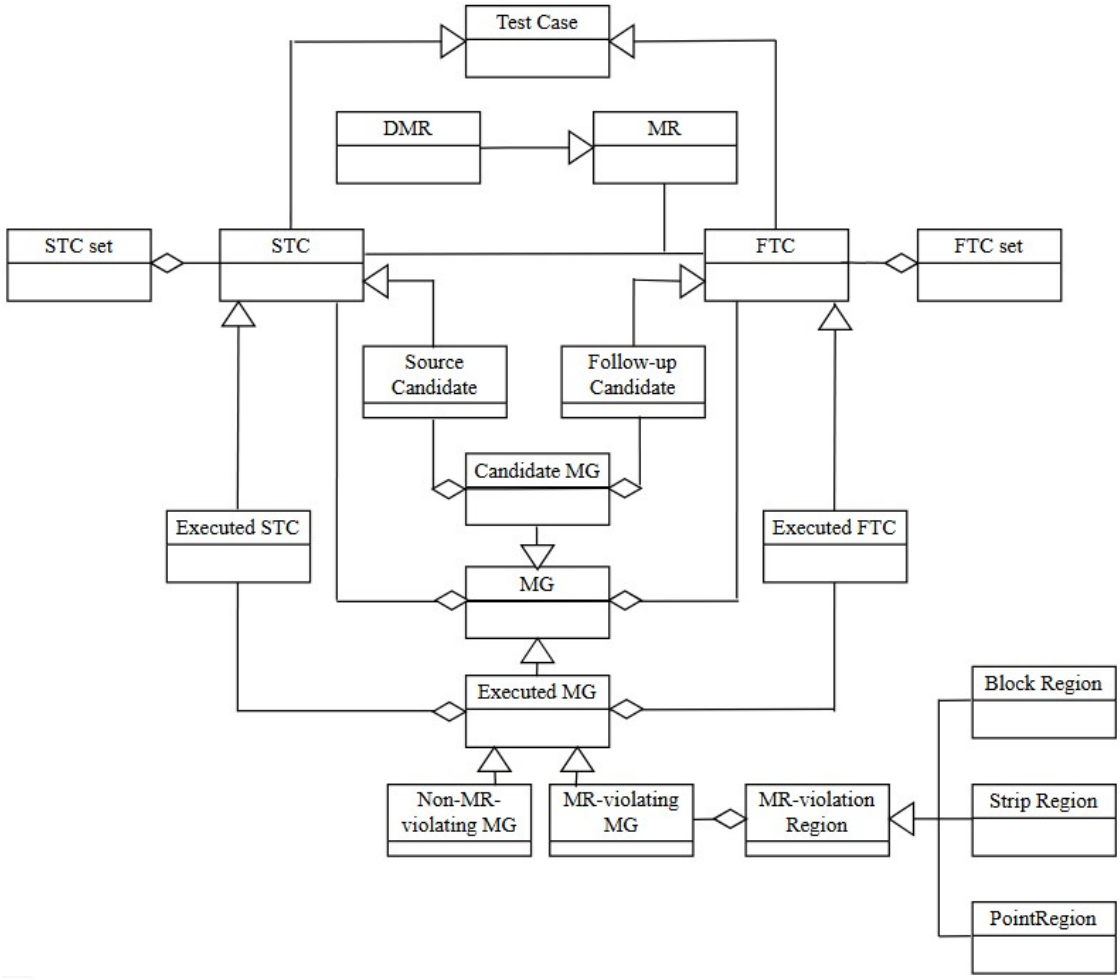


Figure 2: An MT Class Diagram

rate software reliability estimation, and better test efficiency and effectiveness. Despite the existence of several remaining challenges for ART that require resolution (i.e., the computational overhead problem, the high dimension problem, and the edge preference problem), ART generally represents a superior option for test case generation compared to RT.

2.2 METAMORPHIC TESTING

2.2.1 *Overview*

Metamorphic Testing (MT) introduces a novel approach to test case generation and test result verification, through the use of Metamorphic Relations (MRs). MRs represent the essential properties (denoting the conditions logically inferred from) the SUT, involving at least two inputs and their corresponding outputs [52, 64]. In order to clearly elucidate the interconnections among various MT concepts, Fig. 2 depicts an MT class diagram. MT comprises precisely two sets of test cases: STCs and FTCs. In contrast to the conventional approach of individually verifying each test case's outcome using a test oracle, MT assesses the satisfaction of the relationship between the STCs and the FTCs as well as their outputs

[52, 64]. Given an MR, Metamorphic Groups (MGs) denote the combination of correlated STCs and FTCs [64]. MR-violating MGs indicate the MGs that violate an MR, while those satisfying an MR are denoted as non-MR-violating MGs [64]. According to the quantity of STCs and FTCs, MRs generally can be classified into four categories, listed as follows [64]:

- 1-1 MR: Only one STC is employed to produce a single FTC.
- 1-N MR: Only one STC is employed to produce N FTCs ($N > 1$).
- M-1 MR: M STCs ($M > 1$) is employed to produce a single FTC.
- M-N MR: M STCs ($M > 1$) is employed to produce N FTCs ($N > 1$).

It is noteworthy that certain MG-generation algorithms [143, 144] do not directly generate new MGs. Instead, they firstly generate STCs from the input domain as source candidates, and subsequently derive FTCs as follow-up candidates based on the MRs and source candidates. Finally, these algorithms select source and follow-up candidates for execution according to specific criteria.

MT has been employed to alleviate the oracle problem in various fields, including ML classifiers [214, 218, 281], network scanning [289], embedded systems [44, 151, 165], security [68], and web services [45, 263, 285]. Furthermore, over the past few decades, besides software testing, MT has been successfully applied in many different fields, such as software debugging [85, 152] and education [269, 285].

2.2.2 *Metamorphic Relation Patterns*

A great number of MRs have been identified across various application fields; however, the majority of published MRs have been identified arbitrarily and ad-hoc, lacking a systematic mechanism [64, 246]. Consequently, systematic MR identification remains a significant challenge for the MT community [64, 246].

In order to tackle this challenge, Zhou et al. [294] introduced the concept of general MRs, which are abstract representations of MRs and serve as guides for identifying concrete MRs. They reported on a case study applying the general MRs to guide concrete MR identification, with the experimental results indicating that the identified MRs effectively detected software failures.

Segura et al. [247] proposed the formal concept of Metamorphic Relation Output Pattern (MROP), the definition for which is as follows:

- *Definition 2 (Metamorphic Relation Output Pattern (MROP))*: An abstract relation among multiple source outputs and corresponding follow-up outputs that can guide concrete MR identification of when combined with proper input relations.

More recently, Zhou et al. [292] advanced this research by proposing the formal concept of Metamorphic Relation Pattern (MRP) and Metamorphic Relation Input Pattern (MRIP). Their formal definitions are:

- *Definition 3 (Metamorphic Relation Pattern (MRP))*: An abstraction that encapsulates a group of (potentially infinitely many) MRs [292].
- *Definition 4 (Metamorphic Relation Input Pattern (MRIP))*: An abstraction that describes the relationships among various STCs and FTCs within a group of (potentially infinitely many) MRs [292].

Constructing an MRP involves identifying the important factors of a given set of MRs while ignoring all the details [292]. The relations among multiple MRPs may form a hierarchy [191, 292]. In particular, MRPs vary in abstraction levels, with higher levels being more abstract and lower levels being more concrete. MRs are the specific relations derived from these abstractions. For instance, a human may have the following different levels of abstraction: (1) Organism; (2) Human; (3) Age/Gender/Nationality; and (4) Infancy/Childhood/Adolescence/Adulthood.

Here is an example to illustrate the differences between an MR and an MRP. Using a simple program $F(x)$ as an instance. This program calculates the square value of an input x. The following two MRs are identified for the program $F(x)$ as example.

In the domain of $F(x)$ Function
assuming that

- The input contains one parameter: $x(x > 0)$.
- The output consists of one parameter: $F(x)$.

the following MR(s) should hold

- MR_a:

  if $x_2 = x_1 + 1$,
  then $F(x_2) > F(x_1)$.

- MR_b:

  if $x_2 = x_1 + P(P > 0)$,
  then $F(x_2) > F(x_1)$.

MR_b can also be used to identify new MRs by varying the values of P, such as MR_a when P equals 1. However, MR_b does not belong to MRPs. In particular, MRPs concentrate solely on important factors, and the identification of potential MRs using MRPs still demands some degree of creative thought. On the other hand, for the aforementioned two MRs, one important factor suitable for constructing MRPs, for instance, is monotonicity. In this scenario, one of the specific elements overlooked by MRPs but accounted by MR_b, for instance, is how to achieve the monotonicity. That is, there could be various methods to achieve monotonicity, necessitating a degree of creative thinking, and these methods should not be taken into account during the construction of MRPs; rather, they should be incorporated when identifying MRs. For example, altering the method of achieving monotonicity can produce the following two common MRs:

In the domain of $F(x)$ Function
assuming that

- The input contains one parameter: $x(x > 0)$.
- The output consists of one parameter: $F(x)$.

the following MR(s) should hold

- MR_c:

  if $x_2 = x_1 * Q(Q > 1)$,
  then $F(x_2) > F(x_1)$.

- MR_d:

  if $x_2 = x_1 / R(R > 1)$,
  then $F(x_2) < F(x_1)$.

In summary, MRPs can provide guidance for the identification of MRs, but the use of MRPs still requires the user's innovative thinking; instead, MR_b does not provide any guidance, and identifying new MRs through MR_b does not require innovative thinking: It is quite obvious how to identify new MRs according to MR_b.

2.2.2.1 *Existing MRPs in the Literature*

Zhou et al. [292] introduced the *Symmetry* MRP and the *Change Direction* MRIP (Metamorphic Relation Input Pattern), drawing from symmetry concepts evident in diverse fields like nature and mathematics. *Symmetry* MRP guides the identification of MRs based on the perspectives from which the SUT appears identical. Some concepts, like "symmetry", require no explicit explanation in the definition. When applying this MRP to a specific field to construct concrete MRs, testers may interpret these concepts diversely. For example, an e-commerce system should produce the identical amount of items for queries that vary solely based on the sorting rule. *Change Direction* MRIP guides the identification of MRs by altering the direction of source inputs to derive follow-up inputs. For instance, e-commerce systems commonly have two sorting rules that contain directional elements: Price from low to high and price from high to low.

Wu et al. [280] introduced a new MRP, termed *Noise* MRP, which was inspired by the *Symmetry* MRP: It asserts that minimal interference (noise) should not have a severe impact on the behavior/output of a robust system. Similarly, testers may interpret certain concepts (such as "performance" and "noise") diversely when applying this MRP to a specific field.

An input query defines how data is retrieved and displayed. Users are able to use queries to search for information and to filter or sort outputs based on particular rules. For instance, on Amazon, the possible values for filter function include "Women's Fashion Department" and "Men's Fashion Department", while the possible values for sorting function are "Price from Low to High" and "Price from High to Low". Segura et al. [245] term subject systems supporting queries as query-based systems and introduced the following MRPs for such systems:

- *Input Equivalence*: A robust query-based system should yield identical outputs for inputs accepting fully identical values expressed differently (i.e., “1 minutes” compared with “60 seconds”).
- *Shuffling*: Reversing the sorting rule in the inputs (i.e., “price from low to high” compared with “price from high to low”) should not affect the quantity of the items in the outputs.
- *Conjunctive Conditions*: If incorporating a new conjunctive condition into the source inputs to generate follow-up inputs (i.e., a query “shoes” without filters compared with a query “shoes” filtered by a specific brand like “Nike”), then the follow-up outputs should be contained in the source outputs.
- *Disjunctive Conditions*: If incorporating a series of new disjunctive conditions into the source inputs to generate follow-up inputs (i.e., a query “PC” compared with a query “‘PC’ OR ‘mobile phone’”), then the follow-up outputs should contain the source outputs.
- *Disjoint Partitions*: If partitioning the input domain of one or more parameters of the source inputs to generate follow-up inputs, this MRP represents the MRs that the source and follow-up outputs are pairwise disjoint. For instance, a robust query-based system should return completely different outputs for the a query “shoes” filtered by a brand like “Adidas” and the same query filtered by a different brand like “Nike”.
- *Complete Partitions*: If partitioning the input domain of one or more parameters of the source inputs to generate follow-up inputs, this MRP represents the MRs that the follow-up outputs are contained in the source outputs. For instance, the outputs of the query "shoes" without filters on Amazon should include the outputs of the same query "shoes" filtered by "Men’s Fashion".
- *Partition Difference*: If partitioning the input domain of one or more parameters of the source input to generate follow-up inputs, this MRP represents the MRs that the follow-up outputs are pairwise disjoint from each other, while their union equals the source output. For instance, the source input is the query "shoes" filtered by "‘Nike’ and ‘Adidas’", while the two follow-up inputs are the query "shoes" filtered by "Nike" and "Adidas" separately.

REpresentational State Transfer (REST) is a software architecture style used in distributed hypermedia systems [110]. Computer systems are able to communicate with each other based on the rules defined by an API (Application Programming Interface). Web APIs following architectural constraints is capable of securely facilitating the exchange of information among computer systems [110]. Segura et al. [247] proposed six MROPs specifically for this kind of Web API:

- *Equivalence*: Both the source output and the follow-up output contain identical content irrespective of order.
- *Equality*: Both the source output and the follow-up output contain identical content with the same order.
- *Subset*: The source output contains all follow-up outputs.
- *Disjoint*: The source output and the follow-up output are pairwise disjoint to each other.
- *Complete*: Both the source output and the collective output of all follow-ups contain identical content irrespective of order.
- *Difference*: The intersection of the source output and the follow-up output must differ from their combination.

Recently, in addition to the concept of MRP, several studies have concentrated on summarizing the shared properties of multiple MRs. For instance, Hui et al. [142] introduced a formal model to describe MRs using predicate logic. They reported that an MR can be defined as a composed relation in predicate logic and proposed a decomposition model.

$$MR = [IR, SF, OR], \tag{1}$$

- Input-Relation (IR) represents the relation between source and follow-up inputs.
- Self-Relation (SR) represents the SUT.
- Output-Relation (OR) represents the relation between source and follow-up outputs.

The problem is that some MRs cannot be subdivided into input-only and output-only sub-relations [64]. For instance, given a system $S(x)$ that calculates the square value of an integer x ($x > 0$). The following are two MRs identified for this SUT:

- MR_1: If $x_2 = x_1 + 1$, then $S(x_2) > S(x_1)$.
- MR_2: If $x_2 = x_1 + S(x_1)$, then $S(x_2) > x_1 + S(x_1)$.

MR_1 can be divided into input-only and output-only sub-relations, and described using the decomposition model from Hui et al. [142]. However, MR_2 cannot be divided into input-only and output-only sub-relations like MR_1, as the calculation of the follow-up input x_2 relies on $S(x_1)$ (the output of the source input x_1), and the relation between outputs also involves the source input x_1. In this context, the decomposition model is only applicable to a specific subset of MRs, not all. This proves that not all MRs as composed of relations in predicate logic. This differs from MRPs, as each MRP possesses varying abstraction levels, and various MRPs are able to form a hierarchy.

2.2.3 *Metamorphic Exploration and Metamorphic Robustness Testing*

Zhou et al. [292] recently introduced the concept of Metamorphic Exploration (ME): It aims to improve users' SUT understanding, enabling its improved utilization without the requirement of a comprehensive user manual. The authors further noted in the same paper that in ME, MRs need not necessarily be properties of the SUTs; rather, they can be properties hypothesized by users, termed hypothesized MRs (HMRs). HMR violations can provide users with a novel means to comprehend SUT behavior and empower them to take measures to achieve their desired outcomes. For example, users can generate specific STCs and FTCs for a particular MR/HMR. Additionally, testers can apply ME to ascertain whether or not a function is designed to meet the needs of users. In summary, ME offers users a novel approach to exploring the SUT, aiming to enhance users' comprehension of its behavior/output and consequently improve its utilization.

More recently, the concept of Metamorphic Robustness Testing (MRT) [295] has emerged. MRT can assess the robustness of an SUT even in the presence of the oracle problem. Robustness, as a core component of SQA, signifies the ability of an SUT to manage invalid test cases or operate in unforeseen environments [272, 295]. In studies related to MRT [170, 295], MRs are identified to assess the robustness of SUTs rather than to detect software failures, specifically evaluating the SUT's capacity to manage erroneous inputs [117, 272]. These MRs are termed Metamorphic Relations for Robustness (MRRs).

2.2.4 *Advantages and Disadvantages of Metamorphic Testing*

The advantages of MT are listed as follows:

1. MT is known for its simplicity and ease of use. The structure and definition of MT are simple, making it accessible for users to comprehend and implement. Liu et al. [181] and Poon et al. [228] demonstrated that the testers unfamiliar with MT can learn to identify basic MRs and implement MT within several hours across various systems. Le et al. [169] reported the successful identification of a great number of bugs in two widely-used compilers using a straightforward MR, which can be easily identified by even MT beginners.

2. Automating the primary components of MT and developing MT tools is straightforward. In addition to MR identification, testers can easily automate other core components of MT, such as test case generation, implementation, and validation. Existing MT-related studies have extensively explored the generation of STCs. Previous studies indicated that test case implementation is readily automatable [64]. Various methods have been designed and implemented to automate the validation process [64, 246]. Nevertheless, there is still one challenge: MR identification remains dependent on human participants. Significant effort has been dedicated to addressing this challenge, leading to the development of several techniques to mitigate it. MRP [292] is a typical example. Zhou et al. [292] also reported on an empirical study involving

65 popular electronic commercial websites, which revealed many MR violations and failures. Furthermore, some studies [263, 298] have indicated that developing MT tools is straightforward and uncomplicated for testers.

3. Broad applicability. Segura et al. [246] conducted an exhaustive survey of 119 MT-related publications, revealing its widespread adoption across various application domains. They reported that MT is a popular approach for addressing the oracle problem and has been employed in diverse fields.

4. Low costs (compared with conventional software testing techniques). One cost-saving aspect is the identification of MRs. As reported by Chen et al. [64], although the involvement of human participants in the process of MR identification may lead to additional computational overheads, they are generally unavoidable and acceptable. Similar efforts are common in conventional software testing, such as the assertion identifications. Additionally, Chen et al. [64] noted that the additional overheads of test output validation, such as verifying outputs against MRs, are lower than those associated with validation using the oracle.

The disadvantages of MT are listed as follows:

1. The performance of MT largely depends on the MRs, but the identification of MRs is often a difficult and time-consuming task [64, 246]: It can require a good understanding of the SUT, creative thinking and some knowledge of MRPs.

2. Traditional software testing can verify the correctness of individual test cases. MT, without an oracle, cannot determine which test case (STC or FTC, or both) are producing the wrong output [64, 246]. It can only identify that there is an MR violation. Likewise, if the MG's test cases produce the wrong outputs, but (for example) both the wrong STC output and the wrong FTC output are the same (with an equality relation), then MT will not identify the violation.

2.2.5 *MT Test Case Generation*

Chen et al. [84] integrated fault-based testing [211] into MT to construct new STCs automatically. Gotlieb and Botella [120] introduced a new method called automated MT. This method is able to construct new STCs automatically by translating the initial code of the SUT into an equivalent constraint logic system [147] and then searching for new STCs that can violate the MRs. Alatawi et al. [8] integrated the dynamic symbolic execution into MT to construct STCs automatically. More specifically, the proposed method generate STCs by recording the system execution behavior influenced by the MGs and storing the symbolic constraints at each branch point. After the executions, the method constructs new inputs that are able to execute the target system along a different path. Lindvall et al. [177] combined MT with model-based testing methods to construct MGs automatically. Batra et al. [21] designed a genetically enhanced MT method for generating a small set

of efficient STCs by combining MT with genetic algorithms. Saha et al. [238] validated the performance of code-based software testing methods (e.g., line coverage, branch coverage and weak mutation) on generating STCs for MT. Barus et al. [19] applied FSCS-ART to generate STCs for systems taking non-numerical inputs, and used the category-choice framework [223] to compute the distance between the inputs. Cao et al. [37] demonstrated a robust and statistically significant correlation between the effectiveness of MRs and the dissimilarity between STCs and FTCs, particularly when employing branch coverage.

Hui et al. [143] introduced ART into MT to generate MGs for MT and proposed an MG-generation algorithm termed Metamorphic Distance-based ART. This algorithm aims to enhance the fault-detection effectiveness of MT through ensuring an even distribution of STCs across the entire input domain. Additionally, its time complexity is $O(n^2)$ [143, 144].

Subsequently, Hui et al. [144] reported that considering FTCs during the MG-generation process can lead to better MT performance. In this context, Hui et al. [144] introduced a novel MG-generation algorithm called Metamorphic Testing-based Adaptive Random Testing (MT-ART), as an improved iteration of metamorphic distance-based ART. Both metamorphic distance-based ART and MT-ART aim to improve the performance of MT (especially the test effectiveness) from the perspective of black-box testing. In particular, MT-ART aims to improve MT performance by achieving an uniform distribution of both STCs and FTCs throughout the entire input domain. For this purpose, Hui et al. [144] proposed distance metrics to facilitate this algorithm. In particular, they categorized the distance between the test cases in MT into three types, referred to as metamorphic distances:

d_1: The distance between a source candidate and all executed STCs and FTCs.

d_2: The distance among (source and follow-up) candidates from the same candidate MG, including the distance between two source candidates, the distance between two follow-up candidates, and the distance between a source candidate and a follow-up candidate.

d_3: The distance between a follow-up candidate and all executed STCs and FTCs.

The computation of metamorphic distances uses the Euclidean distance.

In the MG-generation process, MT-ART firstly partitions the entire input domain into subdomains with identical sizes according to the number of all executed STCs and executed FTCs. Subsequently, it selects a sequence of STCs from the empty subdomains and generates FTCs based on the chosen STCs and the specified MR, with the aim of obtaining a set of MGs. Then, MT-ART calculates the metamorphic distances between test cases and chooses the MG with the greatest distance for execution. As reported by Hui et al. [144], its time complexity is $O(n^4)$. The computation of metamorphic distances can be performed in three ways: The Maximum (Max), Average (Avg), or Minimum (Min) distance. Additionally, two strategies are available for selecting the next MG: Strategy 1 (prioritizing d_1 and d_3 followed by d_2) and Strategy 2 (prioritizing d_2 followed by d_1 and d_3). As a result, there exist six MT-ART algorithms: MT-ART with Strategy 1 and Max distance;

MT-ART with Strategy 1 and Avg distance; MT-ART with Strategy 1 and Min distance; MT-ART with Strategy 2 and Max distance; MT-ART with Strategy 2 and Avg distance; and MT-ART with Strategy 2 and Min distance. Generally, MT-ART surpasses metamorphic distance-based ART based on effectiveness, efficiency, and code coverage [144].

2.2.6 *MR and MG Selection*

The following metrics have been introduced to define the properties of effective MR:

- Effective MRs should aim to maximize the difference between the behavior of STC execution and that of corresponding FTC execution [56]. This metric has garnered support from various individual studies [14, 37, 99, 101, 167, 181, 282]. Asrafi et al. [14], for example, reported that improving the combined code coverage of the STCs and the FTCs may increase the difference between SUT execution behaviors. Through a large-scale empirical study, Cao et al. [37] reported that from a white-box-coverage perspective, the test effectiveness (i.e., fault-detection capability) of MRs has a strong and statistically significant correlation with the dissimilarity between the STCs and the FTCs.
- The MRs identified according to specific components of the SUT are more likely to outperform those identified according to the entire SUT in terms of test effectiveness (i.e., fault-detection capability) [153, 154]. This metric was subsequently confirmed by Xie et al. [282].
- Chen et al. [56] proposed the role of white-box considerations in guiding MR identification. This metric was subsequently confirmed and formally presented by Mayer and Guderlei [201].
- Mayer and Guderlei [201] proposed four general metrics for quickly (but approximately) assessing potential MR performance. These four metrics can guide, not only the identification, but also the selection, of MRs.

According to these metrics, several techniques have been designed and proposed to guide the selection of effective MRs. Ding et al. [99] introduced a technique known as self-checked MT to assess MR quality. This approach quantifies the code coverage of STCs and FTCs throughout the MT process by integrating MT with structural testing. The rationale behind self-checked MT aligns with one of the aforementioned metrics: MRs exhibiting higher code coverage are likely to be more effective in identifying software failures. They reported on a case study validating the performance (i.e., fault-detection capability) of self-checked MT on a cellular image processing application. Gagandeep and Singh [253] introduced a dynamic system for automating MT implementation. This system employs unified modeling language [227] to guide MR selection. They conducted an empirical study on a banking system to assess the performance of the SUT. Spieker and Gotlieb [256] proposed a new approach called adaptive MT to dynamically adjusts MR selection

throughout the MT process. The rationale behind adaptive MT is to convert the problem of MR selection into a reinforcement learning problem.

Recently, Sun et al. [260] introduced a novel algorithm named feedback-directed MT, designed for the selection of effective MRs and STCs based on the SUT's execution behaviors/outputs. This algorithm prioritizes the following aspects: (1) The violated MRs; and (2) the STCs that are "close" to the STCs from MR-violating MGs. Empirical experiments have been carried out to assess and compare the performance of this algorithm with MT-RT. The experimental results demonstrated that this algorithm surpassed MT-RT in terms of fault-detection capability.

2.3 EVALUATION METRICS

2.3.1 *Test Effectiveness (F-measure and F-ratio)*

F-measure and F-ratio serve as two widely-used metrics for assessing the failure-detection capability in both conventional software testing [81, 140, 210] and MT [64, 246]. The initial F-measure in software testing indicates the count of test case executions required to reveal the first software failure [81, 140, 210], while the initial F-ratio indicates the proportion between the F-measure of a test-case generation algorithm and the F-measure of RT (F_{RT}) [140].

Given that MT identifies software failures through scrutinizing the relationships among STCs and FTCs alongside their corresponding outputs [64, 143, 144, 246], it necessitates redefining both the F-measure and the F-ratio: The F-measure in MT signifies the count of MGs required for identifying the first MR violation, while the F-ratio in MT represents the proportion between the F-measure of an MG-generation algorithm and the F-measure of MT with RT (F_{MT-RT}). A smaller MT F-measure value indicates a reduced count of MGs required for identifying the first MR violation; similarly, a smaller MT F-ratio value indicates enhanced fault-detection capability.

Despite the existence of other metrics for evaluating test effectiveness, like the E-measure [70] and the P-measure [70], this thesis opt for F-measure and F-ratio due to the following rationales:

1. All experiments were iterated 10,000 times to calculate the mean F-measure, generation time, Discrepancy, and Dispersion results. It should be pointed out that given an algorithm, the mean F-measure does not represent the average F-measure of many SUTs. Instead, the mean F-measure represents the average F-measure obtained after multiple executions of a specific algorithm on one SUT and one MR. Specifically, in the experiment, for each SUT and an MR, an algorithm was executed to get the F-measure result. This step was then repeated 10000 times, and then the mean F-measure was calculated. This is because all the three algorithms under test contain randomness (e.g., the selection of STCs as candidates). Running the algorithm only a few times is not reliable.

2. The reason for using the F-measure and F-ratio: F-measure is able to assess the capability of algorithms on detecting the first failure. In other words, when a failure is identified for the first time. The F-ratio facilitates a comparative analysis of the proposed algorithm against the baseline algorithm, MT-RT. Both F-measure and F-ratio serve as popular metrics for assessing test effectiveness, and have been widely employed in prior studies related to MT [143, 144, 260, 262, 296] and ART [81, 140, 210, 249, 265].

3. The reason for using the *mean* F-measure: (1) It has been commonly used in many MT-related studies [143, 144, 260, 262, 296]. For instance, Hui et al. [144] proposed MT-ART, which was also included in the experiments in the thesis, and used the mean F-measure to measure the performance of MT-ART; and (2) this thesis also used the effect size (Cohen's *d*) to compare different algorithms, and the calculation of effect size requires the mean F-measure values.

4. F-measure can measure the fault-detection capability of MG-generation algorithms without the detection of prior faults. Since software testing may reveal multiple faults from a SUT during a test session, the evaluation of MG-generation algorithms after the detection of the first fault may be included in the future work. In particular, the F2-measure may be considered, which can be used to validate the fault-detection capability of MG-generation algorithms after some fault(s) have already been revealed.

2.3.2 *Test Effectiveness (Cohen's d)*

effect sizes are designed and developed to measure the size of effects in a population [121]. Additionally, the effect size (Cohen's *d* [90]) was considered in the empirical studies to visually compare the performance of various algorithms. Generally, the effect size serves as a metric illustrating the strength of the relation between two variables within a population [121]. Various effect size concepts have been formulated and employed in statistical practice, categorized into different families on the basis of the experimental design, including the difference family (measuring the standardized difference between two means) [90, 121], categorical family (assessing risk) [157], and correlation family (evaluating the strength of association between two variables) [237].

Cohen's *d* [90] was included in the empirical studies of this thesis, which is a widely-used effect size within the difference family. In general, Cohen's *d* values denote the number of standard deviations between two populations. Its selection is based on the following considerations: (a) Its capability to measure the standardized difference between two population averages, which correspond to the F-measure results in the empirical studies; and (b) its widespread application in the empirical studies related to MT [262, 296] and ART [249, 265]. Typically, larger Cohen's *d* values signify greater difference between the two means. Table 1 displays the range of Cohen's *d* values corresponding to various effect size magnitudes, initially delineated by Cohen [90] and later further expanded by Sawilowsky [239]. The values of Cohen's *d* can be computed using the following formula:

Table 1: Strengths of effect sizes in different ranges

Effect Size Strengths	Cohen's d Values
Very Small	0.01
Small	0.2
Medium	0.5
Large	0.8
Very Large	1.2
Huge	2.0

$$\begin{aligned} Cohen's \quad d &= \frac{Mean \quad Difference}{Pooled \quad Standard \quad Deviation} \\ &= \frac{M_2 - M_1}{\sqrt{\frac{SD_1^2 + SD_2^2}{2}}} \end{aligned} \tag{2}$$

- M_1 denotes the average of the first set of data.
- M_2 denotes the average of the second set of data.
- SD_1 denotes the standard deviation of the first set of data.
- SD_2 denotes the standard deviation of the second set of data.

2.3.3 *Test Efficiency (Generation Time)*

Two metrics, generation time and execution time [137, 140], are commonly employed to measure and compare the test efficiency of conventional test-case generation algorithms: The generation time denotes the computational cost associated with constructing a specified number of test cases and serves as the main indicator of testing efficiency, while the execution time signifies the duration needed to execute a given number of test cases against the SUT [137, 140].

In the empirical experiments of this thesis, generation time was included as the principal metric for assessing test efficiency, as it typically has a greater influence on the overall computational costs of software testing in comparison to execution time [137]. It is noteworthy that, owing to the characteristics of MT, the concept of generation time in the empirical studies was redefined as the duration required to produce a specified number of MGs: Specifically, the CPU time (measured in seconds) for generating 10,000 MGs was recorded. Given the fact that RT is likely to contain the minimal computational overhead in test case generation, it was anticipated that its generation time would be shorter than that of all other algorithms examined in the experiments.

2.3.4 *Test-Case Diversity (Dispersion)*

Dispersion was considered in the empirical studies to assess the diversity of test cases generated by various algorithms. Dispersion assesses the test-case diversity by determining

the presence of significant empty regions (void of executed test cases) within the input domain [60]. Dispersion can be measured using the following metrics:

- Min: The minimum distance between any two test cases within the test-case set [140].
- Max: The maximum distance between any two test cases within the test-case set [140].
- Max-Min: The difference between the maximum distance (Max) and the minimum distance (Min) within the test-case set [140]. A value approaching zero indicates a more uniformly distributed set of test cases.

2.3.5 *Test-Case Diversity (Discrepancy)*

Discrepancy serves as another metric for evaluating the diversity of test cases in the empirical studies. Discrepancy assesses the test case diversity by determining if various subdomains within the input domain exhibit an equal density of test cases. Eq. 3 [60] is employed to calculate the Discrepancy, illustrated as follows:

$$Discrepancy = \left| \frac{|E_i|}{|E|} - \frac{|D_i|}{|D|} \right| \tag{3}$$

In Eq. 3, D represents the input domain, and E denotes the executed test case set. D_i represents the i^{th} subdomain within D, characterized by randomly defined positions and sizes, while E_i signifies the i^{th} subset of executed test cases located within D_i. Typically, the number of subdomains is set to 1000 to balance between computational efficiency and accuracy [140], which is also the configuration adopted in the empirical studies of this thesis.

- i: $1 \leqslant i \leqslant n$.
- n: The total number of subdomains.
- Min: The minimum value of Eq. 3 obtained for a set of test cases E within D.
- Max: The maximum value of Eq. 3 obtained for a set of test cases E within D.
- Max-Min: The difference between the maximum value (Max) and the minimum value (Min) obtained from Eq. 3. A value approaching zero indicates a more uniformly distributed set of test cases.

2.3.6 *Receiver Operating Characteristics (ROC) and Area Under the ROC Curve (AUC)*

The ROC graph serves as a visualization method for assessing the performance of classification models [108]. It represents a curve depicted on a two-dimensional graph. The ROC curve illustrates the model's capacity to differentiate between positive and negative test

cases effectively. In the context of a model, there exist two types of test cases and four potential outcomes. To determine the ROC curve of a given model, it is necessary to calculate the following rates:

- Positive (P): The positive test case.
- Negative (N): The negative test case.
- True Positive (TP): The positive test case that are predicted to be positive.
- False Negative (FN): The positive test case that are predicted to be negative.
- True Negative (TN): The negative test case that are predicted to be negative.
- False Positive (FP): The negative test case that are predicted to be positive.
- True Positive Rate (TPR): The ratio of TP to the total number of P.
- False-Positive Rate (FPR): the ratio of FP to the total number of N.

For a given model, a pair of points (TPR, FPR) can be computed based on its performance on a set of test cases. The horizontal axis of the ROC graph denotes the FPR, while the vertical axis denotes the TPR. By adjusting the model's threshold, a curve can be generated passing through points $(0,0)$ and $(1,1)$, representing the ROC curve of the model. Typically, an ROC curve should lie above the line connecting points $(0,0)$ and $(1,1)$. To assess and visualize the quality and performance of a model, the Area Under the ROC Curve (AUC) was proposed [26, 124]. The value of AUC represents the size of the area under the ROC curve, with values typically ranging from 0.5 to 1.0. Typically, a larger AUC value indicates better test performance. Both ROC and AUC have been widely employed in evaluating credit risk models [23, 27, 46, 173, 189, 212].

Although other validation metrics, such as the Kolmogorov-Smirnoff statistic, are widely-used in credit scoring validation, this thesis selected AUC for the following two main reasons: (1) It is more suitable for imbalanced classification scenarios than the typical Accuracy metric [27]; and (2) it is widely aadopted within the credit scoring community [27], as well as the Gini coefficient [106], which is merely a transformation of AUC ($Gini = 2 \times AUC - 1$).

2.4 EXPERIMENTS SETUP

In order to thoroughly explore the efficiency and effectiveness of MG-generation algorithms, a diverse array of subject programs was chosen, varying in both size and dimensionality. Artificial faults were manually introduced into the SUTs to produce mutants, in accordance with the following specified mutation operators [149]: Constant RePlacement (CRP); Arithmetic Operator Replacement (AOR); Return Statement Replacement (RSR); and Relational Operator Replacement (ROR). All experimental procedures were performed under uniform hardware conditions: A Dell PC equipped with a 3.00GHz Intel(R) Core(TM) i5-9500 processor and 16.0GB RAM. Details about the SUTs, such as the

Table 2: Information of the Experimental SUTs and MRs

SUT	Sin	tanh	Erf	BesselJ	BesselJ	sncndc	TriSquare	TriSquarePlus	rj	PntLinePos
Input Dimensions	1	1	1	2	2	2	3	3	4	6
Input Domain Ranges	(0,1000)	(0,1000)	(0,1000)	((1,1), (100,100))	((1,1), (100,100))	((0,0), (100,100))	((0,0,0), (100,100,100))	((0,0,0), (100,100,100))	((0,0,0,0), (100,100, 100,100))	((0,0,0,0,0,0), (100,100,100, 100,100,100))
Size (LOC)	120	18	763	140	1211	64	38	31	175	23
Number of MRs	12	8	8	3	3	8	11	11	6	8
Fault Types	CRP, ROR, RSR, AOR									

name of the SUTs, the range of input domains, the Lines Of Code (LOC), and the number of MRs, and the mutation operators used, are outlined in Table 2, with comprehensive explanations provided below.

2.4.0.1 *Sin*

The first SUT, *Sin*, is programmed to execute the sine function, with its source code retrievable from Chen et al. [67]. Due to the oracle problem associated with most inputs for the sine function, it has garnered frequent attention in MT-related studies [52, 67, 143, 144]. Twelve MRs (refer to Appendix 1) were identified based on sine identities or insights from previous MT-related studies [52, 67, 143, 144].

2.4.0.2 *tanh*

The subsequent SUT, *tanh*, computes the ratio of Sinh to Cosh. This choice was because of its popularity in ART-related studies [13, 136, 137, 139]. In total, ten MRs were identified for the empirical experiments based on common *tanh* identities, with detailed information provided in Appendix 1.

2.4.0.3 *Erf*

Erf [276] was selected as the third SUT. Its implements the Gauss error function illustrated in Eq. 4. The version 3.6.1 of *Erf* from the Apache Commons Math framework[1] was selected for the empirical experiments, and a total of eight MRs were identified based on Eq. 4 or the monotonicity of *Erf*, as detailed in Appendix 1.

$$Erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{4}$$

$$Erf(x) = -Erf(-x) \tag{5}$$

2.4.0.4 *BesselJ*

The subsequent SUT, *BesselJ*, was designed to calculate the Bessel function of the first kind $J_v(x)$ for each element in array x. Various implementations of the Bessel function exist, and

1 https://commons.apache.org/proper/commons-math/

this thesis chose the SUT within the Apache Commons Math framework (version 3.6.1)[1] for the empirical experiments. This SUT has been widely employed in ART-related studies [276]. A total of three MRs were identified (listed in Appendix 1) based on the Bessel function or the common relation (Eq. 6).

$$J_{v-1}(x) + J_{v+1}(x) = 2 * v * J_v(x)/x \tag{6}$$

2.4.0.5 *BesselJ*

The next SUT, *BesselJ*, was also designed to implement the Bessel function according to Steed's method [143, 144, 229, 230]. Developed using C++, its source code is available in Numerical Recipes [229, 230]. This selection was motivated by its popularity in MT-related studies [143, 144]. For consistency, the same three MRs as those used by *BesselJ* were employed in the experiments for *BesselJ*.

2.4.0.6 *sncndn*

The sixth SUT is *sncndn*, tasked with implementing the Jacobi elliptic function [229, 230]. It is a system frequently employed in ART-related studies [13, 137, 193]. This system is also developed using C++, and its source code is available in Numerical Recipes [229, 230]. Eight MRs were derived based on the following relationships satisfied by the Jacobi elliptic function:

$$k^2 * sn^2 + dn^2 = 1, k^2 = 1 - y. \tag{7}$$

2.4.0.7 *TriSquare*

TriSquare, another widely-used system, was designed to determine if three given double-precision numbers greater than 0.0 can form a triangle and subsequently compute its size [49, 100, 102, 143, 144]. The system ends if the given numbers are incapable of forming a triangle; otherwise, it proceeds to calculate the its size. Eleven MRs were identified (shown in Appendix 1), derived based on previously-published MT-related studies [100].

2.4.0.8 *TriSquarePlus*

The eighth SUT, *TriSquarePlus*, was sourced from Dong [100] and designed to determine if three given double-precision numbers greater than 0.0 can form a triangle. If so, TriSquarePlus proceeds to identify the type of triangle and calculate its size. In the empirical experiments, the same eleven MRs identified for *TriSquare* were used.

2.4.0.9 *rj*

The next SUT in the empirical experiments is *rj*, sourced from Numerical Recipes [229, 230]. It implements the Carlson's elliptic integral of the third kind, as illustrated in Eq. 8. This SUT represents an enhanced iteration of *cel* and *el2*, both of which are widely-used

in ART-related studies [13, 62, 136, 137, 139, 193]. In the experimental setup, six MRs were identified, as detailed in Appendix 1, on the basis the following formula:

$$r_j(x, y, z, p) = \frac{3}{2} \int_0^\infty \frac{dt}{(t+p)\sqrt{(t+x)(t+y)(t+z)}} \tag{8}$$

2.4.0.10 *PntLinePos*

PntLinePos, serving as the final SUT, was selected due to its popularity in ART-related studies [13, 136, 137, 139, 193]. This SUT determines the relations between a point and a line segment, and outputs values representing the following:

- 0: Indicates that the point lies to the left of the line.
- 1: Indicates that the point lies on the line segment.
- 2: Indicates that the point lies on the extension of the line segment.
- 3: Indicates that the point lies to the right of the line.

In total, eight MRs were identified for the empirical experiments, as introduced in Appendix 1.

2.5 MACHINE LEARNING

2.5.1 *Neural Networks*

Neural networks are among the most widely-used predictive models, with significant research dedicated to their application in the credit scoring domain [27, 31, 93, 97, 148, 171, 190, 212, 225, 255, 273, 278]. A simple form of neural network (consisting of an individual neuron) is given to illustrate neural networks. This neuron serves as a computational unit, accepting two inputs x_1, x_2, two corresponding weights w_1, w_2, and a bias b_1, and producing an output y. The weights determine the strength of inputs, while the bias enables an activation function f to adjust the decision boundary [2, 105, 160]. The absence of bias may render the neural network inactive, impeding information transmission. That is to say, the bias regulates the activation function's triggering threshold. Common activation functions are sigmoid, tanh, and ReLU functions [2, 105, 160]. Here, f represents the activation function and $\hat{y}$ denotes the neural network estimate of y. The formula for computing this individual neuron is:

$$\hat{y} = f(x_1 w_1 + x_2 w_2 + b_1)$$

Neural networks are constructed by interconnecting individual neurons, enabling the output of one neuron to serve as the input for another [2, 105, 160]. Suppose that there is a dataset consisting of three attributes (x_1, x_2, y), with x_1 and x_2 known and y unknown.

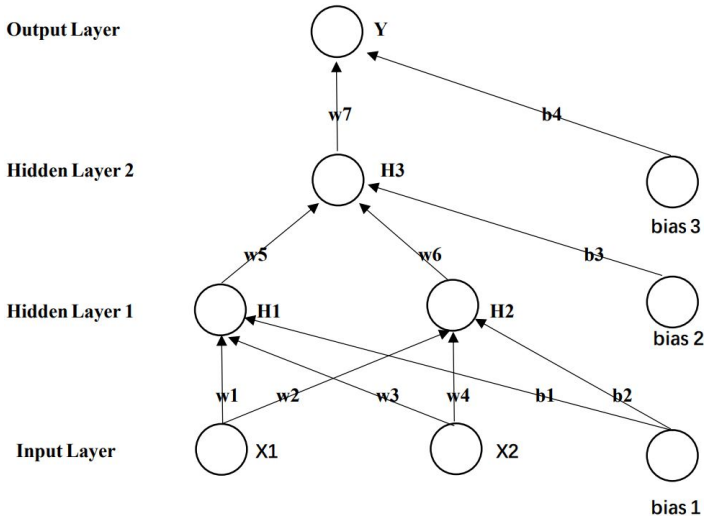


Figure 3: A multi-layer feed-forward neural network example

The objective is to predict the unknown attribute y using the two known attributes. Various types of neural networks exist, and in this instance, a basic multi-layer feed-forward neural network model is constructed, as illustrated in Fig. 3. This model contains three components: An input layer of neurons (used for the acceptance of inputs in various formats), two layers of hidden neurons (used for the identification of hidden features), and one layer of output neurons (used for the generation of final outputs) [2, 105, 160]. In particular, this neural network can be calculated using (where x represents an input, w represents a weight, h represents a hidden neuron, and b represents a bias):

$$h_1 = f(x_1 w_1 + x_2 w_3 + b_1), \quad h_2 = f(x_1 w_2 + x_2 w_4 + b_2)$$

$$h_3 = f(h_1 w_5 + h_2 w_6 + b_3)$$

$$\hat{y} = f(h_3 w_7 + b_4).$$

As a common type of neural network, the back-propagation neural network [36, 119, 172] operates as a multi-layer feed-forward network trained using the error back-propagation algorithm [2, 105, 160]. The back-propagation algorithm, a supervised learning algorithm, employs a loss function that quantifies the differences between predictions $\hat{y}$ and true labels y on training data, with the aim of adjusting the weights of network connections. In general, a back-propagation neural network can be built based on the following steps:

- Forward propagation of the operational signal requires the transmission of the input signal from the input layers through the hidden layers to the output layers. Throughout this process, the weights and biases remain constant, and the state of neurons in each layer solely influences the state of neurons in the subsequent layer. If the anticipated output is not obtained at the output layer, then the process transitions to error signal back-propagation; otherwise, the process ends and the results are reported.
- The back-propagation of the error signal involves the transmission of the error signal, representing the differences between the actual and expected outputs of a

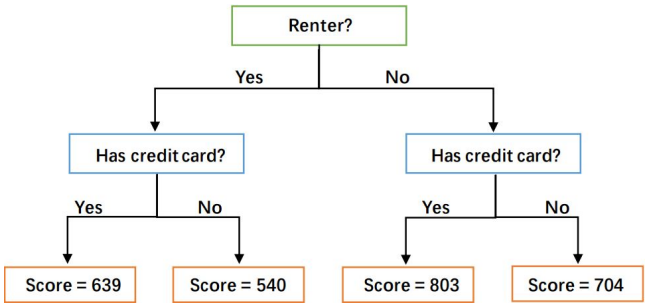


Figure 4: A Simple Decision Tree Example for Credit Scoring

neural network, from the output layers through the hidden layers to the input layers. Throughout this process, the weights are adjusted based on the error feedback. Back-propagation neural networks iteratively adapt the weights and biases to progressively align the network's actual output with the expected output.

2.5.2 *Decision Trees*

A decision tree, a parameter-free supervised ML-based model, serves as a versatile tool for both regression (distinguishing the data into continuous real values) and classification (separating the data into multiple categorical classes) tasks [217, 254]. It represents a tree structure formed by a root node, branches, internal nodes, and leaf nodes [217, 254]. The implementation of decision trees in credit scoring dates back to the 1940s [94, 115]. For instance, in the assessment of a customer's credit score, the decision tree illustrated in Fig. 4 serves as an illustrative example: Renters typically receive lower scores, while individuals with credit cards are assigned higher scores. The decision tree commences with a root node devoid of incoming branches. The pathway from the root node to each leaf node corresponds to a sequence of decisions. Each internal node (or decision node) represents an attribute, serving as a judgment condition containing the dataset subset that satisfies all conditions from the root node to the respective node. The leaf node, also referred to as the terminal node, consists of all potential outcomes and is indivisible.

Commonly-used decision tree types include the classification and regression tree [187], which is available for both classification and regression tasks: The former is applicable when predicting discrete data outcomes, while the latter is available for the scenarios where the predicted results contain real numbers. The classification and regression tree model typically consists of three key steps: Feature selection (highlighting the differences between classification and regression analyses), decision tree generation (formulating a decision tree according to the training dataset), and decision tree pruning (utilizing the validation dataset to refine the generated tree and identify the best sub-tree).

2.5.3 *Gradient Boosting Decision Trees*

Gradient boosting represents a widely-used and powerful ML algorithm employed in constructing predictive models within the field of credit scoring [27, 31, 171, 255]. It represents an ensemble of weak learners (predictive models), which is available for both regression and classification tasks [27, 114, 126]. In general, weak learners represent the models slightly outperforming random guessing, while strong learners denote the models with good accuracy [104]. Specifically, when these weak learners are constituted by decision trees (particularly the classification and regression tree [187]), the resultant models are termed gradient boosting decision trees [27, 126]. Belonging to the domain of ensemble learning, boosting combines a set of models to generate predictions [46, 126]. Within boosting decision trees, an ensemble of weak learners (individual decision trees) is amalgamated and trained sequentially. Each subsequent model learns from the errors of its predecessor to decrease the collective error of these weak learners, with the aim of forming a strong learner [46]. Friedman [114] introduced the concept of gradient boosting decision trees as an extension to boosting decision trees, focusing on descending the loss function in the previous model along its gradient direction. A gradient boosting decision tree typically consists of three components:

- The loss function to be optimized, which is designed according to the problem type.
- The weak learner (decision tree) responsible for predictions.
- The additive model (gradient descent) that amalgamates weak learners to minimize the loss function.

2.5.4 *Random Forests*

The Random forest stands as a widely-used predictive model within the field of credit scoring [6, 27, 31, 93, 107, 171, 212, 255, 270, 277]. As a powerful and favored ML-based model, it represents a parallel combination of decision trees available for both classification and regression analyses [30]. It users both bagging [29] and random feature selection [30] to construct a forest of uncorrelated decision trees. Bagging, or Bootstrap aggregating, is a renowned ensemble learning algorithm that typically combines and trains a set of weak models. Each model learns from a random subset of the original dataset, after which all models are amalgamated to produce an appropriate outcome. In contrast to bagging, random forests employ distinct random subsets of features to train a series of decision trees, with the aim of ensuring minimal correlation between them.

2.5.5 *Machine Learning in Credit Risk Assessment*

Brown et al. [31] investigated the credit scoring performance of eight classifiers, including neural networks, gradient-boosting decision trees and random forests. The empirical

experiments revealed that gradient-boosting decision trees and random forests typically exhibited better performance compared to other classifiers when using test samples with substantial class imbalances. Lessmann et al. [171] reported on an exhaustive study examining the credit scoring effectiveness of 41 different classifiers (such as neural networks, gradient-boosting decision trees, and random forests) across eight real-world datasets. The empirical results revealed that, in general, random forests were able to provide more precise predictions compared to individual classifiers such as neural networks. Wang et al. [277] employed random forest models to assess the probability of default and the specific default time of loan applicants in Peer-to-Peer lending scenarios. Through empirical experiments, they found that the random forest models were capable of outperforming other models, including logistic regression, in terms of AUC. Song et. al. [255] introduced a novel ensemble algorithm and reported on an empirical study to exploring and comparing it with six state-of-the-art algorithms, such as decision trees, gradient-boosting decision trees, random forests, and multi-layer perceptrons. Dusimana et. al. [107] reported on a case study validating the credit risk assessment performance of three popular ML-based models (including logistic regression, decision trees, and random forests) based on various cross-validation methods. Their experimental results indicated that the random forests demonstrated better performance compared to other models. Pandey et. al. [225] conducted empirical experiments to investigate and compare the performance of various credit-risk-related algorithms in assessing credit-risk datasets, such as neural networks, decision trees, multi-layer perceptrons, extreme learning machines, and support vector machines. The experimental results revealed that the extreme learning machines were capable of outperforming other algorithms. Trivedi [270] examined the optimal combination of feature selection methodologies and ML-based models through three commonly-used feature-selection techniques (including chi-square, information gain, and gain ratio) as well as five ML-based models (including Bayesian, Naive Bayes, support vector machines, decision trees, and random forests). The experimental results revealed that combining random forest and chi-square tended to contain better performance. Moscato et al. [212] introduced a bench-marking study examining the probability of loan repayment in a Peer-to-Peer platform through the combination of commonly-used ML-based algorithms (such as random forests and multi-layer perceptrons) and sampling methodologies. Breeden [27] examined the credit risk assessment performance of various ML-based classifiers (including random forests, neural networks, and gradient-boosting decision trees). The experimental results indicated that choosing the optimal classifier among the diverse options is exceedingly challenging. Agrawal et al. [6] presented a credit-risk calculator-driven loan-eligibility prediction tool using ML-based algorithms: It is capable of forecasting credit scores based on ML-based algorithms (including random forests and decision trees). The empirical experiments of algorithmic prediction accuracy revealed logistic regression as the top performer in terms of precision. Davis et al. [93] investigated and contrasted the credit-score prediction performance of various ML-based models (including neural networks and random forests) through a real-world residential loan dataset published by the Fair Isaac COrpora-

tion (FICO). The empirical results revealed that neural networks generally exhibited better performance compared to other linear and nonlinear models.

Predictive models find widespread application in the industrial sector as well. Equifax[2] holds numerous patents granted in the United States, and NeuroDecision Technology (NDT) is presently a widely-used international application developed by Equifax: It represents a regulatory-compliant neural network approach applicable to risk decision-making scenarios and has been widely used in predictive models requiring risk-assessment reason codes [205, 206]. McBurnett et al. [205] examined the interpretability of ML-based models and revealed significant distinctions in model scores and reason codes between NeuroDecision credit-risk models and unconstrained neural-network credit-risk models. McBurnett et al. [206] employed logistic regression to investigate the differences between the cause code patterns generated by the NeuroDecision model and a proxy model, in order to compare their of their accuracy. Quell et al. [233] evaluated the performance of ML-based models (including neural networks, gradient-boosting decision trees, and random forests) in model-risk and credit-score assessment. Regarding selection criteria, Quell et al. [233] stressed the importance of not only assessing model performance (credit risk assessment capability) but also other additional factors (including model interpretability, as well as the cost and effort associated with model application, maintenance, and monitoring). FICO[3], a popular analytics software enterprise assisting businesses across more than 90 countries in making decisions, conducted a comparison between the credit-score prediction performance of the proposed model (termed FICO Score analytic model) and prevalent ML-based models (including neural networks and gradient-boosting decision trees) [109]. The experimental results suggested that the proposed model were capable of outperforming other models.

Traditional linear models often rely on business intuition for model selection. The estimated model coefficients should meet the business expectations. For instance, if we see a positive coefficient estimate for credit score in a default model, we may be suspicious of the model and may not use it [28]. Because there are no coefficients on linear terms, and the effects are commonly non-linear on the factors, this procedure cannot be used for ML-based models. The emergence of ML-based models with high complexity highlights the importance of Explainable Artificial Intelligence (XAI) in credit risk assessment [4]. XAI concentrates on making the decision-making process of ML-based models transparent and understandable [3, 4, 176]. Consumers are able to better understand why the model makes certain decisions and how it works. For example, by deeply understanding the factors that affect the credit score, consumers are able to take proactive measures to improve their credit score. In general, XAI consists of two primary components [176]: Interpretability and explainability.

Interpretability is an important evaluation metric for ML-based credit risk models [38, 118, 288], and is generally associated with the rationale behind the outputs of a model [3]: It indicates the degree to which the output of a model can be understood and interpreted by humans. Baesens et al. [16] was arguably the first to highlight the predictive-accuracy-

2 https://www.equifax.com/ (7 April 2024)
3 https://www.fico.com/en (7 April 2024)

interpretability trade-off in credit risk assessment, which was later further elaborated by Martens et al. [194] and Coussement et al. [92]. On the other hand, explainabilityis often related to the internal logic and mechanisms within ML-based models [3, 176]: A model that is highly explainable represents a deep human understanding of the internal processes that occur during model training or decision-making.

Model-agnostic methods can be used for XAI, which can be roughly divided into two types: Global methods and local methods [209]. A Partial Dependence Plot (PDP) [122] is a typical example of a global method: It enables the relative effect size of values of each feature against the target variable. In contrast, local methods work by interpreting the feature effects inside the local neighborhoods of the data. For instance, Local Interpretable Model-agnostic Explanations (LIME) [236] is one of the most commonly-used methods: It can generate a local linear model in the neighborhood of an example, and offer coefficient estimates for the model as explanations of model behavior at that example. SHapley Additive exPlanations (SHAP) is another commonly-used method for constructing local explanations according to the concept of Shapley value [250]. Although LIME and SHAP can be used with MT/ME (since they are able to offer measures at local level), we are unaware of any previous related works. They were not used in the thesis since they may not necessarily express business intuitions in one specific format, and thus, the HMRs were directly constructed according to business rules instead of using generic local XAI methods.

3

A SIMULATION FRAMEWORK FOR THE PROCESS OF METAMORPHIC TESTING

Publications delivered from this chapter

1. **Zhihao Ying**, Anthony Bellotti, Dave Towey, Tsong Yueh Chen, and Zhi Quan Zhou. Using Metamorphic Relation Violation Regions to Support a Simulation Framework for the Process of Metamorphic Testing. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2022, pp. 1722-1727, doi: 10.1109/COMPSAC54236.2022.00274.

3.1 INTRODUCTION AND MOTIVATION

As a well-developed software testing technique, Simulation evaluates software testing performance by employing artificial failure regions, and assessing whether or not the generated test cases are located within these regions [140]. Due to their ability to mitigate the challenges and time constraints associated with evaluating and comparing software testing techniques, simulations have seen widespread adoption in the development of traditional testing methods [140]. Test cases inducing unexpected SUT behavior or output are termed as failure-causing test cases. Failure regions, as the core of simulations, denote the areas (i.e., test cases) of the input domain that can lead to SUT failures (as determined by the oracle) [140]. Simulations do not require SUT executions; instead, they ascertain software failures based on the location of test cases.

However, to the best of our knowledge, no simulation framework has been proposed for MT. This may be caused by the fact that simulations are often designed designed and applied under the availability and applicability of an oracle. However, a key strength of MT lies in its demonstrated effectiveness in software testing, without the requirement for an available oracle. Furthermore, MT identifies software failures by detecting MR violations, rather than test case failures. These factors make traditional simulations less applicable to MT. MT performance evaluation can prove challenging and time-consuming (particularly the identification of MRs), involving the following steps:

1. Identify an SUT and generate mutants by injecting artificial faults into the SUT, either manually or using an automatic mutation tool [149].

2. Identify MRs for the SUT, either manually or through an automatic tool.

3. Generate MGs using MG-generation algorithms.
4. Execute the generated MGs against the mutants.
5. Examine the outputs and check for MR violations.

The absence of inexpensive and swift MT simulation tools are likely to impede the exploration, development, and application of MT. This chapter refers to this problem as the MT-performance evaluation challenge.

This chapter reports on proposing the concept of MR-Violation Regions (MRVRs) [285] as a solution to this challenge, demonstrating their capability in developing MT simulation tools. Such MT simulations simplify the process of evaluating MT performance, allowing testers and researchers to conduct MT-related experiments with greater efficiency and speed. For instance, numerous previous studies on MG-generation have highlighted the potential influence of MG quality on MT performance [19, 143, 144]. However, assessing the efficiency and effectiveness of MG-generation algorithms has consistently been challenging and time-consuming [246]. The use of MT simulations are expected to address these challenges and provide users with a faster and more comprehensive approach for the investigation of MG-generation algorithms. This chapter reports on a case study investigating the presence of different types of MRVRs in programs with numerical input domains, drawing from previously-published MT-related or ART-related studies [13, 52, 62, 67, 100, 136, 137, 139, 143, 144, 193, 276]. Additionally, this chapter also analyzes and explores the differences between MRVRs and traditional (oracle-based) failure regions. It is hoped that the proposed MT simulation framework could facilitate the in-depth exploration and wider adoption of MT.

3.2 AN MT SIMULATION FRAMEWORK

This section introduces the concept of MRVRs and a framework to facilitate the MT simulation for a specific category of MRs: The Deterministic MRs (DMRs). While the concept of DMRs has been applied to some extent, it remains without a formal definition. Considering this, this section formally proposes definitions for both DMRs and non-DMRs.

3.2.1 *Deterministic Metamorphic Relations (DMRs)*

Using a program, *Square* (calculating the square of a number) as an example. The following are two possible examples of 1-1 MR described based on the template proposed by Segura et al. [244]:

In the domain of *Square* Program
assuming that

- The input contains one parameter: x ($x > 0$).
- The output consists of one parameter: *Square(x)*.

the following metamorphic relation(s) should hold

- $MR_{square1}$:

if source inputs are increased by an integer P ($P = 1$) to construct follow-up inputs,

then the follow-up outputs should be larger than the source outputs.

- $MR_{square2}$:

if source inputs are increased by a positive integer Q to construct follow-up inputs,

then the follow-up outputs should be larger than the source outputs.

It is evident that for the first 1-1 MR type, each STC can generate a unique FTC, while for the second 1-1 MR type, varying FTCs can be constructed from a single STC by using different P values. This chapter refers to the first kind of 1-1 MRs as *Deterministic 1-1 MRs* (*DMRs[1-1]*). With this consideration, $MR_{square1}$ can be written as $DMR[1\text{-}1]_{square1}$. This chapter generalizes this concept to formally define a *Deterministic M-N MR* (*DMR[M-N]*) as follows.

- *Definition 3.1 (Deterministic M-N MR (DMR[M-N]))*: *An M-N MR such that for any set of M STCs (<STC_1, STC_2, ..., STC_M>) in a given valid MG, there is one and only one determined set of N corresponding FTCs (<FTC_1, FTC_2, ..., FTC_N>).*

3.2.2 *Metamorphic Relation Violation Regions (MRVRs)*

If conducting MT on an SUT with a *DMR[1-1]*, all available STCs within the input domain are chosen and the relevant FTCs can be automatically generated, to identify the subset of those STCs (as well as their respective FTCs) that violate the specified MR. Thus, selecting an STC from this test set will certainly result in the violation of this MR. In this scenario, further consideration of FTCs is unnecessary, as they are entirely determined by the STCs and the DMR: Once the STCs and the DMR are decided, the FTCs cannot be altered. In this context, an MT simulation can be employed for (D)MRs: Without executing the SUT, violation of the specified (D)MR can be detected by identifying any STC that leads to an MR violation — these STCs are referred to as (D)MR-violating STCs. Specifically, the following formal definitions are presented:

- *Definition 3.2 (MR-violating STCs)*: *The set of STCs from MR-violating MGs for an DMR.*
- *Definition 3.3 (MR-Violation Region (MRVR))*: *The regions of MR-violating MGs, including corresponding MR-violating STCs and MR-violating FTCs*
- *Definition 3.4 (MR-Violation Rate)*: *The number of MR-violating STCs as a proportion of all possible STCs.*

This chapter further refers to the MRVR-S as the STC-only component of the MRVR and the MRVR-F as the FTC-only component of the MRVR, and their formal definitions are:

Algorithm 1: MT Simulations

```
Choose a processed SUT with a suitable identified MR-violation rate and regions;
while none of the stopping conditions are triggered do
    Employ an MG-generation algorithm to generate a new STC;
    Determine the MR violation by verifying whether or not the new STC falls
     within an MRVR-S;
end
```

- *Definition 3.5 (MRVR-S)*: *The regions within the input domain from which any selected and executed STC will lead to a violation of the MR.*
- *Definition 3.6 (MRVR-F)*: *The regions within the input domain identified based on the MRVR-S and an MR.*

At this stage, a tester or researcher is able to ascertain whether or not an MR is violated by examining whether or not the generated STC belongs to the set of MR-violating STCs (within the relevant MRVR-S). Building upon the concepts of DMRs and MRVRs, this thesis summarizes the procedure of implementing MT simulations in Algorithm 1. Additionally, it is crucial to emphasise that the MRVRs will vary for different MRs.

For the second 1-1 MR type (such as $MR_{square2}$), identifying the relevant MRVR-S necessitates determining the Q value: when $Q = 1$, one MRVR-S can be identified (equivalent to $MR_{square1}$); and when $Q = 2$, a new MRVR-S can be identified. Each Q value can be regarded as defining an input domain dimension, or alternatively, the entire SUT can be considered to have an additional input domain dimension characterized by different Q values. Further discussion and analysis of identifying MRVRs for other types of MRs (such as non-*DMR[1-1]*) is beyond the scope of this research, which may be included in the future work.

3.2.3 *Relationship between MRVRs and Failure Regions*

It should be noted that, within a given faulty SUT, the failure regions and the MRVRs are anticipated to manifest dissimilar characteristics. Typically, in scenarios where the oracle is absent (with the presence of the oracle problem), the localization of failure regions becomes indeterminate; however, the determination of MRVRs remains feasible. Even with an available oracle and the same SUT, as previously indicated, one specific MR has its own MRVR, and different MRs have different MRVRs. In particular, within the context of the same SUT, the localization of failure regions is immutable, while different MRs are anticipated to contain different MRVRs.

In the context of a faulty SUT equipped with an accessible oracle and a 1-1 MR, three different types of MGs can be defined based on the correlation between the location of MGs and the location of failure regions:

1. Either the STC or the FTC lies within the failure region.

2. Both the STC and the FTC lie within the failure region.

3. Neither the STC nor the FTC lies within the failure region.

The selection of MGs categorized under Types (1) or (2) produces the identification of at least one failure-causing test case: Either the STC or the FTC for Type (1); or both for Type (2). Nonetheless, it is important to emphasize that the MGs classified under Types (1) or (2) do not inevitably lead to MR violations; in contrast, the MGs categorized under Type (3) may still reveal MR violations.

The use of diverse combined MRs has been reported to perform better than any individual MRs [64, 74, 181, 246, 297]. Although this concept may be applicable in practical software testing scenarios, the scope of this study includes the design of simulations for specific individual MRs, as exemplified by *DMRs[1-1]*. Testers or researchers might prefer to apply MT simulations when MR-violating STCs or FTCs also trigger failures, that is, the MRVR-S/MRVR-F and the failure region are identical. This is an additional extension for the proposed MT simulation framework.

Drawing from the existence of failure regions in conventional (oracle-based) software testing [83], it is logical to predict the existence of similar (MT-oriented) MRVRs: block, strip and point. For instance, the following MRVRs can be identified for the SUTs (in reality or simulations) with a *DMR[1-1]*:

- *Definition 3.7 (Block MRVRs): The MR-violating STCs or FTCs roughly concentrate within a singular contiguous jagged rectangular region.*
- *Definition 3.8 (Strip MRVRs): The MR-violating STCs or FTCs roughly concentrate within a slender line connecting adjacent boundaries.*
- *Definition 3.9 (Point MRVRs): The MR-violating STCs or FTCs are less obviously clustered, and are more diversely spread throughout the entire input domain in minor clusters.*

3.3 EMPIRICAL EXPERIMENTS

In this section, an investigation was conducted into the presence of various types of MRVR in subject programs of varying sizes and dimensions. Table 3 summarizes brief information about the SUTs and their mutants, while Appendix A presents details of all the (D)MRs used. All SUTs have previously been studied and published in MT-related or ART-related studies [13, 52, 62, 67, 100, 136, 137, 139, 143, 144, 193, 276]. All MRs used in the experiments were DMRs, derived from previous MT-related studies [67, 100, 143, 144] or manually identified. Artificial faults were inserted into the SUTs to create mutants through mutation operators [149] — the description of the mutation operators is given in Section 2.4. In total, ten mutants were created for each SUT. The steps for creating mutants can be summarized as follows:

Table 3: Information of the Experimental SUTs and MRs

SUTs	Sin	tanh	Erf	BesselJ	BesselJ	sncndc	TriSquare	TriSquarePlus	rj	PntLinePos
Input Dimensions	1	1	1	2	2	2	3	3	4	6
Input Domain Ranges	(0,1000)	(0,1000)	(0,1000)	((1,1), (100,100))	((1,1), (100,100))	((0,0), (100,100))	((0,0,0), (100,100,100))	((0,0,0), (100,100,100))	((0,0,0,0), (100,100, 100,100))	((0,0,0,0,0,0), (100,100,100, 100,100,100))
Size (LOC)	120	18	763	140	1211	64	38	31	175	23
Number of MRs	12	8	8	3	3	8	11	11	6	8
Number of Mutants	10	10	10	10	10	10	10	10	10	10
Fault Types	CRP, ROR, RSR, AOR									
Existence of Block MRVRs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Existence of Strip MRVRs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Existence of Point MRVRs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

1. An artificial fault is randomly inserted into the SUT to create a mutant using mutation operators, either manually or through a tool.
2. MT-RT is executed against the mutant using each MR to check whether or not the MR is violated.
3. Repeat Step 2 10,000 times, and if a violation is detected, then include this MR in the experiments; otherwise, skip to Step 1.
4. Repeat Steps 1-3 until ten mutants have been created.

The steps of identifying MRVRs (Section 3.2.2) were repeated for each mutant and its corresponding MR to identify the MRVRs. The detailed experimental steps were:

1. Select a system.
2. If a stopping condition is not triggered, do:
   a) Select a mutant and an MR.
   b) Execute the MGs in the input domain.
   c) Check and mark the part of MGs that violate the MR.
   d) Examine and categorize the MRVR.

The stopping condition was that for the selected system: (A) all three types of MRVRs had been identified; or (B) each pair of mutants and MRs had been examined. For instance, using the Sin system as an example: It has 12 MRs and 10 mutants, and thus, there are a total of $12 * 10 = 120$ pairs of mutant and MR. Then each pair of mutant and MR was selected and examined, until at least one block MRVR, one point MRVR and one strip MRVR had been identified, or all the 120 pairs of mutant and MR had been examined. Therefore, the number of instances for the three kinds of MRVRs were not recorded. This is because the purpose of the experiment is to demonstrate the existence of the three MRVR types, rather than their proportions. Given the very large time commitment (more than a

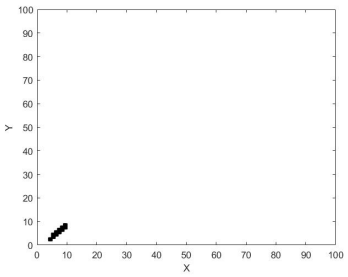


Figure 5: Block MRVR-S

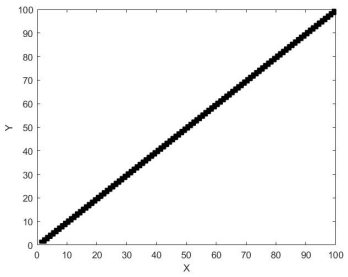


Figure 6: Strip MRVR-S

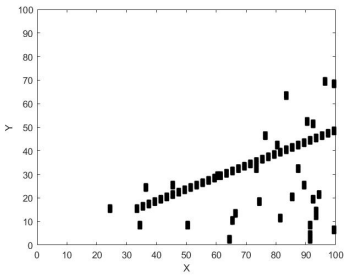


Figure 7: Point MRVR-S

few weeks) that would be necessary to revise the experiment and gather the data, it seems that would not be feasible. It will be interesting future work to examine the proportions of each MRVR type.

The last three rows of Table 3 present the experimental results indicating the presence or absence of the three types of MRVR. For instance, the shapes of the instances found in the 2D to 4D input domains are listed as follows:

1. In 2D input domains:
   - Block MRVR: A complete/incomplete square, and a complete/incomplete rectangle.
   - Point MRVR: Point (individual test case), square, and rectangle.
   - Strip MRVR: A slender line (e.g., trapezium).
2. In 3D input domains:
   - Block MRVR: A complete/incomplete cube, and a complete/incomplete cuboid.
   - Point MRVR: Point (individual test case), cube, and cuboid.
   - Strip MRVR: A flat surface with a certain thickness.
3. In 4D input domains:
   - Block MRVR: A complete/incomplete tesseract, and a complete/incomplete hypercube.
   - Point MRVR: Point (individual test case), tesseract, and hypercube.
   - Strip MRVR: A hyperplane.

It is not difficult to infer the corresponding MRVR shapes in higher dimensional input domains. It should be noted that the complete square means that the square is conventional, while the incomplete square represents that this square contains some extra or miss some parts. For instance, as shown in Fig. 5, this polygon can be regarded as a rectangle that contain/miss some small triangles. In addition, since the purpose of this experiment was to explore and analyze whether or not the three kinds of MRVRs exist, a detailed analysis of their shapes was not the main objective of this thesis. Nevertheless, this is something that I hope to address in future work.

Typical examples of the three MRVR-S (STC-only component of the MRVR) types for the $BesselJ(x, y)$ program with $MR_{BesselJ1}$ are illustrated in Figs. 5, 6 and 7:

- Block MRVR-S: The MR-violating STCs roughly concentrate within a singular contiguous jagged rectangular region.
- Strip MRVR-S: The MR-violating STCs roughly concentrate within a slender line connecting adjacent boundaries.
- Point MRVR-S: The MR-violating STCs are less obviously clustered, and are more diversely spread throughout the entire input domain in minor clusters.

3.4 CONCLUSION

With the rising popularity of MT, there is an increasing need to easily and quickly evaluate its performance, including efficiency and effectiveness. Some software testing techniques, such as ART [140], have always benefited from using simulations to facilitate a quick and efficient evaluation of testing parameters or configuration settings. Over the past decades, the absence of a simulation framework for MT may have hindered certain MT research efforts. Traditional simulations are not readily applicable to MT due to its requirement for an available oracle and the reduced demand for an oracle in MT. In this chapter, the concept of MR-Violation Regions (MRVR) was proposed to address this issue. A new framework was proposed to support MT simulations for a specific MR type, labeled Deterministic MRs (DMRs), based on the application of MRVRs. This, to the best of our knowledge, is the first MT simulation framework in the literature. Specifically, the creation of MRVR-S (the STC-only component of MRVR) enables the application of MT simulations for 1-1 DMRs (MRs that have exactly one STC and one FTC), with MR violations determined by checking if the generated STCs are within MRVR-S without further exploration of FTCs or SUT executions.

This chapter has proposed three types of MRVRs: block, strip, and point. The relationship between oracle-oriented failure regions and MRVRs was discussed and analyzed, highlighting that they are not expected to be identical. A case study was reported, investigating and supporting the identification and classification of the three possible types of MRVRs in certain SUTs and (D)MRs from previously-published studies [13, 52, 62, 67, 100, 136, 137, 139, 143, 144, 193, 276]. A limitation of this study has been its focus solely on one type of MRs (the DMRs). Therefore, future work may involve further investigation into the preparation of simulations for other MR types (i.e., non-*DMR[1-1]*), as well as the exploration of different MRVR types.

The next chapter will introduce how to use MRVR to explore and validate the performance of previously-published MG-generation algorithms. That is, through the application of MRVR, the next chapter effectively identifies the existence of potential challenges that could impact the performance of MG-generation algorithms. A solution will also be proposed to address these challenges, as well as a new MG-generation algorithm aimed at further enhancing MT performance.

4

ADDRESSING THE PROBLEMS IN METAMORPHIC GROUP GENERATION ALGORITHMS

Papers delivered from this chapter (Under Review)

1. **Zhihao Ying**, Dave Towey, Anthony Bellotti, Tsong Yueh Chen, and Zhi Quan Zhou. SFIDMT-ART: A Metamorphic Group Generation Method Based on Adaptive Random Testing Applied to Source and Follow-up Input Domains. *Information and Software Technology*, 2024, pp. 107528.

4.1 INTRODUCTION AND MOTIVATION

The successful application and performance of MT is mainly dependent upon the MRs and MGs [64, 246]. Nevertheless, the construction of effective MGs still needs further investigation and exploration [64, 246]. Prior MT-related studies has mainly concentrated on STCs, somewhat neglecting FTCs, to a certain extent [64, 246]. It has been reported that the most widely-used MG-generation algorithm is MT-RT [64, 246]. However, MT-RT sometimes cannot achieve satisfactory test effectiveness as it does not utilize any features of the SUT [140, 246]. Recently, some studies [144, 291] have suggested considering FTCs in the MT process, which could enhance MT performance. In essence, although some studies have emphasized the significance of FTCs, there remains a dearth of research on how to enhance their quality: All prior MG-generation algorithms treated FTCs in a manner identical to STCs. This may have an impact on the performance of MG-generation algorithms, explained as follows: In conventional software testing, there exists only one set of test cases and one input domain. In contrast, MT comprises at least two sets of test cases (STCs and FTCs). This thesis refers to the set of STCs associated with an MR as the source input domain and the set of FTCs as the follow-up input domain. In this thesis, if no ambiguity exists, the source input domains and the follow-up input domains may be simply referred to as input domains. The entire input domain denotes the input domain of the SUT, containing both the source input domain and the follow-up input domain. MRs can be categorized into two types based on the positional relationship between the source input domain and the follow-up input domain:

1. The MR in which the source input domain and the follow-up input domain are identical.

2. The MR in which the source input domain and the follow-up input domain are different.

However, all previously-published MG-generation studies have ignored the impact of input domains on MT performance. In particular, the differences between the source input domain and the follow-up input domain are likely to have an impact on the efficiency and effectiveness of MG-generation algorithms. This thesis refers to this issue as the input-domain difference problem.

This chapter reports on a study applying MRVRs to examine a particular type of MG-generation algorithms: The algorithms aim to enhance MT performance by achieving a balanced distribution of both STCs and FTCs throughout the entire input domain [143, 144]. Through empirical experiments, this chapter discovers that the algorithm is likely to encounter the input-domain difference problem, as explained below: When the source input domain differs from the follow-up input domain for a given MR, the STCs and FTCs generated by the algorithm may not be able to achieve an even distribution, potentially leading to adverse effects on MT performance [74].

This chapter introduces a novel approach to enhance the quality of both STCs and FTCs: Treating FTCs separately from STCs during the MG-generation process. On the basis of the proposed approach, this chapter presents a novel STC and FTC allocation principle for MT to tackle the input-domain difference problem: Ensuring an even distribution of STCs over the relevant source input domain and FTCs over the relevant follow-up input domain.

Subsequently, this chapter introduces the following two MT-distance measures, known as Input-Domain-based Metamorphic Distances (ID-MDs), to facilitate the application of the proposed novel STC and FTC allocation principle:

- The distance between STCs and executed STCs.
- The distance between FTCs and executed FTCs.

It is important to note that the executed STCs and relevant executed FTCs represent the test cases from the same executed and non-MR-violating MGs. On the basis of the proposed STC and FTC allocation principle and MT-distance measures, this thesis subsequently propose a novel MG-generation algorithm named SFIDMT-ART (MT-based ART applied to Source and Follow-up Input Domains) from black-box testing perspective. Empirical studies were conducted, and the experimental results demonstrated that the proposed algorithm significantly outperforms previously-published MG-generation algorithms in terms of test effectiveness and efficiency.

4.2 SFIDMT-ART ALGORITHM

4.2.1 *Motivation, Problem and Solution*

Different from traditional software testing techniques (such as ART), where one test case is generated for the SUT at a time, MT uses at least two sets of test cases (STCs and

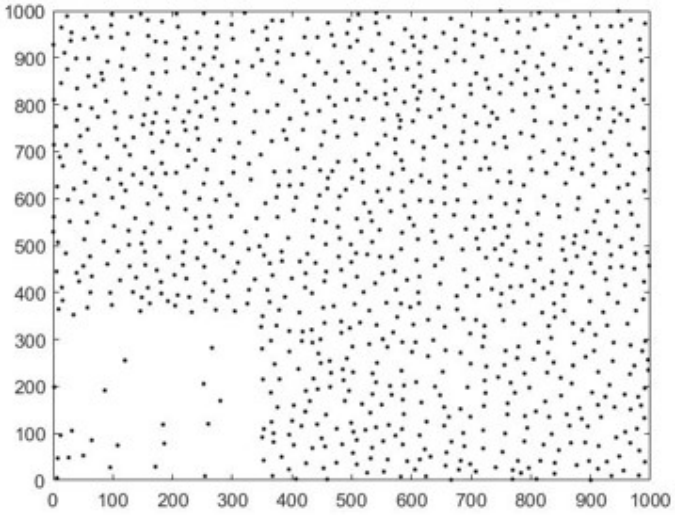


Figure 8: 1000 STCs generated by MT-ART-Min using $MR_{Product}$ for *Product*

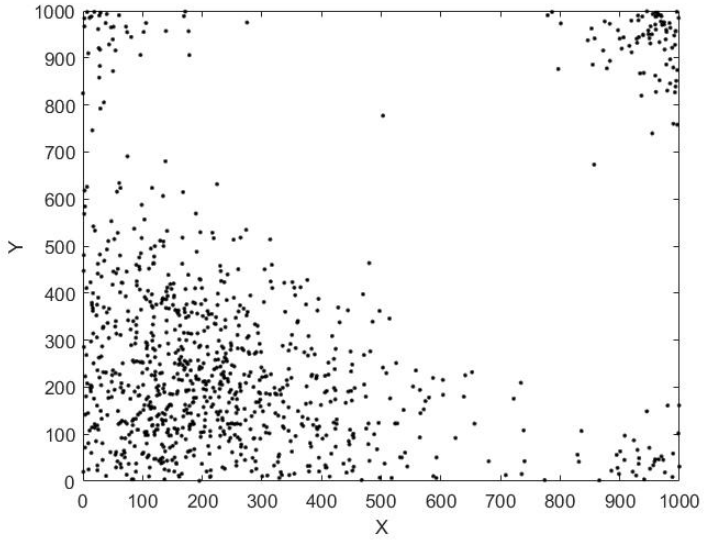


Figure 9: 1000 STCs generated by MT-ART-Max using $MR_{Product}$ for *Product*

FTCs). Because of this difference, traditional ART is not readily applicable for generating test cases for MT. Furthermore, some recent studies [144, 291] have suggested that the MG-generation process should take into account not only STCs but also FTCs, with the aim of enhancing MT performance. Nevertheless, comprehensive and in-depth research on how to improve the quality of FTCs is still lacking. In particular, previous MG-generation studies treated FTCs the same as STCs and disregarded the differences between them. One notable difference, for example, is their corresponding input domains. Specifically, given an MR, the source input domain (the input domain of the STCs) may differ from the follow-up input domain (the input domain of the FTCs). This difference may result in the input-domain difference problem, potentially having a negative impact on the performance of MG-generation algorithms, particularly ART-based ones [143, 144], a factor that has been overlooked in previous MG-generation studies.

For instance, consider the application of MT-ART to a small program *Product*, and a *DMR[1-1]* named $MR_{Product}$. *Product* calculates the product of two double-precision numbers, ranging from 0 to 1000. According to $MR_{Product}$, dividing each source input by three to generate follow-up inputs results in the follow-up output being one-ninth of the source output. Analysis of the MR reveals that the source input domain spans from 0 to 1000, while the follow-up input domain ranges from 0 to 333.33. Evidently, the source input domain is notably larger and contains the entirety of the follow-up input domain. Subsequently, MT-ART-Min was applied across the entire input domain, producing a total of 1000 STCs, as illustrated in Fig. 8: MT-ART-Min selectively picked only a small number of STCs from the region where the source input domain intersects with the follow-up input domain. This may occur since MT-ART attempts to produce STCs and FTCs that are evenly spread throughout the entire input domain, if a majority of the FTCs are concentrated in the area where the source input domain intersects with the follow-up input domain, then fewer STCs will be chosen from that region. Consequently, if MR-violating STCs are located within this region, the performance of MT-ART may be significantly adversely impacted. In particular, although the STCs and FTCs may appear evenly distributed across the entire input domain from a broader standpoint, the individual test case sets (the STC/FTC sets) are actually clustered in distinct regions rather than being uniformly distributed within their respective input domains.

From the aforementioned discussion, it can be inferred that the primary reason MT-ART faces the input-domain difference problem is its treatment of FTCs in the same manner as STCs. Given the difference between the source input domain and the follow-up input domain, this treatment adversely affects the testing performance of MT-ART. Hence, this thesis introduces a novel approach to enhance the quality of STCs and FTCs during MG generation: The MG-generation algorithm can handle FTCs independently of STCs. Subsequently, this thesis introduces a novel principle for distributing STCs and FTCs to tackle the input-domain difference problem of MT-ART in accordance with the proposed approach: Ensure an even distribution of STCs across the respective source input domain and FTCs across the corresponding follow-up input domain. It is noteworthy that not all MG-generation algorithms may encounter the input-domain difference problem: Since the difference between STCs and FTCs may not only be limited to the input domain difference, certain algorithms may be influenced by other specific differences.

In addition to the input domain difference problem, there are other problems present in ART-related studies [140]. Hence, to further explore whether or not MT-ART would be impacted by other problems, MT-ART-Max was applied to generate 1000 STCs for the *Product* program with $MR_{Product}$, and the results are illustrated in Fig. 9. The STCs appear to cluster towards the edges of the entire input domain (particularly in the bottom-left corner), instead of being evenly spread. Traditional ART algorithms may also face this issue, called the edge preference problem [140]. This problem may arise because, with the maximum distance criterion, edge test cases may be farther from the executed test cases than center test cases. As a consequence, MT-ART-Max will prioritize the selection of test

cases from the edge over those from the center. MT-ART-Avg encounters the same problem, while only MT-ART-Min can circumvent it.

This issue may arise from the utilization of the maximum distance criterion and the average distance criterion. MT-ART uses the Euclidean distance to measure the distance between test cases [144]. However, in studies related to ART [140], the use of the Euclidean distance is limited to computing the minimum distance between test cases; in contrast, alternative distance calculation methods are used to assess the average or maximum distance criterion. For instance, the average distance criterion was formulated for test cases prioritization [150]: It was computed on the basis of the Jaccard distance, which quantifies the dissimilarity between test cases. Both Chen et al. [48] and Ciupa et al. [89] employed the maximum and average distance criteria to compute the object distance between the test cases within object-oriented programs. Considering this, SFIDMT-ART only takes the minimum distance criterion into account to mitigate the edge-preference problem.

4.2.2 *Distance Measurements and SFIDMT-ART Algorithm*

In order to implement the proposed STC and FTC allocation principle (achieving an even distribution of STCs and FTCs across their respective input domains), this thesis introduces two novel MT-distance metrics termed Input Domain-based Metamorphic Distances (ID-MDs), detailed as follows:

- *Definition 4.7 (Source Metamorphic Distance (SMD))*: *The distance between a source candidate and all the respective executed STCs.*
- *Definition 4.8 (Follow-up Metamorphic Distance (FMD))*: *The distance between a follow-up candidate and all the respective executed FTCs.*

Based on the ID-MDs, this thesis introduces a novel MG-generation algorithm named MT-based ART applied to Source and Follow-up Input Domains (SFIDMT-ART). It aims to enhance MT performance by: 1) Ensuring the even distribution of STCs across the respective source input domain; and 2) ensuring the even distribution of FTCs across the corresponding follow-up input domain. The steps for implementing SFIDMT-ART are summarized in Algorithm 2 and explanations for each step are described as follows:

- Step 1: SFIDMT-ART is provided with an MR which can be either identified from scratch or directly selected from existing ones.
- Step 2: SFIDMT-ART identifies the scope of source and follow-up input domains based on the provided MR.
- Step 3: SFIDMT-ART records the number of executed STCs as M. The rationale behind this step is that SFIDMT-ART partitions the source input domain according to the quantity of respective executed STCs within that input domain.
- Steps 4-5: SFIDMT-ART initializes all executed test-case sets to be empty. The rationale behind this step lies in the presumption that the input-domain difference

Algorithm 2: SFIDMT-ART for 1-N MRs

```
Given one 1-N MR;
Determine the scope of the source input domain and follow-up input domains
  based on the provided 1-N MR;
Record the number of executed STCs as M;
Initialize the set of executed STCs as empty (M = 0);
Initialize the N sets of executed FTCs as empty;
while no stopping conditions are triggered do
    Partition the source input domain into M + 1 equally-sized subdomains from
      the edge to the center;
    Choose k source candidates at random from the empty subdomains;
    for n = 1 → k do
        Choose the n^th source candidate and construct N follow-up candidates based
          on the provided MR;
        Compute the distance (SMD) between this source candidate and its nearest
          executed STC, recorded as sd_n;
        Compute the sum of the distances (FMDs) between each follow-up candidate
          and its nearest executed FTC from the same follow-up input domain,
          recorded as fd_n;
        Denote the sum of sd_n and fd_n as D_n;
    end
    Choose the candidate MG containing the largest D_n value;
    if more than one MGs containing the same maximum D_n value then
        Randomly select one of them;
    end
    Execute the SUT using the selected MG, and inspect for MR violations;
    Append the currently executed test cases to their respective executed test sets
      (M = M + 1);
end
```

problem is caused by an improper treatment of FTC. Hence, in order to mitigate the impact of this problem, this chapter suggests adopting a novel STC and FTC allocation principle (ensuring an even distribution of STCs and FTCs across their respective input domains). To achieve this principle, each input domain is given a distinct set of executed test cases.

- Step 7: SFIDMT-ART partitions the source input domain into $M+1$ equally-sized subdomains from the edge to the centre, for generating source candidates. Fig. 10 illustrates three prototypical examples of partitioning a square/rectangle/triangle input domain into two equally-sized subdomains in a 2D input domain. The rationale behind setting the number of subdomains to $M+1$ is to guarantee the existence of at least one empty subdomain.

- Step 8: SFIDMT-ART maps all STCs in the executed STC set to their corresponding input domain and searches for empty subdomains. SFIDMT-ART then randomly generates k STCs as source candidates from the empty subdomains.

- Step 10: For each source candidate, SFIDMT-ART generates N FTCs as follow-up candidates based on the MR, in order to obtain k MGs as candidate MGs. The use of k candidate MGs is because, in MT, FTCs are fully determined by the STCs and the MR, rather than generated by the testers/researches from scratch. In this context, SFIDMT-ART generates k MGs as candidate MGs, and compares their qualities based on the ID-MDs, with the aim of achieving the proposed allocation principle.
- Steps 11-13: SFIDMT-ART compares the quality of the k candidate MGs by computing and comparing the sum of the ID-MDs.
- Step 15: SFIDMT-ART chooses the candidate MG containing the maximum value of the sum of ID-MDs. The rationale behind this step is to choose the MG that is the furthest from previously-executed MGs among the k candidate MGs.
- Steps 16-18: If more than one candidate MG meets the criterion, SFIDMT-ART then randomly chooses one among them for execution.
- Step 19: SFIDMT-ART executes the selected MG against the SUT and checks whether or not the MR is violated.
- Step 20: SFIDMT-ART adds the STC from the currently executed MG into the respective executed STC set, and the FTCs into their respective executed FTC sets.
- Step 21: If no stopping conditions are met, such as no violations are detected, SFIDMT-ART proceeds to Step 5; otherwise, SFIDMT-ART ends the testing and reports the results.

Algorithm 2 exclusively considers 1-N MRs. The overall framework remains similar for other types of MRs. Applying SFIDMT-ART to 1-1 MRs or M-N MRs, for instance, contains the following distinctions:

(1) In the case of 1-1 MRs: SFIDMT-ART features a single source input domain and its corresponding executed STC set, as well as a single follow-up input domain and its corresponding executed FTC set. While the computation of sd_n (Algorithm 2 Step 11) remains unchanged, the computation of fd_n (Algorithm 2 Step 12) varies: SFIDMT-ART calculates the distance (FMD) from each follow-up candidate to its nearest executed FTC, eliminating the need for summing FMDs.

(2) In the case of M-N MRs: SFIDMT-ART involves M source input domains with their respective executed STC sets, as well as N follow-up input domains with their respective executed FTC sets. While the computation of fd_n (Algorithm 2 Step 12) remains consistent, the computation of sd_n (Algorithm 2 Step 11) differs: SFIDMT-ART computes the sum of distances (SMD) from each source candidate to its nearest executed STC.

As for how to partition a 1D input domain, a typical example is shown as follows. The scope of the input domain is $[0, 100]$. In this example, SFIDMT-ART partitions the input domain into two subdomains from the centre to the edges. Specifically, SFIDMT-ART partitions the input domain into three parts: The scope of the left part is $[0, 25]$; the

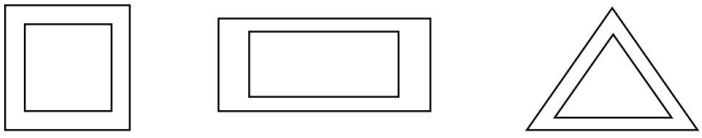


Figure 10: Three examples of dividing a 2D input domain

center part is $[25, 75]$; and the right part is $[75, 100]$. The combination of the left part and the right part denotes a subdomain, and the middle one denotes another subdomain.

SFIDMT-ART can also generate MGs for systems with non-numerical inputs. For example, the category and choice technique, based on a category-partition method [18], can compute the distance between non-numerical MGs. Testers identify input parameters or environmental conditions (referred to as categories) that may influence SUT behavior. Each category includes a series of disjoint partitions (referred to as choices), which represent the potential values of this category. An input is formed by the choices from different categories. As an example, consider a student registration system with three categories: Name, Age, and Nationality. In this scenario, testers can generate inputs such as (Peter, 21, Japanese) and (John, 19, Chinese).

Below is an example of applying SFIDMT-ART to systems with non-numerical inputs: After determining the categories and choices based on the SUT, SFIDMT-ART generates k candidate MGs. SFIDMT-ART selects the n^{th} ($n \leq k$) candidate MG and verifies whether or not the choice of its source candidate matches any of the executed STCs to calculate the SMDs (represented by sd_n); and whether or not the choice of its follow-up candidate matches any of the executed FTCs to compute the FMDs (represented by fd_n). SFIDMT-ART increments the distance by 1 if the respective choices differ. Lastly, SFIDMT-ART calculates the sum of sd_n and fd_n, represented by D_n, and chooses the candidate MG with the highest D_n for execution.

4.2.3 *Characteristics of SFIDMT-ART*

SFIDMT-ART differs from the MT-ART algorithm proposed by Hui et al. [144] in the following aspects:

1. The way of handling STCs and FTCs during the MG-generation process are different. MT-ART treats STCs and FTCs as a unified entity when generating new MGs, aiming to enhance MT performance by achieving an even distribution of STCs and FTCs throughout the entire input domain. However, MT-ART may encounter the input-domain difference problem, which may lead to uneven distribution issues. In order to address this, this thesis suggests that an MG-generation algorithm should treat FTCs separately from STCs. SFIDMT-ART aims to improve MT performance by achieving an even distribution of STCs and FTCs across their respective input domains. In this context, this thesis introduces the SFIDMT-ART algorithm, aiming to enhance MT

performance by achieving an even distribution of STCs and FTCs throughout their respective input domains.

2. SFIDMT-ART and MT-ART employ different methods for computing the distance between test cases. During the MG-generation process, MT-ART concentrates on not only the distance between source/follow-up candidates and all executed test cases (d_1 and d_3), but also the distance among source candidates and follow-up candidates (d_2). In contrast, SFIDMT-ART does not consider the distance among source candidates and follow-up candidates. It focus on only the distance between source candidates (SMD) and executed STCs and the distance between follow-up candidates and executed FTCs (FMD).

   The rationale behind is explained as follows: Firstly, It has been noted that there is a significant correlation between the fault-detection capability of MRs and the dissimilarity between the STC execution profiles and relevant FTC execution profiles (e.g., measured by statement coverage and branch coverage) [37]. These dissimilarity metrics are white-box metrics, and the use of Euclidean distances between test cases on the input domain (as a black-box metric) has not been explored. Secondly, adopting d_2 may not facilitate the even distribution of STCs and FTCs, and sometimes it may even hinder this process. For example, consider MR_{tanh} (Section 4.4) as an example: The distance between the STC and the FTC is $|3 * x - x| = |2x|$. As the absolute value of x increases, the distance between the STC and the FTC also increases, resulting in a higher likelihood of selecting test cases further away from zero. To sum up, the Euclidean distance metric is unsuitable for quantitatively measuring MG dissimilarities. Therefore, SFIDMT-ART does not consider d_2 in the process of MG generation.

3. SFIDMT-ART is applicable to systems with non-numerical inputs, as explained in Section 4.2.2. In contrast, Hui et al. [144] only discuss the application of MT-ART to the systems with numerical input domains.

4. SFIDMT-ART employs only the minimum distance criterion to compute the distance between test cases, while MT-ART includes three criteria (maximum, average, and minimum).

5. MT-ART partitions the entire input domain and selects an empty subdomain for source candidate generation. Nevertheless, as the entire input domain includes the source and follow-up input domains, the selected subdomain may partially or fully extend beyond the source input domain, rendering it unusable for generating new source candidates. Hui et al. [144] did not provide an explanation of how to address this issue. Conversely, SFIDMT-ART avoids this issue by partitioning the source input domain into subdomains and selecting empty ones for source candidate generation.

6. The time complexity of SFIDMT-ART is influenced by the MR types. Suppose k denotes the number of candidate MGs generated in each iteration and n represents the number of MGs to be generated, the time complexity of SFIDMT-ART is $O(n * k) = O(n^2)$ with 1-1 MRs, and is $O(n * k * Max(N, M)) = O(n^3)$ with 1-N

MRs, M-1 MRs or M-N MRs. In this context, a conclusion can be drawn that the time complexity of SFIDMT-ART is much lower than that of MT-ART ($O(n^4)$ [144]).

4.3 RESEARCH QUESTIONS

The following RQs were formulated to provide guidance for the empirical experiments:
RQ1: Do the Input-Domain Difference and Edge-Preference Problems have an impact on the performance of MG-generation algorithms?

- Objective: Investigate whether or not MT-ART and SFIDMT-ART are affected by the input-domain difference and edge-preference problems.
- Motivation: Firstly, given that SFIDMT-ART was introduced to mitigate the input-domain difference problem inherent in MT-ART, an examination of the effect of the input-domain difference problem on both MT-ART and SFIDMT-ART will be conducted. Secondly, since MT-ART was formulated upon the FSCS-ART algorithm, susceptible to the edge-preference problem, an analysis of the influence of this problem on both MT-ART and SFIDMT-ART will be undertaken.
- Methodologies:
  1. Apply MRVR to explore and analyze the impact of the two problems on the performance of MT-ART and SFIDMT-ART.
  2. Determine whether or not MT-ART is affected by these problems and whether or not SFIDMT-ART can mitigate them.

RQ2: What are the performance differences between SFIDMT-ART, MT-ART, and MT-RT?

- Objective: Evaluate whether or not SFIDMT-ART outperforms MT-ART and MT-RT.
- Motivation: MT-RT was selected for the empirical experiments as it is the most popular MG-generation algorithm [64, 246]. Additionally, since SFIDMT-ART is proposed as an enhanced algorithm over MT-ART, MT-ART was also selected for the experiments.
- Methodologies:
  1. Select three performance criteria: Test efficiency, test effectiveness, and MG diversity.
  2. Design sub-RQs to delve deeper into specific aspects of performance.
  3. Determine whether or not SFIDMT-ART performs better across multiple performance metrics compared to MT-ART and MT-RT.

- Sub-RQs:

1. Can SFIDMT-ART achieve better test effectiveness (F-measure) than MT-ART and MT-RT?
2. Can SFIDMT-ART achieve better test efficiency (generation time) than MT-ART?
3. Can SFIDMT-ART achieve better distribution diversity of MGs (Dispersion and Discrepancy) than MT-ART and MT-RT?

RQ3: Does the STC and FTC allocation principle have an impact on the performance of MG-generation algorithms?

- Objective: Investigate whether or not MT-ART and SFIDMT-ART are affected by the input-domain difference and edge-preference problems.
- Motivation: Hui et al. [144] suggested that MT performance can be enhanced by making STCs and FTCs evenly distributed throughout the entire input domain. In this chapter, a novel STC and FTC allocation principle is introduced: Ensuring the even distribution of STCs and FTCs throughout their respective input domains. The aim is to investigate and analyze which criterion could further augment the fault-detection capability of MT, and to ascertain the underlying reasons if the divergent utilization of executed STCs and FTCs during the MG-generation process results in disparate MT performances.
- Methodology:
  1. Analyze differences in MT performance resulting from different approaches to utilizing executed STCs and FTCs.
  2. Determine whether or not the proposed criterion enhances the fault-detection capability of MT and identify any root causes for differences in MT performance.

4.4 EMPIRICAL EXPERIMENTS

4.4.1 *Experimental Setup*

The empirical experiments selected systems of varied sizes and dimensions as well as the corresponding MRs sourced from previously-published studies in the fields of MT or ART, as listed in Table 4. Artificial faults were manually inserted to create mutants according to the mutation operators [149]. The introduction of these SUTs and mutation operators is provided in Section 2, while the introduction of the selected MRs is provided in Appendix A. As SFIDMT-ART and MT-ART with Strategy 1 primarily focus on the distance between (source and follow-up) candidates and executed test cases, while MT-ART with Strategy 2 focuses on the distance among source candidate and follow-up candidates, this chapter sorely adopted MT-ART with Strategy 1 in the experiments. Consequently, this chapter chose the following three MT-ART algorithms: MT-ART with Strategy 1 and Max Distance (MT-ART-Max), MT-ART with Strategy 1 and Avg Distance (MT-ART-Avg), and MT-ART

Table 4: Information of the SUTs and MRs

Systems	Sin	tanh	Erf	BesselJ	sncndc	TriSquare	TriSquarePlus	rj	PntLinePos
Input Dimension	1	1	1	2	2	3	3	4	6
Entire Input Domains	(0,1000)	(0,1000)	(-500,500)	((1,1), (100,100))	((0,0), (100,100))	((0,0,0), (100,100,100))	((0,0,0), (100,100,100))	((0,0,0,0), (100,100, 100,100))	((0,0,0,0,0,0), (100,100,100, 100,100,100))
Source Input Domains	(0,1000)	(0,333.33)	(-500,500)	((2,2), (99,99))	((0,0), (33.33,33.33))	((0,0,0), (33.33,33.33,33.33))	((0,0,0), (100,100,100)), ((0,0,0), (100,100,100))	((0,0,0,0), (100,100, 100,100))	((0,0,0,0,0,0), (33.33,33.33, 33.33,33.33, 33.33,33.33))
Follow-up Input Domains	(0,333.33)	(0,1000)	(-500,500)	((1,1), (99,99), (2,2), (100,100))	((0,0), (100,100))	((0,0,0), (100,100,100))	((0,0,0), (33.33,33.33,33.33)), ((0,0,0), (100,100,100))	((0,0,0,0), (100,100, 100,100))	((0,0,0, 0,0,0), (100,100,100, 100,100,100))
Size (LOC)	120	18	763	1211	64	38	31	175	23
MRs	MR_{Sin13}	MR_{tanh}	MR_{Erf1}	$MR_{BesselJ1}$	$MR_{sncndn2}$	$MR_{TriSquare6}$	$MR_{TriPlus2}$, $MR_{TriPlus12}$	MR_{rj2}	MR_{Pnt2}
Number of Mutants	2	2	2	3	2	3	6	2	2
Fault Type	CRP,ROR	CRP	CRP	CRP,ROR,AOR	CRP,ROR,AOR	CRP,ROR,AOR	CRP,ROR,RSR,AOR	CRP,ROR	CRP,ROR

with Strategy 1 and Min Distance (MT-ART-Min). The source code of all algorithms is publicly-available on Github[1]. In accordance with the principles of ART algorithms, the value of k (representing the number of candidate MGs) was set to 10 for both SFIDMT-ART and MT-ART. A total of 1000 MGs were generated using each algorithm and a corresponding MR, for computing Discrepancy and Dispersion. All experiments were iterated 10,000 times to calculate the average F-measure, generation time, Discrepancy, and Dispersion.

The major contributions of Chapter 3 are: Propose the concept of MRVRs, propose three types of MRVRs (block, point and strip), and propose the simulator. In this chapter, experiments were conducted to evaluate the performance of MG-generation algorithms (using MRVRs) through the following steps.

1. Artificial faults were manually inserted into the systems to create mutants.
2. MGs were generated using given MG-generation algorithms and MRs.
3. The generated MGs were executed against the mutants, and the F-measure values were recorded.
4. Through the experiments, it was found that MT-ART may sometimes perform much worse than MT-RT. In particular, the concept of MRVR was used to investigate the causes of this situation through Step 5.
5. For each mutant, all the STCs and FTCs in the input domain were executed and the outputs were checked against the respective MRs. Then the STCs (as well as the corresponding FTCs) that violate the given MRs were identified, and then they were categorized according to the three types of MRVRs. According to the size and location of input domains and the location of MRVRs, they were divided into several cases, which are shown in Table 6. The detailed processes are shown as follows:

1 *https://github.com/scxzy2/SFIDMT-ART.git*

a) Select one given MR (and a related mutant), and compare the size and location of its source input domain and follow-up input domain. Categorize them according to the following criteria:

   A: The source input domain is inside the follow-up input domain (the source input domain is smaller than the follow-up input domain).

   B: The source input domain is the identical or very similar to the follow-up input domain.

   C: The follow-up input domain is inside the source input domain (the source input domain is larger than the follow-up input domain).

b) For the selected MR (and a related mutant), check the location of its MRVR-S, and categorize it according to the following criteria:

   1: The MRVR-S is close to the center of the source input domain.

   2: The MRVR-S is close to the edge of the source input domain.

   3: The MRVR-S is distributed over the source input domain.

   4: The MRVR-S is close to the center of the follow-up input domain.

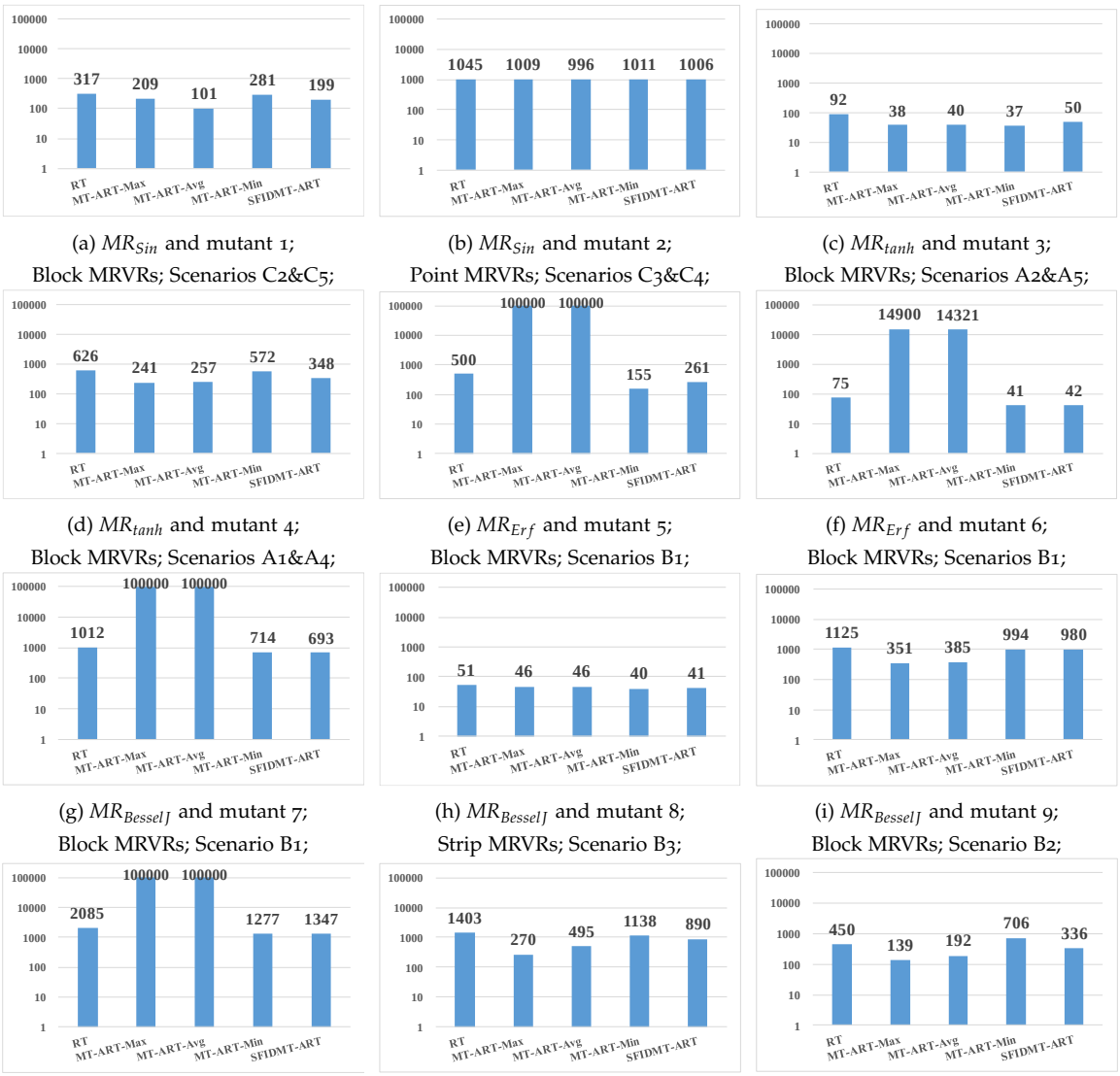
   5: The MRVR-S is close to the edge of the follow-up domain.

c) Combine the above two classifications for each pair of MR and mutant. For instance, for a given MR (and a related mutant), if its source input domain is inside its follow-up input domain, and its MRVR-S is close to the center of the source input domain, then it was categorized as A1.

6. Through the investigation of MT-ART using MRVRs, it was found that the differences between STC input domain and FTC input domain were ignored by MT-ART (since it treats STCs and FTCs as a unified entity), which may have an impact on its performance. This problem was referred to as the input-domain difference problem.

As for the MT simulator, additional MT simulations will be conducted in the future, to further explore the problem and findings from another perspective (e.g., the existence and impact of the input-domain difference problem). The detailed experimental steps will be:

1. Select one MR and one respective previously-identified MRVR-S.
2. Execute MT-ART, SFIDMT-ART or MT-RT to generate MGs.
3. Check if the generated STCs are within the MRVR-S.
4. Compare the results (the outputs and the distribution of generated STCs in the source input domain) to see if the algorithms under test were affected by the input-domain difference problem: Check if the F-measure experimental results of an algorithm are abnormal (very high/low compared with the results of other algorithms) when the source input domain is different from the follow-up input domain



(a) MR_{Sin} and mutant 1; Block MRVRs; Scenarios C2&C5;

(b) MR_{Sin} and mutant 2; Point MRVRs; Scenarios C3&C4;

(c) MR_{tanh} and mutant 3; Block MRVRs; Scenarios A2&A5;

(d) MR_{tanh} and mutant 4; Block MRVRs; Scenarios A1&A4;

(e) MR_{Erf} and mutant 5; Block MRVRs; Scenarios B1;

(f) MR_{Erf} and mutant 6; Block MRVRs; Scenarios B1;

(g) $MR_{BesselJ}$ and mutant 7; Block MRVRs; Scenario B1;

(h) $MR_{BesselJ}$ and mutant 8; Strip MRVRs; Scenario B3;

(i) $MR_{BesselJ}$ and mutant 9; Block MRVRs; Scenario B2;

(j) MR_{sncndn} and mutant 10; Block MRVRs; Scenarios A1&A4;

(k) MR_{sncndn} and mutant 11; Block MRVRs; Scenarios A2&A5;

(l) $MR_{TriSquare}$ and mutant 12; Block MRVRs; Scenarios A2&A5;

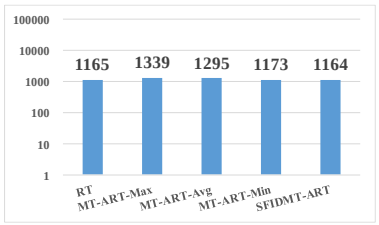
Figure 11: F-measure Experimental Results of the MG-generation Algorithms (Part 1)

4.4.2 *Experimental Results and Discussion*

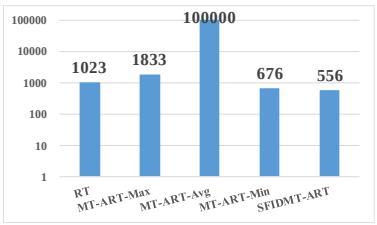
4.4.2.1 *F-Measure and Cohen's* d

The experimental results of F-measure and Cohen's d are presented in Figs. 11 - 5: In Figs. 11 - 12, the vertical axis denotes the F-measure value, while the horizontal axis represents the algorithm under test; and in Fig. 5, the vertical axis denotes the Cohen's d value, while the horizontal axis represents the algorithm under test. During the experiment, all algorithms were terminated when generating and executing 100,000 MGs without detecting any MR violations. Consequently, the Cohen's d values were not computed for these scenarios and were denoted as NaN (Not a Number).

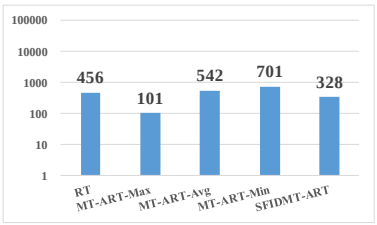
The F-measure and Cohen's d experimental results revealed that under block MRVRs, SFIDMT-ART was capable of consistently outperforming MT-RT, while the three MT-ART



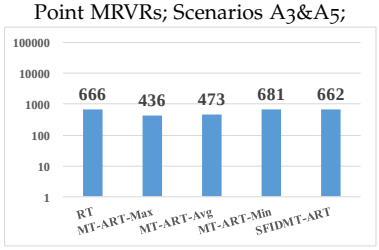
(m) $MR_{TriSquare}$ and mutant 13; Point MRVRs; Scenarios A3&A5;



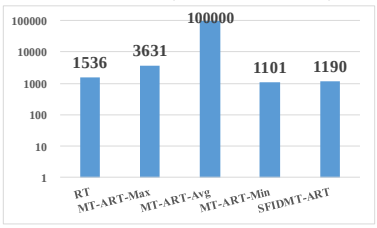
(n) $MR_{TriSquare}$ and mutant 14; Block MRVRs; Scenario A1&E;



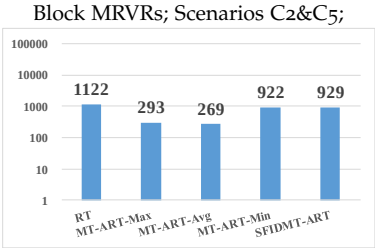
(o) $MR_{TriPlus1}$ and mutant 15; Block MRVRs; Scenarios C2&C5;



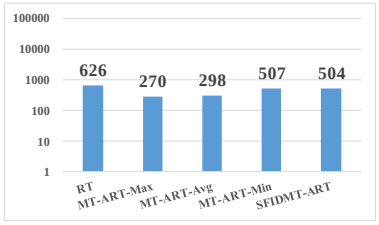
(p) $MR_{TriPlus1}$ and mutant 16; Strip MRVRs; Scenarios C3&C4;



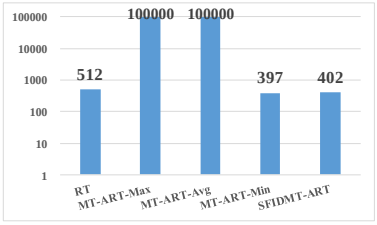
(q) $MR_{TriPlus1}$ and mutant 17; Block MRVRs; Scenarios C1&C4;



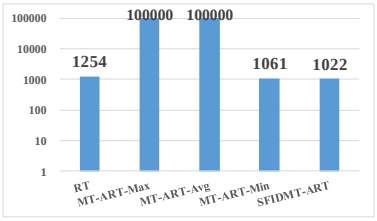
(r) $MR_{TriPlus2}$ and mutant 18; Block MRVRs; Scenario B2;



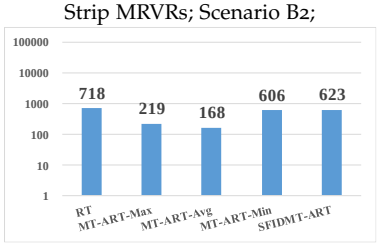
(s) $MR_{TriPlus2}$ and mutant 19; Strip MRVRs; Scenario B2;



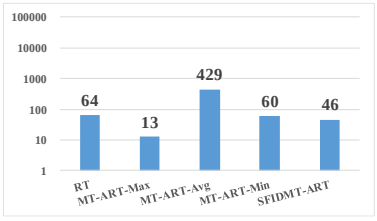
(t) $MR_{TriPlus2}$ and mutant 20; Block MRVRs; Scenario B1;



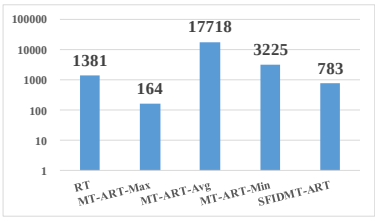
(u) MR_{rj} and mutant 21; Block MRVRs; Scenario B1;



(v) MR_{rj} and mutant 22; Block MRVRs; Scenario B2;



(w) MR_{Pnt} and mutant 23; Block MRVRs; Scenarios A2&A5;



(x) MR_{Pnt} and mutant 24; Block MRVRs; Scenarios A2&A5;

Figure 12: F-measure Experimental Results of the MG-generation Algorithms (Part 2)

algorithms occasionally exhibited superior performance to MT-RT; and under point/strip MRVRs, all the algorithms typically exhibited similar performance.

According to the experimental results for Cohen's d and the strengths of effect sizes in different ranges shown in Table 1, it can be observed that among the cases with block MRVRs, about 30% of the results have values greater than 0.5 (medium effect size strength), and the highest value is 0.64; and about 60% of the results have values less than 0.5 and greater than 0.2 (which means that there are small effect size strengths); and about 10% of the results have values less than 0.2 and greater than 0.01 (which means that there are very small effect size strengths). Among these results, the highest value appeared when using Erf, which was a medium strength; the lowest value appeared when using rj, where the strength value was a very small strength. These F-measure and Cohen's d experimental results indicate that under block MRVRs, SFIDMT-ART was capable of consistently outperforming MT-RT (and sometimes even significantly better than MT-RT),

Table 5: Effect Size (Cohen's *d*) Experimental Results of the MG-generation Algorithms

SUTs	MRs	Mutants	MRVR Type	Statistical Analysis							
				SFIDMT-ART vs RT		SFIDMT-ART vs MT-ART-Max		SFIDMT-ART vs MT-ART-Ave		SFIDMT-ART vs MT-ART-Min	
				p-value	effect size	p-value	effect size	p-value	effect size	p-value	effect size
Sin	MR_{Sin}	mutant 1	Block	0.000	0.480	0.000	0.059	0.000	-0.827	0.000	0.457
		mutant 2	Point	0.006	0.040	0.832	0.005	0.471	-0.009	0.721	0.007
tanh	MR_{tanh}	mutant 3	Block	0.000	0.590	0.000	-0.379	0.000	-0.299	0.000	-0.357
		mutant 4	Block	0.000	0.590	0.000	-0.513	0.000	-0.378	0.000	0.576
Erf	MR_{Erf}	mutant 5	Block	0.000	0.640	NaN	NaN	NaN	NaN	0.000	-0.745
		mutant 6	Block	0.000	0.580	0.000	2.972	0.000	2.628	0.011	-0.041
BesselJ	$MR_{BesselJ}$	mutant 7	Block	0.000	0.390	NaN	NaN	NaN	NaN	0.008	0.030
		mutant 8	Strip	0.000	0.250	0.000	0.125	0.000	0.132	0.060	-0.033
		mutant 9	Block	0.000	0.160	0.000	-0.902	0.000	-0.836	0.472	0.017
sncndn	MR_{sncndn}	mutant 10	Block	0.000	0.460	NaN	NaN	NaN	NaN	0.000	-0.068
		mutant 11	Block	0.000	0.460	0.000	-1.085	0.000	-0.612	0.000	0.266
TriSquare	$MR_{TriSquare}$	mutant 12	Block	0.000	0.290	0.000	-0.823	0.000	-0.547	0.000	0.700
		mutant 13	Point	0.950	0.020	0.000	-0.551	0.000	0.246	0.571	0.042
		mutant 14	Block	0.000	0.600	0.000	1.020	NaN	NaN	0.000	0.250
TriSquarePlus	$MR_{TriSquarePlus1}$	mutant 15	Block	0.000	0.330	0.000	-0.977	0.000	0.497	0.000	0.726
		mutant 16	Strip	0.665	0.010	0.000	-0.398	0.000	-0.325	0.044	0.035
		mutant 17	Block	0.000	0.260	0.000	0.950	NaN	NaN	0.000	-0.092
	$MR_{TriSquarePlus2}$	mutant 18	Block	0.000	0.200	0.000	-1.025	0.000	-1.071	0.549	-0.009
		mutant 19	Strip	0.000	0.220	0.000	-0.585	0.000	-0.502	0.675	0.007
		mutant 20	Block	0.000	0.260	NaN	NaN	NaN	NaN	0.275	-0.016
rj	MR_{rj}	mutant 21	Block	0.000	0.250	NaN	NaN	NaN	NaN	0.006	0.054
		mutant 22	Block	0.000	0.140	0.000	-0.881	0.000	-0.976	0.043	-0.027
PntLinePos	$MR_{PntLinePos}$	mutant 23	Block	0.000	0.340	0.000	-1.140	0.000	1.280	0.000	0.311
		mutant 24	Block	0.000	0.540	0.000	-1.150	0.000	1.621	0.000	1.162

Table 6: The Relationship between MRVRs and Input Domains

Size and Location of Input Domains	MRVR-S Location	Scenarios
The source input domain is inside the follow-up input domain (the source input domain is smaller than the follow-up input domain)	close to the center of the source input domain	A1
	close to the edge of the source input domain	A2
	distributed over the source input domain	A3
	close to the center of the follow-up input domain	A4
	close to the edge of the follow-up domain	A5
The source input domain is the identical or very similar to the follow-up input domain	close to the center of entire input domain	B1
	close to the edge of entire input domain	B2
	distributed over the entire input domain	B3
The follow-up input domain is inside the source input domain (the source input domain is larger than the follow-up input domain)	close to the center of the source input domain	C1
	close to the edge of the source input domain	C2
	distributed over the source input domain	C3
	fully/mostly outside the follow-up input domain	C4
	fully/mostly inside the follow-up input domain	C5

while the three MT-ART algorithms occasionally exhibited superior performance to MT-RT; and under point/strip MRVRs, all the algorithms typically exhibited similar performance.

To further explore the influence of the input-domain difference problem on the MT-ART algorithms, the concept of MRVRs (introduced in Section 3.2) was adopted [284]. The MRVR-S (STC-only components of MRVRs) of all SUTs with relevant MRs have been identified, and the relationship between the input domains and the MRVRs has been categorized based on their relative size and location, as illustrated in Table 6.

Further observations can be made from Figures 11 - 5, as introduced below.

- In contrast to MT-ART-Max and MT-ART-Avg, MT-ART-Min exhibited greater stability: It never required more than 100,000 MGs to detect the initial MR violation and never significantly outperformed SFIDMT-ART in terms of F-measure. The problem is that in Scenarios A2 and C2, the performance of MT-ART-Min markedly lagged behind that of SFIDMT-ART and even MT-RT. The underlying cause of this issue may be the input-domain difference problem, which resulted in the uneven distribution of STC and FTC generated by MT-ART-Min. For example, Fig. 8 illustrates a typical instance of the STCs generated by MT-ART-Min. If MT-ART-Min encounters Scenario A2 — the MRVR-S resides in the bottom-left corner of the source input domain — it would exhibit poor performance. Similarly, MT-ART-Max and MT-ART-Avg may encounter this problem, with SFIDMT-ART being the only algorithm capable of addressing it and achieving significantly better performance.

- MT-ART-Max and MT-ART-Avg occasionally exhibited significantly better performance compared to MT-ART-Min, particularly in Scenarios A2, B2, and C2. Nevertheless, they may occasionally perform quite poorly. For instance, in Scenarios A1, B1, and C1, sometimes 100,000 MGs were generated and executed using MT-ART-Max or MT-ART-Avg, but the first MR violation had not yet been revealed. This could be caused by the influence of both the input-domain difference problem and the edge-preference problem. Due to the edge-preference problem, the STCs and FTCs produced by MT-ART-Max or MT-ART-Avg exhibit a trend of clustering near the boundary of the input domain rather than being uniformly distributed. For instance, in Scenario B2, the MRVR-S was positioned near the boundary of the entire input domain, which may result in the two MT-ART algorithms significantly outperforming all other algorithms. On the contrary, in Scenario B1, where the MRVR-S was close to the center of the entire input domain, both MT-ART algorithms performed poorly.

- Based on the above experimental results and discussions, the following conclusion can be drawn:

  Answer to RQ2.1: SFIDMT-ART generally exhibits relatively stable and superior fault-detection capabilities compared to all other algorithms under test.

4.4.2.2 *Generation Time*

The experimental results of the mean generation time of all algorithms are described in Fig. 13: The vertical axis denotes generation time and the horizontal axis represents the algorithms under test.

It can be observed that SFIDMT-ART consistently required the least amount of time to generate 10,000 MGs, across all scenarios, indicating its superior test efficiency compared to the three MT-ART algorithms. The following observations and conclusions can be made:

1. All three MT-ART algorithms and SFIDMT-ART are affected by the input domain dimension: As the input domain dimension increases, the time required for MT-ART and SFIDMT-ART to generate a certain number of MGs also increases. Between them,

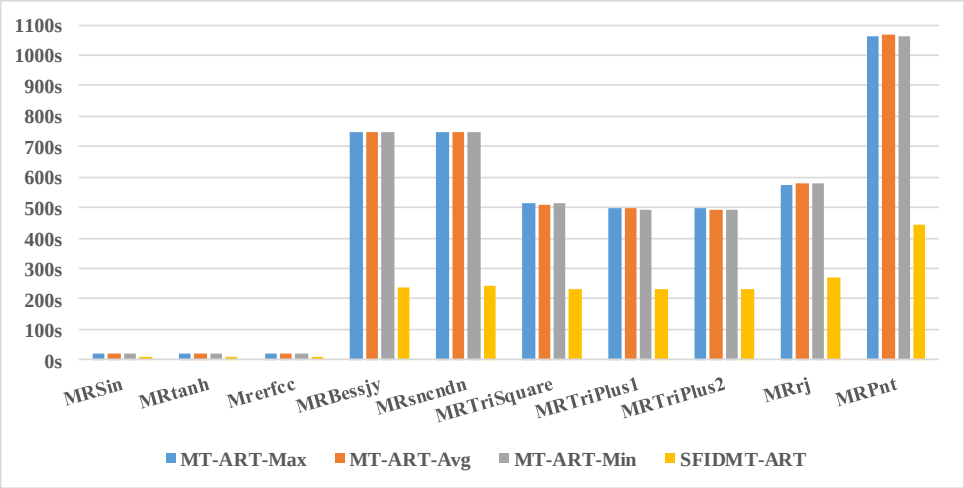


Figure 13: Generation Time Experimental Results of the MG-generation Algorithms

MT-ART is more sensitive to the changes in the input domain dimension. That is, as the input domain dimension increases, the generation time for MT-ART typically increases faster than that of SFIDMT-ART.

2. When using $MR_{BesselJ}$, the generation time of MT-ART was very high, which was even larger than that of MR_{rj}. In addition, the difference in time between SFIDMT-ART and the three MT-ART algorithms was largest when $MR_{BesselJ}$ was used. The possible reason for this phenomenon is that MT-ART is also influenced by the type of MRs: $MR_{BesselJ}$ is a *1-2* MR, while MR_{sncndn} is a *1-1* MR. In other words, given an *M-N* MR, as the value of M or N increases, the generation time for MT-ART also increases. This is different from SFIDMT-ART, which may not be affected by the type of MR: The generation time result of SFIDMT-ART using $MR_{BesselJ}$ is almost the same as that of MR_{sncndn}. Therefore, it is expected that, in theory, for an *M-N* MR, SFIDMT-ART can be capable of outperforming MT-ART in terms of generation time and the time difference between them may increase with higher values of M and N. In the future, empirical studies will be conducted to future investigate the above analysis.

However, there is still room for improvement in both effectiveness and efficiency of SFIDMT-ART. With this consideration, Section 18 introduced SFIDMT-BART, which is an enhanced version of SFIDMT-ART. This algorithm attempts to improve the performance of SFIDMT-ART by combining it with MT-BART (proposed in Chapter 5).

In addition to time complexity, this chapter also aims to investigate and analyze the main reasons why SFIDMT-ART was capable of outperforming the three MT-ART algorithms in terms of generation time from the following perspectives:

- MT-ART typically needs more time for calculating the distance between test cases. In particular, MT-ART computes not only the distance from each candidate to all executed test cases but also the distances among all test cases from a single candidate MG, while SFIDMT-ART only involves the computation of the distance from each source candidate to relevant executed STCs, and the distance from each follow-up candidate to relevant executed FTCs.

Table 7: Dispersion and Discrepancy Experimental Results of the MG-generation Algorithms

MRs	Algorithms	Dispersion						Discrepancy					
		STCs in the STC Input Domain			FTCs in the FTC Input Domain			STCs in the STC Input Domain			FTCs in the FTC Input Domain		
		Min	Max	Max-Min	Min	Max	Max-Min	Min	Max	Max-Min	Min	Max	Max-Min
MR_{Sin}	MT-RT	0.0010	3.7800	3.7790	0.0003	1.2600	1.2597	0	0.0045	0.0045	0	0.0045	0.0045
	MT-ART-Max	0.0100	73.8000	73.7900	0.0033	24.6000	24.5967	0	0.0053	0.0053	0	0.0053	0.0053
	MT-ART-Avg	0.0108	59.2000	59.1892	0.0036	19.7000	19.6964	0	0.0049	0.0049	0	0.0049	0.0049
	MT-ART-Min	0.2640	6.2200	5.9560	0.0880	2.0700	1.9820	0	0.0020	0.0020	0	0.0020	0.0020
	SFIDMT-ART	0.4200	1.6900	1.2700	0.1400	0.5640	0.4240	0	0.0010	0.0010	0	0.0010	0.0010
MR_{tanh}	MT-RT	0.0003	1.2600	1.2597	0.0010	3.7800	3.7790	0	0.0045	0.0045	0	0.0045	0.0045
	MT-ART-Max	0.0034	24.5000	24.4966	0.0101	73.4000	73.3899	0	0.0051	0.0051	0	0.0051	0.0051
	MT-ART-Avg	0.0039	16.7000	16.6962	0.0116	50.0000	49.9884	0	0.0046	0.0046	0	0.0046	0.0046
	MT-ART-Min	0.0903	1.3700	1.2797	0.2710	4.1200	3.8490	0	0.0020	0.0020	0	0.0020	0.0020
	SFIDMT-ART	0.1400	0.5640	0.4240	0.4200	1.6900	1.2700	0	0.0010	0.0010	0	0.0010	0.0010
MR_{Erf}	MT-RT	0.0010	3.7800	3.7790	0.0010	3.7800	3.7790	0	0.0045	0.0045	0	0.0045	0.0045
	MT-ART-Max	0.0119	38.9000	38.8881	0.0119	38.9000	38.8881	0	0.0045	0.0045	0	0.0045	0.0045
	MT-ART-Avg	0.0120	52.5000	52.4880	0.0120	52.5000	52.4880	0	0.0045	0.0045	0	0.0045	0.0045
	MT-ART-Min	0.2250	3.0700	2.8450	0.2250	3.0700	2.8450	0	0.0021	0.0021	0	0.0021	0.0021
	SFIDMT-ART	0.4200	1.6900	1.2700	0.4200	1.6900	1.2700	0	0.0010	0.0010	0	0.0010	0.0010
$MR_{BesselJ}$	MT-RT	0.0692	5.2861	5.2169	0.0692	5.2861	5.2169	0	0.0046	0.0046	0	0.0046	0.0046
	MT-ART-Max	0.0497	10.2096	10.1599	0.0497	10.2096	10.1599	0	0.0077	0.0077	0	0.0077	0.0077
	MT-ART-Avg	0.0531	9.7781	9.7250	0.0531	9.7781	9.7250	0	0.0070	0.0070	0	0.0070	0.0070
	MT-ART-Min	0.8791	4.2297	3.3506	0.8791	4.2297	3.3506	0	0.0029	0.0029	0	0.0029	0.0029
	SFIDMT-ART	1.4170	4.1938	2.7768	1.4170	4.1938	2.7768	0	0.0021	0.0021	0	0.0021	0.0021
MR_{sncndn}	MT-RT	0.0235	1.7900	1.7665	0.0706	5.3702	5.2996	0	0.0045	0.0045	0	0.0045	0.0045
	MT-ART-Max	0.0143	5.0800	5.0657	0.0428	15.2400	15.1972	0	0.0108	0.0108	0	0.0108	0.0108
	MT-ART-Avg	0.0171	3.6500	3.6329	0.0512	10.9498	10.8986	0	0.0081	0.0081	0	0.0081	0.0081
	MT-ART-Min	0.4490	1.7000	1.2510	1.348	5.1010	3.7530	0	0.0022	0.0022	0	0.0022	0.0022
	SFIDMT-ART	0.4780	1.4100	0.9320	1.434	4.2300	2.7960	0	0.0021	0.0021	0	0.0021	0.0021
$MR_{TriSquare}$	MT-RT	0.2345	4.8184	4.5839	0.7035	14.4551	13.7517	0	0.0045	0.0045	0	0.0045	0.0045
	MT-ART-Max	0.1469	7.7325	7.5856	0.4408	23.1975	22.7567	0	0.0120	0.0120	0	0.0120	0.0120
	MT-ART-Avg	0.1660	6.8391	6.6731	0.4980	20.5173	20.0193	0	0.0089	0.0089	0	0.0089	0.0089
	MT-ART-Min	1.8198	4.5100	2.6902	5.4595	13.5300	8.0705	0	0.0027	0.0027	0	0.0027	0.0027
	SFIDMT-ART	1.8334	4.2797	2.4463	5.5002	12.8391	7.3389	0	0.0025	0.0025	0	0.0025	0.0025
$MR_{TriPlus1}$	MT-RT	0.6982	14.4296	13.7314	0.2327	4.8098	4.5771	0	0.0045	0.0045	0	0.0045	0.0045
	MT-ART-Max	0.4362	24.1957	23.7595	0.1454	8.0652	7.9198	0	0.0141	0.0141	0	0.0141	0.0141
	MT-ART-Avg	0.5252	20.0710	19.5458	0.1751	6.6903	6.5153	0	0.0085	0.0085	0	0.0085	0.0085
	MT-ART-Min	5.3961	13.8927	8.4966	1.7987	4.6309	2.8322	0	0.0027	0.0027	0	0.0027	0.0027
	SFIDMT-ART	5.5078	12.8340	7.3262	1.8359	4.2780	2.4421	0	0.0025	0.0025	0	0.0025	0.0025
$MR_{TriPlus2}$	MT-RT	0.6981	14.4124	13.7143	0.5871	14.1467	13.5596	0	0.0045	0.0045	0	0.0045	0.0045
	MT-ART-Max	0.5033	18.8736	18.3703	0.5033	18.8736	18.3703	0	0.0092	0.0092	0	0.0092	0.0092
	MT-ART-Avg	0.5144	19.3271	18.8127	0.5144	19.3271	18.8127	0	0.0084	0.0084	0	0.0084	0.0084
	MT-ART-Min	4.5598	13.5540	8.9942	4.5598	13.5540	8.9942	0	0.0031	0.0031	0	0.0031	0.0031
	SFIDMT-ART	5.5285	12.5366	7.0081	5.5142	12.7878	7.2736	0	0.0025	0.0025	0	0.0025	0.0025
MR_{rj}	MT-RT	2.3200	24.7000	22.3800	2.3200	24.7000	22.3800	0	0.0045	0.0045	0	0.0045	0.0045
	MT-ART-Max	1.7500	25.4000	23.6500	1.7500	25.4000	23.6500	0	0.0086	0.0086	0	0.0086	0.0086
	MT-ART-Avg	1.7700	25.5000	23.7300	1.7700	25.5000	23.7300	0	0.0083	0.0083	0	0.0083	0.0083
	MT-ART-Min	9.9000	23.6000	13.7000	9.9000	23.6000	13.7000	0	0.0038	0.0038	0	0.0038	0.0038
	SFIDMT-ART	11.3000	23.0000	11.7000	11.3000	23.0000	11.7000	0	0.0034	0.0034	0	0.0034	0.0034
MR_{Pnt}	MT-RT	3.7600	31.1000	27.3400	11.2800	93.3000	82.0200	0	0.0013	0.0013	0	0.0013	0.0013
	MT-ART-Max	3.4200	43.9000	40.4800	10.2600	131.700	121.440	0	0.0041	0.0041	0	0.0041	0.0041
	MT-ART-Avg	3.1500	49.6000	46.4500	9.4500	148.800	139.350	0	0.0104	0.0104	0	0.0104	0.0104
	MT-ART-Min	15.5000	29.9000	14.4000	46.5000	89.7000	43.2000	0	0.0015	0.0015	0	0.0015	0.0015
	SFIDMT-ART	15.4000	29.6000	14.2000	46.2000	88.8000	42.6000	0	0.0012	0.0012	0	0.0012	0.0012

- In MT-ART, the number of empty subdomains may have an impact on its test efficiency. Because MT-ART divides the entire input domain based on the number of all executed STCs and executed FTCs, an great number of subdomains may be created, particularly when the source input domain differs from the follow-up input domain. This may lead to the following consequences: (1) A great number of empty subdomains may prolong the search process excessively; and (2) the large number of empty subdomains available for generating source candidates in each testing round may not necessarily enhance test effectiveness. This is mainly because the objective of partitioning the input domain is to facilitate the selection of source candidates distant from executed test cases; however, an excess of subdomains may impede this process, particularly in the case of 1-*N* MRs or *M*-*N* MRs. In contrast, SFIDMT-ART divides the source input domain according to the number of relevant executed STCs, which can improve test efficiency while preserving test effectiveness.

- Based on the above experimental results and discussions on generation time, the following conclusion can be drawn:

  Answer to RQ2.2: SFIDMT-ART demonstrates superior time complexity and outperforms MT-ART in terms of test efficiency.

4.4.2.3 *Discrepancy and Dispersion*

Table 7 presents the experimental results of Dispersion and Discrepancy, from which the following observations can be drawn: SFIDMT-ART typically exhibited the best performance in both Dispersion and Discrepancy, followed by MT-ART-Min. SFIDMT-ART was the only algorithm consistently outperform MT-RT. MT-ART-Max typically demonstrated the poorest performance.

In light of these results, the following conclusions can be inferred:

Answer to RQ2.3: SFIDMT-ART is capable of achieving the most uniform distribution of STCs and FTCs across their respective input domains compared to MT-ART and MT-RT.

Answer to RQ2: Among SFIDMT-ART and the three MT-ART algorithms, SFIDMT-ART is capable of exhibiting better performance in terms of F-measure, Cohen's *d*, generation time, Discrepancy and Dispersion, making it the preferred choice for MG generation.

Answer to RQ1: According to the experimental results and discussions regarding F-measure, Cohen's *d*, generation time, Discrepancy and Dispersion, a conclusion may be drawn that, compared to achieving even distribution of both STCs and FTCs throughout the entire input domain, the performance of MT may be further improved by achieving even distribution of STCs and FTCs throughout their respective input domains.

4.5 CONCLUSION

While MT has repeatedly demonstrated its effectiveness as an SQA technique, there is still room to enhance both its effectiveness and efficiency [64, 246]. Recent studies [144, 291] have suggested that considering the quality of both STCs and FTCs could lead to

further enhancements in MT performance. Nevertheless, previous MG-generation studies treated FTCs in the same manner as STCs. This could make an MG-generation algorithm encounter the input-domain difference problem, severely affecting its effectiveness and efficiency. MT-ART [144], for example, is a recently-released MG-generation algorithm that seeks to enhance MT performance through evenly distributing both STCs and FTCs throughout the entire input domain. This algorithm is likely to encounter the input-domain difference problem, explained as follows: Different from traditional software testing, which operates within a single input domain, MT examines the relationships among STCs and FTCs, resulting in the presence of at least two input domains: The source input domain and the follow-up input domain. However, MT-ART overlooks the difference between the source input domain and the follow-up input domain, treating FTCs in the same manner as STCs. From a broader perspective, while the STCs and FTCs may appear to be evenly distributed throughout the entire input domain, a closer examination focusing solely on STCs or FTCs reveals different outcomes: STCs may cluster in one region, while FTCs cluster in another. Consequently, the performance of MT-ART could be severely impacted. This thesis referred to this issue as the input-domain difference problem.

This thesis have reported on a case study investigating how this problem may affect the performance of a specific MG-generation algorithm, MT-ART [144], using the concept of MRVR. A potential solution to address this problem has also been introduced: Treating the FTCs separately from the STCs during the MG-generation process, instead of treating them as a unified entity. Inspired by the proposed solution, to address the input-domain difference problem existing in MT-ART to enhance its performance further, a new principle for allocating STCs and FTCs has been presented as follows: Ensuring the even distribution of STCs throughout the respective source input domain and FTCs throughout the respective follow-up input domain. Additionally, this thesis have introduced two MT-distance measurements (ID-MDs) to facilitate the implementation of the proposed allocation principle and have introduced a new MG-generation algorithm named SFIDMT-ART to enhance the test efficiency and effectiveness of MT-ART. Empirical experiments have been conducted to evaluate and compare the performance of SFIDMT-ART with that of MT-RT and MT-ART using publicly-available systems from previously-published MT-related or ART-related studies [13, 49, 52, 62, 67, 100, 102, 136, 137, 139, 143, 144, 193, 238]. The experimental results revealed that, in comparison with MT-RT and MT-ART, SFIDMT-ART was capable of exhibiting significantly better performance in terms of fault-detection capability, while also reducing the computational overhead of MT-ART.

The main limitation of SFIDMT-ART is its high computational overhead. Therefore, addressing the issue of generation time cost in SFIDMT-ART is imperative. In this context, future work will include the exploration of approaches to minimize the computational overhead associated with SFIDMT-ART. Since the rationale of SFIDMT-ART is inspired by ART, one possible approach to address this limitation is to introduce some advanced ART algorithms, or the methods designed to improve the efficiency of ART algorithms, into SFIDMT-ART to improve its efficiency.

The following chapter of this thesis will introduce two novel MG-generation algorithms. The rationale behind the next chapter is that while the efficiency of SFIDMT-ART has improved compared to MT-ART, it remains excessively high compared to MT-RT. In light of this, the next chapter will introduce two novel algorithms that exhibit significantly better while maintaining high test effectiveness compared to SFIDMT-ART and MT-ART.

5

METAMORPHIC GROUP GENERATION ALGORITHMS FOR IMPROVING TEST EFFICIENCY AND EFFECTIVENESS

Publications delivered from this chapter

1. **Zhihao Ying**, Dave Towey, T. Y. Chen and Zhi Quan Zhou. MT-PART: Metamorphic-Testing-Based Adaptive Random Testing Through Partitioning. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC'24)*, 2024, IEEE, pp. 1184–1193.

5.1 INTRODUCTION AND MOTIVATION

Previously-published MG-generation algorithms may confront issues relating to test efficiency, explained as follows: Their main focus is directed towards enhancing the test effectiveness of MT (especially the fault-detection capabilities), albeit at the expense of test efficiency (particularly evident in ART-based algorithms) [19, 21, 64, 143, 144, 246]. At this point, the principal objective of this chapter is to design and develop MG-generation algorithms that are capable of achieving a balance between the effectiveness and efficiency of MT.

This chapter introduces a series of novel class of MG-generation algorithms rooted in partition-based ART, referred to as MT-based ART Through Partitioning (MT-PART): These algorithms aim to improve the effectiveness and efficiency of MT by dynamically partitioning the source and follow-up input domains, and generating new STCs and FTCs uniformly distributed throughout their respective input domains. MT-PART consists of two concrete algorithms: MT-based ART by Bisection (MT-BART) and MT-based ART through Iterative Partitioning (MT-IPART). Then, this chapter conducts empirical experiments aimed at assessing and comparing the performance of MT-PART algorithms against other existing MG-generation algorithms in the literature. In particular, this chapter selects SFIDMT-ART (proposed in the last chapter of this thesis), MT-ART (SFIDMT-ART was proposed as its enhanced version) and MT-RT (the most popular MG-generation algorithm) for the empirical experiments. The experimental results revealed that MT-PART algorithms were capable of achieving a marked enhancement in the efficiency of MT, while maintaining a high level of effectiveness.

Table 8: Experimental Results of Generation Time [57]

ART Algorithms	FSCS-ART	RPART	BART	IPART
Time (in seconds)	3645.8	1141.1	6.4	12.0

Table 9: Experimental Results of F-ratio on 2D input domains (under the block failure region) [57]

Failure Rates	F-ratio of FSCS-ART	F-ratio of RPART	F-ratio of BART	F-ratio of IPART
0.01	67%	76%	75%	63%
0.00	66%	77%	74%	61%
0.002	65%	79%	74%	60%
0.001	65%	80%	75%	61%

5.2 MT-PART ALGORITHMS

5.2.1 *Selection of Basic Algorithms*

The primary computational cost associated with FSCS-ART [83] stems from the computation and comparison of distances between test cases. If the size of the candidate set is denoted as c ($c > 1$), the test case to be chosen is denoted as i ($i > 0$), and the number of previously-selected test cases (from the input domain) is denoted as n ($n > 0$), then the number of distance calculations for selecting the i^{th} test case is given by $c * (i - 1)$.

The time complexity of FSCS-ART is $O(n^2)$ [57, 77], a value that can be computed as follows:

$$\sum_{i=1}^{n} e * (i-1) = e * \sum_{i=1}^{n-1} i \in O(n^2) \tag{9}$$

Both RPART [78] and BART [78] exhibit a time complexity of $O(n)$ [53, 57]. Let k ($k > 0$) represent the input dimensionality, the time complexity of IPART [57] can then be measured by:

$$O(3^{k+1}, n^{1+1/k}) \tag{10}$$

In IPART, searching for the empty subdomains that are not surrounded by any non-empty subdomains constitute the main computational overhead.

Chen et al. [57] conducted empirical experiments to measure the actual time required for generating test cases using the ART algorithms, as some certain factors that could potentially impact the real-world experiments may be overlooked in the calculation of time complexity. The experiments were conducted under the following conditions: (a) On an HP Compaq PC equipped with a 2.6 GHz Intel Pentium IV processor and 256M RAM; and (b) using the Microsoft Windows XP SP1 operating system. Table 8 lists the time (in seconds) to run the ART algorithms for 5000 trials.

The experimental results revealed that the generation times of BART and IPART were minimal in comparison to FSCS-ART and RPART. However, while the generation time of RPART is notably lower than that of FSCS-ART, it remains considerably larger than the

other two partition-based algorithms. This may be due to the computational overhead involved in identifying the largest subdomain, despite the avoidance of distance calculations and comparisons.

Additionally, Chen et al. [57] conducted experiments to evaluate and compare the fault-detection capabilities of the four ART algorithms with RT, with the experimental results summarized in Table 9. The experiments included four failure rates (0.01, 0.005, 0.002, and 0.001), with all four ART algorithms demonstrating significantly better fault-detection capabilities compared to RT. Among the ART algorithms, IPART exhibited the highest fault-detection capability overall, followed by FSCS-ART. Although BART exhibited inferior performance compared to FSCS-ART, it was capable of significantly outperforming RPART.

In summary, the inclusion of BART and IPART in the MG-generation algorithms was guided by the following considerations:

1. In terms of fault-detection capability, as indicated by the F-ratio experimental results, IPART generally necessitates the fewest number of test cases among the five algorithms to identify the first failure, followed closely by FSCS-ART. While the fault-detection capability of BART falls below that of IPART and FSCS-ART, it is still capable of significantly outperforming both RPART and RT.

2. In regard to computational efficiency, the generation time experimental results revealed that BART and IPART incur acceptable computational overhead. Their computational overheads are similar to or slightly higher than RT, and meanwhile significantly lower than those of FSCS-ART and RPART.

5.2.2 *MT-based ART by Bisection (MT-BART)*

The steps of MT-BART are summarized in Algorithm 3, with a detailed explanation of each step provided below:

- Step 1: MT-BART assumes the existence of an MR, which can be identified from scratch using specific approaches (such as MRPs) or directly chosen from previously-published MRs.

- Step 2: MT-BART determines the scope of the source input domain and follow-up input domain according to the given MR.

- Step 3: MT-BART initializes the executed STC set and the executed FTC set as empty. The rationale behind this step is that MT-BART aims to enhance MT performance by evenly distributing STCs and FTCs across their respective input domains, thereby two sets of executed test sets are required.

- Steps 5-7: MT-BART checks:

  (1) In the source input domain, if all subdomains are non-empty (covered by the test cases in the executed STC set), MT-BART then bisects these subdomains by drawing horizontal or vertical lines.

Algorithm 3: MT-BART Algorithm for 2D input domains

```
1  Assume the existence of an MR;
2  Determine the scope of the source and follow-up input domains according to the
     given MR;
3  Initialize the executed STC and FTC sets to be empty;
4  while all the stopping conditions are not satisfied do
5      if all subdomains are non-empty or the number of executed STCs/FTCs is the same as
         the number of relevant subdomains then
6          Bisect all the corresponding subdomains using horizontal or vertical lines;
7          Identify the empty subdomains in the source/follow-up input domain as
             source/follow-up candidate subdomains;
8      end
9      Randomly choose one source candidate subdomain and select k source
         candidates;
10     for m = 1 → k do
11         Choose the m^th source candidate and construct follow-up candidates based
             on the given MR;
12         Explore if the follow-up candidates are within follow-up candidate
             subdomains, represented by f_m (If yes, f_m = 1; otherwise, f_m = 0);
13     end
14     Choose the candidate MG with the largest f_m;
15     if more than one MG satisfies the criterion then
16         Choose one at random;
17     end
18     Execute the MG against the SUT and check whether the given MR is violated;
19     Add the executed STC and FTC to their respective executed test sets;
20 end
```

(2) In the follow-up input domain, if all subdomains are non-empty (covered by the test cases in the executed FTC set), or the number of executed FTCs is the same as the number of subdomains, then MT-BART bisects all the subdomains.

The rationale behind these three steps is to generate empty subdomains for future use.

- Step 8: After partitioning the subdomains, MT-BART chooses the empty subdomains in the source input domain as source candidate subdomains, and chooses the empty subdomains in the follow-up input domain as follow-up candidate subdomains. Source candidate subdomains are created for generating STCs, while follow-up candidate subdomains serve the purpose of validating and comparing the generated STCs and FTCs to determine the most suitable MG for execution.

- Step 9: MT-BART chooses one source candidate subdomain at random and selects k STCs from it as source candidates.

- Steps 10-13: MT-BART constructs k FTCs as follow-up candidates according to the given MR and the k source candidates, with the aim of creating k candidate MGs. Subsequently, MT-BART validates and compares the k candidate MGs to get an appropri-

ate one for execution by exploring if the follow-up candidates are within follow-up candidate subdomains.

- Step 14: MT-BART chooses a candidate MG for execution, with all its follow-up candidates situated within follow-up candidate subdomains. It is worth noting that, to achieve the objective within reasonable computational overheads, a finite number of (k) candidate MGs is considered instead of an infinite set. In particular, if none of the k candidate MGs satisfy the predefined criteria, MT-PART ceases the search process and randomly chooses one from the pool of k candidate MGs for execution.
- Steps 15-17: If there are more than one candidate MG containing the same maximum value, then MT-BART chooses one of them for execution at random.
- Step 18: MT-BART executes the generated MG against the SUT and checks for MR violations.
- Step 19: For the currently executed MG, MT-BART appends its STC to the executed STC set, and appends its FTC to the executed FTC set.
- Step 20: If all the stopping conditions are not satisfied (i.e., no MR violations detected), then MT-BART skips to Step 5; otherwise, MT-BART ceases the process and reports the testing outcomes.

5.2.3 *MT-based ART through Iterative Partitioning (MT-IPART)*

The steps of MT-IPART are summarized in Algorithm 4, with a detailed explanation of each step provided as follows:

- Step 1: MT-IPART assumes the existence of an MR.
- Step 2: MT-IPART determines the scope of the source input domain and follow-up input domain according to the given MR.
- Step 3: MT-IPART initializes the executed STC set and the executed FTC set as empty.
- Step 4: MT-IPART divides the source/follow-up input domain based on the value of the relevant partitioning scheme: The source partitioning scheme (p_s) for the source input domain; and the follow-up partitioning scheme (p_f) for the follow-up input domain. In this step, MT-IPART initialize the values of both schemes to 1 ($p_s = 1$ and $p_f = 1$).
- Steps 6-10: MT-IPART checks:

  (1) Within the source input domain, if all subdomains are either non-empty or surrounded by a non-empty subdomain, then MT-IPART increments the source partitioning scheme by 1 ($p_s = p_s + 1$) and divides the source input domain into $p_s \times p_s$ subdomains.

Algorithm 4: MT-IPART Algorithm for 2D input domains

```
1  Assume the existence of an MR;
2  Determine the scope of the source and follow-up input domains according to the
     given MR;
3  Initialize the executed STC and FTC sets to be empty;
4  Set the source and follow-up partitioning schemes to 1 (p_s = 1 and p_f = 1);
5  while all the stopping conditions are not satisfied do
6      if all subdomains are non-empty or are surrounded by a non-empty subdomain; or the
         number of executed STCs/FTCs is the same as the number of relevant subdomains then
7          Increase the corresponding partitioning scheme by 1 (p = p + 1);
8          Partition the corresponding input domain into p × p subdomains;
9          Map all executed STCs/FTCs to the respective input domain to construct
             empty subdomains;
10         Identify the empty subdomains that are also surrounded by empty
             subdomains as source/follow-up candidate subdomains;
11     end
12     Randomly choose one source candidate subdomain and select k source
         candidates;
13     for m = 1 → k do
14         Choose the m^th source candidate and construct follow-up candidates based
             on the given MR;
15         Explore if the follow-up candidates are within follow-up candidate
             subdomains, represented by f_m (If yes, f_m = 1; otherwise, f_m = 0);
16     end
17     Choose the candidate MG with the largest f_m;
18     if more than one MG satisfies the criterion then
19         Choose one at random;
20     end
21     Execute the MG against the SUT and check whether the given MR is violated;
22     Add the executed STC and FTC to their respective executed test sets;
23 end
```

(2) Within the follow-up input domain, if all subdomains are either non-empty or surrounded by a non-empty subdomain, or the number of test cases in the executed FTC set is equal to the number of subdomains in the follow-up input domain, then MT-IPART increments the follow-up partitioning scheme by 1 ($p_f = p_f + 1$) and divides the follow-up input domain into $p_f \times p_f$ subdomains.

- Step 11: After partitioning the subdomains, MT-IPART subsequently identifies the empty subdomains (from the source/follow-up input domain) that are not surrounded by non-empty subdomains as (source/follow-up) candidate subdomains. In addition, due to the fact that the subdomains located at the edges have fewer adjacent subdomains, the probability of these subdomains being selected as source/follow-up candidate subdomains may be higher than the subdomains located at the center. With this consideration, given a subdomain located at the edges, MT-IPART not only checks if its adjacent subdomains are empty, but also checks if the corresponding subdomain located at the edge of the other side of the entire input domain are empty.

- Step 12: MT-IPART chooses one source candidate subdomain at random and chooses k STCs from it as source candidates.
- Steps 13-16: MT-IPART constructs k FTCs as follow-up candidates according to the given MR and the k source candidates, with the aim of creating k candidate MGs. Subsequently, MT-IPART validates and compares the k candidate MGs to get an appropriate one for execution by exploring if the follow-up candidates are within follow-up candidate subdomains.
- Step 17: MT-IPART chooses a candidate MG for execution, with all its follow-up candidates situated within follow-up candidate subdomains.
- Steps 18-20: If there are more than one candidate MG containing the same maximum value, then MT-IPART chooses one of them for execution at random.
- Step 18: MT-IPART executes the generated MG against the SUT and checks for MR violations.
- Step 19: For the currently executed MG, MT-IPART appends its STC to the executed STC set, and appends its FTC to the executed FTC set.
- Step 20: If all the stopping conditions are not satisfied (i.e., no MR violations detected), then MT-IPART skips to Step 5; otherwise, MT-IPART ceases the process and reports the testing outcomes.

5.2.4 *Comparison between MT-BART and MT-IPART*

Suppose k denotes the number of candidate MGs generated in each iteration, and n represents the number of MGs to be generated, the time complexity of MT-BART and MT-IPART is $O(n \times k)$. According to the experimental results and analysis shown in Section 5.4.2, the value of k typically can be set to 3 or 4. Therefore, the complexity is $O(3n)$ or $O(4n)$, both of which are $O(n)$. In this context, a conclusion can be drawn that the time complexity of MT-BART and MT-IPART is much lower than that of SFIDMT-ART and MT-ART.

The major differences between MT-BART and MT-IPART are summarized as follows:

- The major difference lies in the partitioning method. For instance, in the case of a 2D input domain, MT-BART divides it by drawing horizontal or vertical lines, while MT-IPART divides it into $n * n$ subdomains.
- The second difference concerns the conditions for partitioning the input domains. For instance, MT-BART partitions the source input domain when all subdomains are covered by executed STCs, while MT-IPART partitions the source input domain under the condition that all subdomains are either covered by executed STCs or surrounded by a non-empty subdomain.
- The final difference concerns the way of defining source/follow-up candidate subdomains. In particular, MT-BART identifies subdomains not covered by executed

Table 10: Information of the SUTs and MRs

SUTs	Sin	BesselJ	TriSquare	TriSquarePlus
Input Dimensions	1	2	3	3
Input Domain Ranges	(0,1000)	((1,1), (100,100))	((0,0,0), (100,100,100))	((0,0,0), (100,100,100))
Sizes (LOC)	120	1211	38	31
MRs	MR_{Sin13}	$MR_{BesselJ}$	$MR_{TriSquare3}$, $MR_{TriSquare6}$	$MR_{TriPlus2}$, $MR_{TriPlus12}$
Number of MRs	1	1	2	2
Dimension of MRs	1	3	2	2
Number of Mutants	2	3	6	6
Fault Types	CRP, ROR, AOR	CRP, ROR, AOR	CRP, ROR, RSR, AOR	CRP, ROR, RSR, AOR

STCs/FTCs as source/follow-up candidate subdomains, while MT-IPART identifies subdomains that are not only empty but also not surrounded by non-empty subdomains as source/follow-up candidate subdomains.

5.3 RESEARCH QUESTIONS

The following RQs were formulated to provide guidance for the empirical experiments:
RQ1: Can MT-BART and MT-IPART achieve a balance between test effectiveness and efficiency?

- Objective: Evaluate whether or not MT-BART and MT-IPART can outperform SFIDMT-ART, MT-ART and MT-RT and achieve a better balance between test effectiveness and efficiency.
- Motivation: MT-RT is the most popular MG-generation algorithm, but it sometimes cannot achieve satisfactory fault-detection capability as it does not utilize any features of the SUT [64, 246]. MT-ART and SFIDMT-ART were introduced to improve the fault-detection capability of RT; however, the improvement of their fault-detection capabilities are achieved at the cost of sacrificing efficiency. In this context, MT-BART and MT-IPART were proposed, in order to achieve a balance between test effectiveness and efficiency.
- Baseline algorithms:
  1. MT-RT, selected for its widespread adoption — it is the most popular MG-generation algorithm.
  2. MT-ART, selected for its endeavor to bolster MT performance by uniformly dispersing STCs and FTCs across the complete input spectrum
  3. SFIDMT-ART, selected as it is another MG-generation algorithm proposed in this thesis, and it is proposed as an enhanced iteration of MT-ART.
- Methodologies:
  1. Select three performance criteria: Test efficiency, test effectiveness, and MG diversity.

2. Design sub-RQs to delve deeper into specific aspects of performance.
3. Determine whether or not MT-BART and MT-IPART perform better across multiple performance metrics compared to SFIDMT-ART, MT-ART and MT-RT.

- Sub-RQs:
  1. Can MT-BART and MT-IPART achieve better test effectiveness (F-measure) than SFIDMT-ART, MT-ART, and MT-RT?
  2. Can MT-BART and MT-IPART achieve better test efficiency (generation time) than SFIDMT-ART and MT-ART?
  3. Can MT-BART and MT-IPART achieve better distribution diversity of MGs (Dispersion and Discrepancy) than SFIDMT-ART, MT-ART, and MT-RT?

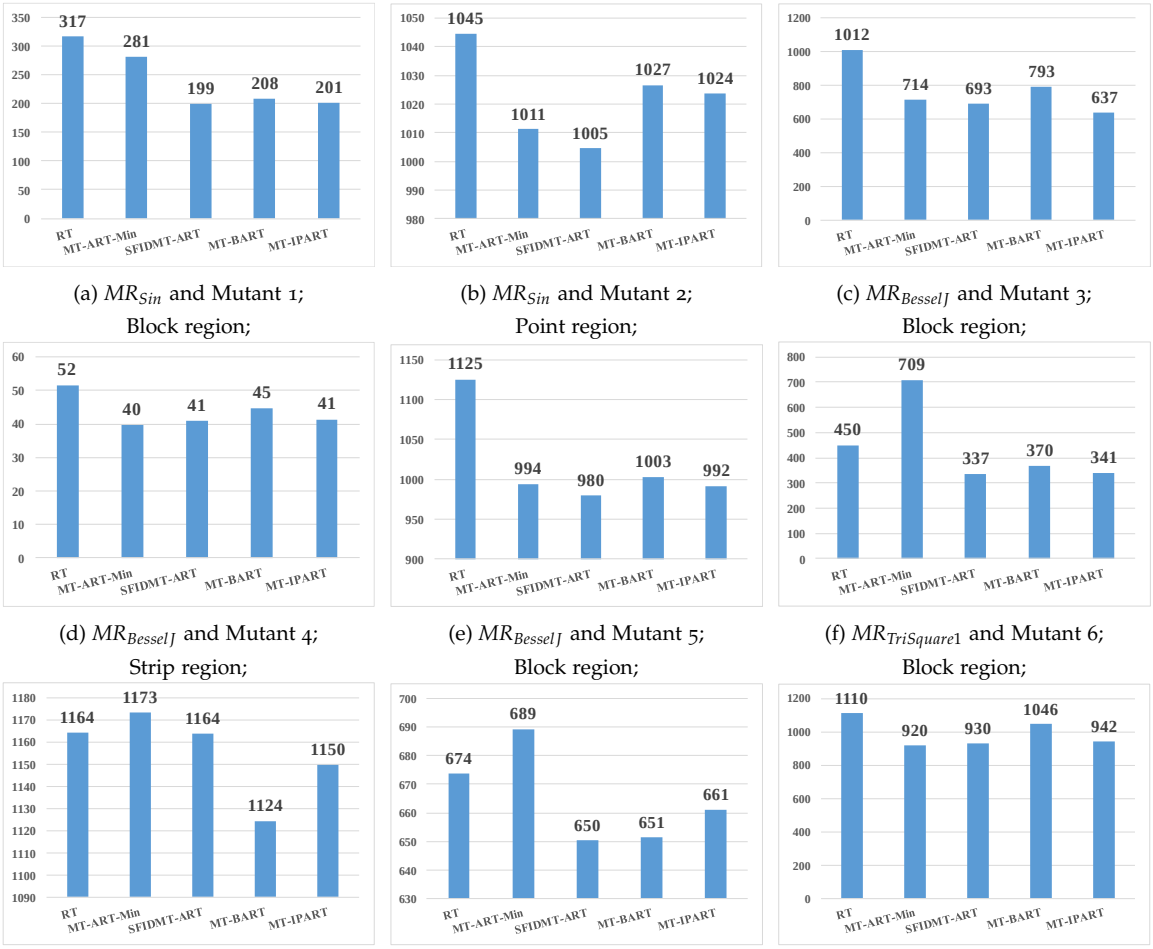
5.4 EMPIRICAL EXPERIMENTS

5.4.1 *Experimental Setup*

In the empirical experiments, systems of various sizes and dimensions were chosen, and artificial faults were introduced to generate mutants. Table 10 provides an overview of the SUTs and MRs, including input dimensions, the scope of the entire input domain, the scope of the source input domains, the scope of the follow-up input domain, the size of the SUTs (measured in Lines Of Codes (LOC)), MRs, number of mutants, and types of faults. A detailed introduction of the SUTs is provided in Section 2.4. Six MRs, all sourced from previously-published MT-related studies [49, 52, 67, 100, 143, 144], were chosen for the SUTs. The descriptions of these MRs are presented in Appendix A.

Since according the the experimental results and conclusions from Section 4.4.2, MT-ART-Min (MT-ART with Strategy 1/Min) is the most stable one among the three MT-ART algorithms (MT-ART-Max, MT-ART-Avg, and MT-ART-Min), to simplify the experimental procedure, only MT-ART-Min was included in the empirical experiments. The following experimental environment was built, mirroring the experiments conducted in Section 4.4:

- The k value for MT-ART-Min and SFIDMT-ART was set to 10;
- 1000 MGs were generated for each MG-generation algorithm and its corresponding MR, facilitating the computation of Discrepancy and Dispersion.
- All experiments were repeated 10,000 times to compute the average F-measure, Discrepancy, and Dispersion.
- The number of subdomains for Discrepancy was set to 1000, aiming to balance computation overheads and accuracy.



(a) MR_{Sin} and Mutant 1; Block region;

(b) MR_{Sin} and Mutant 2; Point region;

(c) $MR_{BesselJ}$ and Mutant 3; Block region;

(d) $MR_{BesselJ}$ and Mutant 4; Strip region;

(e) $MR_{BesselJ}$ and Mutant 5; Block region;

(f) $MR_{TriSquare1}$ and Mutant 6; Block region;

(g) $MR_{TriSquare1}$ and Mutant 7; Block region;

(h) $MR_{TriSquare1}$ and Mutant 8; Point region;

(i) $MR_{TriSquare2}$ and Mutant 9; Strip region;

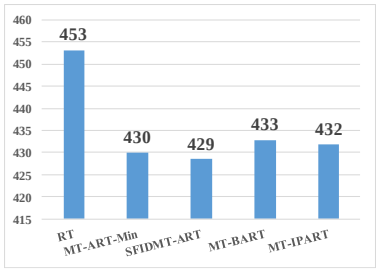
Figure 14: F-measure Experimental Results of the MG-generation Algorithms (Part 1)

5.4.2 *Experimental Results, Discussions, and Conclusions*

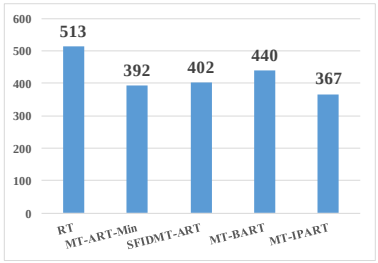
5.4.2.1 *F-measure and Cohen's* d

Figs. 14 - 11 illustrates the F-measure and Cohen's *d* experimental results of the MG-generation algorithms under test, while Table 11 presents the Cohen's *d* experimental results. The following observations can be drawn from the experimental results:

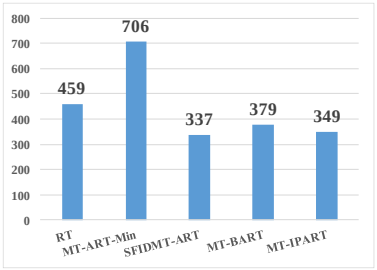
- In block MRVRs: All four algorithms (MT-BART, MT-IPART, SFIDMT-ART, and MT-ART-Min) consistently outperformed MT-RT. Specifically, MT-IPART and SFIDMT-ART exhibited the highest performance, followed by MT-BART and MT-ART-Min. MT-BART generally exhibited performance comparable to MT-ART-Min. For instance, when using mutants 1, 6, and 12, MT-BART significantly outperformed MT-ART-Min; conversely, when using mutants 3, 11, and 15, MT-ART-Min exhibited superior performance. In most cases, they demonstrated similar performance, such as mutants 5, 7, 8, 10, 13, 14, 15, 16 and 17.
- In point/strip MRVRs: The performance of all five algorithms tends to be similar.



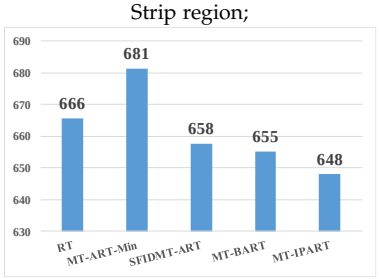
(j) $MR_{TriSquare2}$ and Mutant 10; Strip region;



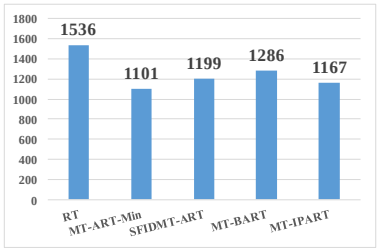
(k) $MR_{TriSquare2}$ and Mutant 11; Block region;



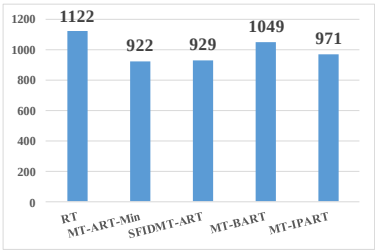
(l) $MR_{TriPlis1}$ and Mutant 12; Block region;



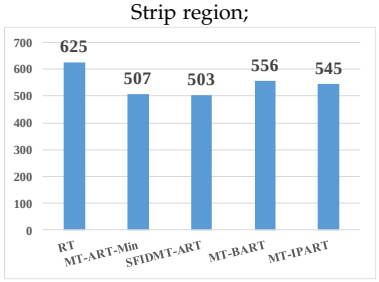
(d) $MR_{TriPlis1}$ and Mutant 13; Strip region;



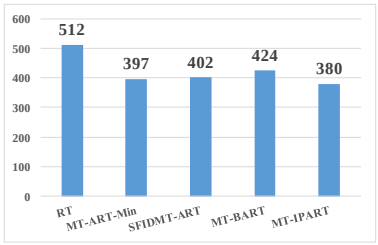
(e) $MR_{TriPlis1}$ and Mutant 14; Block region;



(f) $MR_{TriPlis2}$ and Mutant 15; Block region;



(g) $MR_{TriPlis2}$ and Mutant 16; Strip region;



(h) $MR_{TriPlis2}$ and Mutant 17; Block region;

Figure 15: F-measure Experimental Results of the MG-generation Algorithms (Part 2)

According to the F-measure and Cohen's *d* experimental results and subsequent discussion, a conclusion can be inferred as follows:

Answer to RQ1.1: MT-IPART and SFIDMT-ART typically exhibit the highest test effectiveness among the five algorithms under test, followed by MT-BART and MT-ART.

5.4.2.2 *Generation Time*

Table 12 presents the mean generation time experimental results. MT-BART and MT-IPART consistently required significantly less time to generate 10,000 MGs compared to MT-ART-Min and SFIDMT-ART, which indicate their superior test efficiency. Notably, MT-BART exhibited better test efficiency than MT-IPART.

The underlying reason that MT-BART and MT-IPART outperformed SFIDMT-ART and MT-ART is that, although MT-BART and MT-IPART still require the generation of extra source and follow-up candidates, they do avoid the distance computations and comparisons, which are the major computational overheads in SFIDMT-ART and MT-ART. In particular, MT-BART and MT-IPART only need to check whether or not the follow-up candidates are within empty subdomains, while SFIDMT-ART and MT-ART compute and compare the distance between source candidates and executed STCs and the distance between follow-up candidates and executed FTCs. The computational cost of SFIDMT-ART

Table 11: Effect Size (Cohen's *d*) Experimental Results of the MG-generation algorithms

SUTs	MRs	Mutants	MT-BART vs RT	MT-BART vs MT-ART-Min	MT-BART vs SFIDMT-ART	MT-IPART vs RT	MT-IPART vs MT-ART-Min	MT-IPART vs SFIDMT-ART	MT-BART vs MT-IPART
Sin	MR_{Sin}	Mutant 1	0.38	0.33	-0.13	0.46	0.43	-0.04	-0.10
		Mutant 2	0.01	-0.03	-0.03	0.01	-0.02	-0.03	0.00
BesselJ	$MR_{BesselJ}$	Mutant 3	0.25	-0.13	-0.16	0.46	0.14	0.11	-0.27
		Mutant 4	0.15	-0.12	-0.09	0.22	-0.04	-0.01	-0.08
		Mutant 5	0.12	-0.01	-0.04	0.14	0.01	-0.02	-0.01
TriSquare	$MR_{TriSquare1}$	Mutant 6	0.20	0.64	-0.10	0.28	0.71	-0.01	-0.09
		Mutant 7	0.04	0.04	0.04	0.01	0.02	0.01	0.02
		Mutant 8	0.03	0.06	0.00	0.02	0.04	-0.02	0.02
	$MR_{TriSquare2}$	Mutant 9	0.06	-0.15	-0.13	0.17	-0.04	-0.02	-0.11
		Mutant 10	0.05	-0.01	-0.01	0.05	0.00	-0.01	0.00
		Mutant 11	0.16	-0.13	-0.10	0.35	0.08	0.11	-0.20
TriSquarePlus	$MR_{TriPlus1}$	Mutant 12	0.20	0.62	-0.13	0.28	0.69	-0.04	-0.09
		Mutant 13	0.02	0.04	0.00	0.03	0.05	0.02	-0.01
		Mutant 14	0.18	-0.16	-0.07	0.29	-0.07	0.03	-0.10
	$MR_{TriPlus2}$	Mutant 15	0.07	-0.15	-0.14	0.15	-0.07	-0.06	-0.09
		Mutant 16	0.12	-0.10	-0.11	0.14	-0.08	-0.09	-0.02
		Mutant 17	0.20	-0.08	-0.06	0.32	0.05	0.07	-0.13

Table 12: Generation Time Experimental Results of the MG-generation algorithms

SUTs	MRs	MT-ART-Min	SFIDMT-ART	MT-BART	MT-IPART
Sin	MR_{Sin}	16.873	8.172	0.01	1.38
BesselJ	$MR_{BesselJ}$	218.764	28.226	0.078	2.864
TriSquare	$MR_{TriSquare1}$	110.627	53.743	0.252	2.633
	$MR_{TriSquare2}$	135.291	67.676	0.254	2.589
TriSquarePlus	$MR_{TriPlus1}$	107.551	53.891	0.256	2.64
	$MR_{TriPlus2}$	135.581	67.872	0.255	2.588

and MT-ART becomes particularly high when using M-N MRs or when the number of executed STCs and FTCs increases. Therefore, MT-BART and MT-IPART typically have much lower computational overhead while maintaining a high fault-detection capability.

Thus, based on the mean generation time experimental results and discussions, a conclusion can be inferred as follows:

Answer to RQ1.2: In addition to MT-RT, MT-BART typically demonstrates the highest test efficiency, followed by MT-IPART.

5.4.2.3 *Dispersion and Discrepancy*

Table 13 summarizes the Dispersion and Discrepancy experimental results. It is evident that all four ART-based algorithms achieved a uniform distribution of STCs and FTCs throughout their respective input domains compared to MT-RT. Among these algorithms, MT-IPART and SFIDMT-ART generally demonstrated superior performance, followed by MT-BART and MT-ART-Min. Consistent with the F-measure and Cohen's *d* experimental results, MT-ART and MT-BART exhibited similar performance regarding Dispersion and Discrepancy.

Therefore, according to the Dispersion and Discrepancy experimental results and discussions, a conclusion can be inferred as follows:

Answer to RQ1.3: MT-IPART and SFIDMT-ART are capable of achieving the most even distribution of STCs and FTCs throughout the respective input domains, followed by MT-BART and MT-ART-Min.

Table 13: Discrepancy and Dispersion Experimental Results of the MG-generation algorithms

MRs	Methods	Discrepancy						Dispersion					
		STCs in the Source Input Domain			FTCs in the Follow-up Input Domain			STCs in the Source Input Domain			FTCs in the Follow-up Input Domain		
		Min	Max	Max-Min	Min	Max	Max-Min	Min	Max	Max-Min	Min	Max	Max-Min
MR_{Sin}	MT-RT	0	0.0081	0.0081	0	0.0081	0.0081	0.0009	3.7863	3.7854	0.0003	1.2621	1.2618
	MT-ART-Min	0	0.0020	0.0020	0	0.0020	0.0020	0.2685	4.6971	4.4289	0.0895	1.5657	1.4763
	SFIDMT-ART	0	0.0010	0.0010	0	0.0010	0.0010	0.4197	1.6953	1.2753	0.1399	0.5651	0.4251
	MT-BART	0	0.0020	0.0020	0	0.0020	0.0020	0.0345	1.9310	1.8965	0.0115	0.6437	0.6322
	MT-IPART	0	0.0010	0.0010	0	0.0010	0.0010	0.5068	1.4940	0.9872	0.1689	0.4980	0.3291
$MR_{BesselJ}$	MT-RT	0	0.0046	0.0046	0	0.0046	0.0046	0.0692	5.2861	5.2169	0.0692	5.2861	5.2169
	MT-ART-Min	0	0.0029	0.0029	0	0.0029	0.0029	0.8791	4.2297	3.3506	0.8791	4.2297	3.3506
	SFIDMT-ART	0	0.0021	0.0021	0	0.0021	0.0021	1.4170	4.1943	2.7772	1.4170	4.1943	2.7772
	MT-BART	0	0.0027	0.0027	0	0.0027	0.0027	0.2593	4.3349	4.0756	0.2593	4.3349	4.0756
	MT-IPART	0	0.0020	0.0020	0	0.0020	0.0020	1.5502	4.2302	2.6799	1.5502	4.2302	2.6799
$MR_{TriSquare1}$	MT-RT	0	0.0045	0.0045	0	0.0045	0.0045	0.2345	4.8184	4.5839	0.7035	14.4551	13.7517
	MT-ART-Min	0	0.0027	0.0027	0	0.0027	0.0027	1.7430	7.6376	5.8946	5.2289	22.9128	17.6839
	SFIDMT-ART	0	0.0025	0.0025	0	0.0025	0.0025	1.8334	4.2797	2.4463	5.5002	12.8391	7.3389
	MT-BART	0	0.0033	0.0033	0	0.0033	0.0033	0.4393	4.3873	3.9480	1.3178	13.1618	11.8440
	MT-IPART	0	0.0025	0.0025	0	0.0025	0.0025	1.6506	4.3014	2.6508	4.9519	12.9042	7.9523
$MR_{TriSquare2}$	MT-RT	0	0.0045	0.0045	0	0.0045	0.0045	0.6981	14.4689	13.7708	0.6981	14.4689	13.7708
	MT-ART-Min	0	0.0031	0.0031	0	0.0031	0.0031	4.5597	13.5676	9.0079	4.5597	13.5676	9.0079
	SFIDMT-ART	0	0.0025	0.0025	0	0.0025	0.0025	5.5040	12.8224	7.3183	5.5040	12.8224	7.3183
	MT-BART	0	0.0033	0.0033	0	0.0033	0.0033	1.3235	13.2100	11.8864	1.3235	13.2100	11.8864
	MT-IPART	0	0.0025	0.0025	0	0.0025	0.0025	4.9520	12.9119	7.9598	4.9520	12.9119	7.9598
$MR_{TriPlus1}$	MT-RT	0	0.0045	0.0045	0	0.0045	0.0045	0.6982	14.4296	13.7314	0.2327	4.8099	4.5771
	MT-ART-Min	0	0.0027	0.0027	0	0.0027	0.0027	5.1816	23.0830	17.9014	1.7272	7.6943	5.9671
	SFIDMT-ART	0	0.0025	0.0025	0	0.0025	0.0025	5.5074	12.8353	7.3279	1.8358	4.2784	2.4426
	MT-BART	0	0.0033	0.0033	0	0.0033	0.0033	1.3295	13.1807	11.8512	0.4432	4.3936	3.9504
	MT-IPART	0	0.0025	0.0025	0	0.0025	0.0025	4.9535	12.9046	7.9511	1.6512	4.3015	2.6504
$MR_{TriPlus2}$	MT-RT	0	0.0045	0.0045	0	0.0045	0.0045	0.6981	14.4124	13.7143	0.5871	14.1467	13.5596
	MT-ART-Min	0	0.0031	0.0031	0	0.0031	0.0031	4.5623	13.6858	9.1235	4.4997	13.4354	8.9357
	SFIDMT-ART	0	0.0025	0.0025	0	0.0025	0.0025	5.5288	12.5385	7.0097	5.5140	12.7897	7.2757
	MT-BART	0	0.0033	0.0033	0	0.0033	0.0033	1.3250	13.1809	11.8559	1.3226	13.1891	11.8665
	MT-IPART	0	0.0025	0.0025	0	0.0025	0.0025	4.9390	12.5679	7.6289	4.8874	12.7346	7.8472

According to the experimental results and discussions, it can be concluded that among the five MG-generation algorithms under test, MT-BART and MT-IPART offer a better balance between efficiency and effectiveness, and thus, they are preferable choices for MG generation. In addition, between these two algorithms, MT-BART generally outperforms in efficiency (generation time), while MT-IPART excels in effectiveness (fault-detection capability).

5.5 FUTURE WORK

5.5.1 *MT-based ART by Random Partitioning (MT-RPART)*

In addition to BART and IPART, there are also many other well-known partition-based ART algorithms, including RPART [78].

In Section 5.2.1, aiming to simplify the experimental process and allocate time for other projects, analysis and discussion have been provided to explain why RPART was not selected. Nevertheless, certain features of RPART might have been overlooked during the theoretical analysis. More specifically, although RPART typically exhibits lower efficiency and effectiveness compared to BART and IPART [57], the capability of RPART for design-

Algorithm 5: MT-RPART Algorithm for 2D input domains

```
Assume the existence of an MR;
Determine the scope of the source and follow-up input domains according to the
  given MR;
Initialize the executed STC and FTC sets to be empty;
while all the stopping conditions are not satisfied do
    Identify the empty subdomains in source/follow-up input domain as
      source/follow-up candidate subdomains;
    Choose the largest source candidate subdomain and generate k source
      candidates;
    for m = 1 → k do
        Choose the m^th source candidate and construct follow-up candidates based
          on the given MR;
        Explore if the follow-up candidates are inside the largest follow-up candidate
          subdomain, represented by f_m (If yes, f_m = 1; otherwise, f_m = 0);
    end
    Choose the candidate MG with the largest f_m;
    if more than one MG satisfies the criterion then
        Choose one at random;
    end
    Execute the MG against the SUT and check whether the given MR is violated;
    Partition the two relevant subdomains into four subdomains by drawing
      horizontal and vertical lines according to the currently executed test cases;
    Add the executed STC and FTC to their respective executed test sets;
end
```

ing MG-generation algorithms remains unverified. With this consideration, this section proposes a novel MG-generation algorithm named MT-based ART by Random Partitioning (MT-RPART), based on RPART. Future work include evaluating and comparing MT-RPART with MT-RT, MT-BART, and MT-IPART through empirical experiments. The steps of MT-RPART are outlined in Algorithm 5, with a detailed explanation of each step provided below:

- Step 1: MT-RPART assumes the existence of an MR.
- Step 2: MT-RPART determines the scope of the source input domain and follow-up input domain according to the given MR.
- Step 3: MT-RPART initializes the executed STC set and the executed FTC set as empty.
- Step 5: MT-RPART checks:

  (1) Within the source input domain, MT-RPART identifies the empty subdomains as source candidate subdomains.

  (2) Within the follow-up input domain, MT-RPART identifies the empty subdomains as follow-up candidate subdomains.
- Step 6: MT-RPART chooses the largest subdomain and chooses k STCs from it at random as source candidates.

- Steps 7-10: MT-RPART constructs k FTCs as follow-up candidates according to the given MR and the k source candidates, with the aim of creating k candidate MGs. Subsequently, MT-RPART validates and compares the k candidate MGs to get an appropriate one for execution by exploring if the follow-up candidates are within the largest follow-up candidate subdomain.
- Step 11: MT-RPART chooses a candidate MG for execution, with its follow-up candidates situated within the largest follow-up candidate subdomain.
- Steps 12-14: If there are more than one candidate MG satisfying the criterion, then MT-RPART chooses one of them for execution at random.
- Step 15: MT-RPART executes the generated MG against the SUT and checks for MR violations.
- Step 16: MT-RPART divides the subdomain (within the source input domain) containing the currently executed STC into four subdomains using horizontal and vertical lines. Similarly, it divides the subdomain (within the follow-up input domain) containing the currently executed FTC into four subdomains using horizontal and vertical lines.
- Step 19: For the currently executed MG, MT-RPART appends its STC to the executed STC set, and appends its FTC to the executed FTC set.
- Step 18: If all the stopping conditions are not satisfied (i.e., no MR violations detected), then MT-RPART skips to Step 5; otherwise, MT-RPART ceases the process and reports the testing outcomes.

The major differences among MT-RPART and the other two MT-PART algorithms (MT-BART and MT-IPART) closely resemble those illustrated in Section 5.2.4, including the partitioning method, the conditions for input domain partitioning, and the way to defining source and follow-up candidate subdomains. Further detailed explanations are as follows:

- Regarding the partitioning method: In a 2D input domain scenario, MT-RPART divides the subdomain containing the currently selected STC/FTC into four subdomains through the use of horizontal and vertical lines.
- Regarding the conditions for partitioning the input domains: MT-RPART divides the corresponding subdomains each time an STC/FTC is selected and executed.
- Regarding the way of defining source and follow-up candidate subdomains: MT-RPART designates subdomains not covered by any executed STCs or FTCs as source or follow-up candidate subdomains.

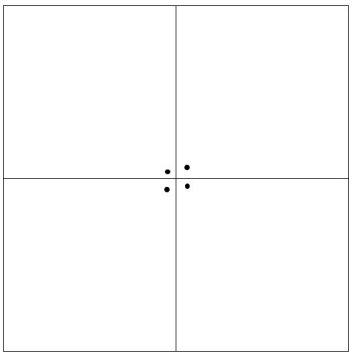


Figure 16: A set of possible nearby STCs generated using MT-BART in a 2D input domain

5.5.2 *A Combination of SFIDMT-ART and MT-BART*

The underlying reason for SFIDMT-ART's poor efficiency is mainly because it requires the generation of extra source and follow-up candidates and the distance computations and comparisons between source/follow-up candidates and executed STCs/FTCs. The computational cost of SFIDMT-ART and MT-ART becomes particularly high when using *M*-*N* MRs or when the number of executed STCs and FTCs increases.

As for MT-BART, its effectiveness also needs further improvement. In particular, the intuition of MT-BART is improve MT performance by achieving a uniform distribution of STCs and FTCs across their respective input domains. However, it has been noted that MT-BART sometimes cannot achieve this: It may suffer from the nearby test-case problem [196, 197]. That is, the MGs generated by MT-BART may be too close to the nearby MGs from neighboring subdomains. Fig. 16 shows a typical STC distribution example in a 2D input domain. It can be observed that STCs tend to cluster in a small area rather than evenly distributed across the input domain, which may negatively affect the performance of MT.

With these considerations, a new MG-generation algorithm, named MT-based BART applied to Source and Follow-up Input Domains (SFIDMT-BART), can be proposed by combining SFIDMT-ART and MT-BART. In particular, after the identification of source/follow-up candidate subdomains (Step 6 of Algorithm 3), SFIDMT-BART identifies the non-empty subdomains that are adjacent to these empty subdomains, as well as the executed STCs/FTCs that are within these non-empty subdomains. Next, instead of checking if the follow-up candidates are within follow-up candidate subdomains (Step 12 of Algorithm 3), SFIDMT-BART focuses on calculating the distance from a source/follow-up candidate to its nearest executed STCs/FTCs, respectively. The intuition for this step is to avoid generating test cases that are very close to each other. In addition, to reduce computational overheads, only the executed STCs/FTCs that are inside adjacent subdomains are considered. That it, SFIDMT-BART calculates the distance from a source/follow-up candidate to the executed STCs/FTCs that are within its adjacent subdomains. Obviously, it may happen that the subdomains adjacent to some source/follow-up candidates are all empty, which means the input domain has a large space that has not been tested. At this point, one test case would be randomly selected from those source/follow-up candidates. Theoreti-

cally, it is expected that SFIDMT-BART can achieve a better balance between effectiveness and efficiency than SFIDMT-ART and MT-BART. Note that SFIDMT-BART is an attempt to alleviate the high computational overhead of SFIDMT-ART. The future work will include the investigation of SFIDMT-BART and further modify or even redesign it based on the experimental results.

5.6 CONCLUSION

MGs have been acknowledged as one of the cornerstones of MT; however, there remains a need for further research on how to generate effective MGs [64, 246]. Previous MG-generation algorithms, including MT-ART [144] and SFIDMT-ART (proposed in the last chapter of this thesis), primarily concentrated on enhancing the effectiveness of MT while neglecting its efficiency. The testing performance of MT, particularly its efficiency, still requires further enhancement.

Taking this into account, this chapter introduced two novel MG-generation algorithms (MT-BART and MT-IPART) according to partition-based ART algorithms, aiming at achieve a balance between the effectiveness and efficiency in MT. The intuition behind the two algorithms is to distribute (1) STCs throughout the respective source input domain and (2) FTCs throughout the respective follow-up input domain. Subsequently, this chapter conducted empirical experiments to assess and compare the performance of the two algorithms with MT-ART, SFIDMT-ART and MT-RT. Programs of various input dimensions and sizes, as well as MRs, were chose from previous MT-related or ART-related studies [13, 49, 52, 62, 67, 100, 102, 136, 137, 139, 143, 144, 193]. The experimental results demonstrated that the proposed algorithms significantly outperformed MT-ART and SFIDMT-ART in terms of test efficiency while maintaining high test effectiveness.

Since in addition to MGs, MR is the second element that have a great impact on the successful implementation and performance of MT, to address the performance challenge of MT, the following chapter of this thesis will introduce a series of MRPs (Metamorphic Relation Patterns) that can be used to guide the identification of effective MRs.

6

METAMORPHIC RELATION PATTERNS, TREES AND FRAMEWORK

Papers delivered from this chapter (Under Review)

1. **Zhihao Ying**, Dave Towey, Anthony Bellotti, Zhi Quan Zhou and T. Y. Chen. Preparing SQA professionals: Metamorphic relation patterns, exploration, and testing for big data. In *Proceedings of the 2021 International Conference on Open and Innovative Education (ICOIE'21)*, 2021, pp. 22–30.

2. **Zhihao Ying**, Dave Towey, Anthony Bellotti, Caslon Chua, and Zhi Quan Zhou. Metamorphic Relation Patterns for Metamorphic Testing, Exploration, and Robustness. Submitted to *Software Testing, Verification and Reliability*, 2023.

6.1 INTRODUCTION AND MOTIVATION

The quality of MRs (and HMRs and MRRs) and MGs profoundly influences MT performance [64, 246]. Nevertheless, systematically identifying MRs remains a significant challenge, demanding innovative thinking and a thorough comprehension of the SUT, particularly for effective MRs [64, 246]. In order to systematically formalize the process of MR identification and provide guidance, Zhou et al. [292] introduced the concept of MRPs (Metamorphic Relation Patterns): They are abstract representations or templates for multiple concrete MRs. A great number of studies [33, 170, 243, 245, 280, 295] have illustrated the effectiveness of MRPs in facilitating the identification of effective MRs. As the number of MRPs continues to increase, the challenge lies in the lack of a formal approach to describe and classify the relationship between the increasing number of MRPs [280]. Users still lack a way to easily and timely find and access the MRPs they want and use them. In addition, there is also a scarcity of MRPs, particularly since some of them are limited to specific application fields [245, 247, 280]. Therefore, the identification and application of MRPs is still at an early stage and necessitate further research.

To formally define the relationship between multiple MRPs, this chapter introduces the concepts of Sub-Metamorphic Relation Pattern (sub-pattern) and Super-Metamorphic Relation Pattern (super-pattern). On the basis of these concepts, this chapter additionally proposes a formal tree structure, termed the MRP tree, for the classification of existing MRPs. Subsequently, this chapter introduces a series of new MRPs for various application fields. Through the collection and categorization of existing MRPs as well as the proposed

MRPs, this chapter presents two new MRP trees. Furthermore, a new MT framework is designed and developed to guide not only the identification but also the application of MRPs. This chapter presents findings from three case studies employing the MRPs and the proposed MT framework to identify MRs for MT, ME or MRT, revealing multiple violations. ME (Metamorphic Exploration) [292] aims at enhancing the understanding of the SUTs, thus providing users with a way to better utilize them without requiring a comprehensive user manual. In ME, MRs are the properties hypothesized by users, termed HMRs (Hypothesized MRs). MRT (Metamorphic Robustness Testing) [295] can validate SUT robustness without requiring an available oracle, and the MRs used in MRT are termed MRRs (Metamorphic Relations for Robustness). Experimental results demonstrate that MRPs can facilitate MR identification for detecting software faults (MT), evaluating software robustness (MRT), and enhancing software understanding (ME), as well as the capacity of the MT framework to guide the identification and application of MRPs. For instance, testers and users can enhance their understanding of the SUTs and improve testing or usage by applying MRPs to guide the identification of HMRs for ME. This is the first study to do so.

6.2 DEFINITIONS

6.2.1 *Sub-MRP (Sub-Pattern) and Super-MRP (Super-Pattern)*

In general, MRPs serve as abstractions/templates for multiple concrete MRs, and are capable of guiding the identification of specific MRs [292]. The process of identifying MRs involves identifying a series of *necessary* attributes of the SUT and describing them formally as MRs; similarly, identifying MRPs involves identifying *important* factors for a group of MRs, overlooking all details, and formally describing them as MRPs. The necessary attributes represent the conditions logically inferable from the SUT [64, 246], while the important factors represent the conditions logically deducible from each MR within a specified MR group [292]. Users can identify and summarize the common important factors from a series of MRs to infer MRPs, and furthermore, employ MRPs to guide the identification of specific MRs.

It has been reported that MRPs exhibit varying abstraction levels, and are capable of forming a hierarchical structure [292]: Those positioned higher possess higher abstraction levels, while those situated lower exhibit lower abstraction levels. On the basis of this observation, the concepts of sub-MRP (also known as sub-pattern) and super-MRP (also known as super-pattern) are introduced to formally define and standardize the relationships among MRPs of differing abstraction levels. The formal definitions of sub-patterns and super-patterns are as follows:

- *Definition 6.1 (sub-pattern and super-pattern)*: Let MRP_A and MRP_B represent two MRPs, where MRP_A contains the important factors $F_A = \{f_1^A, f_2^A, ..., f_k^A\}$ identified from a group of MRs, denoted as $\text{MR}_A = \{\text{MR}_1^A, \text{MR}_2^A, \ldots, \text{MR}_l^A\}$; and MRP_B contains the important factors $F_B = \{f_1^B, f_2^B, ..., f_m^B\}$ identified from a group of MRs,

denoted as $\mathrm{MR}_B = \{\mathrm{MR}_1^B, \mathrm{MR}_2^B, \ldots, \mathrm{MR}_n^B\}$. Note that $k, l, m, n \geq 2$. If $F_B \subseteq F_A$ and $\mathrm{MR}_B \subseteq \mathrm{MR}_A$, then MRP_B is termed a sub-pattern of MRP_A, and conversely, MRP_A is termed a super-pattern of MRP_B.

Generally, a super-pattern exhibits the following properties in comparison to its corresponding sub-pattern:

- The relation between a sub-pattern and its corresponding super-pattern is *transitive*. For instance, given three MRPs: MRP_1, MRP_2, and MRP_3. If MRP_1 is a super-pattern of MRP_2 and MRP_2 is a super-pattern of MRP_3, then MRP_1 is likewise a super-pattern of MRP_3.
- Super-patterns can offer more vague and abstract MR identification guidance for a wider scope of applications, while sub-patterns are able to provide more specific and in-depth MR identification guidance for a smaller scope of applications.
- Super-patterns tend to be more abstraction, while sub-patterns tend to be more concrete.
- Super-patterns contain important factors inferred from larger or identical MR groups.
- Both a sub-pattern and a super-pattern can be any of MRP, MRIP or MROP.

6.2.2 *Metamorphic Relation Pattern Tree*

A MRP family tree (or MRP tree) is tree-structured visual tool showing the MRPs of this family and structuring and visualizing relationships among them. It can provide users with a simple and fast way to find their desired MRPs for reuse, inference, or reference. This section introduces a novel approach to constructing an MRP family tree: Assuming the presence of a set of MRPs, including one MRP and all its super-patterns and sub-patterns. The MRP with the highest abstraction level serves as the root of the tree, with MRPs nearer to the root being more abstract (and less concrete), while those further away are less abstract (and more concrete). It is noteworthy that, according to the transitive property, in an MRP tree, all other MRPs can be considered as sub-patterns of the root, they only appear at the layer corresponding to their lowest abstraction level. Using the same three MRPs as examples: MRP_1, MRP_2, and MRP_3 (MRP_1 is a super-pattern of MRP_2, and MRP_2 is a super-pattern of MRP_3). An MRP tree can be formed based on the three MRPs. The root (the first layer) of this MRP tree is MRP_1. While both MRP_2 and MRP_3 are sub-patterns of MRP_1, MRP_2 appears solely in the second layer, and MRP_3 appears exclusively in the third layer. Identifying the abstraction levels of MRPs typically requires human participants (i.e., MT experts). Therefore, future work involves investigating systematic techniques for automatically measuring the abstraction levels of MRPs. Additionally, a Directed Acyclic Graph [266] represents a more general form of the relationships among MRPs: The tree structure is a subset of it, as it permits a node to have multiple parents. However, due to the absence of systematic techniques for measuring the abstraction level of MRPs, this chapter restricts the focus to tree structures.

6.3 METAMORPHIC RELATION PATTERNS AND TREES

This section proposes three MRPs with varying application scopes: *Sets* MRP, *Similar* MRP and *Irrelevance* MRP. By lowering the levels of abstraction, this section additionally proposes six MRIPs and two MROPs as sub-patterns of the proposed MRPs. To demonstrate the application and effectiveness of the proposed MRPs, this section also gathers and introduces the previously-published MRs [11, 32, 33, 285, 295, 296] that can be easily and quickly identified using the proposed MRPs. In addition, based on the concept of MRP trees, this section constructs two MRP trees by collecting and classifying the previously-published MRPs and the proposed MRPs.

6.3.1 Sets *MRP*

In mathematics, a set represents a group of n ($n \geq 1$) distinct objects, where these objects serve as the elements of the set. Generally, anything can serve as an element, including numbers, words, animals, and mathematical concepts. Sets are typically manipulated using set operations, and the following are typical set operations.

- Union: The symbol $\cup$ denotes the union of two sets: $A \cup B$ is defined as the set consisting of all the elements present in set A or set B. For example, $\{1,2\} \cup \{2,3\} = \{1,2,3\}$.
- Intersection: The symbol $\cap$ represents the intersection of two sets: $A \cap B$ is defined as the set consisting of elements common to both A and B. For example, $\{1,2\} \cap \{2,3\} = \{2\}$.
- Relative Complement: $A - B$ refers to the elements that are in set A, but are not in the set B. For example, $\{1,2\} - \{2,3\} = \{1\}$.
- Symmetric Difference: $A \oplus B$ refers to a new set that contains all the elements present in either of the sets but not in their intersection. For example, $\{1,2\} \oplus \{2,3\} = \{1,3\}$.

Relationships between sets can be categorized into various types based on the size of the set and the elements it contains. For instance, equal, disjoint, and sub-set (and super-set) are common relationships between sets.

- The symbol $=$ indicates that two given sets are equal to each other: They contain exactly the same elements in any order. For instance, $\{1,2\} = \{2,1\}$.
- Disjoint sets denote that two sets (A and B) have no common elements, represented by $A \cap B = \emptyset$. For instance, $\{1,3\} \cap \{2,4\} = \emptyset$.
- Sub-set (and super-set) relations between two sets A and B are defined as follows: $A \subseteq B$ indicates that every element of A is also an element of B, while $A \supseteq B$ indicates that every element of B is also an element of A. For instance, if $A = \{2\}$ and $B = \{1,2\}$, then A is a sub-set of B, and conversely, B is a super-set of A, denoted as $A \subseteq B$.

In addition to the relationships and operations between two sets, there are also the relationships and operations between an element and a set, and the following are two common examples:

- The symbol $\in$ denotes that a particular element is a member of a given set, and $\notin$ that a particular element does not belong to a set. For example, if a given set A consists of an element x, then it is represented by $x \in A$; and if not, then $x \notin A$.

A formal presentation of *Sets* MRP is as follows:

- *Definition 6.2 (*Sets *MRP)*: This MRP represents the MRs for which both inputs and outputs can be represented by sets, and the relationships among inputs and outputs can be denoted through set relations or calculated via set operations.

For a certain application field, such as e-commerce systems, the current MRP set (as well as the respective MRIPs and MROPs introduced in the following sections) may have covered the common set relationships and operations introduced in this section. However, with the development of computing systems and related technologies that support them, in the future, the current set of MRPs may need further revisions or additions. Furthermore, since different kinds of application fields may have different features/properties, the current MRP set may be incomplete for them.

6.3.2 Similar *MRP for Big Data Systems*

This MRP is a sub-pattern of the *Symmetry* MRP and draws inspiration from the *Input Equivalence* MRP: A robust big data system should accurately process not only identical but also similar inputs.

The following presents the formal definition of the *Similar* MRP:

- *Definition 6.3 (*Similar *MRP)*: A robust big data system should be able to return similar outputs for similar inputs.

Furthermore, it is unnecessary to clearly define certain terms in this definition such as "similar inputs" and "similar outputs", as these can be interpreted differently by testers when applying this MRP to a specific filed. For example, this MRP may serve to assess the robustness of SUTs (i.e., the MRs identified in the experiments of the chapter), which is a significant challenge in big data systems, as explained below:

1. Due to the extensive user base, developers of big data systems may lack the opportunity to directly engage with users to fully comprehend and fulfill their requirements. This contrasts with traditional software development, where developers and users can communicate directly [292].
2. The SUT might be used for various (and sometimes even conflicting) purposes [292].
3. The SUT must be able to handle diverse unexpected inputs provided by users.

6.3.3 *MRIPs for Query-based Systems*

With the aim of providing more precise and specific guidance for MR identification in query-based systems, this chapter has additionally introduced two new MRIPs, both of which are sub-patterns of *Similar* MRP, as outlined below:

In the domain of query-based systems
assuming that

- The input contains a set of parameters, including the query, the filter function and the sorting function.

the following patterns should hold

- *Query-Meaning* MRIP: The follow-up input query has the identical meaning as the source input query.
- *Input-Merge* MRIP: The combined values of all follow-up input parameters have the identical meaning as the combined values of all input parameters of the source input.

Within these two MRIPs, the concept of "similar inputs" was defined as inputs sharing identical meaning while diverging slightly in one or more other parameters: *Query-Meaning* MRIP emphasizes a particular input parameter, namely, the query, while *Input-Merge* MRIP concentrates on an combination of all input parameters. Here, *Query-Meaning* MRIP is exemplified to illustrate the application of the proposed MRIPs. By clearly explaining the definition of "elements" and the method of their slight modifications in different ways, three MRs can be identified for e-commerce systems as follows:

- MR_{QM1}: If the source input query is altered sorely through rearranging two words, or by a minor syntactical adjustments, to generate the follow-up input query with the same meaning, then the follow-up output ought to have identical or substantially similar items as the source output, irrespective of their order. For instance, [“shoes for men”] and [“men’s shoes”] are two different input queries, but they contain the same meaning and may be used to search for the same thing. In this context, a robust e-commerce system ought to return identical or highly similar outputs for them, irrespective of their order. This MR was also included in the empirical studies of this chapter, as shown in Section 6.5.2.
- MR_{QM2}: A robust e-commerce system should return identical or substantially similar outputs, irrespective of their order, for two input queries with equivalent meaning but differing solely in the use of singular and plural linguistic forms. For instance, some users always use the query ["hats"], while some may prefer ["hat"]. Since both are valid and used to search for the same thing, an e-commerce system should return identical or highly similar outputs, irrespective of their order.
- MR_{QM3}: A robust e-commerce system should return identical or substantially similar outputs, irrespective of their order, for two input queries with equivalent meaning but differing solely in the presence of a conjunction. For example, [“sweaters

for men"] and ["sweaters men"] are commonly-used input queries and a robust e-commerce system ought to be able to return identical or highly similar outputs for them, irrespective of their order.

6.3.4 *MRIPs for Machine Translation Systems*

In order to offer more precise and in-depth guidance for MR identification in machine translation systems, this section has additionally proposed two new MRIPs, both of which are sub-patterns of *Sets* MRP and *Similar* MRP, as illustrated below:

In the domain of machine translation systems
assuming that

- The input can be represented by a sequenced group of elements.
- The elements can be words, phrases, or punctuation marks.

the following input patterns should hold

- *Addition* MRIP: The follow-up input consists of an additional element compared to the source input.
- *Subtraction* MRIP: The follow-up input consists of one fewer element compared to the source input.
- *Substitution* MRIP: The source and follow-up inputs contain identical length and sorely differ in one of the elements.
- *Combination* MRIP: The ordered combination of multiple follow-up inputs have identical or highly similar elements as the ordered combination of multiple source inputs.

These four MRIPs draws inspiration from both *Sets* MRP and *Similar* MRP: Based on the *Sets* MRP, this chapter divides the inputs into a sequenced group of elements, and subsequently concentrated on the input relationship within the *Similar* MRP to identify the four MRIPs. Specifically, to identify the *Addition* MRIP, the *Subtraction* MRIP, and the *Substitution* MRIP, the concept of "similar inputs" was characterized as two (highly) identical inputs differing solely in one specific element. To identify the *Combination* MRIP, this chapter focused on the combination of n inputs ($n \geq 1$), and additionally defined "similar inputs" as an ordered combination of n source/follow-up inputs. Moreover, in the *Combination* MRIP, the "(highly) identical" inputs can be constructed based on the approaches mentioned within the *Addition* MRIP, the *Subtraction* MRIP, and the *Substitution* MRIP.

6.3.5 Irrelevance *MRP for Big Data Systems*

With the aim of offering more precise and in-depth guidance for MR identification in big data systems, this section has additionally proposed one new MRP, which is the sub-pattern of *Symmetry* MRP, as outlined below:

- *Definition 6.4 (*Irrelevance *MRP)*: A robust big data system should return (highly) identical output for the inputs differing solely in irrelevant conditions.

This MRP draws inspiration from both *Symmetry* MRP and *Input Equivalence* MRP: The behavior/output of the SUT can be influenced by both inputs and environmental conditions, warranting consideration during the MR identification process. Potential irrelevant conditions for a machine translation system, for example, are the factors like the date and medium of SUT execution (such as a laptop or a mobile phone). It is noteworthy that since various SUTs may possess distinct attributes, irrelevant conditions identified for a particular SUT type may not be applicable to other SUT types. In addition, the inputs of *Similar* MRP differ from those of *Irrelevance* MRP: The "similar inputs" represent the inputs that are not only similar but also have an impact on the behavior/output of the SUT. Particularly, *Irrelevance* MRP focuses on the conditions irrelevant to SUT behavior/output, while *Similar* MRP mainly concentrates on the conditions relevant to SUT behavior/output.

6.3.6 *MROPs for Big Data Systems*

With the aim of offering more precise and in-depth guidance for MR identification in big data systems, this section has additionally proposed two new MROPs, both of which are the sub-patterns of *Irrelevance* MRP, as illustrated below:

In the domain of big data systems
assuming that

- The behavior/output can be represented by a series of elements, called sub-outputs.

the following output patterns should hold

- *Fully-Equal* MROP: Each sub-output in the follow-up output is identical to the relevant sub-output in the source output.
- *Partially-Equal* MROP: One or several sub-outputs in the follow-up output are identical to the relevant sub-outputs in the source output.

These two MROPs were identified by concentrating on the output relationship within *Irrelevance* MRP: Further postulating that SUT outputs can be decomposed into discrete elements termed sub-outputs According to this premise, this chapter suggests that an irrelevant condition does not necessarily need to be irrelevant to the entirety of SUT behavior/output; instead, it can be irrelevant to one or more sub-outputs of the SUT behavior/output. In this scenario, different conditions may be considered "irrelevant" to various sub-outputs. Amazon's output, for instance, is a list of items that can be divided into a group of sub-outputs, including the name, quantity and order. One of the inputs to Amazon, the sorting rule, should solely have an impact on the item order and remain "irrelevant" to item names or quantities.

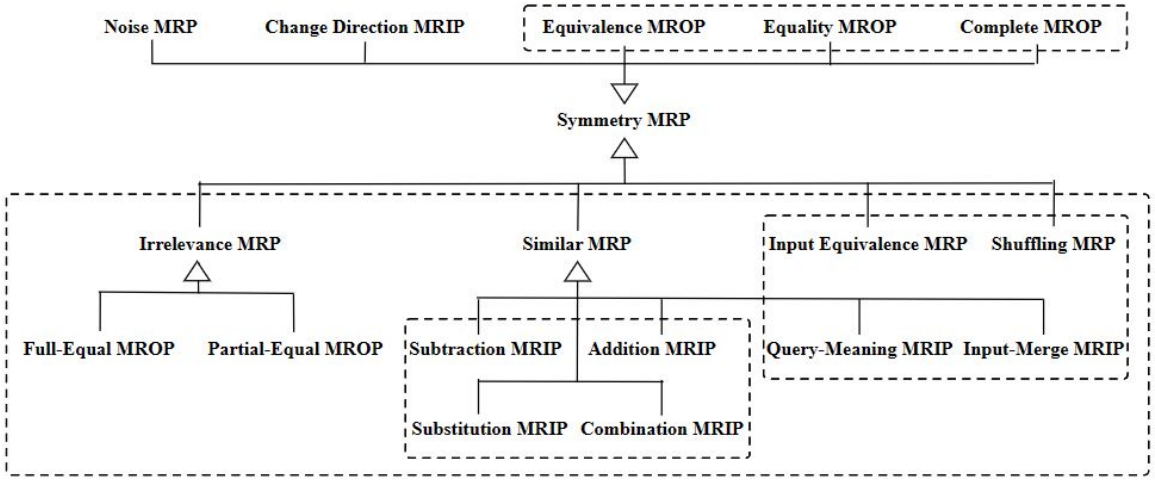


Figure 17: *Symmetry* MRP Tree

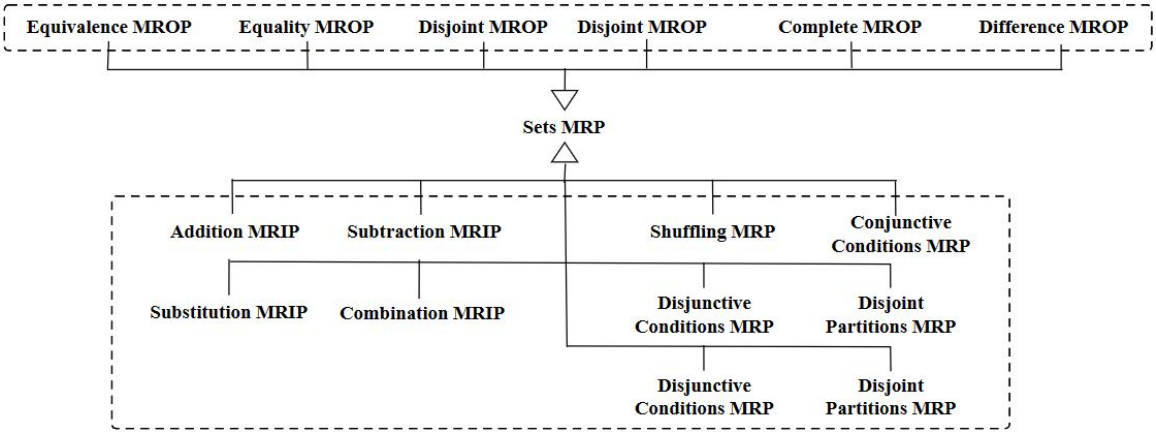


Figure 18: *Sets* MRP Tree

6.3.7 Symmetry *MRP Tree and* Sets *MRP Tree*

In this section, by gathering and categorizing previously-published MRPs as well as the proposed MRPs (and MRIPs and MROPs), two MRP trees were proposed, as shown in Figs. 17 - 18: *Symmetry* MRP tree and *Sets* MRP tree. Within these figures, the MRPs within the same dashed box share identical application scopes, with specific scopes detailed in Table 14. These trees serve to collect and classify MRPs, and offer users a convenient and efficient means to locate MRPs for reuse, inference, or reference. *Symmetry* MRP serves as the root of the first tree, while *Sets* MRP acts as the root of the second tree. MRPs connected by a single line represents the presence of a sub-super relation: MRPs positioned higher possess higher abstraction levels and serve as super-patterns of MRPs positioned lower, while MRPs positioned lower have lower abstraction levels and function as sub-patterns of MRPs positioned higher. It is noteworthy that while some MRPs may have limited application scopes, they are intended to offer guidance for MR identification rather than directly generating new MRs. Consequently, MRPs from other domains can also be employed provided the tester possesses adequate understanding of the SUT.

Table 14: MRP Application Scopes

Application Scope	Total Number of Patterns	MRPs	MRIPs	MROPs
No Restrictions	4	Symmetry, Sets, Noise	Change Direction	-
RESTful Web APIs	6	-	-	Equivalence, Equality, Subset, Disjoint, Complete, Difference
Big Data Systems	4	Irrelevance, Similar	-	Fully-Equal, Partially-Equal
Query-based Systems	9	Query-Meaning, Input-Merge, Input Equivalence, Disjunctive Conditions, Disjoint Partitions, Complete Partitions, Shuffling, Conjunctive Conditions, Disjunctive Conditions	-	-
Machine Translation Systems	4	-	Addition, Subtraction, Substitution, Combination	-

6.3.8 *Existing Application of the Proposed MRPs*

This section has gathered and summarized previously-published MRs that can be easily identified based on the MRPs proposed in Section 6.3. For instance, the following are concrete MRs derived from the proposed *Similar* MRP and its sub-patterns:

- $MR_{MPReverseJD}$ [296]: For a robust search engine, a slight change in the input query should not severely affect the output. Additionally, Signorini et al. [252] used the rationale behind this MR to test semantic search engines according to a set of semantically equivalent queries.

- MR_{SwapJD} [296]: Given an input query consisting of two words. For a robust search engine, swapping the two words to construct a new input query without changing the meaning should not have a severe impact on the output. This MR is designed based on the same rationale as $MR_{MPReverseJD}$. Zhou et al. [296] used the Jaccard Coefficient [221] to measure the overlap percentage between outputs and the similarity between outputs.

- $MR_{MPShuffleJD}$ [11]: For a robust academic search engine, a slight change in the input query should not have a series impact on the output. This MR has been used to assess the stability ranking of academic search engines (such as IEEE, ACM, Springer, IEEE, and ScienceDirect), and is a follow-up of the work by Zhou et al. [296].

- $MR_{Similar}$ [32, 33] was identified on the basis of the rationale that for an industrial map system, a slight modification in the input query should not severely affect the output [296]. Lee et al. [170] applied these kinds of MRs to assess machine translator (such as Google Translator). For instance, MR_{case} presents that for a robust machine translator, minor case-sensitive typos should not have a severe impact on the output. Typical examples of input queries are ["It is A pen"] and ["It is a pen"].

- $MR_{Similar}$ [295]: A robust citation database system ought to return identical or highly similar average citation counts for two slightly different large publication sets (i.e., without any systematic differences in the factors related to citations)

- He et al. [129] employed MT to assess machine translators, positing that a robust machine translator ought to generate outputs with similar structures for similar inputs.

They proposed the following three approaches to construct similar input sentences: Raw target sentences, constituency parse trees, and dependency parse trees. For instance, the process of raw target sentences involves modifying an individual token in a sentence to check its variation in the output.

The specific MRs derived from the proposed *irrelevance* MRP and its sub-patterns are as follows:

- $MR_{Environment}$ [32]: A robust navigation system ought to return a similar output for the same input across various user environments (i.e., APIs and mobile applications).
- $MR_{TrimOff}$ [33]: In a robust map system, altering irrelevant environmental factors should not have a severe impact on the time cost and distance of a returned route.

6.4 A NEW METAMORPHIC TESTING FRAMEWORK

6.4.1 *Introduction and Motivation*

With the growing popularity of MT and the significance of MR, MRP as a technology for MR identification has garnered increased attention. The quantity of MRPs identified for various application fields is rapidly increasing [245, 247, 280, 292]. Therefore, an important question requiring further research is: *How can MRPs be effectively used as their number continues to grow?* With this consideration, this section introduces a new MT framework to tackle this issue: This framework is capable of facilitating the identification and application of MRPs (as well as MRIPs and MROPs). This framework accomplishes its objectives by dividing an MRP into two components: An input-only pattern (MRIP) and an output-only pattern (MROP). When applying an MRIP or an MROP, the user only needs to consider how to satisfy either the input or the output relation, rather than both simultaneously. The pertinent output or input relation can be deduced according to the particular concrete input or output relations, respectively. This can alleviate the challenge of MR identification for users, particularly MT beginners.

6.4.2 *Framework*

Fig. 19 introduces the architecture of the proposed MT framework to guide not only the identification, but also the application, of MRPs (as well as MRIPs and MROPs). It consists of two major components: The "MR-Identification Phase" (to guide concrete MR identification) and the "MRP-Identification Phase" (to guide MRP identification according to existing MRPs). As other MT steps, including the generation and execution of STCs and FTCs, are the same as traditional MT process, they are not included in the framework.

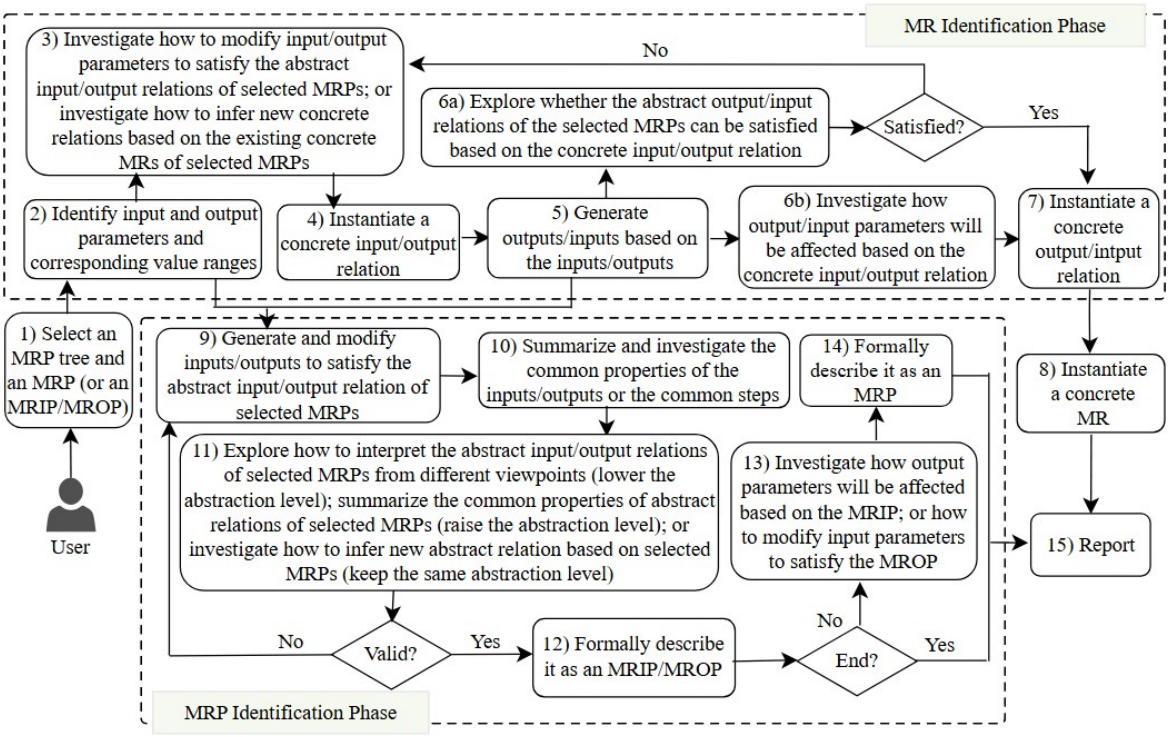


Figure 19: MT Framework Architecture

6.4.3 *Application of the MT Framework*

This section introduces the application of the proposed MT framework via a typical example involving users on the Amazon e-commerce platform.

1. Users are capable of choosing an MRP tree and an MRP (or MRIP/MROP) with preferred abstraction levels, for future application. Using the *Change Direction* MRIP and Amazon as an example.

2. Users can determine input and output parameters as well as their relevant value ranges of the SUT. As an illustration, Amazon's input parameters consist of a query, a sorting rule, and a filter rule; while its output parameters consist of item name, quantity, and order.

3. Users have two potential methods to determine a new relation:

   (a) Explore the modification of one or more input parameters to fulfill the abstract input/output relation of the chosen MRP.

   (b) Investigate previously-published concrete MRs/HMRs of the selected MRP to deduce novel concrete input/output relations. It is noteworthy that the existing concrete MRs/HMRs of the MRPs proposed in this chapter have been gathered and summarized in Section 6.3.8.

   For instance, assuming method 3a is chosen. Within Amazon, the sorting rule values are: "Feature" (default), "Price: Low to High", "Price: High to Low", "Avg. Customer Review", "Newest Arrivals", and "Best Sellers". In the *Change Direction* MRIP, the

meaning of "modifying the direction element" was construed to switch the sorting rule from "Price: High to Low" to "Price: Low to High".

4. Users can try to instantiate a concrete input/output relation. Building upon the aforementioned example, an input relation, MR_{Input}, can be determined thus: The sorting rule is transitioned from "Price: High to Low" to "Price: Low to High" to formulate follow-up inputs.

5. Users are able to construct outputs/inputs for future application in accordance with the inputs/outputs constructed in Step 2.

6. Users have two methods to explore potential relations:

   (a) Users can scrutinize if the abstract output/input relations of the chosen MRPs can be fulfilled according to the currently-identified concrete input/output relation (Step 3a). Users can try to identify a concrete output/input relation if the abstract relations are fulfillable; otherwise, proceed to Step 3.

   (b) Users can examine the ramifications of the currently-identified concrete input/output relation on the output/input parameters. For example, given the currently-identified MR_{Input} and the *Change Direction* MRIP, method 6b is chosen: The quantity of returned items should remain unchanged, while their order is reversed.

7. Users can try to determine a concrete output/input relation. For example, a concrete output relation, MR_{Output}, can be constructed as follows: The item orders in the follow-up outputs ought to be inverted from that of the source outputs.

8. Users are able to ascertain a concrete MR/HMR according to previously-identified concrete input and output relations. Given the currently-identified MR_{Input} and MR_{Output}, a concrete MR/HMR can be constructed: If the sorting rule in the source input transitions from "Price: High to Low" to "Price: Low to High" to generate the follow-up input, then the item order in the follow-up output ought to be inverted from that of the source outputs.

9. In addition to MR/HMR identification, the MT framework is also applicable for identifying MRPs (as well as MRIPs and MROPs). More specifically, users can generate inputs/outputs based on the identified input/output parameters and their value ranges (in Steps 2 and 5), then modify them to investigate how to meet the abstract input/output relations of the chosen MRP (or MRIP/MROP).

10. Users can explore and summarize the common properties of the inputs/outputs, or the common steps taken to meet the abstract relations.

11. The MT framework provides users with three possible methods to deduce new MRPs with varying abstraction levels according to the chosen MRP (or MRIP/MROP).

    (a) Reduce the abstraction level: Investigate various interpretations of the chosen abstract input/output relation.

(b) Raise the abstraction level: Investigate and summarize the common properties of the chosen abstract input/output relations.

(c) Maintain the same abstraction level: Explore ways to modify the chosen MRP to deduce new abstract input/output relations. Examples of this include the *Addition* MRIP and the *Subtraction* MRIP.

12. Users are able to construct inputs/outputs to verify the validity and achievability of the findings. If yes, then users can formally translate the abstract input/output relation of the chosen MRPs into a more concrete/abstract relation as a new MRIP/MROP; otherwise, proceed to Step 9.

13. Users are able to choose whether or not to conclude the MT process. If yes, then proceed to Step 15; otherwise, the MT framework provides users with two possible methods to deduce a new corresponding MROP/MRIP:

    (a) Explore the impact of the currently-identified MRIP on the output parameters.

    (b) Explore the ways of satisfying the currently-identified MROP by modifying the input parameters.

14. Users are able to formally present the previously-identified MRIP and MROP as a new MRP.

15. Users are able to conclude the MT process and report the outcomes.

6.5 A CASE STUDY OF QUERY-BASED SYSTEMS

6.5.1 *Experimental Setup*

Systems Under Test: According to the Global Powers of Retailing 2022 report [96], two popular e-commerce platforms, namely Amazon and JD.com, were selected as the subject programs for the empirical experiments. The experimental studies include both English and Chinese languages. Thus, there are a total of three SUTs: Amazon in its English version, Amazon in its Chinese version, and JD.com. To obviate any potential impact stemming from user-account configurations [292], the experiments were executed without any logged-in sessions to either Amazon or JD.com. It is noteworthy that the empirical experiments were executed in 2023: Since 2024, JD.com requires user authentication prior to accessing its query function. All the experimental results reported in this chapter were reproducible on different days and machines during the experimental process, therefore ruling out the possibility of MR violations were caused by server-side data updates [292].

Test Case Generation: As presented in Figs. 20 - 21, English inputs were constructed based on popular syntactic structures within Amazon: the "noun-'s-noun" structure and the "noun-for-noun" structure. In contrast, for JD.com, as presented in Fig. 22, the "noun noun" structure was employed to construct Chinese inputs. The first set of nouns included common consumer requisites sourced from Amazon, while the second set included two common nouns, "women" ("女") and "men" ("男"). To illustrate, given the nouns "sweaters"

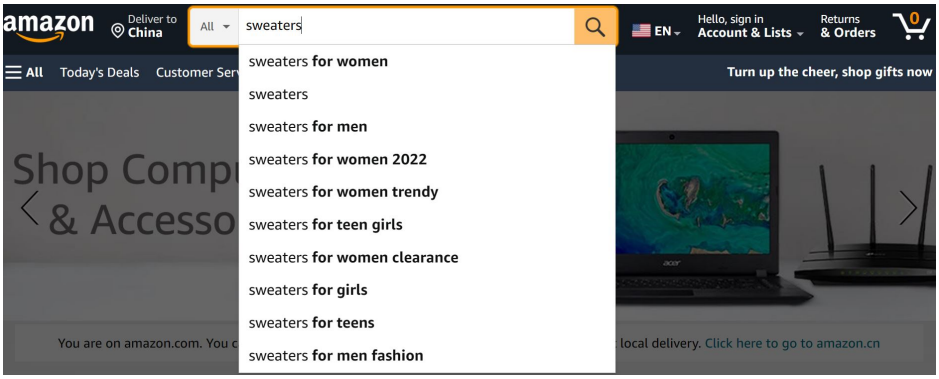


Figure 20: The first recommendation list example from Amazon

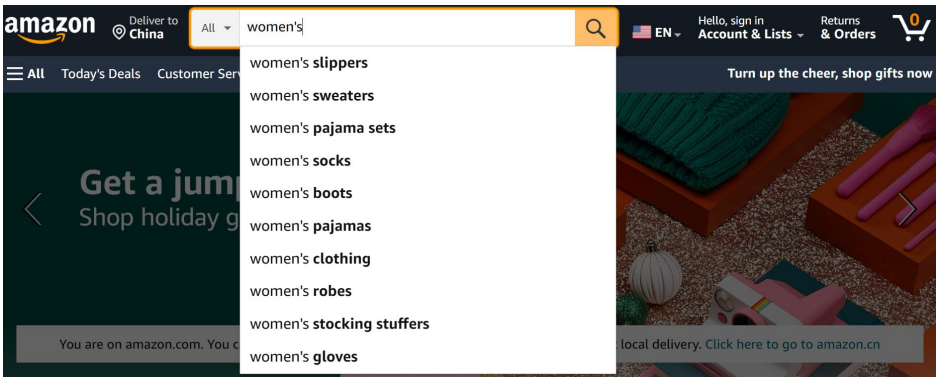


Figure 21: The second recommendation list example from Amazon

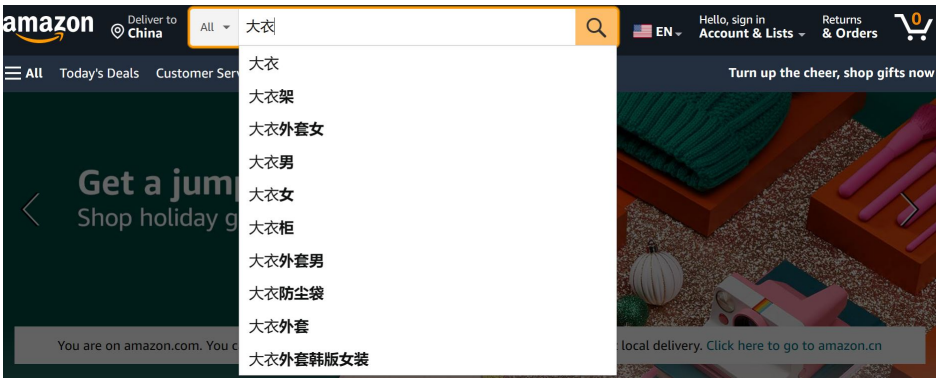


Figure 22: The third recommendation list example from Amazon

("毛衣") and "men" ("男"), two English inputs ([“sweaters for men”] and [“men’s sweaters”]) and two Chinese inputs ([“毛衣男”] and [“男毛衣”]) were constructed. In addition, the first noun set was also directly employed as inputs in the experimental studies.

The Selection of Query Function and the Number of Returned Items: The query function was chosen for the empirical experiments, with the number of returned items serving as the outputs. That is, in this chapter, the output of an e-commerce system typically refers to the number of items. The query function was singled out owing to its popularity across e-commerce platforms [292]. Users are able to derive insights through the identification and application of MRPs and HMRs, in order to enhance their understanding of the behavior and output of the query function. HMR violations enable users to evaluate the performance of e-commerce systems in meeting their requirements. User are able to make

better decisions regarding subsequent actions, including adopting more precise queries or choosing other e-commerce platforms.

The rationales behind the adoption of the number of returned items as the evaluation metric are:

- A significant difference between the source output and the follow-up output may have a impact on user experience. For instance, as presented in Figs. 25 - 26, when a sorting rule is used, although users may not browse every returned item, a significant omission in the returned items may lead to the loss of some top-ranked items, which may prevent users from finding their desired things.
- Even a small difference in returned item counts can have a severe impact on user experience. For instance, as illustrated in Figures 23 - 24, the absence of over 940 items, relative to the initial 66 items returned, could result in users being unable to find their desired item, which may prompt them to choose alternative e-commerce platforms.

6.5.2 *Relations*

In total, one MR and three HMRs were identified for e-commerce systems following the guidance of MRPs and the MT framework, as illustrated below.

MR1: The sorting rule should not have an impact on the output (the number of returned items) of a robust e-commerce system. In the empirical experiments, the default rule and the price sorting rule were chosen due to their common usage among users [91, 95, 258]. This MR exemplifies the *Partially-Equal* MROP: Since the empirical experiments focus on solely one specific type of output (the number of returned items), new concrete MRs can be deduced by considering the use of functions that have no effect on it.

HMR1: If a specific filter rule is applied to the source input to generate the follow-up input, then the follow-up output (the number of returned items) should be greater than or equal to the source output (the number of returned items). This HMR was derived based on the *Complete Partitions* MRP [245]. For example, when an input query like ["sweaters"] is filtered by the default rule (["All Department"]), if further filtered by a specific rule, such as ["Men's Fashion"], a robust e-commerce system should return identical or more items for the first input compared to the second.

HMR2: If only the word order is altered, or an additional grammatical change is made to the source input query to generate the follow-up input query without altering the meaning, then the follow-up output (the number of returned items) should be equal to the source output (the number of returned items). This HMR can be easily and quickly identified based on the *Query-Meaning* MRIP (and also its super-pattern, the *Similar* MRP). For example, consider two English input queries involving structural changes: ["men's sweaters"] and ["sweaters for men"]. Both query forms are commonly used by users, as evidenced by the recommendation lists from Amazon (see Figs. 20 - 21). Although the two input queries are not the same, they have the same meaning and are used to search for the same

things. In this context, a robust e-commerce system ought to return identical or highly similar outputs for them. Notably, in Chinese, rather than using conjunctions like "for", users prefer directly combining Chinese words (see Fig. 22). For example, the two English input queries (["men's sweaters"] and ["sweaters for men"]) can be respectively translated to Chinese as ["男士毛衣"] and ["毛衣男士"].

HMR3: If a user employs two different inputs to search for the same item — the first input relies solely on the query function, while the second one includes both the query and filter functions — then a robust e-commerce system should return identical or highly similar outputs (the number of returned items) for both inputs. Since the source and follow-up input queries contain the same meaning, users are likely seeking the same item via different inputs, and a robust e-commerce system should correctly handle such scenarios. This HMR can be easily and quickly identified based on the *Input-Merge* MRIP (and also its super-pattern, the *Similar* MRP). For instance, when searching for women's sweaters on Amazon, some users may employ a straightforward query like ["women's sweaters"], with the filter function defaulting to ["All Departments"]. Conversely, some users might prefer the query ["sweaters"] filtered by ["Women's Fashion"]. In both cases, Amazon should return identical or highly similar outputs.

The reason for having a set of HMR1 to HMR3: The purpose of the experiment is to use MRPs, the MT framework and MRP trees to guide the identification of MRs. During the experiment, a set of MRs and HMRs were identified, and after a literature review, the duplicate MRs/HMRs were removed. Finally, three HMRs were identified and used for e-commerce systems. In addition, there is no sub/super relation among HMR1 to HMR3 (and other relations stated in Section 6.6 and Section 6.7). To illustrate the sub/super relation between two MRs, suppose we are testing a simple program F(x) that computes the value of 2 to the x^{th} power. The following two MRs can be identified for the program F(x):

- MRa: If x2 = x1 + 1, then F(x2) > F(x1).
- MRb: If x2 = x1 + P (P > 0), then F(x2) > F(x1).

We can see that MRb can also be used to construct new MRs by setting different values of P (including MRa when P = 1). At this point, MRa can be regarded as the sub-relation of MRb. Thus, as we can see, there is no sub/super relation among the relations stated in the thesis.

6.5.3 *Experimental Results, Evaluation and Discussion*

The experimental results indicating the numbers of MR/HMR violations for Amazon and JD.com are presented in Tables 15 - 17: MR1 and all HMRs were violated by all three SUTs. It is worth noting that Amazon and JD.com did not provide exact numbers of returned items. Therefore, parenthesized numbers were considered to represent the range of differences between a source output and its corresponding follow-up output. For instance, "Number of Violations $[10^0 - 10^1)$" represents the number of MR/HMR violations where the difference range between the source output and its corresponding follow-up output is

Table 15: The number of MGs and MR/HMR violations for English Amazon

MRs	Number of Metamorphic Groups	Number of Violations (Total)	Number of Violations $[10^0 - 10^1)$	Number of Violations $[10^1 - 10^2)$	Number of Violations $[10^2 - 10^3)$	Number of Violations $[10^3 - 10^4)$	Number of Violations $[10^4 - 10^5)$	Number of Violations $[10^5 - 10^6)$	Number of Violations $[10^6 - 10^7)$
MR1	1747	1532	31	71	84	505	708	133	0
HMR1	1747	159	14	22	61	39	20	3	0
HMR2	1747	982	127	183	51	318	292	11	0
HMR3	1747	1681	11	120	331	703	502	14	0

Table 16: The number of MGs and MR/HMR violations for Chinese Amazon

MRs	Number of Metamorphic Groups	Number of Violations (Total)	Number of Violations $[10_0 - 10_1)$	Number of Violations $[10_1 - 10_2)$	Number of Violations $[10_2 - 10_3)$	Number of Violations $[10_3 - 10_4)$	Number of Violations $[10_4 - 10_5)$	Number of Violations $[10_5 - 10_6)$	Number of Violations $[10_6 - 10_7)$
MR1	2621	2361	190	220	401	1001	469	40	0
HMR1	2621	164	40	68	37	15	4	0	0
HMR2	2621	1884	309	312	427	575	234	27	0
HMR3	2621	2557	12	218	730	1135	398	14	0

Table 17: The number of MGs and MR/HMR violations for JD.com

MRs	Number of Metamorphic Groups	Number of Violations (Total)	Number of Violations $[10^0 - 10^1)$	Number of Violations $[10^1 - 10^2)$	Number of Violations $[10^2 - 10^3)$	Number of Violations $[10^3 - 10^4)$	Number of Violations $[10^4 - 10^5)$	Number of Violations $[10^5 - 10^6)$	Number of Violations $[10^6 - 10^7)$
MR1	2621	668	10	12	175	10	421	40	0
HMR1	2621	20	0	0	1	2	6	4	7
HMR2	2621	1275	77	175	688	159	21	126	11
HMR3	2621	1909	41	167	708	282	436	265	10

within $[10^0 - 10^1)$. As illustrated in Fig. 27, the source output exceeds 100,000 items, while the relevant follow-up output shown in Fig. 28 exceeds 200,000 items. Consequently, the difference between them amounts to 100,000, falling within the range $[10^5 - 10^6)$. Given that both the outputs and the differences are not precise numbers, the MR/HMR violations numbers presented in Tables 15 - 17 are approximations. For instance, the actual values of the two outputs may be 199,999 and 200,001, meaning the true difference between them is merely two, as opposed to 100,000. Due to the lack of access to the precise numbers and the internal source code of the SUTs, this study solely relies on the approximate numbers provided by Amazon and JD.com. In addition, since investigating the root causes of violations and delving into the internal architecture of Amazon and JD.com are not the primary objectives of this study, only the instances of MR/HMR violations of (the English version of) Amazon are exclusively presented and analyzed.

6.5.3.1 *Violations of MR1*

All the three SUTs violated MR1. Figs. 23 - 24 illustrate an instance of an MR1 violation on the (English version of) Amazon website: The source input was ["iPhone"] with ["Sort by: Featured"], and its corresponding output exceeded 1000 items; however, the follow-up input was ["iPhone"] with ["Sort by: Price Low to High"], and its corresponding output contained only 60 items. In this scenario, the estimated difference amounted to 940, markedly exceeding the follow-up output. Because only the sorting rule in the source input was altered to construct the follow-up input, and the source output significantly differed from the follow-up output, this was a clear MR1 violation.

Figure 23: An MR1 STC example for Amazon in English

Figure 24: An MR1 FTC example for Amazon in English

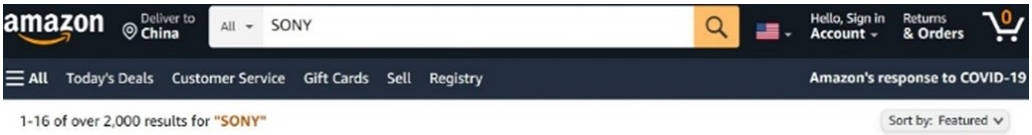


Figure 25: An HMR1 STC example for Amazon in English

Figure 26: An HMR1 FTC example for Amazon in English

MR1 is related to the functional correctness of Amazon and JD.com. Functional correctness denotes the ability of the SUT to deliver the anticipated outcomes with the necessary precision [1]. In this context, the violation of MR1 reveals the functional deficiency of the SUTs: At least one output contains incomplete results.

Since the internal code and design of SUT cannot be accessed, it is not possible to ascertain the causes of MR1 violations and the functional deficiency. Nonetheless, MR1 violations present both pros and cons from the user experience standpoint: On the one hand, an overly detailed and precise output may require excessive preparation time, while a faster but less precise output may cater to users who prioritize shorter loading times. For instance, Kohavi et al. [159] presented that certain users exhibit high sensitivity to delays, and even minor delays may prompt them to switch to alternative e-commerce platforms: For example, they reported that a 100 millisecond increase in the page load time cost 1% in sales, while similar research from Google shown that a 500 millisecond increase in the search results display time cost 1% in revenue [159]. A potential drawback could be that the absence of items may impact the user experience, as some users prioritize an complete and accurate list of results despite longer loading times [234].

6.5.3.2 *Violations of HMR1*

All the three SUTs violated HMR1. Figs. 25 - 26 illustrate an instance of an HMR1 violation on the (English version of) Amazon: The source input was [“SONY”] with [“All Departments”], and its corresponding output exceeded 2000 items; however, the follow-up input was [“SONY”] with [“Women’s Fashion Department”], and its corresponding output exceeded 200,000 items. In this scenario, the estimated difference amounted to

Figure 27: An HMR2 STC example for Amazon in English

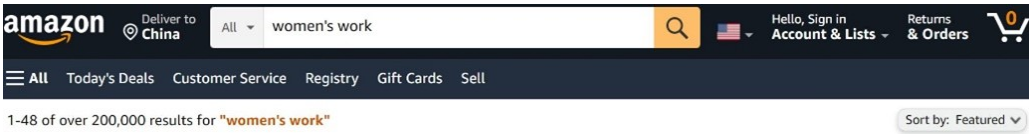


Figure 28: An HMR2 FTC example for Amazon in English

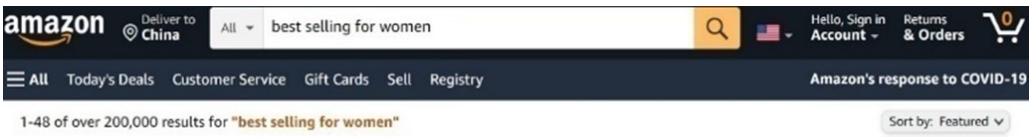


Figure 29: An HMR3 STC example for Amazon in English

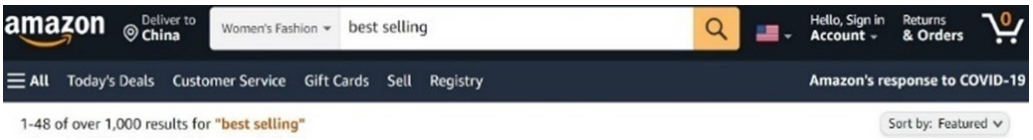


Figure 30: An HMR3 FTC example for Amazon in English

198,000, markedly exceeding the source output. Given that the source output drastically differs from the follow-up output (as the [“All Departments”] category should contain the [“Women’s Fashion Department”]) and the difference is substantial, this is definitely an HMR1 violation.

Since the internal code and design of SUT cannot be accessed, it is impossible to ascertain the reasons of HMR1 violations. Nonetheless, these violations contravene the fundamental functionalities of e-commerce systems and could potentially impact the user experience: The Amazon web system explicitly specifies that the [“All Departments”] should contain all other departments. The users of e-commerce systems may not be familiar with the technical design details of the SUTs. The HMR1 violations indicate that when a default setting is used (such as "All Departments"), users may not be able to get a comprehensive result. This could potentially make it difficult for certain users to find their desired item and thus, the user experience may be affected.

6.5.3.3 *Violations of HMR2*

All the three SUTs violated HMR2. Figs. 27 - 28 illustrate an instance of an HMR2 violation on the (English version of) Amazon: The source input was [“work for women”], and its corresponding output exceeded 100,000 items; however, the follow-up input was [“women’s work”], and its corresponding output exceeded 200,000 items. In this scenario, the estimated difference amounted to 100,000, which was quite substantial. Given that these two input queries serve the same purpose of searching for identical items, and the difference between the outputs is considerable, this is definitely an HMR2 violation.

6.5.3.4 *Violations of HMR3*

All the three SUTs violated HMR3. Figs. 29 - 30 illustrate an instance of an HMR3 violation on the (English version of) Amazon: The source input was ["best selling for women"] with ["All Departments"], and its corresponding output exceeded 200,000 items; however, the follow-up input was ["best selling"] with ["Women's Fashion Department"], and its corresponding output exceeded 1000 items. In this scenario, the estimated difference amounted to 199,000, markedly surpassing the follow-up output. Given that these two input queries are employed to search for identical items, and the difference between the outputs is considerable, this is definitely an HMR3 violation.

While the HMR3 violations may not contravene the fundamental functionalities of e-commerce systems, they could still impact the user experience, as the users of e-commerce systems may not familiar with the technical design details of the SUTs. The HMR3 violations indicate that at least one of the outputs is not complete. As different users may adhere to individual typing habits and employ preferred methods to search for their desired items, they may struggle to accomplish their objectives, and thus, their user experience may be affected.

6.6 A CASE STUDY OF MAP SYSTEMS

6.6.1 *Experimental Design*

Systems Under Test: Map systems provide a function for planning a route (producing the most cost-effective and efficient route from an origin to a destination). They are not only among the most widely-used Internet applications but also the mobile applications with the highest number of downloads globally [32]. The underlying geographic data exemplifies another instance of big data, making map systems ideal SUTs for this study. In the empirical experiments, MRPs and the proposed MT framework were employed to alleviate the challenge of identifying MRs for map systems. Two popular map systems were selected for this experiment: Google Maps [32, 33] and Baidu Maps [174]. To eliminate the probability of MR violations being caused by the user-account configurations [292], the experiment was conducted without logging into any accounts. Moreover, to mitigate the influence of server-side data updates, all experimental results presented in this chapter are reproducible on different days and machines throughout the experimental process.

Test Case Generation: Locations in both Google Maps and Baidu Maps were randomly chosen as source inputs, and this process was repeated to generate 100 valid source inputs.

6.6.2 *Relations*

In total, one MR and one HMR were identified for map systems following the guidance of MRPs and the MT framework, as presented below.

MR2: A robust map system ought to produce identical outputs for identical inputs within very short timeframes. The timeframe represents the duration taken by a map system to execute an input once, typically approximately one second. This MR draws inspiration from the *Irrelevance* MRP, where the short timeframes represents an irrelevant environment. Considering the same origin and destination through different system functions should result in identical or highly similar outputs.

HMR4: Whether by clicking on the map or inputting location names to determine a route between the same origin and destination, a robust map system should produce identical or highly similar routes. This HMR draws inspiration from the *Similar* MRP: Users have two methods for selecting origins and destinations, namely, inputting location names or clicking on the map. Using the same origin and destination through different system functions should result in identical or highly similar outputs.

6.6.3 *Evaluation and Discussion*

Throughout the experiment, it was observed that both SUTs satisfied MR2 while violating HMR4. Out of the 100 generated MGs, four violated HMR4 on Google Maps and five on Baidu Maps. For brevity, solely the HMR4 violation instances of Google Maps were presented in this section. It is essential to acknowledge that since software testing can solely ascertain the existence of failures rather than their absence [98], the absence of violations in the experimental results for MR2 does not guarantee that MR2 will never be violated.

Figs. 31 - 32 illustrate an instance of HMR4 violation on Google Maps: The source input involved typing the destination name (["Jubilee Park"] in London), resulting in a route of 0.8 miles long and estimated to take 15 minutes; the follow-up input entailed clicking on the destination (["Jubilee Park"]) on the map (situated very close to the source input destination with no apparent obstacles in between), resulting in a route of 0.2 miles long and estimated to take 5 minutes. In this scenario, the time difference was 10 minutes and the distance difference was 0.6 miles, both of which are substantial. Given that these two inputs share the same origin and their destinations are in close proximity to each other with no apparent obstacles in between, this is unquestionably an HMR4 violation.

HMR4 violations may not contravene the basic functions of map systems as the exact destinations are not identical. Nevertheless, map systems are designed for a wide user base and offer various methods for selecting origins and destinations, given that different users may opt for different interaction methods with the map system, a robust map system should correctly handle inputs that vary but serve the same purpose. In this context, significant differences in outputs may adversely affect the user experience. For instance, as shown in Figs. 31 - 32, if the user simply searches for the shortest path to Jubilee Park without considering the exact destination within the park, violating HMR4 could mean that the SUT fails to meet the user's requirements, which may prompt this user to choose alternative map systems.

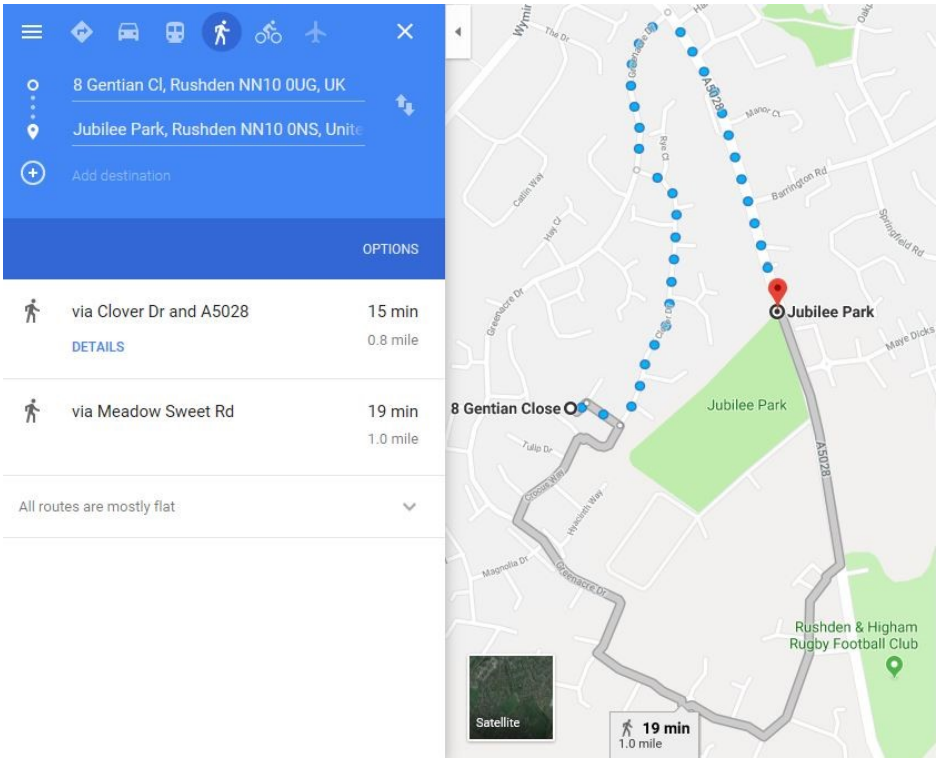


Figure 31: An HMR4 STC example for Google Maps

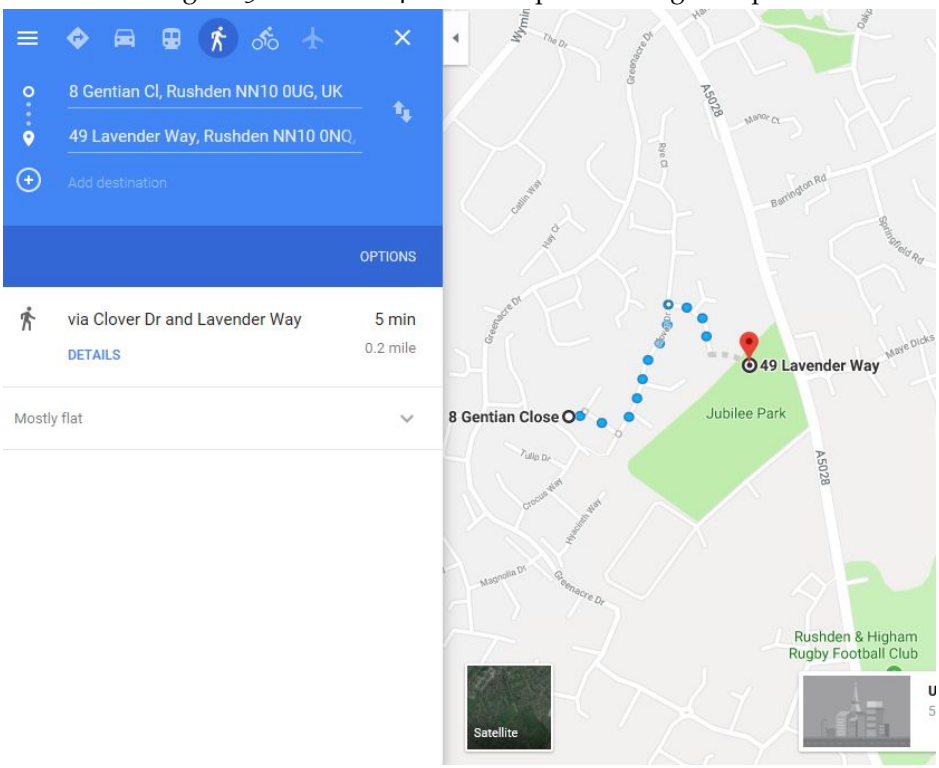


Figure 32: An HMR4 FTC example for Google Maps

6.7 A CASE STUDY OF MACHINE TRANSLATION SYSTEMS

6.7.1 *Experimental Design*

Systems Under Test: Despite the growing popularity of machine translation systems, the complexity and flexibility of natural language make it frequently impracticable to assess translation accuracy solely through traditional approaches (such as human translation): This reveals the presence of an oracle problem within machine translation systems [226]. While MT has been employed for evaluating these systems [129, 226, 264], the process of MR identification may still encounter hurdles. Given these considerations, this section presented a case study employing MRPs in conjunction with the proposed MT framework to identify MRs within machine translation systems. For this experiment, three widely-used machine translation systems were chosen: Google Translator, Microsoft Bing Translator, and Baidu Translator [264].

Reason for Applying MRT: Machine translation systems has been repeatedly proven to be effective in translating clean and in-domain texts [257]. Nonetheless, the majority of user-generated content, particularly in the social media field, is surrounded by noise such as typographical errors, idiosyncrasies, dialects, and slang. These noise types are apt to significantly reduce the precision of machine translation systems. In light of this context, this section presented a case study utilizing MRPs and the proposed MT framework to guide the identification of MRs in MRT, aiming to assess the robustness of machine translation systems.

Test Case Generation: Simple sentences were manually created as source inputs by (1) consulting English/Chinese dictionaries; and (2) iteratively repeating this procedure to generate a total of 100 valid source inputs.

6.7.2 *Relations*

In total, one MR, one HMR and eight MRRs were identified for machine translation systems following the guidance of MRPs and the proposed MT framework, as presented below.

MR3: MR3 states that a robust machine translation system should consistently produce identical or highly similar outputs for the same input, irrespective of when it is running. This MR can be easily and quickly identified based on the *Fully-Equal* MRIP (and also its super-pattern, the *Irrelevance* MRP). In particular, the irrelevant condition in this MR is when to run the machine translation system.

HMR5: If a widely-known abbreviation substitutes a word in the source input to create the follow-up input without altering the meaning of the inputs, then the follow-up should be identical or highly similar to the source output. In this study, two popular abbreviations were selected: “as soon as possible” was used to replace “ASAP”, and “Mister” was used to replace "Mr.". This HMR can be easily and quickly identified based on the *Substitution* MRIP (as well as its super-patterns, the *Similar* MRP and the *Sets* MRP). In particular, the

source input and the follow-up input in this HMR have the same meaning and only differ in one specific element: Whether or not an abbreviation is used instead of the complete word.

MRR1: If an additional blank space is appended to the end of a source input to generate the follow-up input without altering the meaning of the inputs, then the follow-up output should be identical or highly similar to the source output. In the experiment, an additional blank space was inserted between the sentence and the period. This MRR can be easily and quickly identified based on the *Addition* MRIP (as well as its super-patterns, *Similar* MRP and the *Sets* MRP). In particular, the notion of "one more component" was further defined as appending an additional blank space to the end of a sentence (i.e., between the sentence and the period) to create this MRR. In machine translation systems, it is common for users to unconsciously type an additional blank space, which may even go unnoticed. When a space appears within a word, such as when "maybe" is mistakenly typed as "may be", its meaning can change. However, when a space appears between the last word and the period in a sentence, the translation outcome should not be significantly affected.

MRR2: If removing a blank space between two sentences from the source input to construct the follow-up input without altering the meaning of the inputs, then the follow-up output should be identical or highly similar to the source output. This MRR can be easily and quickly identified based on the *Subtraction* MRIP (and also its super-patterns, *Similar* MRP and the *Sets* MRP). In particular, the notion of "one less component" was further defined as eliminating the blank space between two complete sentences, which is also a common error in machine translation systems, in order to construct this MRR.

MRR3: If the plural morpheme "-s" of a word from the source input is omitted to construct the follow-up input without altering the meaning of the inputs, then the follow-up output should be identical or highly similar to the source output. This MRR can be easily and quickly identified based on the *Substitution* MRIP (as well as its super-patterns, *Similar* MRP and the *Sets* MRP). When using machine translation systems, the misuse of the singular/plural form of words is a common error that may go unnoticed, particularly for words where the plural form is formed by simply adding the letter "s". In this context, to construct this MRR, the notion of "differ in only one component" was further defined as an erroneous usage of the singular and plural forms of a word. It is worth noting that in the experiment, in order to increase the similarity between the source and follow-up inputs, only the words whose plural form could be obtained by simply adding a letter "s" were included.

MRR4: If a punctuation mark between two complete sentences from the source input is omitted to construct the follow-up input without altering the meaning of the inputs, then the follow-up output should be identical or highly similar to the source output. This MRR can be easily and quickly identified based on the *Substitution* MRIP (as well as its super-patterns, *Similar* MRP and the *Sets* MRP). Incorrect usage of punctuation marks is common in machine translation systems, and a robust machine translation system ought to be capable of properly handling such errors. In this context, the notion of "one less

component" was further defined as eliminating a punctuation mark between two complete sentences to construct this MRR.

MRR5: If an additional identical punctuation mark is inserted between two complete sentences in the source input to construct the follow-up input without altering the meaning of the inputs, then the follow-up output should be identical or highly similar to the source output. This MRR can be easily and quickly identified based on the *Addition* MRIP (as well as its super-patterns, *Similar* MRP and the *Sets* MRP). In this context, the notion of "one more component" was further defined as adding an additional identical punctuation mark between two complete sentences.

MRR6: If a punctuation mark between two complete sentences from the source input is erroneously used to construct the follow-up input without altering the meaning of the inputs, then the follow-up output should be identical or highly similar to the source output. This MRR can be easily and quickly identified based on the *Substitution* MRIP (as well as its super-patterns, *Similar* MRP and the *Sets* MRP). In this context, the notion of "differ in only one component" was further defined as modifying the punctuation mark between two complete sentences, and in this experiment, a comma was used to replace a period.

MRR7: If the uppercase/lowercase character at the beginning of a complete sentence from the source input is erroneously used to construct the follow-up input without altering the meaning of the inputs, then the follow-up output should be identical or highly similar to the source output. This MRR can be easily and quickly identified based on the *Substitution* MRIP (as well as its super-patterns, *Similar* MRP and the *Sets* MRP). Incorrect usage of uppercase/lowercase characters in complete sentences is common in machine translation systems, and a robust machine translation system should be capable of handling such mistakes appropriately. In this context, the notion of "differ in only one component" was further defined as changing the first character of a complete sentence from uppercase to lowercase.

MRR8: If an crucial component that has an impact on the structure/meaning of the sentence is removed from the source input to construct the follow-up input, then the follow-up output should differ from the source output. This MRR can be easily and quickly identified based on the *Subtraction* MRIP (as well as its super-patterns, *Similar* MRP and the *Sets* MRP), but in a different way than previous MRRs: MRR8 requires that the outputs must be different from each other, rather than similar. Omission of words in sentences is also common in machine translation systems, particularly in lengthy texts. In particular, if a fundamental component of one complete sentence is omitted, the two sentences are likely to contain different meanings. Therefore, a robust machine translation system should be capable of producing different outputs for these two inputs. An illustrative example is the absence of a verb component in a complete sentence with a subject-verb-object structure. In this context, the notion of "one less component" was further defined as omitting one of the essential components of a complete sentence.

Table 18: Overall Statistical Results of the Experiment

		Google Translator	Baidu Translator	Microsoft Bing Translator
Number of metamorphic groups		100	100	100
Number of MRR1 violations		4	0	6
Number of MRR1 violations with different fault type	Type A	0	0	1
	Type B	3	0	5
	Type C	1	0	0
Number of MRR2 violations		11	1	1
Number of MRR2 violations with different fault type	Type A	4	1	1
	Type B	4	0	0
	Type C	3	0	0
Number of MRR3 violations		4	4	1
Number of MRR3 violations with different fault type	Type A	1	3	1
	Type B	3	1	0
	Type C	0	0	0
Number of MRR4 violations		6	51	13
Number of MRR4 violations with different fault type	Type A	2	51	10
	Type B	3	0	3
	Type C	1	0	0
Number of MRR5 violations		7	4	3
Number of MRR5 violations with different fault type	Type A	3	2	1
	Type B	3	2	2
	Type C	1	0	0
Number of MRR6 violations		6	43	8
Number of MRR6 violations with different fault type	Type A	3	42	3
	Type B	3	0	4
	Type C	0	1	1
Number of MRR7 violations		4	1	3
Number of MRR7 violations with different fault type	Type A	2	1	2
	Type B	1	0	1
	Type C	1	0	0
Number of MRR8 violations		18	24	10
Number of MRR8 violations with different fault type	Type A	0	0	0
	Type B	0	0	0
	Type C	18	24	10

6.7.3 *Evaluation and Discussion*

The MRR violation experimental results of the three machine translation systems are presented in Table 18: All eight MRRs were violated by all the SUTs, while both MR3 and HMR5 were satisfied by all the SUTs. Because all the SUTs satisfied MR3 and HMR5, only the overall statistical outcomes for the eight MRRs are provided in this section. Furthermore, to deeply explore and analyze the experimental results, this section conducted an analysis and categorization of how these MRR violations occurred, and outlined the following three types of faults:

A) Certain words/sentences in the source/follow-up input have not been translated.

B) The identical component in the source/follow-up input has been translated into varying meanings.

C) Certain words/sentences not present in the source/follow-up input have been translated.

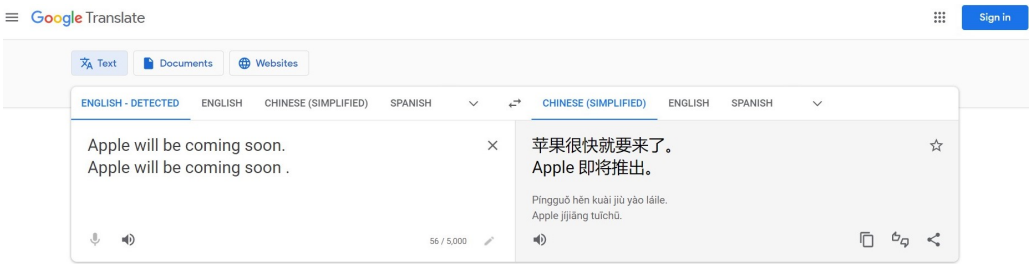


Figure 33: An MRR1 violation example for Google Translator

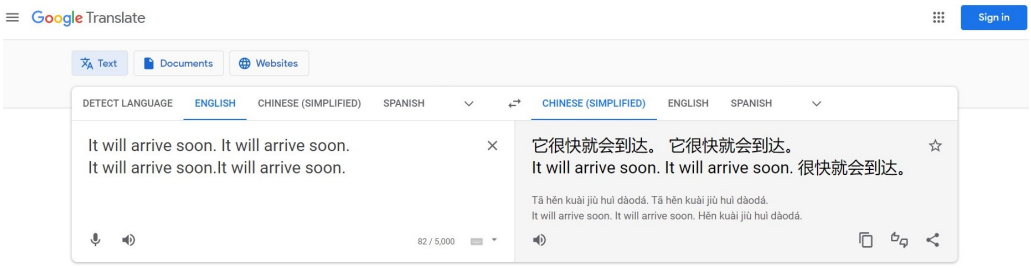


Figure 34: An MRR2 violation example for Google Translator

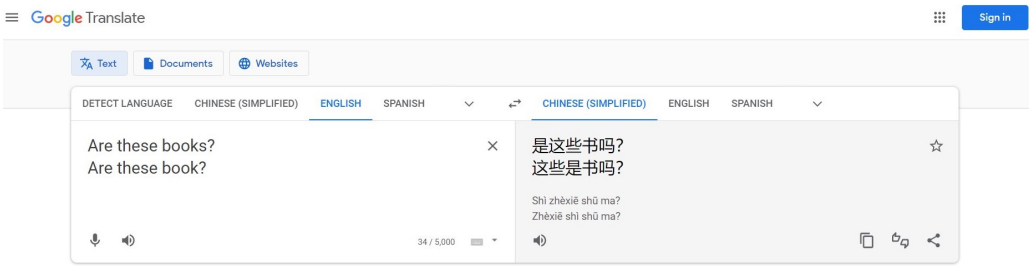


Figure 35: An MRR3 violation example for Google Translator

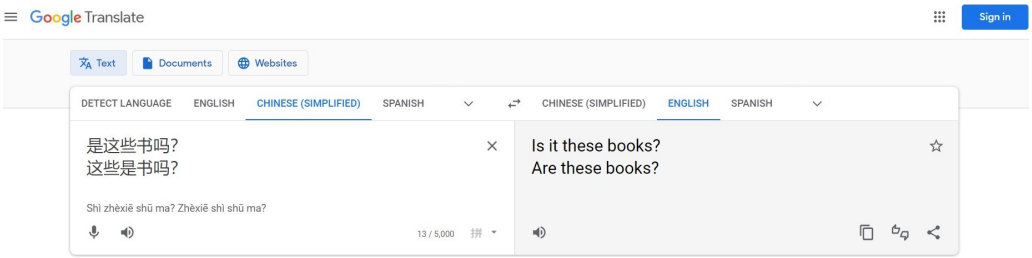


Figure 36: An MRR3 violation example for Google Translator

The experimental results revealed that MRR4 and MRR6 identified the highest number of software robustness failures, followed by MRR8, while MRR7 detected the fewest violations. MRR4 and MRR6 mainly focus on the incorrect use of punctuation marks, while MRR8 centers on the omission of important elements in complete sentences. Therefore, it can be inferred that among the various types of errors in this experiment, the three translators may excel in handling mistakes related to punctuation marks and omitted important elements. Notably, Baidu Translator exhibited notably poor performance in Type A. Typical violation examples are provided for each MRR, accompanied by discussion and analysis of the experimental results. Additionally, due to the uniformity of MRR violations across all three machine translation systems, for brevity, only the experimental results from Google Translator are presented in this chapter.

6.7.3.1 *Violations of MRR1*

Each of the three SUTs was found to violate MRR1. Fig. 33 illustrates an instance of MRR1 violations with Google Translator. In this case, an additional blank space was introduced between the word "soon" and the period in the source input to generate the follow-up input. It is notable that while the source input was accurately translated, the follow-up input was mistranslated: Google Translator failed to translate the word "Apple". Given that the two inputs carry identical meanings but the translations are different, this is definitely an MRR1 violation, revealing a failure in robustness.

6.7.3.2 *Violations of MRR2*

Each of the three SUTs was found to violate MRR2. Fig. 34 illustrates an instance of MRR2 violations with Google Translator. In this instance, a blank space between two complete sentences in the source input was eliminated to generate the follow-up input. It is notable that while the source input was accurately translated, the follow-up input was mistranslated: Google Translator duplicated the first input sentence instead of translating it and omitted the word "It" ("它") in the second input sentence. Given that the two inputs carry identical meanings but the translations are different, this is definitely an MRR2 violation, revealing a failure in robustness.

6.7.3.3 *Violations of MRR3*

Each of the three SUTs was found to violate MRR3. Figs. 35 - 36 illustrate an instance of MRR2 violations with Google Translator. In this instance, the word "books" was altered to "book" in the source input to create the follow-up input. To investigate the translations provided by Google Translator, round-trip translation was employed [7, 158]. This approach involves translating a text into another language and then back again to ascertain if the final output maintains a similar meaning to the original text. It is evident that the source input was correct but was mistranslated, while the follow-up input was incorrect (due to a missing "s") but its relevant output matched the expected source output. In other words, Google Translator ought to return “这些是书吗?” for the source input “Are these books?”. Given that the two inputs carry identical meanings but the translations are different, this is definitely an MRR3 violation, revealing a failure in robustness.

6.7.3.4 *Violations of MRR4*

Each of the three SUTs was found to violate MRR4. Figs. 37 - 38 illustrate two representative instances of MRR4 violations with Google Translator, as explained below.

- In the first scenario shown in Fig. 37, a period between two sentences in the source input was omitted to create the follow-up input. It is notable that while the source input was accurately translated, the follow-up input was mistranslated: The word "It" ("它") was omitted in the translation. Given that the two inputs carry identical mean-

Figure 37: The first MRR4 violation example for Google Translator

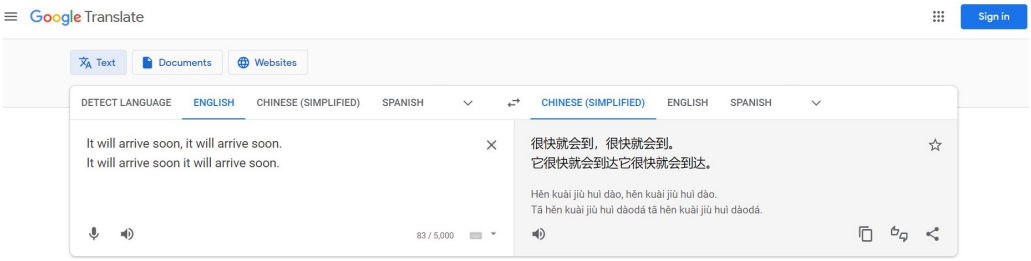


Figure 38: The second MRR4 violation example for Google Translator

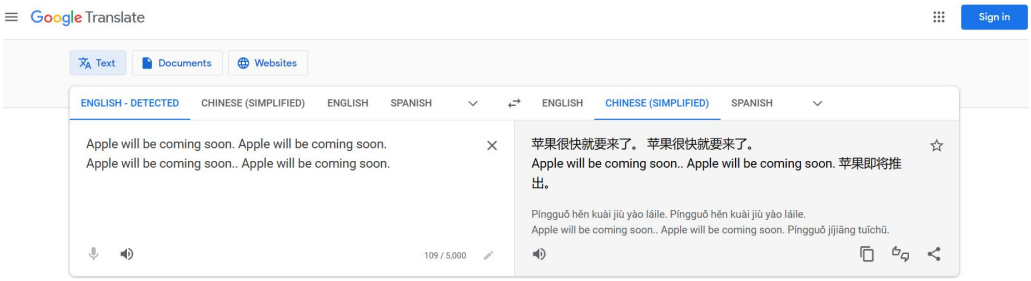


Figure 39: An MRR5 violation example for Google Translator

ings but the translations are different, this is definitely an MRR4 violation, revealing a failure in robustness.

- In the second scenario shown in Fig. 38, a comma between two sentences in the source input was removed to create the follow-up input. It is notable that the source input was a grammatically correct sentence; however, its corresponding output was incorrect due to the omission of the word "It". In the follow-up input, after the comma between the two sentences was removed, Google Translator provided an accurate translation matching the source output. In other words, Google Translator offered an incorrect translation for the correct sentence, and corrected the (plural) error and offered a correct translation for the wrong sentence. Given that the two inputs carry identical meanings but the translations are different, this is definitely an MRR4 violation, revealing a failure in robustness.

6.7.3.5 *Violations of MRR5*

Each of the three SUTs was found to violate MRR5. Fig. 39 illustrates a representative instance of MRR5 violations with Google Translator. In this scenario, the period between two complete sentences in the source input was duplicated to create the follow-up input. It is evident that while the source input was a grammatically correct sentence and was

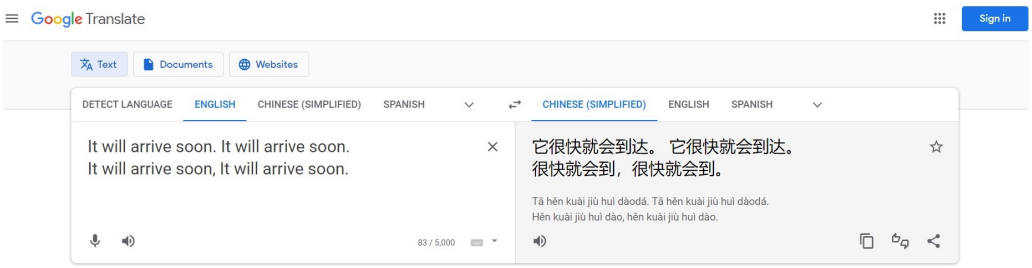


Figure 40: An MRR6 violation example for Google Translator.

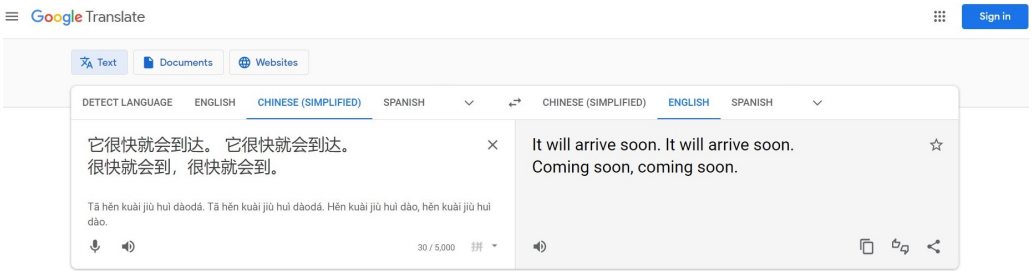


Figure 41: An MRR6 violation example for Google Translator

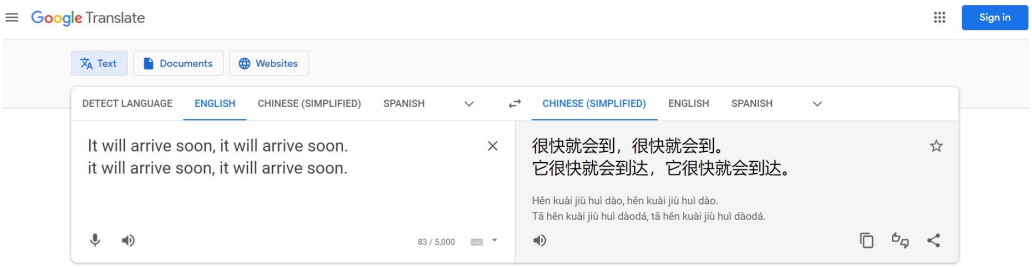


Figure 42: An MRR7 violation example for Google Translator

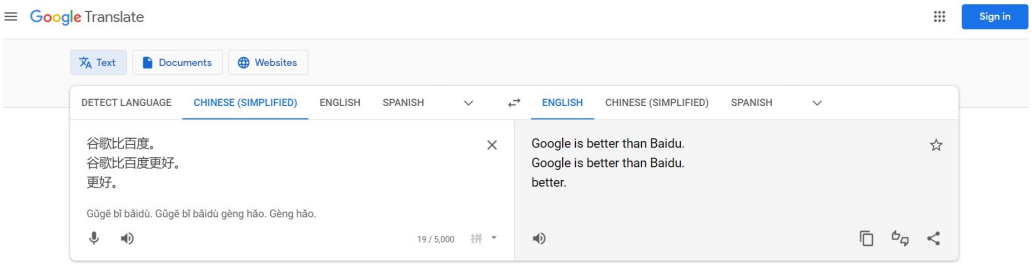


Figure 43: An MRR8 violation example for Google Translator

accurately translated, the follow-up input was mistranslated: Although the follow-up input contained only two sentences, the follow-up output comprised three sentences. It appears that Google Translator firstly copied and repeated the follow-up input, and then proceeded to translate its second sentence. Given that the two inputs carry identical meanings but the translations are different, this is definitely an MRR5 violation, revealing a failure in robustness.

6.7.3.6 *Violations of MRR6*

Each of the three SUTs was found to violate MRR6. Figs. 40 - 41 illustrate a representative instance of MRR6 violations with Google Translator. In this instance, the period in the source input was replaced with a comma to create the follow-up input. It is evident that

while the source input was a grammatically correct sentence and was accurately translated, the follow-up input was mistranslated: The word "It" is missing. Given that the two inputs carry identical meanings but the translations are different, this is definitely an MRR6 violation, revealing a failure in robustness.

6.7.3.7 *Violations of MRR7*

Each of the three SUTs was found to violate MRR7. Fig. 42 illustrates a representative instance of MRR7 violations with Google Translator. In this scenario, the word "It" in the source input was changed to "it" to create the follow-up input. It is evident that while the source input was a grammatically correct sentence, it was mistranslated: The word "It" is missing. In the follow-up input, the lowercase letter "i" in the word "it" is an error; however, its output corresponds to the expected source output. In particular, Google Translator offered an incorrect translation for the correct sentence; and corrected the lowercase letter "i" and offered a correct translation for the incorrect sentence. Given that the two inputs carry identical meanings but the translations are different, this is definitely an MRR7 violation, revealing a failure in robustness.

6.7.3.8 *Violations of MRR8*

Each of the three SUTs was found to violate MRR8. Fig. 43 illustrates an instance of MRR8 violations with Google Translator. In this instance, the characters “更好” (“better”) were removed from the source input to create the follow-up input. It is evident that while the source input was a grammatically correct sentence and was accurately translated, the follow-up input was mistranslated: A significant portion of the source input was omitted to create the follow-up input, resulting in a completely different meaning from the source input; nevertheless, their outputs remain identical. Google Translator may automatically fill in the incomplete sentences and translate them. Given that the two inputs carry identical meanings but the translations are different, this is definitely an MRR8 violation, revealing a failure in robustness.

6.8 CONCLUSION

The quality of MRs and MGs significantly impacts MT performance, yet identifying effective MRs remains a significant challenge, which demands innovative thinking, knowledge of MRPs, and a thorough understanding of the SUT [64, 246]. As an advanced MR-identification technique, MRP has repeatedly demonstrated its effectiveness in guiding MR identification [33, 170, 243, 245, 280, 295]. However, research on MRP is still in its early stages, requiring further improvement [64, 246, 280].

This chapter has introduced the concepts of sub-patterns, super-patterns, and MRP family trees to systematically define the relationships among multiple MRPs at different abstraction levels. Then this chapter has introduced two MRP trees, created by categorizing previously published MRPs as well as the proposed MRPs, to offer users an easily and

quickly means of obtaining their desired MRPs for reuse, reference, or inference. In total, three MRPs, six MRIPs, and two MROPs have been introduced, each tailored to different application fields. Additionally, this chapter has introduced a novel MT framework designed to facilitate the identification and application of MRPs. Given the current lack of research on systematically identifying new MRPs, this is the first framework that are capable of guiding not only the identification but also the application of MRPs (as well as MRIPs and MROPs). Three case studies have been presented, demonstrating the capabilities of the MRPs and the proposed MT framework to guide the identification of MRs for MT/ME/MRT. The empirical experiments were conducted using three kinds of popular big data systems:

- E-commerce Systems: Amazon and JD.com.
- Map Systems: Google Map and Baidu Map.
- Machine Translation Systems: Google Translator, Microsoft Bing Translator and Baidu Translator.

Through the experiments, multiple MR violations have been successfully detected. Experimental results indicate that (1) MRPs can effectively guide the identification of MRs not only for MT but also for ME and MRT; and (2) both the proposed MRPs and the MT framework offer users direction and guidance in identifying effective MRs and MRPs. A major limitation of the proposed MRPs and MRP trees is the absence of a systematic method for measuring the level of abstraction, which will be studied in future work.

Furthermore, MRPs may serve as valuable tools for educating and training future software quality assurance professionals [269, 285]. It has been reported that many computer science students complain that the materials they use in class and independent learning may appear too theoretical, and they prefer more practical and useful resources, reflecting the tensions in education and training methods, such as traditional higher education and vocational and professional education and training [267, 268].

The subsequent chapter of this thesis will introduce a novel MR-MG pair selection algorithm. The rationale behind the next chapter is that since this chapter has proposed MRPs to facilitate the identification of effective MRs from scratch, and Chapters 4 and 5 have proposed MG-generation algorithms to construct effective MGs from scratch, the upcoming chapter will concentrate on selecting effective MRs and MGs from existing ones. With this consideration, the next chapter will introduce a novel MR-MG pair selection algorithm that exhibit better effectiveness while maintaining high test efficiency compared to MT-RT, which is currently the most commonly-used MR and MG selection algorithm [64, 246].

7

METAMORPHIC RELATION AND GROUP SELECTION ALGORITHM

Papers delivered from this chapter (Under Review)

1. **Zhihao Ying**, Dave Towey, Anthony Bellotti, and Zhi Quan Zhou. MRGS-ART: Metamorphic Relation and Group Selection based on Adaptive Random Testing. Submitted to *Software Testing, Verification and Reliability*, 2023.

7.1 INTRODUCTION AND MOTIVATION

Although a great number of studies from various perspectives have conducted to enhance MT performance, there still exist some challenges in MT [64]. Notably, one challenge relates to the selection of MRs and MGs from existing ones [64]. Over recent years, MRs and MGs, have been recognized as the core elements of MT, have garnered significant attention within the MT community. Many techniques have been proposed to guide the identification of MRs [82, 103, 155, 156, 182, 232, 247, 261, 271, 287, 292] or the generation of MGs [8, 19, 21, 84, 120, 143, 147, 177, 238]. Therefore, a large number of MRs and MGs can be constructed for various kinds of SUTs. Additionally, it has been reported that considering all available MRs during the MT process could improve the performance of MT [67]. Given the limited resources available for software testing [246, 260], algorithms are necessary to guide the selection of effective MRs and MGs from multiple existing ones. A key question, therefore, is: How can a tester choose the "most appropriate" MR-MG pair in each round of MT for execution? Nonetheless, there remains a deficiency in metrics or algorithms to guide the selection of effective MRs and MGs for execution. Some studies aimed to address the selection issue, albeit by focusing solely on MRs [99, 253, 256] or MGs [143, 144], rather than both simultaneously.

This chapter introduces a novel MR-MG distribution metric for selecting effective MR-MG pairs from a black-box perspective: An effective MR, when paired with an MG, should aim to maximize the distance between this MG and previously-executed ones. This is the the first study to define the properties of effective MR-MG pairs. Additionally, this metric can serve as a guiding principle for the design and development of novel MR-MG pair selection algorithms. The rationale behind this metric aligns with that of the MG-generation algorithms proposed in previous chapters (Chapters 4 - 5): The performance of MT can be improved by achieving a balanced distribution of STCs and FTCs throughout their respective input domains. Previous algorithms only considered the case in which a single

MR was present. This chapter takes a step further and examine the scenario where multiple MRs are present. Furthermore, this chapter concentrates on the selection of effective MRs and MGs, as opposed to their generation. In this context, this chapter introduces a novel MR-MG pair selection algorithm, named Metamorphic Relation and Group Selection based on Adaptive Random Testing (MRGS-ART), which is inspired by the proposed metric. MRGS-ART can automatically and dynamically choose suitable MR-MG pairs from existing ones for execution. Empirical experiments were conducted to assess and compare the performance of MRGS-ART against other existing MR-MG pair selection algorithms in the literature. Through experiments, this chapter demonstrates the performance of MRGS-ART and provides guidance on how to effectively use it.

7.2 METRIC AND ALGORITHM

7.2.1 *MR-MG Distribution Metric*

ART refers to a family of software test-case generation algorithms aimed at improving the fault-detection capability of RT by evenly distributing test cases over the input domain [77, 83, 140]. In the previous chapters of this thesis, the concept of ART has been employed to design and develop MG-generation algorithms. In particular, the proposed MG-generation algorithms aim to enhance the performance of MT by evenly distributing STCs and FTCs throughout their respective input domains. This chapter shifts focus to a new perspective: The selection of effective MR-MG pairs from existing ones. Specifically, by leveraging ART principles and MT features, this chapter introduces a novel metric, termed MR-MG Distribution Metric, from a black-box perspective, to guide the selection of effective MR-MG pairs: An effective MR-MG pair is characterized by the MR maximizing the distance between the currently executing MG and previously executed MGs.

The following example illustrates the application of the proposed metric:

1. Consider two STCs and N ($N > 1$) MRs. Each MR possesses its own executed STC set and executed FTC set, as opposed to a shared executed test set among all MRs. Consequently, there are a total of N executed STC sets and N executed FTC sets. The rationale of this step is to avoid the impact of the input-domain difference problem (introduced in Chapter 4).

2. The tester should choose a distance measure based on SUT features and input types.

3. The tester should generate N FTCs based on the first STC and the N MRs. Each MR corresponds to one MG in this scenario, resulting in a total of N MGs.

4. For each MR, the tester can use the chosen distance measure to calculate the sum of:

   (a) The "distance" between the STC and the nearest executed STC.

   (b) The "distance" between the FTC and the nearest executed FTC.

5. The tester should choose the MR with the largest "distance" to form an MR-MG pair.

6. The tester should repeat the aforementioned steps for the second STC, choosing the MR with the largest “distance” to form a new MR-MG pair.

7. The tester should compare the two “distances” and choose the MR-MG pair with the largest "distance" for execution.

8. Following the execution of the MR-MG pair against the SUT, if no stopping conditions are triggered, the tester should add the executed test cases to their respective executed test sets.

Various methods can be used to quantify the "distance" between one MG and another. For instance, the distance between numerical test cases can be quantified using Euclidean distance [77, 140], while the dissimilarity between two sets can be measured using Jaccard distance [150]. This chapter assesses the "distance" by dividing the input domain and choosing test cases from vacant subdomains to ensure that the selected test cases are not very close to each other.

7.2.2 *Selection of Basic Algorithm*

A discussion and analysis of various partition-based ART algorithms was given in Section 5.2.1. As IPART is likely to encounter the boundary effect problem [57], while BART does not, this chapter only chooses BART for designing the algorithm. In particular, with IPART, the subdomains close to the boundary have fewer neighbors compared to those near the centre. Consequently, subdomains near the boundary typically have a lower probability of neighboring non-empty subdomains compared to those near the center, making them more likely to be selected for generating new test cases. On the other hand, BART does not suffer from this problem since each subdomain has an equal probability of being chosen for generating new test cases.

7.2.3 *MRGS-ART Algorithm*

In this section, based on the MR-MG distribution metric, a new MR-MG pair selection algorithm was introduced, termed Metamorphic Relation and Group Selection based on Adaptive Random Testing (MRGS-ART). This algorithm aims to enhance MT performance by selecting appropriate MR-MG pairs based on the information provided by the executed and non-MR-violating MGs. In particular, MRGS-ART was designed based on the following rationale: The fault-detection capability of MT should be improved through an even distribution of STCs and FTCs across the respective input domains for all the MRs included during the process of MT. Algorithm 6 outlines the process of applying MRGS-ART on n 1-1 MRs, accompanied by detailed explanations as follows:

- Step 1: The starting point of MRGS-ART assumes the existence of n MRs, along with either an STC generation algorithm or a repository of STC datasets. MRs can be manually or automatically identified through advanced techniques such as MRPs.

Algorithm 6: MRGS-ART for n 1-1 MRs

```
Assume the existence of n 1-1 MRs;
Determine the scope of the source and follow-up input domains according to the
  given MR;
Each MR is given one executed STC set and one executed FTC set;
Initialize the n executed STC sets and n executed FTC sets to be empty;
Randomly generate one STC for each MR;
Generate FTCs based on the MRs and the STCs;
for i = 1 → n do
    Randomly choose an MR containing empty executed STC and FTC sets;
    Execute the MG against the SUT and check whether the MR is violated;
    Add the STC and the FTC to the respective executed test sets;
end
while all stopping conditions are not satisfied do
    if (a) the quantity of executed STCs/FTCs reaches a threshold or (b) all the subdomains
      contain at least one executed STC/FTC then
        Bisect all the subdomains in the relevant source/follow-up input domain;
    end
    Assume the existence of k (k > 0) source candidates;
    for m = 1 → n do
        for i = 1 → k do
            Choose the i^th source candidate and construct the follow-up candidate
              based on the m^th MR;
            Count the sum of the quantity of source and follow-up candidates located
              within empty subdomains, represented by NG_i;
        end
        Choose the candidate MG with the largest NG_i, represented by (NG_i)_m;
        if multiple candidate MGs contain the maximum NG_i value then
            Choose one at random;
        end
    end
    Choose the MR-MG pair with the largest (NG_i)_m;
    if multiple MR-MG pairs contain the maximum (NG_i)_m value then
        Choose one at random;
    end
    Execute the MG against the SUT and check whether the MR is violated;
    Add the executed STC and FTC to their respective executed test sets;
end
```

Another premise is that all MRs used by MRGS-ART must include an identical scope of source input domains, irrespective of their categorization as 1-1 MRs, M-1 MRs, or M-N MRs. In particular, if an STC is applicable to one MR employed by MRGS-ART, then it should also be applicable to all other MRs used by MRGS-ART. Within the framework of MRGS-ART, the quantity of executed STC sets and FTC sets is determined by the type of MR. That is, a 1-1 MR has one executed STC set and one executed FTC set; an M-1 MR has M executed STC sets and one executed FTC set; and an M-N MR has M executed STC sets and N executed FTC sets. In this scenario, there are n executed STC sets and n executed FTC sets for n 1-1 MRs. The rationale

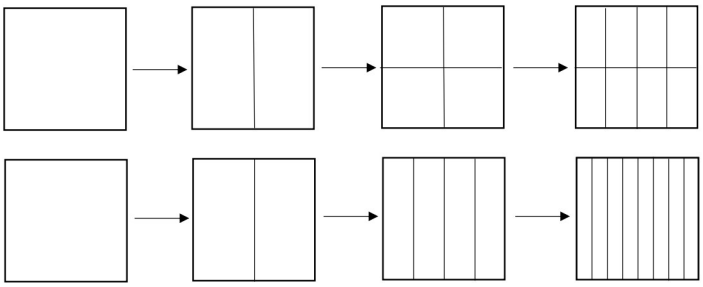


Figure 44: Examples of Bisecting Subdomains in 2D input domains

of this statement is to circumvent the input-domain difference problem (introduced in Chapter 4): If all MRs share the same executed test set, the differences between the source and follow-up input domains may be ignored, which may negatively affect the even distribution of STCs and FTCs.

- Step 2: MRGS-ART initializes all executed STC sets and executed FTC sets as empty.
- Steps 3-8: MRGS-ART gives each MR an STC in random order, followed by the generation of the corresponding FTC based on the MR and the STC. The rationale of these steps is to ensure that each MR is given at least one available MG. Then, MRGS-ART sequentially executes each MR-MG pair against the SUT and checks for violations. If no stopping condition is satisfied (i.e., no violation is identified), MRGS-ART appends each STC to the respective executed STC set and each FTC to the respective executed FTC set.
- Step 10: If either (A) the quantity of executed STCs/FTCs reaches a threshold; or (B) all source/follow-up subdomains are occupied by executed STCs/FTCs, MRGS-ART proceeds to bisect all subdomains within the respective input domain. Condition b is derived from BART. In order to ensure that the difficulty of achieving Condition 10A is similar to the difficulty of achieving Condition 10B, the threshold can be defined as the total number of subdomains. The rationale behind Condition 10A stems from the random generation of candidate MGs, which could potentially lead to a scenario where all candidate MGs reside within non-empty subdomains, particularly when the k value is low. Thus, it is necessary to consider Condition 10A to ensure the input domain will be partitioned in a timely manner. Two examples illustrating the application of bisection partitioning to 2D input domains are presented in Fig. 44.
- Step 13: MRGS-ART chooses k ($k > 0$) STCs as source candidates from an STC dataset or via a specific STC-generation algorithm, subsequently generating FTCs as follow-up candidates based on the provided MR and source candidates. It is noteworthy that MRGS-ART does not construct new MGs or MRs from scratch; rather, it chooses the "appropriate" MR-MG pair among existing ones for execution. The k value plays an important role in MRGS-ART. The rationale of using the parameter k is that because MRGS-ART chooses the "appropriate" MR-MG pair from various MRs and MGs in each MT iteration, a higher number of candidate MGs expands the selection options, which may lead to greater diversity in MG distribution. However, given that larger

k values may escalate computational overhead, in Section 7.4, empirical experiments have been conducted to explore the impact of various k values, aiming to ascertain the optimal balance between the efficiency and effectiveness of MRGS-ART.

- Steps 15-18: For the i^{th} candidate MG, MRGS-ART calculates the total number of source and follow-up candidates located within empty subdomains, represented by NG_i. The rationale behind this step is that if an MG contains a higher number of test cases (STCs/FTCs) within empty subdomains, choosing this MG may lead to a more balanced distribution of MGs across the relevant input domain.

- Step 19: Following the computation of NG_i values for all candidate MGs based on the first MR, MRGS-ART proceeds to choose the candidate MG exhibiting the highest NG_i value to get a pair of MR-MG. The associated number of valid candidates is represented by $(NG_i)_1$. The rationale behind this step is that an MG containing a greater NG_i value indicates a larger number of source and follow-up candidates situated within empty subdomains. Such an MG is likely to be further away from the previously-executed and non-MR-violating MGs.

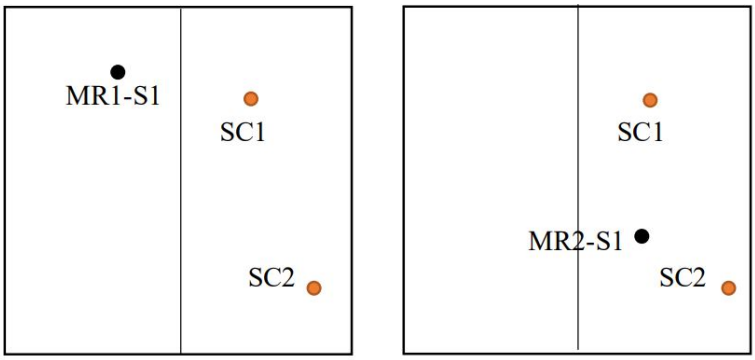
- Steps 20-22: In the event that multiple candidate MGs possess identical maximum NG_i values, MRGS-ART proceeds to randomly choose one from them.

- Step 24: At this point, MRGS-ART has completed the calculating of valid source and follow-up candidates for all candidate MGs according to the first MR. More specifically, MRGS-ART has identified the MG that is capable of achieving a more balanced distribution of MGs for the first MR from the k candidate MGs. Subsequently, MRGS-ART repeats the aforementioned steps for all other MRs, in order to get the MR-MG pair exhibiting the highest $(NG_i)_m$ value. The rationale behind this step is to assess the distribution diversity of all MR-MG pairs and choose the one demonstrating a more even distribution of STCs and FTCs.

- Steps 25-27: If multiple pairs have the same maximum $(NG_i)_m$ value, then MRGS-ART randomly chooses one from them.

- Steps 28-29: MRGS-ART proceeds to execute the selected MR-MG pair against the SUT and examine for MR violations. If no stopping condition is satisfied (i.e., no violation is identified), then MRGS-ART appends each STC to the relevant executed STC set and each FTC to the relevant executed FTC set, and returns to Step 9; alternatively, if any of the stopping conditions are met, then MRGS-ART reports the experimental results and ends the process.

It is important to emphasize that MRGS-ART with $k = 1$ differs from MT-RT. This arises from the fact that MRGS-ART computes and compares distribution diversity in two distinct rounds. The first round is within Steps 15-22 of Algorithm 6. For an MR, MRGS-ART calculates and compares the distribution diversity of all relevant candidate MGs, with the aim of choosing the one exhibiting the greatest distribution diversity to obtain a pair of MR-MG. The second round is within Steps 14-27, wherein MRGS-ART evaluates the



(a) source input domain of MR1. (b) source input domain of MR2.

Figure 45: Executed STCs Distribution (source candidates are represented by red points, and executed STCs are represented by black points)

(c) follow-up input domain of MR1. (d) follow-up input domain of MR2.

Figure 46: Executed FTCs Distribution (follow-up candidates are represented by green points, and executed FTCs are represented by black points)

distribution diversity among all MR-MG pairs, with the aim of choosing the one with the greatest distribution diversity for execution.

7.2.4 *Application of MRGS-ART*

This section will illustrate the application of MRGS-ART through a simple example. The SUT takes two inputs. Two 1-1 MRs (MR_1 and MR_2) are identified for the SUT, with k (the number of source candidates) set to 2. When every MR has been executed once using an MG and the relevant input domains have been partitioned, the distribution of the executed STCs and executed FTCs is illustrated in Figs. 45 - 46. Each MR has one executed STC and one executed FTC, denoted as black points. The red points represent source candidates (SC_1 and SC_2), and the green points represent follow-up candidates ($MR_1 - FC_1$, $MR_1 - FC_2$, $MR_2 - FC_1$ and $MR_2 - FC_2$). At the same time, only MR_1-FC_2 is located within an empty subdomain, while MR_1-FC_1 is inside a non-empty subdomain. With this consideration, MRGS-ART chooses SC_2 and MR_1-FC_2 for MR_1. MRGS-ART then repeats the same steps for MR_2, and chooses SC_2 and MR_2-FC_2. In this context, two MR-MG pairs can be obtained: (MR_1, (SC_2, MR_1-FC_2)) and (MR_2, (SC_2, MR_2-FC_2)). MRGS-ART then compares these two MR-MG pairs: SC_2 (from MR_2) is located within a non-empty subdomain. At this point, MRGS-ART chooses (MR_1, (SC_2, MR_1-SC_2)) for execution in this iteration. After

the execution of the selected MR-MG pair against the SUT, MRGS-ART appends the currently executed STC and FTC to the respective executed test sets if no stopping condition is satisfied (i.e., no MR is violated).

In addition, there are two points to note.

- Given that MRGS-ART solely involves MR-MG pair selection, integrating any existing MR-identification methods or MG-generation algorithms into the MRGS-ART framework can further enhance MT performance.
- While MRGS-ART is primarily intended for MR-MG pair selection, it also has the capability to individually choose one of them: When provided with an MG, the tester can employ MRGS-ART to choose an effective MR; in contrast, when provided with an MR, the tester can apply MRGS-ART to choose an effective MG.

7.2.5 *Advantages*

The following paragraphs offer comparisons between the proposed methodologies (MR-MG distribution metric and MRGS-ART) and previously-published work:

- From the Viewpoint of Metrics: (1) Previously-published metrics are mainly designed from the perspective of white-box testing and necessitate access to the SUT source code. In contrast, the proposed metric was designed from a black-box testing perspective, without the need for the source codes of the SUT.

  (2) Previous metrics typically require the execution of multiple MGs to acquire SUT execution behaviors/outputs for guiding the selection of a fixed number of MRs, which may increase computational overhead. In contrast, the proposed metric only focuses on the calculation of MG-distribution diversity, and enables adaptive MR selection throughout each round of MT.

  (3) Previous metrics concentrate on the properties of an effective MR, while the proposed metric centers on the properties of an effective MR-MG pair.

- From the Viewpoint of Algorithms: (1) While techniques for MR selection have been proposed, the majority either choose a fixed number of MRs or necessitate human participants [64, 246]; only a minority can autonomously and adaptively choose appropriate MRs [256]. In contrast, MRGS-ART is able to autonomously and adaptively choose an appropriate MR-MG pair in each MT iteration, without the need for human participants.

  (2) The majority of previously-published algorithms mainly concentrate on MR selection while neglecting MG selection.

  (3) Since MRGS-ART was derived from the MR-MG distribution metric, it similarly does not necessitate access to the source codes of the SUT.

  (4) MRGS-ART concentrates on the quality of MGs (including both STCs and FTCs). In particular, MRGS-ART aims to enhance MT performance by ensuring a uniform

distribution of STCs and FTCs throughout their respective input domains. Different from MRGS-ART, feedback-directed MT solely takes into account the quality of STCs (as introduced in Section 2.2.6, and it is the only algorithm that can adaptively select MR-STC pairs) [260].

(5) Since MRGS-ART solely involves selecting MR-MG pairs from existing ones, any MR-identification or MG-generation algorithm can theoretically be integrated into MRGS-ART to further improve MT performance.

7.3 RESEARCH QUESTIONS

The following RQs were formulated to provide guidance for the empirical experiments:
RQ1: Can MRGS-ART achieve a balance between test effectiveness and efficiency?

- Objective: Evaluate whether or not MRGS-ART can outperform MT-RT and achieve a better balance between test effectiveness and efficiency.
- Motivation: MT-RT is the most popular MR-MG pair selection algorithm, but it sometimes cannot achieve satisfactory fault-detection capability as it does not utilize any features of the SUT [64, 246]. In this context, MRGS-ART was proposed, in order to achieve a better balance between test effectiveness and efficiency.
- Baseline algorithms: MT-RT, selected for its widespread adoption — it is the most widely-used MR-MG pair selection algorithm.
- Methodologies:
  1. Select three performance criteria: Test efficiency, test effectiveness, and MG diversity.
  2. Design sub-RQs to delve deeper into specific aspects of performance.
  3. Determine whether or not MRGS-ART can perform better across multiple performance metrics compared to MT-RT.

- Sub-RQs:
  1. Can MRGS-ART achieve better test effectiveness (F-measure) than MT-RT?
  2. Can MRGS-ART achieve better test efficiency (generation time) than MT-RT?
  3. Can MRGS-ART achieve better distribution diversity of MGs (Dispersion and Discrepancy) than MT-RT?

RQ2: How to effectively apply MRGS-ART?

- Objective: Study the impact of different parameters on the performance of MRGS-ART, as well as the optimal values of these parameters that can achieve a better balance between testing effectiveness and efficiency.

Table 19: SUTs and MRs

Programs	Input Dimensions	Input Domains		Size (LOC)	Number of MRs	Number of Mutants	Fault Types
		From	To				
Sin	1	0.0	1000.0	120	12	3	CRP, ROR, AOR
Erf	1	0.0	1000.0	763	8	2	CRP
sncndn	2	(0.0,0.0)	(100.0,100.0)	64	8	4	CRP, ROR, RSR, AOR
BesselJ	2	(1.0,1.0)	(100.0,100.0)	1211	3	2	CRP, ROR
TriSquarePlus	3	(0.0,0.0,0.0)	(100.0,100.0,100.0)	31	11	3	CRP, ROR, AOR
rj	4	(0.0,0.0, 0.0,0.0)	(100.0,100.0, 100.0,100.0)	175	6	2	CRP, AOR
PntLinePos	6	(0.0,0.0,0.0, 0.0,0.0,0.0)	(100.0,100.0,100.0, 100.0,100.0,100.0)	23	8	1	CRP

- Motivation: Some of the parameters may have an impact on both test effectiveness and efficiency of MRGS-ART.
- Methodologies:
  1. Select the following three parameters for the experiments: (1) The number of source candidates (the value of k); (2) the number of MRs; and (3) the SUT input dimension.
  2. Explore whether or not these parameters may have an impact on the performance of MRGS-ART.
  3. Explore the optional values of these parameters, with the aim of achieving a better balance between testing effectiveness and efficiency.

7.4 EMPIRICAL EXPERIMENTS

7.4.1 *Experimental Setup*

The information of the SUTs and MRs used in the experiments are listed in Table 19. The experimental settings (the values of the parameters) for investigating RQ2 were as follows:

(1) In order to examine the influence of the number of source candidates (the value of k) on MRGS-ART performance, the value of k ranged from 1 to 5.

(2) To assess the impact of the number of MRs, different numbers of MRs were chose.

(3) To investigate the effect of the SUT input dimension, systems of varying sizes and input dimensions were selected.

A total of 10,000 trials were performed to measure the mean F-measure, F-ratio, Cohen's d, generation time, Discrepancy, and Dispersion. For calculating Discrepancy and Dispersion, MRGS-ART and MT-RT were employed to produce a total of 1000 MR-MG pairs. While theoretically, any MR-identification or MG-generation algorithm could be integrated into MRGS-ART to further improve its performance, the major objective of this empirical study is to examine the performance of the original MRGS-ART. Consequently, the MRs were derived from previously-published MRs and the MG-generation algorithm employed in MRGS-ART was MT-RT (the given source candidates were generated randomly in each round). As software testing cannot conclusively verify the absence of faults [98], and to prevent an unreasonable expenditure of time, the algorithms were terminated

Table 20: MR-violation Regions and Rates of Each MR for Sin (10,000 trials per algorithm)

Programs	Type	MR_{Sin1}	MR_{Sin2}	MR_{Sin3}	MR_{Sin4}	MR_{Sin5}	MR_{Sin6}	MR_{Sin7}	MR_{Sin8}	MR_{Sin9}	MR_{Sin10}	MR_{Sin11}	MR_{Sin12}
Sin mutant 1	MR-Violation Rate	$8.0*10^{-4}$	$1.1*10^{-3}$	$8.0*10^{-4}$	$1.6*10^{-3}$	$1.2*10^{-3}$	$8.0*10^{-4}$	$1.6*10^{-3}$	$<1*10^{-6}$	$1.1*10^{-3}$	$1.9*10^{-3}$	$1.2*10^{-3}$	$1.2*10^{-3}$
	F-measure of MT-RT	1258.53	939.91	1255.50	625.18	812.73	1243.33	632.70	>1000000	954.62	529.13	826.97	813.18
	MR-Violation Region Type	Block	Block	Block	Block	Block	Block	Block	Block	Block	Block	Block	Block
Sin mutant 2	MR-Violation Rate	$1.0*10^{-3}$	$9.8*10^{-4}$	$1.0*10^{-3}$	$1.0*10^{-3}$	$1.7*10^{-3}$	$<1*10^{-6}$	$<1*10^{-6}$	$1.3*10^{-3}$	$2.0*10^{-3}$	$1.7*10^{-3}$	$2.7*10^{-3}$	$2.7*10^{-3}$
	F-measure of MT-RT	981.81	1021.22	998.16	991.60	593.74	>1000000	>1000000	785.55	499.21	577.77	364.46	366.98
	MR-Violation Region Type	Point	Point	Point	Point	Point	Point	Point	Point	Point	Point	Point	Point
Sin mutant 3	MR-Violation Rate	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$4.0*10^{-3}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$4.0*10^{-3}$	$4.0*10^{-3}$
	F-measure of MT-RT	>1000000	>1000000	>1000000	>1000000	251.37	>1000000	>1000000	>1000000	>1000000	>1000000	249.96	247.30
	MR-Violation Region Type	NaN	NaN	NaN	NaN	Block	NaN	NaN	NaN	NaN	NaN	Block	Block

Table 21: MR-violation Regions and Rates of Each MR for Erf (10,000 trials per algorithm)

Programs	Type	MR_{Erf1}	MR_{Erf2}	MR_{Erf3}	MR_{Erf4}	MR_{Erf5}	MR_{Erf6}	MR_{Erf7}	MR_{Erf8}
Erf mutant 1	MR-Violation Rate	$9.97*10^{-3}$	$5.97*10^{-3}$	$5.04*10^{-3}$	$<1*10^{-6}$	$<1*10^{-6}$	$1.98*10^{-3}$	$1*10^{-3}$	$1*10^{-4}$
	F-measure of MT-RT	100.30	167.47	198.29	>1000000	>1000000	503.81	995.89	10004.08
	MR-Violation Region Type	Block	Block	Block	NaN	NaN	Block	Block	Block
Erf mutant 2	MR-Violation Rate	$3.97*10^{-2}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$
	F-measure of MT-RT	25.18	>1000000	>1000000	>1000000	>1000000	>1000000	>1000000	>1000000
	MR-Violation Region Type	Block	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 22: MR-violation Regions and Rates of Each MR for sncndn (10,000 trials per algorithm)

Programs	Type	MR_{sn1}	MR_{sn2}	MR_{sn3}	MR_{sn4}	MR_{sn5}	MR_{sn6}	MR_{sn7}	MR_{sn8}
sncndn mutant 1	MR-Violation Rate	$8.9*10^{-4}$	$4.0*10^{-4}$	$2.2*10^{-4}$	$1.4*10^{-4}$	$9.9*10^{-5}$	$3.9*10^{-4}$	$2.4*10^{-3}$	$3.6*10^{-3}$
	F-measure of MT-RT	1118.12	2486.41	4474.78	7032.57	10045.5	2508.91	403.29	276.47
	MR-Violation Region Type	Block	Block	Block	Block	Block	Block	Block	Block
sncndn mutant 2	MR-Violation Rate	$8.9*10^{-4}$	$8.8*10^{-4}$	$8.8*10^{-4}$	$8.8*10^{-4}$	$8.9*10^{-4}$	$8.9*10^{-4}$	$8.9*10^{-4}$	$8.9*10^{-4}$
	F-measure of MT-RT	1119.87	1130.53	1127.61	1131.13	1115.84	1116.53	1112.77	1122.54
	MR-Violation Region Type	Point	Point	Point	Point	Point	Point	Point	Point
sncndn mutant 3	MR-Violation Rate	$1.2*10^{-3}$	$5.4*10^{-4}$	$3.0*10^{-4}$	$1.9*10^{-4}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$
	F-measure of MT-RT	817.13	1852.71	3281.06	5165.17	>1000000	>1000000	>1000000	>1000000
	MR-Violation Region Type	Block	Block	Block	Block	Block	Block	Block	Block
sncndn mutant 4	MR-Violation Rate	$1.5*10^{-3}$	$1.3*10^{-3}$	$1.2*10^{-3}$	$1.2*10^{-3}$	$2.0*10^{-3}$	$1.9*10^{-3}$	$1.9*10^{-3}$	$2.0*10^{-3}$
	F-measure of MT-RT	658.12	751.49	786.96	827.53	494.14	501.49	501.67	494.44
	MR-Violation Region Type	Strip	Strip	Strip	Strip	Strip	Strip	Strip	Strip

Table 23: MR-violation Regions and Rates of Each MR for BesselJ (10,000 trials per algorithm)

Programs	Type	$MR_{BesselJ1}$	$MR_{BesselJ2}$	$MR_{BesselJ3}$
BesselJ mutant 1	MR-Violation Rate	$2.77*10^{-2}$	$<1*10^{-6}$	$<1*10^{-6}$
	F-measure of MT-RT	36.12	>1000000	>1000000
	MR-Violation Region Type	Block	NaN	NaN
BesselJ mutant 2	MR-Violation Rate	$1.23*10^{-2}$	$<1*10^{-6}$	$<1*10^{-6}$
	F-measure of MT-RT	81.02	>1000000	>1000000
	MR-Violation Region Type	Block	NaN	NaN

Table 24: MR-violation Regions and Rates of Each MR for TriSquarePlus (10,000 trials per algorithm)

Programs	Type	MR_{Tri1}	MR_{Tri2}	MR_{Tri3}	MR_{Tri4}	MR_{Tri5}	MR_{Tri6}	MR_{Tri7}	MR_{Tri8}	MR_{Tri9}	MR_{Tri10}	MR_{Tri11}
TriSquarePlus mutant 1	MR-Violation Rate	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$1.9*10^{-3}$	$5.7*10^{-4}$	$2.4*10^{-4}$	$1.2*10^{-4}$	$7.2*10^{-5}$	$4.5*10^{-5}$	$3.0*10^{-5}$
	F-measure of MT-RT	>1000000	>1000000	>1000000	>1000000	520.24	1744.09	4037.96	7995.4	13754.2	21953.7	32915.3
	MR-Violation Region Type	NaN	NaN	NaN	NaN	Block	Block	Block	Block	Block	Block	Block
TriSquarePlus mutant 2	MR-Violation Rate	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$9.9*10^{-4}$	$1.0*10^{-3}$	$1.1*10^{-3}$	$1.1*10^{-3}$	$1.1*10^{-3}$	$1.1*10^{-3}$	$1.1*10^{-3}$
	F-measure of MT-RT	>1000000	>1000000	>1000000	>1000000	1008.27	917.35	896.12	889.05	884.19	882.38	881.29
	MR-Violation Region Type	NaN	NaN	NaN	NaN	Block	Block	Block	Block	Block	Block	Block
TriSquarePlus mutant 3	MR-Violation Rate	$7.8*10^{-4}$	$<1*10^{-6}$	$7.8*10^{-4}$	$7.8*10^{-4}$	$2.9*10^{-4}$	$3.5*10^{-4}$	$3.6*10^{-4}$	$3.7*10^{-4}$	$3.8*10^{-4}$	$3.8*10^{-4}$	$3.8*10^{-4}$
	F-measure of MT-RT	1273.66	>1000000	1274.53	1271.81	3360.01	2842.48	2720.29	2657.63	2623.42	2607.16	2592.07
	MR-Violation Region Type	Strip	NaN	Strip	Strip	Strip	Strip	Strip	Strip	Strip	Strip	Strip

after generating and executing 1,000,000 MGs. As a consequence, the values of Cohen's *d* for these cases were not calculated, represented by NaN (Not a Number).

7.4.2 *Experimental Results and Discussion*

This section explores and analyzes the experimental results, discusses the findings, and answers the RQs from Section 7.3.

7.4.2.1 *F-ratio and Cohen's* d

In order to comprehensively explore and compare the performance of MRGS-ART and MT-RT, experiments were conducted to compute the MR-violation regions and rates for each MR and SUT, as illustrated in Tables 20-26: F_{MT-RT} denotes the F-measure of MT-RT, indicating the number of MGs required to detect the first MR violation using MT-RT. Table 27 presents the experimental results of F-ratio, while Table 28 presents the experimental results of Cohen's *d*: ◯ indicates the presence of the specific type of MRVR in the given mutant and MR, while × indicates its absence. It is important to note that abbreviations have been used in the MR column to denote the MRs employed. For example, MR_{Sin1}-MR_{Sin3} means that the following three MRs were included: MR_{Sin1}, MR_{Sin2}, and MR_{Sin3}.

Table 25: MR-violation Regions and Rates of Each MR for rj (10,000 trials per algorithm)

Programs	Type	MR_{rj1}	MR_{rj2}	MR_{rj3}	MR_{rj4}	MR_{rj5}	MR_{rj6}
rj mutant 1	MR-Violation Rate	$7.87*10^{-3}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$
	F-measure of MT-RT	126.94	>1000000	>1000000	>1000000	>1000000	>1000000
	MR-Violation Region Type	Block	NaN	NaN	NaN	NaN	Block
rj mutant 2	MR-Violation Rate	$5.95*10^{-3}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$
	F-measure of MT-RT	167.88	>1000000	>1000000	>1000000	>1000000	>1000000
	MR-Violation Region Type	Strip	NaN	NaN	NaN	NaN	NaN

Table 26: MR-Violation Regions and Rates of Each MR for PntLinePos (10,000 trials per algorithm)

Programs	Type	MR_{Pnt1}	MR_{Pnt2}	MR_{Pnt3}	MR_{Pnt4}	MR_{Pnt5}	MR_{Pnt6}	MR_{Pnt7}	MR_{Pnt8}
PntLinePos mutant 1	MR-Violation Rate	$4.1*10^{-3}$	$3.6*10^{-4}$	$6.4*10^{-5}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$	$<1*10^{-6}$
	F-measure of MT-RT	243.91	2794.2	15619.8	>1000000	>1000000	>1000000	>1000000	>1000000
	MR-Violation Region Type	Block	Block	Block	NaN	NaN	NaN	NaN	NaN

The following observations can be made from the viewpoint of k values:

- *When* $k = 1$: With block MRVRs, MRGS-ART with k=2 was capable of performing at least similarly to, and sometimes significantly better than, MT-RT; while with point/strip MRVRs, the two algorithms typically performed similar to each other.
- *When* $k = 2$: With block MRVRs, the performance of MRGS-ART with k=2 was significantly better than that of MT-RT and MRGS-ART with k=1; while with point/strip MRVRs, both algorithms generally performed similarly.
- *When* $k = 3$: With block MRVRs, the performance of MRGS-ART continues to improve as the k value increases from 2 to 3; while with point/strip MRVRs, the two algorithms typically performed similar to each other.
- *When* $k \geq 4$: When the k value increases from 3 to 4, the performance of MRGS-ART slightly improves or remains similar with all three kinds of MRVRs; and when $k \geq 4$, its performance remains relatively stable.
- According to the Cohen's d experimental results, it can be observed that among the cases with block MRVRs, about 15% of the results have values greater than 0.5 (which means that there are medium effect size strengths), and the largest value is 0.659; about 50% of the results have values less than 0.5 and greater than 0.2 (which means that there are small effect size strengths); and about 35% of the results have values between 0.01 and 0.20 (which means that there are very small effect size strengths). These results indicate that, under block MRVRs, MRGS-ART can always perform better than MT-RT, and sometimes even significantly better than MT-RT. However, there is still room for improvement in the effectiveness of MRGS-ART. With this consideration, Section 7.5.1 introduced MRGS-ART+, which is an improved version of the original MRGS-ART.

Table 27: Mean F-ratio Experimental Results (10,000 trials per algorithm)

Programs	MRs	Existence of							
		Block MRVR	Point MRVR	Strip MRVR	k=1	k=2	k=3	k=4	k=5
Sin mutant 1	MR_{Sin1}-MR_{Sin3}	◯	×	×	99.37%	88.45%	85.27%	83.50%	82.78%
	MR_{Sin1}-MR_{Sin7}	◯	×	×	97.66%	90.64%	88.97%	88.88%	88.34%
	MR_{Sin1}-MR_{Sin12}	◯	×	×	97.57%	89.51%	88.12%	88.01%	87.86%
Sin mutant 2	MR_{Sin1}-MR_{Sin3}	×	◯	×	99.74%	99.03%	100.81%	99.71%	99.82%
	MR_{Sin1}-MR_{Sin7}	×	◯	×	100.27%	99.21%	99.38%	100.44%	99.48%
	MR_{Sin1}-MR_{Sin12}	×	◯	×	100.72%	100.69%	99.35%	100.72%	101.43%
Sin mutant 3	MR_{Sin1}-MR_{Sin2}, MR_{Sin5}	◯	×	×	73.37%	60.30%	56.22%	54.02%	53.44%
	MR_{Sin5}, MR_{Sin11}-MR_{Sin12}	◯	×	×	100.00%	83.48%	80.63%	76.92%	75.77%
	MR_{Sin1}-MR_{Sin7}	◯	×	×	63.08%	54.61%	52.50%	51.50%	51.10%
	MR_{Sin1}-MR_{Sin5}, MR_{Sin11}-MR_{Sin12}	◯	×	×	86.42%	77.33%	75.56%	74.67%	73.55%
	MR_{Sin1}-MR_{Sin12}	◯	×	×	82.50%	76.45%	75.30%	74.81%	75.63%
Erf mutant 1	MR_{Erf1}-MR_{Erf3}	◯	×	×	96.60%	81.05%	76.36%	75.20%	73.20%
	MR_{Erf1}-MR_{Erf5}	◯	×	×	88.04%	79.24%	73.56%	74.61%	74.54%
	MR_{Erf1}-MR_{Erf8}	◯	×	×	86.35%	78.43%	76.78%	77.69%	75.68%
Erf mutant 2	MR_{Erf1}-MR_{Erf3}	◯	×	×	76.05%	63.95%	60.60%	58.95%	57.44%
	MR_{Erf1}-MR_{Erf5}	◯	×	×	69.31%	60.44%	57.18%	55.64%	55.23%
	MR_{Erf1}-MR_{Erf8}	◯	×	×	65.09%	57.93%	56.94%	55.79%	56.45%
sncndn mutant 1	MR_{sn1}-MR_{sn3}	◯	×	×	84.29%	76.22%	73.92%	71.95%	71.82%
	MR_{sn1}-MR_{sn5}	◯	×	×	81.16%	74.56%	72.91%	73.21%	73.14%
	MR_{sn1}-MR_{sn8}	◯	×	×	89.07%	85.90%	85.60%	85.06%	84.89%
sncndn mutant 2	MR_{sn1}-MR_{sn3}	×	◯	×	100.07%	99.66%	99.12%	98.91%	98.38%
	MR_{sn1}-MR_{sn5}	×	◯	×	99.96%	97.96%	98.77%	99.44%	97.15%
	MR_{sn1}-MR_{sn8}	×	◯	×	99.27%	100.96%	99.80%	98.66%	99.80%
sncndn mutant 3	MR_{sn1}-MR_{sn3}	◯	×	×	87.47%	80.00%	78.56%	78.22%	77.81%
	MR_{sn1}-MR_{sn5}	◯	×	×	84.67%	80.09%	79.51%	78.27%	78.01%
	MR_{sn1}-MR_{sn8}	◯	×	×	83.72%	80.12%	78.92%	78.69%	78.24%
sncndn mutant 4	MR_{sn1}-MR_{sn3}	×	×	◯	101.23%	99.95%	100.85%	101.45%	99.30%
	MR_{sn1}-MR_{sn5}	×	×	◯	98.70%	99.57%	99.43%	98.83%	97.94%
	MR_{sn1}-MR_{sn8}	×	×	◯	100.16%	101.11%	101.85%	99.26%	99.56%
BesselJ mutant 1	$MR_{BesselJ1}$-$MR_{BesselJ3}$	◯	×	×	78.52%	68.43%	65.80%	65.08%	63.38%
BesselJ mutant 2	$MR_{BesselJ1}$-$MR_{BesselJ3}$	◯	×	×	85.74%	76.54%	74.84%	74.99%	73.23%
TriSquarePlus mutant 1	MR_{Tri1}-MR_{Tri3}	◯	×	×	79.87%	68.71%	65.93%	64.03%	63.36%
	MR_{Tri1}-MR_{Tri7}	◯	×	×	72.62%	65.81%	63.74%	63.28%	62.70%
	MR_{Tri1}-MR_{Tri11}	◯	×	×	71.09%	66.41%	65.66%	65.57%	65.14%
TriSquarePlus mutant 2	MR_{Tri1}-MR_{Tri3}	◯	×	×	99.49%	90.31%	87.98%	87.32%	85.24%
	MR_{Tri1}-MR_{Tri7}	◯	×	×	97.70%	91.64%	89.98%	87.44%	88.44%
	MR_{Tri1}-MR_{Tri11}	◯	×	×	97.48%	91.73%	89.73%	89.02%	88.49%
TriSquarePlus mutant 3	MR_{Tri1}-MR_{Tri3}	×	×	◯	100.31%	100.09%	100.75%	99.29%	99.00%
	MR_{Tri1}-MR_{Tri7}	×	×	◯	99.06%	100.08%	98.86%	99.55%	98.43%
	MR_{Tri1}-MR_{Tri11}	×	×	◯	100.20%	98.90%	100.73%	98.23%	100.25%
rj mutant 1	MR_{rj1}-MR_{rj3}	◯	×	×	89.87%	80.15%	76.95%	76.91%	75.34%
	MR_{rj1}-MR_{rj6}	◯	×	×	83.33%	76.86%	75.68%	75.04%	74.98%
rj mutant 2	MR_{rj1}-MR_{rj3}	×	×	◯	99.41%	96.40%	97.48%	96.83%	97.49%
	MR_{rj1}-MR_{rj6}	×	×	◯	100.96%	98.08%	97.55%	98.14%	97.85%
PntLinePos mutant 1	MR_{Pnt1}-MR_{Pnt3}	◯	×	×	95.07%	91.36%	90.87%	90.45%	90.19%
	MR_{Pnt1}-MR_{Pnt5}	◯	×	×	92.89%	90.80%	90.08%	88.54%	88.87%
	MR_{Pnt1}-MR_{Pnt8}	◯	×	×	93.46%	88.99%	89.52%	88.74%	88.18%

Table 28: Cohen's *d* Experimental Results (10,000 trials per algorithm)

Programs	MRs	Existence of			MRGS-ART vs MT-RT				
		Block MRVR	Point MRVR	Strip MRVR	k=1	k=2	k=3	k=4	k=5
Sin mutant 1	MR_{Sin1}-MR_{Sin3}	◯	×	×	0.006	0.127	0.167	0.189	0.199
	MR_{Sin1}-MR_{Sin7}	◯	×	×	0.024	0.102	0.120	0.122	0.128
	MR_{Sin1}-MR_{Sin12}	◯	×	×	0.005	0.087	0.130	0.132	0.134
Sin mutant 2	MR_{Sin1}-MR_{Sin3}	×	◯	×	0.006	0.018	0.002	0.007	0.006
	MR_{Sin1}-MR_{Sin7}	×	◯	×	-0.003	0.008	0.006	-0.004	0.005
	MR_{Sin1}-MR_{Sin12}	×	◯	×	0.008	-0.013	0.006	-0.007	-0.006
Sin mutant 3	MR_{Sin1}-MR_{Sin2}, MR_{Sin5}	◯	×	×	0.319	0.518	0.586	0.622	0.631
	MR_{Sin5}, MR_{Sin11}-MR_{Sin12}	◯	×	×	0.002	0.190	0.229	0.279	0.296
	MR_{Sin1}-MR_{Sin7}	◯	×	×	0.465	0.602	0.636	0.652	0.659
	MR_{Sin1}-MR_{Sin5}, MR_{Sin11}-MR_{Sin12}	◯	×	×	0.152	0.271	0.296	0.308	0.323
	MR_{Sin1}-MR_{Sin12}	◯	×	×	0.200	0.28	0.295	0.301	0.291
Erf mutant 1	MR_{Erf1}-MR_{Erf3}	◯	×	×	0.035	0.222	0.285	0.302	0.33
	MR_{Erf1}-MR_{Erf5}	◯	×	×	0.129	0.242	0.315	0.305	0.306
	MR_{Erf1}-MR_{Erf7}	◯	×	×	0.18	0.313	0.344	0.33	0.364
Erf mutant 2	MR_{Erf1}-MR_{Erf3}	◯	×	×	0.286	0.463	0.518	0.544	0.568
	MR_{Erf1}-MR_{Erf5}	◯	×	×	0.376	0.513	0.562	0.587	0.594
	MR_{Erf1}-MR_{Erf7}	◯	×	×	0.438	0.55	0.568	0.586	0.577
sncndn mutant 1	MR_{sn1}-MR_{sn3}	◯	×	×	0.175	0.283	0.315	0.343	0.348
	MR_{sn1}-MR_{sn5}	◯	×	×	0.216	0.305	0.329	0.327	0.329
	MR_{sn1}-MR_{sn8}	◯	×	×	0.12	0.156	0.161	0.167	0.168
sncndn mutant 2	MR_{sn1}-MR_{sn3}	×	◯	×	-0.001	0.003	0.009	0.011	0.016
	MR_{sn1}-MR_{sn5}	×	◯	×	0	0.021	0.012	0.006	0.029
	MR_{sn1}-MR_{sn8}	×	◯	×	0.007	-0.010	0.002	0.014	0.002
sncndn mutant 3	MR_{sn1}-MR_{sn3}	◯	×	×	0.139	0.233	0.255	0.259	0.265
	MR_{sn1}-MR_{sn5}	◯	×	×	0.171	0.231	0.240	0.255	0.260
	MR_{sn1}-MR_{sn8}	◯	×	×	0.184	0.229	0.245	0.250	0.254
sncndn mutant 4	MR_{sn1}-MR_{sn3}	×	×	◯	-0.012	0.001	-0.008	-0.014	0.007
	MR_{sn1}-MR_{sn5}	×	×	◯	0.013	0.004	0.006	0.012	0.021
	MR_{sn1}-MR_{sn8}	×	×	◯	-0.002	-0.011	-0.018	0.007	0.004
BesselJ mutant 1	$MR_{BesselJ1}$-$MR_{BesselJ3}$	◯	×	×	0.247	0.384	0.426	0.437	0.461
BesselJ mutant 2	$MR_{BesselJ1}$-$MR_{BesselJ3}$	◯	×	×	0.158	0.275	0.298	0.299	0.32
TriSquarePlus mutant 1	MR_{Tri1}-MR_{Tri3}	◯	×	×	0.236	0.397	0.444	0.475	0.486
	MR_{Tri1}-MR_{Tri7}	◯	×	×	0.329	0.432	0.464	0.472	0.48
	MR_{Tri1}-MR_{Tri11}	◯	×	×	0.355	0.428	0.441	0.443	0.449
TriSquarePlus mutant 2	MR_{Tri1}-MR_{Tri3}	◯	×	×	0.005	0.106	0.132	0.14	0.164
	MR_{Tri1}-MR_{Tri7}	◯	×	×	0.021	0.088	0.106	0.137	0.124
	MR_{Tri1}-MR_{Tri11}	◯	×	×	0.026	0.087	0.109	0.118	0.124
TriSquarePlus mutant 3	MR_{Tri1}-MR_{Tri3}	×	×	◯	-0.003	-0.001	0.027	0.007	0.010
	MR_{Tri1}-MR_{Tri7}	×	×	◯	0.010	-0.001	0.012	0.005	0.016
	MR_{Tri1}-MR_{Tri11}	×	×	◯	-0.002	0.011	-0.007	0.018	-0.002
rj mutant 1	MR_{rj1}-MR_{rj3}	◯	×	×	0.109	0.227	0.272	0.274	0.292
	MR_{rj1}-MR_{rj6}	◯	×	×	0.188	0.275	0.291	0.301	0.3
rj mutant 2	MR_{rj1}-MR_{rj3}	×	×	◯	0.006	0.036	0.025	0.032	0.026
	MR_{rj1}-MR_{rj6}	×	×	◯	-0.01	0.019	0.025	0.019	0.021
PntLinePos mutant 1	MR_{Pnt1}-MR_{Pnt3}	◯	×	×	0.082	0.122	0.127	0.132	0.136
	MR_{Pnt1}-MR_{Pnt5}	◯	×	×	0.093	0.116	0.124	0.142	0.138
	MR_{Pnt1}-MR_{Pnt8}	◯	×	×	0.068	0.118	0.111	0.120	0.126

Table 29: Mean Time (in seconds) Experimental Results (10,000 trials per algorithm)

Programs	MRs	MT-RT	MRGS-ART				
			k=1	k=2	k=3	k=4	k=5
Sin	MR_{Sin1}-MR_{Sin3}	0.004s	0.034s	0.037s	0.040s	0.044s	0.047s
	MR_{Sin1}-MR_{Sin7}		0.055s	0.061s	0.065s	0.071s	0.076s
	MR_{Sin1}-MR_{Sin12}		0.098s	0.104s	0.115s	0.125s	0.135s
Erf	MR_{Erf1}-MR_{Erf3}	0.004s	0.034s	0.037s	0.041s	0.044s	0.048s
	MR_{Erf1}-MR_{Erf5}		0.045s	0.049s	0.053s	0.058s	0.062s
	MR_{Erf1}-MR_{Erf8}		0.061s	0.067s	0.071s	0.078s	0.083s
sncndn	MR_{sn1}-MR_{sn3}	0.005s	0.034s	0.039s	0.044s	0.049s	0.054s
	MR_{sn1}-MR_{sn5}		0.039s	0.046s	0.051s	0.058s	0.065s
	MR_{sn1}-MR_{sn8}		0.054s	0.061s	0.070s	0.078s	0.087s
BesselJ	$MR_{BesselJ1}$-$MR_{BesselJ3}$	0.005s	0.034s	0.039s	0.043s	0.048s	0.055s
TriSquarePlus	MR_{Tri1}-MR_{Tri3}	0.006s	0.033s	0.040s	0.047s	0.056s	0.062s
	MR_{Tri1}-MR_{Tri7}		0.056s	0.067s	0.080s	0.092s	0.103s
	MR_{Tri1}-MR_{Tri11}		0.076s	0.091s	0.108s	0.127s	0.143s
rj	MR_{rj1}-MR_{rj3}	0.007s	0.025s	0.034s	0.041s	0.048s	0.055s
	MR_{rj1}-MR_{rj6}		0.034s	0.044s	0.054s	0.065s	0.075s
PntLinePos	MR_{Pnt1}-MR_{Pnt3}	0.013s	0.048s	0.062s	0.075s	0.090s	0.106s
	MR_{Pnt1}-MR_{Pnt5}		0.057s	0.077s	0.095s	0.114s	0.132s
	MR_{Pnt1}-MR_{Pnt8}		0.077s	0.104s	0.138s	0.153s	0.178s

- From these observations, the following conclusions can be inferred: With block MRVRs, the fault-detection capability of MRGS-ART generally increases as the number of source candidates(the k value) increases, stabilizing within a certain range after reaching a threshold; while with point/strip MRVRs, the number of source candidates typically does not have an impact on the fault-detection capability of MRGS-ART.

- An observation can be obtained that with block MRVRs, regardless of the number of MRs used, the fault-detection capability of MRGS-ART firstly improves with the number of source candidates and then stabilizes within a certain range after reaching a threshold; meanwhile, with point/strip MRVRs, the number of source candidates typically does not have an impact on the fault-detection capability of MRGS-ART. In addition, a conclusion can be inferred that with block MRVRs, regardless of the number of MRs used, MRGS-ART was able to perform at least no worse, and sometimes significantly better, than MT-RT.

- Considering that different MRs for an SUT are likely to have various MRVRs and MR-violation rates, directly comparing the performance of MRGS-ART by solely varying the numbers of MRs may not be fair. With this consideration, in this empirical study, experiments were not conducted to validate the effect of varying numbers of MRs on MRGS-ART performance.

- The F-measure was used to quantify the fault-detection capability of MRGS-ART. However, the F-measure only focuses on the detection of the first failure. In the future work, the F2-measure may be used to further evaluate the performance of MRGS-ART for detecting multiple software failures. The F2-measure represents the

Table 30: Mean Dispersion (Max-Min) Experimental Results (10,000 trials per algorithm)

Programs	MRs	RT	MRGS-ART				
			k=1	k=2	k=3	k=4	k=5
Sin	MR_{Sin1}-MR_{Sin3}	1.3083	1.0064	0.8096	0.7484	0.7200	0.7037
	MR_{Sin1}-MR_{Sin2}, MR_{Sin5}	2.4303	1.8701	1.5042	1.3887	1.3371	1.3064
	MR_{Sin5}, MR_{Sin11}-MR_{Sin12}	2.9735	2.2875	1.8406	1.7019	1.6378	1.6012
	MR_{Sin1}-MR_{Sin7}	7.2047	4.9276	4.1546	3.9195	3.7814	3.7058
	MR_{Sin1}-MR_{Sin5}, MR_{Sin11}-MR_{Sin12}	10.3602	7.0830	5.9722	5.6285	5.4357	5.3267
	MR_{Sin1}-MR_{Sin12}	6.6527	4.2969	3.7550	3.6425	3.5962	3.5776
Erf	MR_{Erf1}-MR_{Erf3}	1.3082	1.0065	0.8096	0.7484	0.7200	0.7037
	MR_{Erf1}-MR_{Erf5}	2.0603	1.4289	1.1598	1.0713	1.0304	0.9994
	MR_{Erf1}-MR_{Erf8}	24.9000	16.7000	14.0005	13.0019	12.5022	12.1318
sncndn	MR_{sn1}-MR_{sn3}	6.2075	5.6827	5.2959	5.1723	5.1144	5.0835
	MR_{sn1}-MR_{sn5}	9.6650	8.4786	8.0795	7.9091	7.8284	7.8050
	MR_{sn1}-MR_{sn8}	16.0814	14.6660	14.0682	13.9585	13.9327	13.8855
BesselJ	$MR_{BesselJ1}$-$MR_{BesselJ3}$	3.1119	2.8389	2.6491	2.5885	2.5573	2.5438
TriSquarePlus	MR_{Tri1}-MR_{Tri3}	17.5900	11.3675	9.9335	9.5332	9.4116	9.3623
	MR_{Tri1}-MR_{Tri7}	17.7757	16.5473	16.0524	15.6877	15.6600	15.6221
	MR_{Tri1}-MR_{Tri11}	32.1005	29.5955	28.5023	28.0895	28.0450	27.9895
rj	MR_{rj1}-MR_{rj3}	23.7159	22.7200	22.1583	21.8950	21.7450	21.6917
	MR_{rj1}-MR_{rj6}	23.4650	22.6233	22.0592	21.9267	21.8575	21.7917
PntLinePos	MR_{Pnt1}-MR_{Pnt3}	81.9418	62.2007	61.2372	60.2981	60.1115	60.1009
	MR_{Pnt1}-MR_{Pnt5}	81.4503	52.6431	51.9593	51.0566	50.8129	50.8101
	MR_{Pnt1}-MR_{Pnt8}	75.2119	47.3819	46.798	46.2186	46.1252	46.1264

Table 31: Mean Discrepancy (Max-Min) Experimental Results (10,000 trials per algorithm)

Programs	MRs	RT	MRGS-ART				
			k=1	k=2	k=3	k=4	k=5
Sin	MR_{Sin1}-MR_{Sin3}	0.0021	0.0017	0.0011	0.0009	0.0008	0.0008
	MR_{Sin1}-MR_{Sin2}, MR_{Sin5}	0.0021	0.0017	0.0011	0.0009	0.0008	0.0008
	MR_{Sin5}, MR_{Sin11}-MR_{Sin12}	0.0021	0.0017	0.0011	0.0009	0.0008	0.0008
	MR_{Sin1}-MR_{Sin7}	0.0033	0.0022	0.0015	0.0013	0.0012	0.0011
	MR_{Sin1}-MR_{Sin5}, MR_{Sin11}-MR_{Sin12}	0.0033	0.0022	0.0015	0.0013	0.0012	0.0011
	MR_{Sin1}-MR_{Sin12}	0.0049	0.0031	0.0021	0.0017	0.0015	0.0014
Erf	MR_{Erf1}-MR_{Erf3}	0.0021	0.0017	0.0011	0.0009	0.0008	0.0008
	MR_{Erf1}-MR_{Erf5}	0.0029	0.0023	0.0016	0.0013	0.0012	0.0011
	MR_{Erf1}-MR_{Erf8}	0.0037	0.0025	0.0018	0.0016	0.0015	0.0014
sncndn	MR_{sn1}-MR_{sn3}	0.0021	0.0017	0.0013	0.0012	0.0012	0.0011
	MR_{sn1}-MR_{sn5}	0.0037	0.0028	0.0024	0.0023	0.0023	0.0023
	MR_{sn1}-MR_{sn8}	0.0040	0.0031	0.0027	0.0026	0.0026	0.0026
BesselJ	$MR_{BesselJ1}$-$MR_{BesselJ3}$	0.0021	0.0017	0.0013	0.0012	0.0012	0.0011
TriSquarePlus	MR_{Tri1}-MR_{Tri3}	0.0021	0.0018	0.0015	0.0015	0.0014	0.0014
	MR_{Tri1}-MR_{Tri7}	0.0033	0.0027	0.0025	0.0025	0.0025	0.0024
	MR_{Tri1}-MR_{Tri11}	0.0047	0.0038	0.0035	0.0035	0.0035	0.0035
rj	MR_{rj1}-MR_{rj3}	0.0021	0.0019	0.0018	0.0018	0.0018	0.0018
	MR_{rj1}-MR_{rj6}	0.0031	0.0028	0.0026	0.0026	0.0026	0.0026
PntLinePos	MR_{Pnt1}-MR_{Pnt3}	0.4993	0.002	0.0018	0.0018	0.0018	0.0018
	MR_{Pnt1}-MR_{Pnt5}	0.7991	0.0028	0.0027	0.0026	0.0026	0.0026
	MR_{Pnt1}-MR_{Pnt8}	0.7989	0.0034	0.0033	0.0033	0.0033	0.0033

number of additional MGs needed to discover the second failure after the detection of the first failure. In particular, the generalization of the F2-measure can be used to validate the capability of MRGS-ART to reveal the $(i+1)^{th}$ failure after i failure(s) have already been revealed.

7.4.2.2 *Generation Time*

The time taken to select 10,000 MR-MG pairs for each algorithm is presented in Table 29. The following observations can be made based on the generation time experimental results:

- Typically, the computational overhead of MRGS-ART is affected by two elements: The number of source candidate (the k value) and the number of MRs. In particular, the computational overhead of MRGS-ART increases as both the number of source candidates and the number of MRs increase.
- It can be evident that the generation time of MT-RT is consistently shorter than that of MRGS-ART in all scenarios, as anticipated. Nonetheless, given that the generation time of MRGS-ART is either comparable to or sorely marginally higher than that of MT-RT, the computational overhead of MRGS-ART is deemed acceptable.
- The generation time was used to measure the computational overhead of MRGS-ART, which measures the time required to generate a certain number of MGs, regardless of the number of failures, or whether or not the failures are detected. That is, the generation time is generally not affected by the presence or number of software failures, and thus, it can also be used to reflect the computational overhead of MRGS-ART for detecting multiple software failures.

7.4.2.3 *Dispersion and Discrepancy*

Since MRGS-ART aims to enhance MT performance by making STC and FTCs evenly distribution throughout their respective input domains for each MR used during the MT process, this empirical experiment computed the Dispersion and Discrepancy for all the MRs used. In order to compare the STC and FTC distribution diversity among different algorithms, the mean Dispersion and mean Discrepancy of all the MRs used in the MT procedure were computed, as illustrated in Table. 30 - 31.

The following observations can be obtained based on the Dispersion and Discrepancy experimental results:

- *When* $k = 1$: MRGS-ART with $k = 1$ was able to significantly outperform MT-RT.
- *When* $k = 2$: MRGS-ART with $k = 2$ was able to significantly outperform both MT-RT and MRGS-ART with $k = 1$.
- *When* $k = 3$: As the k value increases from 2 to 3, the Dispersion and Discrepancy performance of MRGS-ART shows a slight improvement. In particular, the improvement

in Dispersion and Discrepancy between MRGS-ART with $k = 2$ and MRGS-ART with $k = 3$ is typically lower than the improvement between MRGS-ART with $k = 1$ and MRGS-ART with $k = 2$.

- *When $k \geq 4$*: When $k \geq 4$, the Dispersion and Discrepancy performance of MRGS-ART remains relatively stable.

- Similar conclusions to those drawn from F-ratio and Cohen's d experimental results can be inferred: With block MRVRs, the STC and FTC distribution diversity of MRGS-ART initially increases as the number of source candidates (the k value) increases and then stabilizes within a certain range after reaching a threshold; while with point/strip MRVRs, the number of source candidates typically does not have an impact on the STC and FTC distribution diversity of MRGS-ART.

7.4.3 *Answer to Research Question*

This section answers the research questions listed in Section 7.3 based on the experimental results and discussions.

7.4.3.1 *Answer to RQ1*

Experimental results on F-ratio, Cohen's d, Dispersion, and Discrepancy indicate that, with block MRVRs, MRGS-ART was able to significantly outperform MT-RT in terms of test effectiveness and test-case distribution diversity; while with point/strip MRVRs, both algorithms generally performed similarly to each other. In this context, a conclusion can be inferred that MRGS-ART is capable of performing at least no worse, and sometimes significantly better, than MT-RT. This may be attributed to the fact that MRGS-ART inherits the characteristics of ART algorithms: They tend to excel in block failure regions. The experimental results on generation time indicated that MRGS-ART, compared to MT-RT, required an acceptable computational overhead to choose a certain number of MR-MG pairs among existing ones. Thus, based on the experimental results, it can be concluded that MRGS-ART is typically a preferable option for selecting MR-MG pairs than MT-RT.

7.4.3.2 *Answer to RQ2*

According to the experimental results, the following suggestions can be provided to guide the application of MRGS-ART:

- The threshold for the number of candidate MGs (the k value) is affected by the dimensionality of the input domain and the number of MRs used during the MT process: It decreases as the dimension of the input domain and the number of MRs increase. For instance, the threshold can be set to 3 for programs with 3D input domains and three MRs. This differs from traditional ART algorithms, where the number of candidates is typically set to 10 [77, 140].

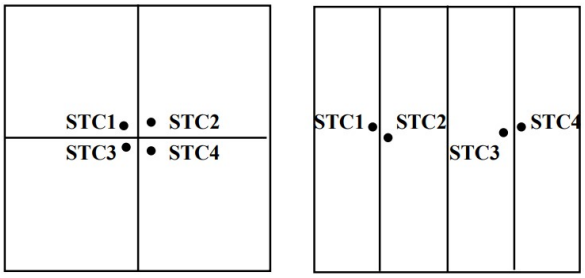


Figure 47: Two sets of possible nearby STCs selected by MRGS-ART in 2D input domains

- According to the experimental results on F-ratio, Cohen's *d*, Dispersion, and Discrepancy, it can be observed that when the *k* value increases from 3 to 4, the performance of MRGS-ART remains similar in most cases, with occasional slight improvements; and when the value increases from 4 to 5, its performance remains relatively stable in all cases. Based on the experimental results for the generation time, it can be concluded that the computational overheads for MRGS-ART increase as the value increases. Therefore, a conclusion can be drawn that if the primary concern is the fault-detection capability of MRGS-ART, setting the *k* value to 4 is sufficient for most scenarios; alternatively, since the fault-detection capability of MRGS-ART generally remains unchanged, and sometimes has a little improvement, when the *k* value increases from 3 to 4, if achieving a balance between test effectiveness and efficiency is prioritized, the *k* value can also be set to 3. In addition, some potential future work may also include exploring the impact of additional *k* values (such as 6 to 10) on the performance of MRGS-ART to further support the above conclusions.

7.5 FUTURE WORK

7.5.1 *An Enhanced Version of MRGS-ART*

7.5.1.1 *Introduction and Motivation*

Section 7.2 has proposed and introduced a novel MR-MG pair selection algorithm, termed MRGS-ART. Nevertheless, empirical experiments have revealed a potential performance-affecting problem, which prompts further investigation and resolution in future work. Specifically, the following two questions remain to be investigated and answered:

1. Can the MRGS-ART algorithm consistently ensure an even distribution of selected STCs and FTCs across their respective input domains?
2. Are there methods available to enhance the even distribution of STCs and FTCs, in order to further enhance the fault-detection capability of MRGS-ART?

The original MRGS-ART algorithm aims to improve MT performance by evenly distributing STCs and FTCs throughout their respective input domains for all MRs employed during the MT process. In order to achieve the even distribution of STCs and FTCs, MRGS-

ART partitions the input domain into subdomains and selects MGs from empty ones. However, it has been observed that the MGs selected by MRGS-ART from empty subdomains may still be in close proximity to other executed MGs, leading to a potential challenge referred to as the nearby test-case problem. BART may also encounter this problem [196, 197]. In particular, the MGs chosen via MRGS-ART may closely neighbor the MGs from adjacent non-empty subdomains. For instance, Fig. 47 illustrates two instances of STC distribution in a 2D input domain, revealing the uneven distribution of STCs selected by MRGS-ART, which may negatively affect the performance of MRGS-ART.

7.5.1.2 *An Enhanced Version of MRGS-ART (MRGS-ART+)*

To address the nearby test-case problem, MRGS-ART was extended to MRGS-ART+. In MRGS-ART+, subdomains are categorized according to their locations relative to executed MGs, as follows:

- Empty Subdomains: Subdomains devoid of any executed STCs or FTCs.
- Occupied Subdomains: Subdomains containing at least one executed STCs or FTCs.
- Adjacent Subdomains: Empty subdomains adjacent to occupied subdomains.

The steps of applying MRGS-ART+ to 2D input domains with n 1-1 MRs are outlined in Algorithm 7. In particular, MRGS-ART+ not only assesses whether or not a candidate MG is located within an empty subdomain, but also counts its neighboring occupied subdomains and measures the distance from this candidate MG to its nearest occupied subdomain. The rationale behind these steps is to guarantee a basic distance between the selected MG and all other executed MGs. The underlined parts are steps that are different from the original MRGS-ART. The specific explanation is as follows:

- Step 18: When considering the i^{th} candidate MG, MRGS-ART+ identifies the subdomain wherein the source/follow-up candidate resides, subsequently calculating the cumulative count of neighboring occupied subdomains. The underlying rationale of this step is to assess whether or not the given source/follow-up candidate is located within adjacent subdomains.
- Steps 19-21: Given the i^{th} candidate MG, if its source/follow-up candidate is located within an adjacent subdomain, then MRGS-ART+ calculates the distances from it to all neighboring occupied subdomains and chooses the shortest one, represented by dis_i.
- Steps 24-29: MRGS-ART+ chooses one candidate MG for execution according to one of the following criteria:

  1) If there exists exactly one candidate MG with the maximum NG_i value, then MRGS-ART+ chooses this candidate MG.

  2) If multiple candidate MGs have the same maximum NG_i value, then MRGS-ART+ chooses the one with the maximum dis_i value.

Algorithm 7: MRGS-ART+ for n 1-1 MRs

Assume the existence of n 1-1 MRs;
Determine the scope of the source and follow-up input domains according to the given MR;
Each MR is given one executed STC set and one executed FTC set;
Initialize the n executed STC sets and n executed FTC sets to be empty;
Randomly generate one STC for each MR;
Generate FTCs based on the MRs and the STCs;
for $i = 1 \rightarrow n$ **do**
    Randomly choose an MR containing empty executed STC and FTC sets;
    Execute the MG against the SUT and check whether the MR is violated;
    Add the executed STC and FTC to their respective executed test sets;
end
while *all stopping conditions are not satisfied* **do**
    if *(a) the quantity of executed STCs/FTCs reaches a threshold* ***or*** *(b) all the subdomains contain at least one executed STC/FTC* **then**
        Bisect all the subdomains in the relevant source/follow-up input domain;
    end
    Assume the existence of k ($k > 0$) source candidates;
    for $m = 1 \rightarrow n$ **do**
        for $i = 1 \rightarrow k$ **do**
            Choose the i^{th} source candidate and construct the follow-up candidate based on the m^{th} MR;
            Count the sum of the quantity of source and follow-up candidates located within empty subdomains, represented by NG_i;
            Count the sum of the quantity of neighboring occupied subdomains, represented by NS_i;
            if $NG_i > 0$ & $NS_i > 0$ **then**
                Compute the average distance from the source and follow-up candidates to their nearest occupied subdomain, represented by dis_i;
            end
        end
        Choose the candidate MG with the largest NG_i, represented by $(NG_i)_m$;
        if *multiple pairs contain the maximum* $(NG_i)_m$ *value* **then**
            Choose the one with the smallest NS_i, represented by $(NS_i)_m$;
            if *multiple pairs contain the maximum* $(NS_i)_m$ *value* **then**
                Choose the one with the largest dis_i, represented by $(dis_i)_m$;
            end
        end
    end
    Choose the MR-MG pair with the largest $(NG_i)_m$;
    if *multiple pairs contain the maximum* $(NG_i)_m$ *value* **then**
        Choose the pair with the smallest $(NS_i)_m$;
        if *multiple pairs contain the maximum* $(NS_i)_m$ *value* **then**
            Choose the pair with the largest $(dis_i)_m$;
        end
    end
    Execute the MG against the SUT and check whether the MR is violated;
    Add the executed STC and FTC to their respective executed test sets;
end

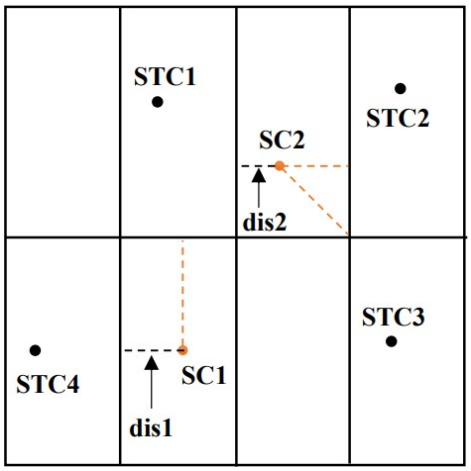


Figure 48: Distribution of executed STCs and source candidates

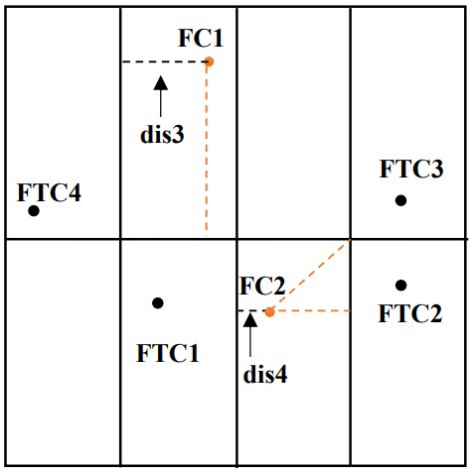


Figure 49: Distribution of executed FTCs and follow-up candidates

3) If multiple candidate MGs possess both the maximum NG_i value and the maximum dis_i value, then MRGS-ART+ randomly chooses one from them.

After completing the computation and comparison of all candidate MGs based on the first MR, MRGS-ART+ proceeds to iterate through the above steps for all remaining MRs to determine an appropriate MR-MG pair.

7.5.1.3 *Application of MRGS-ART+*

To exemplify the application of MRGS-ART+ to an SUT with a 2D input domain, a simplified scenario is presented where only one 1-1 MR is employed, and the value of k is set to 2. After four iterations of MRGS-ART+, the input domain and the distribution of test cases are illustrated in Figs. 48-49. The black points represent executed STCs and FTCs, while the red points represent two source candidates (SC_1 and SC_2) and two follow-up candidates (FC_1 and FC_2). It can be observed that all candidates are situated within adjacent subdomains. Subsequently, MRGS-ART+ calculates the number of neighboring occupied subdomains for each candidate STC/FTC.

- SC_1 is adjacent to two occupied subdomains (the ones containing STC1 or STC4); and the distance to its nearest occupied subdomain is dis1, denoted as a black line.
- SC_2 is adjacent to three occupied subdomains (the ones containing STC1, STC2 or STC3); and the distance to its nearest occupied subdomain is dis2, denoted as a black line.
- FC_1 is adjacent to two occupied subdomains (the ones containing FTC1 or FTC4); and the distance to its nearest occupied subdomain is dis3, denoted as a black line.
- FC_2 is adjacent to three occupied subdomains (the ones containing FTC1, FTC2 or FTC3); and the distance to its nearest occupied subdomain is dis4, denoted as a black line.

The first candidate MG (SC_1, FC_1) is selected by MRGS-ART+ for the given MR due to its fewer neighboring occupied subdomains. Additionally, if two candidate MGs possess an equal quantity of neighboring occupied subdomains, then MRGS-ART+ compares the values of $(dis1 + dis3)/2$ and $(dis2 + dis4)/2$ and chooses the one with the greater distance (the first candidate MG) for execution.

7.5.2 *Metamorphic Relation and Group Selection based on ART Through Iterative Partitioning (MRGS-IPART)*

In addition to analyzing and comparing the performance of various traditional partition-based ART algorithms and explaining the rationale of selecting BART to design MR-MG selection algorithm in Section 7.2.2, this section intends to examine whether or not the MR-MG pair selection algorithms designed according to IPART and RPART can achieve satisfactory effectiveness and efficiency. With this consideration, this section proposes a new algorithm so-called Metamorphic Relation and Group Selection based on ART Through Iterative Partitioning (MRGS-IPART). The steps of MRGS-IPART with n 1-1 MRs and 2D input domains are outlined in Algorithm 8. The underlined parts are steps that are different from the original MRGS-ART. The specific explanation is as follows:

- Step 3: In this step, MRGS-IPART set the value of the partitioning schema (p) to 1. The rationale of using the partitioning schema is that MRGS-IPART partitions the input domain based on its value.
- Steps 10-13: If either (a) the number of executed STCs/FTCs reaches a threshold; or (b) all source/follow-up subdomains are covered by executed STCs/FTCs or surround a subdomain covered by executed STCs/FTCs, then MRGS-IPART adds the partitioning schema by 1 ($p = p + 1$) and partitions the corresponding input domain into $p * p$ equally-sized subdomains. In contrast to MRGS-ART, where only non-empty subdomains are excluded from generating new test cases, MRGS-IPART avoids generating test cases in both non-empty subdomains and subdomains surrounded by non-empty subdomains.

Algorithm 8: MRGS-IPART for n 1-1 MRs with 2D input domains

Assume the existence of n 1-1 MRs;
Determine the scope of the source and follow-up input domains according to the given MR;
Each MR is given one executed STC set and one executed FTC set;
Initialize the n executed STC sets and n executed FTC sets to be empty;
Randomly generate one STC for each MR;
Generate FTCs based on the MRs and the STCs;
Initialize the value of the partitioning schema (p) to 1;
for $i = 1 \rightarrow n$ **do**
    Randomly choose an MR containing empty executed STC and FTC sets;
    Execute the MG against the SUT and check whether the MR is violated;
    Add the executed STC and FTC to their respective executed test sets;
end
while *all stopping conditions are not satisfied* **do**
    if *(a) the quantity of executed STCs/FTCs reaches a threshold* ***or*** *(b) every the subdomains contains at least one executed STC/FTC or is surrounded by non-empty subdomains* **then**
        $p = p + 1$;
        Partition the relevant input domain into $p * p$ equally-sized subdomains;
    end
    Assume the existence of k ($k > 0$) source candidates;
    for $m = 1 \rightarrow n$ **do**
        for $i = 1 \rightarrow k$ **do**
            Choose the i^{th} source candidate and construct the follow-up candidate based on the m^{th} MR;
            Count the sum of the quantity of source and follow-up candidates that are located within an empty subdomain and not surrounded by any non-empty subdomain, represented by NG_i;
        end
        Choose the candidate MG with the largest NG_i, represented by $(NG_i)_m$;
        if *multiple candidate MGs contain the maximum NG_i value* **then**
            Choose one at random;
        end
    end
    Choose the MR-MG pair with the largest $(NG_i)_m$;
    if *multiple MR-MG pairs contain the maximum $(NG_i)_m$ value* **then**
        Choose one at random;
    end
    Execute the MG against the SUT and check whether the MR is violated;
    Add the executed STC and FTC to their respective executed test sets;
end

- Step 18: Given the i^{th} candidate MG, MRGS-ART counts the sum of the number of source and follow-up candidates that are (1) within empty subdomains; and (2) not surrounded by non-empty subdomains, denoted as NG_i. The rationale behind this step is to measure and compare the quality of the candidate MG, with the aim of selecting the "most appropriate" one for execution.

Algorithm 9: MRGS-RPART for n 1-1 MRs with 2D input domains

```
1  Assume the existence of n 1-1 MRs;
2  Determine the scope of the source and follow-up input domains according to the
     given MR;
3  Each MR is given one executed STC set and one executed FTC set;
4  Initialize the n executed STC sets and n executed FTC sets to be empty;
5  Randomly generate one STC for each MR;
6  Generate FTCs based on the MRs and the STCs;
7  for i = 1 → n do
8  |  Randomly choose an MR containing empty executed STC and FTC sets;
9  |  Execute the MG against the SUT and check whether the MR is violated;
10 |  Add the executed STC and FTC to their respective executed test sets;
11 end
12 while all stopping conditions are not satisfied do
13 |  Assume the existence of k (k > 0) source candidates;
14 |  for m = 1 → n do
15 |  |  for i = 1 → k do
16 |  |  |  Choose the i^th source candidate and construct the follow-up candidate
   |  |  |    based on the m^th MR;
17 |  |  |  Map the source and follow-up candidates into the relevant subdomains
   |  |  |    and calculate the size of the subdomains, represented by NG_i;
18 |  |  end
19 |  |  Choose the candidate MG with the largest NG_i, represented by (NG_i)_m;
20 |  |  if multiple candidate MGs contain the maximum NG_i value then
21 |  |  |  Choose one at random;
22 |  |  end
23 |  end
24 |  Choose the MR-MG pair with the largest (NG_i)_m;
25 |  if multiple MR-MG pairs contain the maximum (NG_i)_m value then
26 |  |  Choose one at random;
27 |  end
28 |  Execute the MG against the SUT and check whether the MR is violated;
29 |  Add the executed STC and FTC to their respective executed test sets;
30 |  Partition the relevant subdomain by drawing horizontal and vertical lines based
   |    on the currently-executed STC/FTC;
31 end
```

7.5.3 *Metamorphic Relation and Group Selection based on ART Through Random Partitioning (MRGS-RPART)*

Algorithm 9 presents the steps of MRGS-RPART with n 1-1 MRs and 2D input domains. The underlined parts are steps that are different from the original MRGS-ART. The specific explanation is as follows:

- Step 14: MRGS-RPART maps the source candidate into the source input domain and maps the follow-up candidate into the follow-up input domain. Then MRGS-RPART computes the sum of the size of all the occupied subdomains, represented by NG_i. The rationale behind this step is to measure and compare the quality of the candidate MG, with the aim of selecting the "most appropriate" one for execution.
- Step 27: MRGS-RPART maps the currently-executed STC into the source input domain and partitions the occupied subdomain by using horizontal and vertical lines based on the executed STC; subsequently, MRGS-RPART maps the currently-executed FTC into the follow-up input domain and partitions the occupied subdomain by using horizontal and vertical lines based on the executed FTC.

7.6 CONCLUSION

It has been reported that the quality of MRs and MGs has a great impact on the performance of MT [64, 246]. As the previous chapters (Chapters 4, 5 and 6) of this thesis have concentrated on the identification of MRs and the generation of MGs from scratch, this chapter mainly considers the selection of MR-MG pairs from existing ones, which is also an area that have been neglected by prior studies in MT.

This chapter has integrated partition-based ART algorithms into MT and introduced a novel black-box testing metric for selecting MR-MG pairs: The MR-MG pair is chosen to maximize the distance between the current MG and executed and non-MR-violating MGs. In particular, the rationale of this metric is to improve MT performance by making STCs and FTCs evenly distributed throughout their respective input domains for all the MRs included during the process of MT. At this point, a novel MR-MG pair selection algorithm, MRGS-ART, was introduced, which is capable of automatically and adaptively selecting effective MR-MG pairs according to executed and non-MR-violating MGs. BART, a widely-used PART algorithm, was chosen to design the algorithm due to its ability to effectively balance test efficiency and effectiveness [78]. Empirical experiments were conducted to assess the performance of MRGS-ART on systems of varying complexities and input dimensions. The experimental results have indicated that, compared to MT-RT (the most commonly-used MR-MG pair selection algorithm [64, 246]), MRGS-ART was able to detect the first MR violation faster and achieved a more even distribution of STCs and FTCs throughout their respective input domains, while maintaining an acceptable computational overhead to select MR-MG pairs from existing ones. Section 7.5.1 introduced MRGS-ART+, which is an improved version of the original MRGS-ART. Sections 7.5.2 -

7.5.3 presented two additional MR-MG selection algorithms based on other widely-used partition-based ART algorithms (IPART and RPART). Future work will involve evaluating the performance of these algorithms and comparing them with MRGS-ART.

The subsequent chapter of this thesis will introduce the application of MT and ME as supplementary approaches for model testing, verification, and selection in the domain of credit risk assessment.

8

METAMORPHIC TESTING FOR VALIDATING CREDIT SCORE ASSESSMENT MODELS

Papers delivered from this chapter (Under Review)

1. **Zhihao Ying**, Anthony Bellotti, Joe Breeden and Dave Towey. Metamorphic Exploration for Machine Learning Validation and Model Selection. 1. Abstract from Credit Scoring and Credit Control Conference, Edinburgh, United Kingdom, 2023.

2. **Zhihao Ying**, Anthony Bellotti, Joseph Lynn Breeden, and Dave Towey. Metamorphic Testing and Exploration for Machine Learning Credit Score Models. Submitted to *Applied Soft Computing*, 2024.

8.1 INTRODUCTION AND MOTIVATION

Digital decision tools have found widespread application in automating decision-making within the financial services industry, supplanting human participants [112]. At the heart of these tools are prediction models, forecasting individuals' future behaviors based on their past information and actions. For instance, credit risk models serve as statistical tools extensively employed to estimate creditworthiness and forecast the probability of default on credit commitments [225]. Credit bureaus and lenders rely on credit risk models to evaluate the lending risk associated with individuals or businesses. Typically, credit risk models express the probability of default through a credit score, a three-digit number representing the creditworthiness of an individual or business. A higher score corresponds to a lower probability of default, indicating a higher probability of being a responsible borrower. Generally speaking, credit risk models function as binary classifiers designed according to debt repayment [133, 274], used to distinguish between default (outstanding loans) and non-default (repaid loans) scenarios. Given that default cases are always much less common than non-default cases, credit risk models may encounter the highly imbalanced classification challenge [31].

Employing credit risk models can have various benefits for both lenders and borrowers, as described below:

- Fast and Cost-effective Decision Making: Automation of the assessment process through credit risk models enables lenders to make lending decisions more rapidly and economically compared to human counterparts.

- Enhance Accuracy and Avoid Risk: Credit risk models utilize statistical algorithms to analyze large amounts of data, in order to identify potential risks and uncertainties and provide more precise credit risk predictions compared to manual underwriting processes.
- Consistency and Fairness: Credit risk models apply consistent standards to assess creditworthiness, reducing the risk of bias/discrimination (i.e., based on race, gender, or ethnicity) in loan decisions.
- Tailored Solutions: Lenders have the flexibility to customize credit risk models based on specific needs (such as lending policies and target markets), enhancing the adaptability of credit decision-making.
- Fraud Prevention: Credit risk models can integrate fraud detection algorithms to identify and avoid suspicious/fraudulent activities, with the aim of reducing risks.

Credit risk models are therefore of great value to both lenders and borrowers, which highlights the need for efforts to determine whether or not the credit risk models developed and applied within the financial services industry meet the needs of users. The fundamental model evaluation approach needs to ensure a satisfactory model-fit, characterized by a clear distinction between the default and non-default categories. Common measures employed for this assessment include the Area Under the ROC Curve (AUC), Kolmogorov-Smirnov statistic or the Gini score, which can be used to measure the discrimination power of the models, thus helping in model selection [26, 124]. Evaluation is performed on an independent test set or through methodologies like cross-validation in order to alleviate the risk of model over-fitting.

The implementation of ME represents an optimal strategy for any commercial application of prediction algorithms such as credit scoring. A notably evident business rationale, for instance, suggests that increasing the credit scores of individuals or businesses (with other factors held constant) ought to enhance their likelihood of obtaining a loan. The anticipated association between credit score and risk is described as follows:

- "Your credit score is important. The higher your credit rating, the better your chances of being accepted for credit at the best rates. It can influence your ability to get things like credit cards, loans, mortgages, mobile contracts and more"[1].
- "If you have a low or 'bad' credit score, you're more likely to be turned down when you apply to borrow money, or offered less favourable interest rates, in which case you should take steps to improve your score"[2].

An HMR can be identified based on the anticipated association between credit score and risk, subsequently followed by empirical validation to assess the quality and performance of models.

1 https://www.experian.co.uk/consumer/guides/improve-credit-score.html (3 March 2023)
2 https://www.which.co.uk/money/credit-cards-and-loans/credit-scores/how-to-improve-your-credit-score-am4S05w3aUIE (3 March 2023)

This chapter introduces a novel approach for empirically assessing the quality and performance of credit risk models on the basis of MT and ME: Users are able to identify HMRs according to business expectations hypothesized by users, and forecast the potential impact of specific input modifications on the output. This, to the best of our knowledge, is the first exploration of using ME in evaluating credit risk models. Conventional ML-based model evaluation methodologies typically compare the predicted outputs with the actual results to determine the accuracy of the test set classification. In contrast, the proposed approach assesses whether or not the outputs conform to the anticipated changes, as declared in the HMRs. This chapter presents a case study investigating and comparing the application of conventional evaluation metrics with the proposed approach for assessing ML-based credit risk models. Specifically, this chapter applies the proposed approach to evaluate the performance of models chosen via conventional model-fit evaluation metrics. Empirical experimentation revealed that the models under test frequently violated the HMRs, and the number of violations increases with the grow of model size/complexity. Thus, it might be advantageous to employ the proposed approach for re-validating existing credit risk models and incorporate it into the credit risk model testing, evaluation and selection process. Additionally, given that HMR identification and ME implementation do not require expertise or familiarity with software testing, the proposed approach is generally straightforward and user-friendly.

8.2 RESEARCH QUESTIONS

The following RQs were formulated to provide guidance for the empirical experiments:
RQ1: What are the relations between conventional evaluation metrics and ME in evaluating ML-based credit risk models?

- Objective: Explore whether or not HMR violations can be identified on the models chosen based on conventional evaluation metrics of model-fit (such as AUC).
- Motivation: This chapter presents a novel approach for empirically assessing the quality and the performance of credit risk models according to ME, an examination of the relations between conventional evaluation metrics and ME in evaluating ML-based credit risk models should be conducted.
- Methodologies: Apply ME to evaluate the models chosen through conventional evaluation metrics of model-fit (i.e., AUC)

RQ2: What factors may have an impact on the performance of ME in evaluating credit risk models?

- Objective:
    1. Explore the impact on lenders and customers of deploying credit risk models based on ME or conventional evaluation metrics (such as AUC).

2. Provide lenders and customers with suggestions for the effective selection, application and validation of credit risk models.

- Motivation: Since this chapter presented proposed to apply ME to empirically assess the quality and the performance of credit risk models, an investigation of how to effectively apply ME should be conducted.

- Methodologies:

1. Select three possible factors that are likely to have an impact on the performance of ME:

a) The size/complexity of a model, as it is a factor that has an impact on the performance of the model on traditional evaluation metrics (i.e., AUC) [26, 124] and interpretability [16]: Complex models are models with a non-linear structure and a larger number of parameters and hence more flexibility in fitting the training data. For example, the complexity of a neural network is based on the number of layers and neurons [2, 105, 160], while for a gradient-boosting decision tree, it is the number and depth of trees [114], amongst other possible parameters. Classically, the size/complexity and accuracy of a model are often positively correlated, and a model with a larger size or higher complexity can mean that the model is likely to have lower interpretability. In general, overly complex models can lead to over-fitting. In the context of MT, it is expected that overly complex models may also lead to higher chance of HMR violations, since any particular decision on a new customer may be based on idiosyncratic fit to training data or influenced by noise in training data, rather than expressing a general rule genuinely related to the business area and expressed in the HMRs.

b) The specific models used, as it is also a factor that has an impact on the performance of the model on traditional evaluation metrics (illustrated in Section 2.5.5).

c) The specific HMRs used, as previous researches [64, 246] on MRs and HMRs have repeatedly demonstrated their impacts on the performance of MT and ME.

2. Modify the values of the factors that may have an impact on the performance of ME to investigate the best performance of ME.

3. Design sub-RQs to delve deeper into specific aspects of performance.

- Sub-RQs:

1. *Will the performance of ME in evaluating credit risk models be affected by the size or complexity of the models?* The major factor considered in this sub-RQ is the size/complexity of the models. Loosely speaking, the number of neurons was modified to change the size/complexity of neural networks, and the numbers of trees in gradient-boosting trees and random forests were modified to change the sizes/complexities.

2. *Will the performance of ME in evaluating credit risk models be affected by the specific HMRs used?* This sub-RQ investigates whether or not some particular HMRs are more likely to be violated than others. Specifically, five different HMRs were identified and explored.
3. *Will the performance of ME in evaluating credit risk models be affected by the specific models used?* This sub-RQ investigates that given the same set of HMRs and datasets, whether or not different models would lead to different numbers of HMR violations and which ones were more or less vulnerable to HMR violations.

8.3 CASE STUDY OF CREDIT RISK MODELS

8.3.1 *Experimental Setup*

The publicly available dataset of Freddie Mac US mortgages[3] in 2016 and 2018 was selected for these experiments. The user guide for Freddie Mac mortgage loan data [188] provides a detailed introduction to the dataset The rationale for choosing the 2018-oriented dataset is that when the project began in 2022, it was the latest dataset allowing for a 3-year observation period. In the experiments, to ensure that the experimental results were not affected by the COVID-19 outbreak, the 2016-oriented dataset (with a 3-year observation period), which does not contain the COVID-19 outbreak, was also considered. In these two datasets, default is defined as at least 90 consecutive days of delinquent on mortgage repayments or all real estate acquisitions within 3 years of account opening. The raw 2016-oriented dataset consists of 43,232 clients with 1,360 default loans, while the raw 2018-oriented dataset consists of 44,672 clients with 1,467 default loans. In each iteration of the experiments, the two datasets were randomly divided into training and testing sets, with a segmentation ratio of 3 : 1, where 75% of the dataset were added to the training set and 25% were added to the testing set.

These experiments include two types of tests:

1. Timely and out-of-sample testing: Test a model independently to avoid over-fitting and validate the generalization ability of the model [127, 283]. Over-fitting represents when a model can only give accurate predictions for training data but performs poorly on unseen data [127, 283]. By employing an independent test set for model validation, the training set can be used to train the model, and the test set can be used to validate the performance of a model on unseen data, in order to detect and prevent over-fitting. The testing set and training set have the same distribution.
2. Predictive analytics [161, 208]: Evaluate the forecast performance of a model, which is a realistic scenario that matches how the finance practitioners use the model.

It should be noted that the class imbalance problem (that is, lower default rate) is likely to have an impact on the performance of models. In order to alleviate the problem and im-

3 https://www.freddiemac.com/research/datasets/sf_loanlevel_dataset.page (9 August 2023)

prove the experimental performance, re-sampling was considered [31]. That is, the defaults in the training dataset were tripled in order to re-sample the datasets. For example, the raw 2018-oriented training dataset consists of 32,404 non-defaults and 1,101 defaults. After re-sampling (tripling the defaults), the new training dataset consists of 32,404 non-defaults and $3 * 1101 = 3303$ defaults. Additionally, the testing dataset was not re-sampling, with the aim of ensuring that the true distribution of the results were reflected. Table 32 describes the detailed information of the training and testing datasets.

Based on the findings of previous researchers on machine learning used for credit scoring (as illustrated in Section 2.5.5), three popular ML-based models were chosen, including neural networks, gradient-boosting decision trees and random forests. The functions from Scikit-learn in Python[4] were adopted to build models. Particularly, the following steps were adopted to build models with various sizes/complexities:

- The number of hidden neurons and the number of hidden layers were modified to vary the sizes/complexities of multi-layer neural networks. More specifically, the MLPClassifier function was chosen to build multi-layer neural networks, and the values of the *hidden_layer_sizes* parameter in the MLPClassifier function were edited to vary model sizes/complexities. The number of hidden neurons in each layer includes 1, 2, 3, 4 and 5; while the number of hidden layers includes 1, 2 and 3. In this context, there are a total of 15 sets of multi-layer neural networks of different sizes/-complexities. It is noteworthy that a neural network with no hidden layers and only one output node corresponds to logistic regression: This scenario was not considered as it is essentially a linear model, if weights from each parameter are in the correct direction [28], no HMR violations should be detected.

- The number of trees was modified to vary the sizes/complexities of gradient-boosting decision trees. In particular, the GradientBoostingClassifier function was used to build gradient-boosting decision trees, and the values of the *n_estimators* parameter in this function were edited to vary model sizes/complexities. The input values of *n_estimators* included 1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 500 and 1000.

- The number of trees was modified to vary the sizes/complexities of random forests. Particularly, the RandomForestClassifier function was applied to build random forests and the values of *n_estimators* parameter in this function were edited to change model sizes/complexities. The input values of *n_estimators* included 1, 50, 100, 500, 1000, 5000, 10000 and 20000. Note that in our experiments, only the values of above-mentioned parameters were modified, while all other parameters were default values.

With the aim of robustly investigating the performance of credit-risk models, all the experiments were repeated 100 times to compute the average number of HMR violations, average AUC value, and average standard deviation value. The rationale behind is to account for randomness in the model-building process, such as initial parameter settings

4 `https://scikit-learn.org` (9 August 2023)

Table 32: Credit Dataset Size

Year	Dataset	Test Cases				
		Total	Defaults		Non-Defaults	
			Number	Percentage	Number	Percentage
2016	Training Set	34,464	1020*3=3060	8.88%	31,404	91.12%
	Testing Set	10,808	340	3.15%	10,468	96.85%
2018	Training Set	35,707	1101*3=3303	9.25%	32,404	90.75%
	Testing Set	11,167	366	3.28%	10,801	96.72%

or bootstrap sampling. The percentage of HMR violations was used in order to clearly illustrate the magnitude of the problem, rather than absolute numbers.

One-hot encoding [242] is a widely-used methodology in the field of credit-risk assessment [107, 255]: It uses an indicator variable for each category to encode categorical variables. The following categorical parameters from the datasets were one-hot encoded: "Loan Purpose", "Property Type", "Channel", "Property State", "Occupancy Status", and "First Time Home-buyer". After one-hot encoding, the 2016-oriented dataset consists of 80 input parameters, while the 2018-oriented dataset consists of 79 input parameters.

In order to facilitate data processing, the remaining parameters that were not one-hot encoded were standardized based on the following formula:

$$x' = \frac{x - \mu}{\sigma} \tag{11}$$

- x: The input parameters.
- μ: The sample mean in the training set for x.
- σ: The standard deviation in the training set for x.
- x': The value of the parameter after standardization.

8.3.2 *Input Parameters*

A total of 15 attributes were included as input parameters for the ML-based models, with their basic information summarized below.

- *Credit Score*: This represents the creditworthiness of borrowers, and forecasts their potential ability to repay future debts punctually. The possible range of the input parameter is $[300, 850]$, where 9999 indicates unavailable.
- *Loan-To-Value (LTV)*: Assuming the acquisition of a mortgage loan, this LTV ratio can be computed by dividing the original mortgage loan amount by the appraised value of the mortgaged property. The possible range of the input parameter is $[1\%, 98\%]$, where 999 indicates unavailable.
- *Debt-To-Income (DTI)*: This denotes the percentage of the monthly income that goes to debt payments. The possible range of the input parameter is $[0\%, 65\%]$, where 999 indicates unavailable.

- *Original Loan Term*: This denotes the terms involved when borrowing money. For instance, the period (expressed in months or years) over which a home-buyer should pay back the initial principal balance plus interest.
- *Number of Borrowers*: This reflects the number of borrowers responsible for repaying the mortgage note secured by the property. The possible range of the input parameter is $[1, 10]$, where 99 indicates missing data.
- *Original Interest Rate (OIR)*: This denotes the initial interest rate specified in the mortgage agreement. The possible range of the input parameter is $[2.5, 6]$.
- *Unpaid Principal Balance (UPB)*: This represents the UPB of the mortgage as of the note date.
- *Mortgage Insurance Rate*: This represents the rate of coverage for loan losses. The possible range of the input parameter is $[1\%, 55\%]$, where 999 indicates unavailable.
- *Number of Units*: This represents the property-units numbers. The possible range of the input parameter is $[1, 4]$, where 99 represents unavailable.
- *First Time Home-buyer*: This represents whether or not a borrower (1) acquires the mortgaged property; (2) intends to use it as a primary residence; and (3) has not owned a residence within three years before acquiring the mortgaged property. The possible values of the input parameter contain Y (Yes) and N (No), where 9 indicates unavailable.
- *Occupancy Status*: This represents the category of the mortgage loan. The possible values of the input parameter contain P (Primary Residence), I (Investment Property), and S (Second Home), where 9 indicates unavailable.
- *Channel*: This indicates the involvement of a broker or correspondent in the mortgage loan origination process. The possible values of the input parameter contain R (Retail), B (Broker), and C (Correspondent), where 9 indicates unavailable.
- *Property State*: This represents the territory/state of the mortgaged property, represented by a two-letter abbreviation (i.e., AL for Alabama, FL for Florida, and LA for Louisiana). There are a total of 56 potential values for this input parameter, consisting of 50 states, one district (D.C. for Washington), and five territories (including the U.S. Virgin Islands, Puerto Rico, Northern Mariana Islands, Guam, and American Samoa).
- *Property Type*: This represents the category of property subject to the mortgage. The possible values of the input parameter contain CO (Condominium), PU (Planned Unit Development), MH (Manufactured Home), SF (Single-Family Home), and CP (Cooperative Share), where 99 denotes unavailable.
- *Loan Purpose*: This indicates the category of mortgage loan. The possible values of the input parameter contain P (Purchase Mortgage), C (Cash-Out Refinance Mortgage),

N (No Cash-Out Refinance Mortgage), and R (Not Specified Refinance Mortgage), where 9 denotes unavailable.

8.3.3 *HMRs*

In the experiment, this thesis identified a total of five HMRs for the models. This thesis used different operators (additive and multiplicative) to construction different HMRs, in order to increase the diversity of the HMRs. The definitions of the five HMRs are as follows.

- **HMR1**: If multiplying the credit score in the source inputs using a specific percentage $p(p > 100\%)$ to construct the follow-up inputs, the probability of default is expected to decrease or remain the same. In this experiment, the value of P was set to 110%. This MR was identified according to the credit score business rationale analyzed in in the Section 8.1.

- **HMR2**: If multiplying the DTI in the source inputs using a specific percentage $P(P < 100\%)$ to construct the follow-up inputs, then the probability of default is expected to decrease or remain unchanged. In this experiment, the value of P was set to 90%. The business rationale behind this HMR is that a lower debt corresponds to less risk.

- **HMR3**: If multiplying the LTV in the source inputs using a specific percentage $P(P < 100\%)$ to construct the follow-up inputs, then the probability of default is expected to decrease or remain unchanged. In this experiment, the value of P was set to 90%. The business rationale behind this HMR is that a higher LTV denotes higher credit risk. Different from credit score, in general, to be eligible for the best mortgage products, a home purchaser would be advised by a mortgage advisor to keep a low LTV.

- **HMR4**: If multiplying the credit score in the source inputs using a specific percentage $P_1(P_1 > 100\%)$, and multiplying the LTV in the source inputs using a specific percentage $P_2(P_2 < 100\%)$ to construct the follow-up inputs, then the probability of default is expected to decrease or remain unchanged. In the experiment, the value of P_1 was set to 110% and the value of P_2 was set to 90%. HMR4 is identified based on the composition [182] of HMR1 and HMR3.

- **HMR5**: If multiplying the credit score in the source inputs using a specific percentage $P_1(P_1 > 100\%)$, multiplying the LTV in the source inputs using $P_2(P_2 < 100\%)$, and multiplying the DTI using $P_3(P_3 < 100\%$, to generate follow-up inputs, then the probability of default is expected to decrease or remain unchanged. In this experiment, the value of P_1 was set to 110%, the value of P_2 was set to 90%, and the value of P_3 was set to 90%. HMR5 is identified based on the composition [182] of HMR1, HMR2 and HMR3.

HMR4 and HMR5 are both composite MRs used to identify violations under joint conditions [182]. This allows for tests that are more realistic, since testers would expect multiple variable values to change simultaneously.

8.3.3.1 *Difference Range in HMR Violations*

In order to explore the range of differences between the source and follow-up outputs, an instance of gradient-boosting decision tree with 1000 trees was trained according to HMR1 and the 2018-oriented dataset, as illustrated in Fig. 50: The vertical axis represents the range of differences between the source and follow-up outputs, while the horizontal axis denotes the number of violations within different ranges. It can be seen that most of the violations fall within the range of $[10^{-3}, 10^{-0}]$. Since a change of 10^{-2} is relatively large in the probabilities of default and an HMR violation with a difference of less than 10^{-2} between the source and follow-up outputs may not have a significant impact, the concepts of strong violations and weak violations are defined as follows:

- *Definition 8.1 (Strong (HMR) Violations)*: The (HMR) violations with a difference of no less than 10^{-2} between the source output and subsequent outputs.
- *Definition 8.2 (Weak (HMR) Violations)*: The (HMR) violations with a difference of less than 10^{-2} between the source output and subsequent outputs.

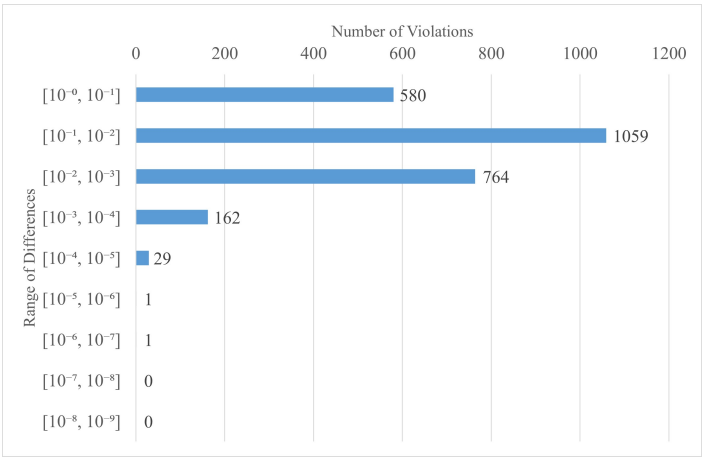


Figure 50: An instance of the number of HMR violations in different ranges

8.3.3.2 *A Strong HMR Violation Example*

The following is a typical example of a strong HMR violation. A gradient-boosting decision tree with 1000 trees was built using HMR1 and the 2018-oriented dataset. The description of this model and the experimental results are as follows:

- Non-standardized values of the input parameters from the STC: Credit Score=661, Mortgage Insurance Rate=0, Number of Units=1, DTI=32, UPB=227000, LTV=67, OIR=5.875, Original Loan Term=360, and Number of Borrowers=1.
- Values of the input parameters from the FTC: Keep unchanged, except for Credit Score = 661*110% $\approx$ 727.

- Source output: 0.150.
- Expected follow-up output: ≤ 0.150.
- Actual follow-up output: 0.177.

This HMR1 violation indicates that increasing the value of the credit score resulted in an unexpected increase in the value of probability of default.

8.3.4 *Experimental Results*

This section provides the experimental results of the three ML-based models using the 2016-oriented and 2018-oriented datasets, as illustrated in Table 33 - 38. It can be observed that, in all experiments, the AUC standard deviation is relatively low than the mean, indicating that the models are stable and the experimental results are reliable.

8.3.4.1 *Neural Network Experimental Results*

The neural network experimental results are illustrated in Tables 33 - 34. It can be observed that, in both datasets, one layer with three neurons achieved the largest AUC value, with $AUC = 0.728$ for 2016 and $AUC = 0.759$ for 2018. Except for HMR3, the proportion of HMR violations is usually low (no more than 1%). The proportion of HMR violations usually increases with the number of layers and neurons. Even in the worst-case scenario $(5 * 3)$, the proportion of HMR violations is still greater than 1%.

8.3.4.2 *Gradient-boosting Decision Tree Experimental Results*

The gradient-boosting decision trees experimental results are illustrated in Tables 35 - 36. The following observations can be made:

- In the 2016-oriented dataset: The proportion of HMR violations was high, with strong violations exceeding 1%. The highest AUC value (0.731) was achieved for 80 trees.
- In the 2018-oriented dataset: The proportion of HMR violations was typically lower than that of the 2016-oriented dataset. The highest AUC value (0.760) was achieved for 90 trees.
- In both scenarios, the proportion of HMR violations usually increases with the number of trees.

8.3.4.3 *Random Forests Experimental Results*

The random forests experimental results are illustrated in Tables 37 - 38. The following observations can be made:

Table 33: Back-propagation neural networks experimental using the 2016-oriented dataset (n*m = (number of neurons in each layer) * (number of layers))

Hidden Layer Size		1*1	2*1	3*1	4*1	5*1	1*2	2*2	3*2	4*2	5*2	1*3	2*3	3*3	4*3	5*3
HMR1	Strong	0.01%	0.10%	0.07%	0.23%	0.36%	0%	0.25%	0.27%	0.76%	1.60%	0%	0.01%	0.59%	1.88%	3.41%
	Weak	0.02%	0.19%	0.11%	0.31%	0.45%	0.03%	0.39%	0.30%	0.96%	1.87%	0%	0.07%	0.89%	2.11%	3.57%
HMR2	Strong	0%	0.03%	0.13%	0.18%	0.29%	0%	0.05%	0.38%	0.84%	1.87%	0%	0.09%	0.51%	1.43%	3.40%
	Weak	0.01%	0.12%	0.74%	0.97%	1.39%	0%	0.51%	1.00%	3.29%	5.15%	0%	0.42%	1.78%	5.20%	8.21%
HMR3	Strong	0%	0.14%	0.74%	0.85%	1.83%	0%	0.40%	1.04%	3.05%	4.93%	0%	0.37%	1.52%	4.29%	5.98%
	Weak	0.01%	0.58%	2.48%	4.42%	7.92%	0%	1.81%	4.88%	9.03%	13.02%	0%	2.13%	7.43%	13.66%	15.23%
HMR4	Strong	0.01%	0.08%	0.08%	0.15%	0.27%	0%	0.21%	0.16%	0.60%	1.28%	0%	0.01%	0.49%	1.58%	2.96%
	Weak	0.02%	0.14%	0.08%	0.21%	0.30%	0.03%	0.38%	0.31%	0.66%	1.48%	0%	0.08%	0.76%	2.03%	2.71%
HMR5	Strong	0.01%	0.05%	0.02%	0.08%	0.17%	0%	0.15%	0.05%	0.42%	0.85%	0%	0.01%	0.35%	1.22%	2.20%
	Weak	0.02%	0.11%	0.10%	0.13%	0.19%	0.02%	0.15%	0.18%	0.48%	1.13%	0%	0.09%	0.62%	1.73%	2.32%
AUC	Mean	0.685	0.718	0.728	0.722	0.717	0.635	0.693	0.721	0.709	0.703	0.585	0.673	0.700	0.704	0.699
	Standard Deviation	0.096	0.051	0.004	0.007	0.007	0.115	0.085	0.011	0.031	0.012	0.112	0.097	0.056	0.014	0.013

Table 34: Back-propagation neural networks experimental using the 2018-oriented dataset (n*m = (number of neurons in each layer) * (number of layers))

Hidden Layer Size		1*1	2*1	3*1	4*1	5*1	1*2	2*2	3*2	4*2	5*2	1*3	2*3	3*3	4*3	5*3
HMR1	Strong	0%	0.03%	0.03%	0.02%	0.02%	0%	0.02%	0.16%	0.26%	0.47%	0%	0.02%	0.34%	0.52%	1.42%
	Weak	0%	0.05%	0.05%	0.02%	0.06%	0%	0.15%	0.09%	0.25%	0.48%	0%	0.25%	0.20%	0.56%	1.29%
HMR2	Strong	0%	0.08%	0.21%	0.56%	0.67%	0%	0.10%	0.64%	1.09%	2.48%	0%	0.07%	0.73%	1.67%	3.96%
	Weak	0%	0.31%	1.46%	1.69%	2.70%	0%	0.39%	1.87%	3.91%	6.70%	0%	0.84%	2.05%	4.33%	8.53%
HMR3	Strong	0%	0.14%	0.75%	1.14%	2.11%	0%	0.22%	0.98%	2.85%	4.79%	0%	0.19%	1.58%	3.10%	5.29%
	Weak	0%	1.11%	2.75%	5.92%	8.21%	0%	1.17%	4.45%	10.39%	14.05%	0%	1.05%	4.48%	13.26%	13.48%
HMR4	Strong	0%	0.01%	0.01%	0.01%	0.01%	0%	0.01%	0.15%	0.24%	0.42%	0%	0.02%	0.28%	0.43%	1.27%
	Weak	0%	0.03%	0.04%	0.02%	0.01%	0%	0.15%	0.09%	0.27%	0.45%	0%	0.24%	0.30%	0.45%	1.23%
HMR5	Strong	0%	0.01%	0.01%	0%	0%	0%	0%	0.11%	0.18%	0.31%	0%	0.01%	0.20%	0.36%	0.86%
	Weak	0%	0.02%	0.04%	0.01%	0%	0%	0.13%	0.09%	0.16%	0.32%	0%	0.24%	0.20%	0.44%	0.92%
AUC	Mean	0.747	0.755	0.759	0.753	0.748	0.659	0.733	0.748	0.747	0.740	0.570	0.682	0.744	0.741	0.731
	Standard Deviation	0.059	0.038	0.003	0.005	0.006	0.133	0.080	0.006	0.010	0.009	0.118	0.123	0.048	0.012	0.012

- In the 2016-oriented dataset: The proportion of HMR violations was very high (greater than 10%). The best AUC value (0.696) was achieved when the number of trees was 5000.
- In the 2018-oriented dataset: The proportion of HMR violations was very high (greater than 10%). The best AUC value (0.721) was achieved when the number of trees was 5000.
- In both scenarios, the proportion of HMR violations usually increases with the number of trees and stabilizes after reaching 5000.
- Gradient-boosting decision trees and random forests are likely to perform well for number of HMR violations until the number of trees reaches a certain threshold. For instance, the experimental results shown in Table 35 indicate that when the number of trees was less than 70, the number of strong HMR violations was less than 2%, and the number of strong HMR violations increases with the number of trees, typically greater than 14%.
- In all experiments, HMR3 was able to achieve the highest proportion of HMR violations, followed by HMR2. HMR1 has the lowest proportion of HMR violations, as well as the composite HMR4 and HMR5.

Table 35: Gradient-boosting decision trees experimental using the 2016-oriented dataset

Number of Trees		1	10	20	30	40	50	60	70	80	90	100	500	1000
HMR1	Strong	0%	0.02%	0.02%	2.05%	2.11%	1.89%	1.62%	26.07%	24.82%	24.20%	23.96%	22.15%	14.76%
	Weak	0%	0.02%	0.96%	23.66%	20.64%	19.94%	19.57%	13.79%	14.75%	14.65%	14.95%	12.94%	8.91%
HMR2	Strong	0%	0.03%	0.19%	0.11%	0.15%	0.21%	0.25%	0.24%	0.25%	0.35%	0.36%	3.94%	5.46%
	Weak	0%	0.46%	0.46%	0.44%	0.76%	0.54%	0.62%	1.22%	1.21%	1.21%	1.20%	6.11%	11.91%
HMR3	Strong	0%	0%	0.01%	0.03%	0.19%	0.26%	0.17%	0.26%	0.98%	0.96%	1.17%	19.45%	33.28%
	Weak	0%	0%	0%	0.01%	0.41%	0.52%	0.61%	0.68%	0.56%	0.57%	0.57%	31.47%	23.94%
HMR4	Strong	0%	0.02%	0.03%	1.75%	1.55%	1.24%	0.99%	24.64%	23.33%	22.65%	22.34%	24.18%	20.31%
	Weak	0%	0.02%	0.88%	23.40%	20.12%	19.46%	19.01%	14.27%	14.90%	14.80%	14.93%	12.54%	10.89%
HMR5	Strong	0%	0.03%	0.03%	0.39%	0.34%	0.25%	0.22%	18.98%	17.86%	17.10%	16.62%	16.59%	13.88%
	Weak	0%	0.01%	0.12%	14.79%	12.78%	11.90%	11.63%	11.55%	11.94%	11.94%	12.24%	11.72%	9.47%
AUC	Mean	0.680	0.712	0.721	0.726	0.727	0.729	0.730	0.730	0.731	0.730	0.730	0.707	0.679
	Standard Deviation	$1*10^{-16}$	$1*10^{-16}$	$1*10^{-16}$	$1*10^{-16}$	$1*10^{-16}$	$6*10^{-7}$	$8*10^{-7}$	$7*10^{-7}$	$8*10^{-7}$	$9*10^{-7}$	$9*10^{-6}$	$3*10^{-5}$	$4*10^{-5}$

Table 36: Gradient-boosting decision trees experimental using the 2018-oriented dataset

Number of Trees		1	10	20	30	40	50	60	70	80	90	100	500	1000
HMR1	Strong	0%	0.17%	0.23%	0.29%	0.28%	0.33%	0.32%	0.35%	0.36%	0.37%	0.36%	0.86%	1.76%
	Weak	0%	0.01%	0.04%	0.14%	0.27%	0.07%	0.09%	0.34%	0.33%	0.80%	0.84%	0.43%	1.13%
HMR2	Strong	0%	0.18%	0.38%	0.40%	0.42%	0.51%	0.55%	0.61%	0.62%	0.68%	0.66%	5.88%	8.51%
	Weak	0%	1.46%	0.19%	0.57%	0.59%	0.59%	1.97%	1.97%	1.94%	1.97%	1.01%	10.23%	14.93%
HMR3	Strong	0%	0%	0.06%	0.37%	0.49%	0.47%	0.44%	0.43%	0.46%	0.45%	0.52%	4.35%	7.20%
	Weak	0%	0%	0.27%	1.92%	1.76%	1.28%	1.24%	0.86%	0.84%	0.85%	0.89%	12.53%	14.78%
HMR4	Strong	0%	0.17%	0.22%	0.32%	0.33%	0.31%	0.31%	0.32%	0.33%	0.33%	0.33%	0.67%	1.49%
	Weak	0%	0.01%	0.17%	0.70%	0.77%	0.47%	0.46%	0.40%	0.42%	0.55%	0.57%	0.60%	1.56%
HMR5	Strong	0%	0.23%	0.35%	0.35%	0.30%	0.37%	0.37%	0.36%	0.39%	0.41%	0.38%	0.82%	1.60%
	Weak	0%	0.02%	0.09%	0.61%	0.71%	0.55%	0.61%	0.60%	0.57%	0.66%	0.60%	0.57%	1.44%
AUC	Mean	0.707	0.733	0.746	0.752	0.753	0.755	0.757	0.758	0.759	0.760	0.759	0.749	0.734
	Standard Deviation	$1*10^{-16}$	$1*10^{-16}$	$1*10^{-6}$	$1*10^{-6}$	$1*10^{-6}$	$2*10^{-6}$	$2*10^{-6}$	$2*10^{-6}$	$2*10^{-6}$	$2*10^{-6}$	$5*10^{-6}$	$1*10^{-4}$	$2*10^{-4}$

8.3.5 *Forecast Performance*

To comprehensively investigate the performance of the models, the forecast performance of the models was also considered, as it is common in the development of credit risk models: In forecasting, historical information is used to predict a range of probable future outcomes. In particular, the models were trained according to the 2016-oriented dataset and then tested according to the 2018-oriented dataset. Because the gradient-boosting decision tree achieved the best performance in terms of AUC, it is selected as the baseline model. The experimental results are illustrated in Table 39. It can be observed that multiple HMR violations were detected. It is worth further comparing the forecast results with those forecasts for 2018 based on training in the same year (Table 36). The 80 tree gradient-boosting decision tree will be selected if the best AUC for 2016 is given priority. Nevertheless, it can be observed that this will lead to an increasing number of HMR violations. Alternatively, 20 trees would be a good choice if prioritizing reducing the number of HMR violations as well as improving AUC values. In general, the forecast performance of the models is highly identical to that of the real-time testing model. This indicates that it is a good idea to choose a credit risk model with fewer HMR violations.

Table 37: Random forests experimental results using the 2016-oriented dataset

Number of Trees		1	50	100	500	1000	5000	10000	20000
HMR1	Strong	3.21%	42.45%	45.88%	42.35%	41.74%	41.67%	41.55%	41.59%
	Weak	0%	0%	2.96%	12.37%	13.76%	14.65%	14.79%	14.81%
HMR2	Strong	1.27%	18.89%	19.50%	11.03%	10.13%	9.34%	9.20%	9.10%
	Weak	0%	0%	3.54%	18.63%	21.38%	24.15%	24.55%	24.75%
HMR3	Strong	1.47%	26.47%	28.80%	20.39%	19.01%	18.26%	18.11%	18.07%
	Weak	0%	0%	4.64%	24.21%	27.98%	31.80%	32.53%	32.86%
HMR4	Strong	3.87%	44.79%	47.87%	44.53%	43.90%	43.83%	43.75%	43.77%
	Weak	0%	0%	2.78%	11.15%	12.36%	13.23%	13.46%	13.52%
HMR5	Strong	3.47%	40.29%	42.93%	38.57%	37.76%	37.70%	37.59%	37.66%
	Weak	0%	0%	2.49%	11.26%	12.64%	13.57%	13.72%	13.77%
AUC	Mean	0.517	0.660	0.677	0.691	0.695	0.696	0.696	0.696
	Standard Deviation	$8*10^{-3}$	$9*10^{-3}$	$6*10^{-3}$	$3*10^{-3}$	$2*10^{-3}$	$9*10^{-4}$	$7*10^{-4}$	$5*10^{-4}$

Table 38: Random forests experimental results using the 2018-oriented dataset

Number of Trees		1	50	100	500	1000	5000	10000	20000
HMR1	Strong	1.49%	16.20%	16.94%	9.44%	8.46%	7.86%	7.76%	7.69%
	Weak	0%	0%	2.83%	16.80%	19.37%	22.60%	23.11%	23.56%
HMR2	Strong	1.60%	19.94%	20.23%	11.78%	10.69%	9.91%	9.76%	9.74%
	Weak	0%	0%	3.81%	18.45%	21.24%	24.41%	24.90%	25.21%
HMR3	Strong	1.44%	26.45%	28.54%	19.98%	18.81%	17.72%	17.48%	17.44%
	Weak	0%	0%	4.85%	24.32%	28.56%	32.90%	33.69%	34.28%
HMR4	Strong	2.00%	22.10%	23.43%	14.88%	13.60%	12.70%	12.41%	12.38%
	Weak	0%	0%	2.89%	17.04%	19.56%	22.06%	22.37%	22.59%
HMR5	Strong	2.05%	20.83%	21.68%	12.66%	11.27%	10.33%	10.11%	10.05%
	Weak	0%	0%	2.41%	15.14%	17.23%	19.26%	19.57%	19.81%
AUC	Mean	0.527	0.693	0.707	0.718	0.719	0.721	0.721	0.721
	Standard Deviation	$7*10^{-3}$	$7*10^{-3}$	$6*10^{-3}$	$2*10^{-3}$	$2*10^{-3}$	$1*10^{-3}$	$7*10^{-4}$	$4*10^{-4}$

8.3.6 *Answers to Research Questions*

8.3.6.1 *Answer to RQ1*

The following conclusions can be inferred on the basis of the experimental results and discussions:

- HMR violations still can be detected even when employing ME on the model exhibiting the highest AUC value.
- The model with minimal HMR violations (0 violations) did not necessarily demonstrate the highest AUC value. These models include a neural network containing one hidden layer and one hidden neuron, a neural network containing two hidden layers with one hidden neuron each, a neural network containing three hidden layers and one hidden neuron in each layer, and a gradient-boosting decision tree containing one tree.

Table 39: The forecast experimental results of gradient-boosting decision trees

Number of Trees		1	10	20	30	40	50	60	70	80	90	100	500	1000
HMR1	Strong	0%	0%	0.02%	3.81%	3.04%	3.05%	2.79%	21.11%	19.65%	18.71%	18.49%	18.82%	12.64%
	Weak	0%	0.04%	1.11%	18.98%	16.30%	15.17%	15.05%	13.25%	14.26%	14.40%	14.49%	9.75%	7.74%
HMR2	Strong	0%	0.08%	1.79%	1.81%	2.07%	2.29%	2.35%	2.27%	2.34%	3.84%	3.81%	6.94%	8.41%
	Weak	0%	0.38%	0.54%	0.44%	0.88%	0.66%	0.71%	1.87%	1.80%	2.53%	2.52%	8.60%	18.64%
HMR3	Strong	0%	0%	0%	0.01%	0.24%	0.39%	0.34%	0.67%	1.54%	1.55%	1.77%	18.97%	32.05%
	Weak	0%	0%	0%	0.01%	0.41%	0.91%	0.89%	0.82%	0.80%	0.80%	0.81%	29.55%	24.69%
HMR4	Strong	0%	0%	0.02%	3.35%	2.15%	2.05%	1.73%	19.46%	18.25%	16.91%	16.62%	20.89%	18.11%
	Weak	0%	0.04%	1.05%	18.90%	16.24%	15.37%	15.19%	13.87%	14.33%	14.39%	14.46%	9.72%	10.19%
HMR5	Strong	0%	0.01%	0.81%	2.27%	2.24%	2.16%	1.98%	15.54%	14.51%	13.68%	13.21%	14.22%	13.12%
	Weak	0%	0.01%	0.51%	11.14%	9.17%	8.58%	8.52%	10.54%	10.70%	10.22%	10.39%	9.03%	8.70%
AUC	Mean	0.715	0.747	0.753	0.754	0.754	0.755	0.754	0.753	0.753	0.750	0.749	0.719	0.695
	Standard Deviation	$1*10^{-16}$	$2*10^{-16}$	$2*10^{-16}$	$2*10^{-16}$	$3*10^{-16}$	$1*10^{-3}$	$1*10^{-3}$	$1*10^{-3}$	$2*10^{-3}$	$2*10^{-3}$	$2*10^{-3}$	$2*10^{-3}$	$2*10^{-3}$

- The relation between the number of HMR violations and the sizes/complexities of models suggests that simpler models are less likely to violate HMRs.
- In this context, it can be concluded that the credit risk models selected according to the number of HMR violation and the values of AUC are not the same.

8.3.6.2 *Answer to RQ2.1*

Based on the experimental results regarding the number of HMR violations and the values of AUC, when keeping models, datasets, and HMRs unchanged, the number of detected HMR violations varied significantly among models of differing sizes/complexities. In this context, it can be inferred that the performance of ME is affected by the sizes/complexities of credit risk models. Given an HMR, more complex models typically cause a higher number of HMR violations, exhibiting more deviations from business rationale.

8.3.6.3 *Answer to RQ2.2*

Based on the experimental results on the number of HMR violations and the values of AUC, when keeping models and datasets unchanged, the number of detected HMR violations varied significantly when different HMRs were used. HMR3 (constructed by modifying the credit score) typically exhibits the highest number of HMR violations, while HMR1 (constructed by modifying the LTV) demonstrates the lowest. Consequently, it is apparent that, on the whole, LTV serves as a less dependable parameter in the models, given its propensity to produce more HMR violations. In this context, it can be inferred that the performance of ME is affected by the specific HMR used, which highlights the importance of HMR identification and selection.

8.3.6.4 *Answer to RQ2.3*

Based on the experimental results regarding the number of HMR violations and the values of AUC, when keeping HMRs and datasets unchanged, the number of detected HMR violations varied significantly when different models were used. The following conclusions can be drawn on the basis of the experimental results:

- Among the three models under test, gradient-boosting decision trees typically had the highest AUC performance; however, it did not contain the lowest number of HMR violations.
- In general, neural networks contained slightly lower AUC values, and had a significantly fewer number of HMR violations.
- Random Forests exhibited the poorest performance across both metrics.
- In this context, it can be inferred that the performance of ME is affected by the specific HMR used, which highlights the importance of HMR identification and selection.

8.3.6.5 *Answer to RQ2*

The following conclusions can be inferred on the basis of the experimental results and discussions:

- More specifically, it can be concluded that adopting an ML-based model solely based on optimal model-fit is likely to result in the model with substantial deviations from business rationale.
- The default patterns vary among models. That is, across numerous iterations of gradient-boosting decision trees and random forests, the number of HMR violations remains relatively constant, with a minute standard deviation. However, in the case of neural networks, the number of HMR violations exhibits significant variability. For instance, in the case of the neural network with $3 * 1$ neurons, no HMR violations were detected in 85 out of 100 trials; in contrast, in instances where HMR violations did occur, their number was notably increased. This suggests that the stability of the neural network decreases with variations in the orientation of network weights across different iterations. Consequently, this observation proposes the possibility of selecting neural networks according to the lower numbers of HMR violations, or ideally, no violations.
- When applying ME to a model with the highest AUC value, HMR violations can still be identified. This implies that a model exhibiting strong AUC performance may struggle to consistently forecast the probability of default for customers with similar profiles, which may pose significant implications for both lenders and borrowers. Consider two loan applicants with comparable profiles, for example, one boasting a higher credit score. Nevertheless, the experimental results revealed that the adoption of an AUC-centric model may lead to a scenario where the applicant with the larger credit score is denied the loan, while the other applicant, despite comparable profiles but with a lower credit score, secures the loan. Such outcomes could significantly impact customer satisfaction. In addition, extending loans to individuals with lower likelihood of repayment may increase the risk of capital loss.
- In conclusion, the credit risk models constructed according to the number of HMR violations are different from those constructed according to the values of AUC, which

could significantly affect both lenders and borrowers. Furthermore, the application of ME necessitates no specialized software-testing experience or expertise, making it advantageous for re-validating existing credit risk models and informing the selection of such models.

8.4 CONCLUSION AND FUTURE WORK

With the increasing popularity of digital financial services, there is an increasingly urgent need for efficient and effective methods for evaluating credit risk models. This chapter has introduced a new systematic approach for users to easily and effectively evaluate ML-based credit risk models from the business rationale perspective. Then this chapter reported on a case study exploring the implementation of traditional evaluation metrics and the proposed approach for ML-based credit risk model evaluation and the relationships and differences between them. In the experiment, a total of three ML-based models have been chosen: Neural networks, gradient-boosting decision trees and random forests. In total, five HMRs have been identified these models, and multiple HMR violations have been detected amongst a significant minority of cases. This is the first study on the topic of using MT and ME to evaluate credit risk models. Since the process of identifying HMRs and implementing ME typically does not require (1) knowledge or experience in software testing; and (2) an available oracle (no need to know the outcome of the loan), the proposed approach can be used as an effective *live* testing procedure as loans are originated and existing credit risk models can be easily re-validated using it. Users and programmers are able to obtain an in-depth understanding of the models under test by using the proposed approach, making it easy and effective to validate and select credit risk models.

The empirical experiments revealed that the credit risk models selected using ME and traditional model evaluation techniques (such as AUC) are different: A model that performs well in terms of predictive performance (i.e., AUC on the independent test set) may still cause a large number of HMR violations. Since the HMRs used in the empirical experiments focus on slightly modifying the feature values to simulate customers with similar situations, a conclusion may be drawn that a model with well-predictive performance may have difficulty predicting the probability of default for customers with similar situations. This issue may have a serious impact on the user experience of both lenders and customers. For example, it may happen that a customer with a higher credit score may not be able to obtain a loan, while another customer with almost identical circumstances but a lower credit score may obtain a loan. In this way, by using both traditional model evaluation techniques and the proposed approach, potential problems can be detected much faster than using only the methods that rely on measuring the outcomes. In this context, this chapter recommends ME as an additional selection indicator for ML-based credit risk model-selection in the development process. In addition, experimental results indicated that although more complex models may be able to better fit an independent test set, they generally result in more HMR violations. Based on the results, this chapter also recom-

mends ME as an additional evaluation indicator in the validation stage of the ML-based credit risk models.

The experimental results suggested that ME should be considered as an additional model-selection and validation indicator when applying ML-based credit risk models in financial services. Currently, credit scorecard developers can examine model coefficient estimates as a way to validate linear models against business rationale [28]. This study presented how this approach can be extended to non-linear ML-based models using ME. From the perspective of recommendations for validation procedures, the following two methods can be considered.

- Identify the models that is within the same AUC confidence interval band as the best model (based on AUC) and then select the model with the fewest HMR violations.
- Determine an acceptable HMR violation proportion threshold and reject any model that exceeds this threshold.

The next chapter of this thesis will conclude the whole thesis.

9 CONCLUSION, CONTRIBUTION AND FUTURE WORK

9.1 DISCUSSION AND CONCLUSION

The field of computer science is experiencing complexity and sophistication; however, the advancement of SQA tools and processes has not been able to match the rapid evolution of these intricate systems, which has consequently led to an increasing number of challenges [17, 81, 246, 293]. Software testing constitutes a necessary component of SQA, requiring comprehensive consideration: This includes the execution of an SUT to assess the behavior/outputs. As an advanced property-based software-testing methodology, MT uses MRs and MGs to identify the existence of potential faults within the SUT. Previous research on MT has repeatedly proven the effectiveness of MT in not only alleviating the oracle problem, but also providing a novel approach for test-case generation and test-output verification [64, 246, 292]. Nevertheless, despite the growing popularity of MT, its performance still requires further research and improvement [64, 246].

Through a comprehensive examination of the literature on MT, this thesis identified the absence of a straightforward and quick method to assess MT performance, which was referred to as the MT-performance evaluation challenge. In contrast, simulations are frequently employed to verify the performance of conventional software testing methodologies, with the aim of reducing the difficulties and time overheads that may be encountered during the verification process. Nevertheless, a challenge lies in the inapplicability of traditional simulations to MT due to: (1) Their reliance on an accessible oracle, which is a premise not assumed by MT; and (2) MT detects software failures through the identification of MR violations, rather than conventional test-case failures. To address this challenge, the concept of MR-Violation Region (MRVR) was introduced in Chapter 3. Subsequently, Chapter 3 developed and presented an MT simulation framework based on MRVRs. The proposed MT simulation framework has the potential to reduce the computational complexity and time overheads associated with MT performance assessment, making it easier and faster for testers to set up MT-related experiments.

The performance of MT, particularly its fault-detection capability, is highly dependent on the MGs and MRs. By conducting a thorough review of the literature on MG generation, the concept of the input-domain difference problem was identified and introduced in Chapter 4. This problem may have an impact on the performance of MG-generation algorithms. Then, a case study was reported in which the proposed MT simulation framework

was applied to investigate the impact of this problem on the performance of a previously-published MG generation algorithm (MT-ART [144]). Empirical studies revealed that the potential cause of the input-domain difference problem lies in neglecting the differences between STCs and FTCs: Previous MG-generation algorithms process the FTCs in the same manner as the STCs. Taking this into account, this thesis introduced a solution to alleviate this issue: Treating FTCs differently from STCs. Building upon this solution, this thesis subsequently introduced a novel MG-generation algorithm, SFIDMT-ART, with new distance metrics to facilitate this algorithm. This algorithm is designed as an improved version of MT-ART. The rationale behind this algorithm is to achieve even spread STCs and FTCs over their respective input domains. In particular, SFIDMT-ART prioritizes the generation of MGs with the following features: 1) Making the STCs evenly distributed across the corresponding source input domain; and 2) making the FTCs evenly distributed across the corresponding follow-up input domain. Empirical experiments were conducted and the results demonstrated that SFIDMT-ART surpassed the original algorithm (MT-ART) in terms of test effectiveness and efficiency.

Nonetheless, there is still one problem with the proposed MG-generation algorithm: The test efficiency. The time complexity of SFIDMT-ART and MT-ART is excessively high, potentially resulting in considerable overheads for MG generation. To address this problem, Chapter 5 integrated partition-based ART into MT and proposed a family of two MG-generation algorithms, MT-BART and MT-IPART. Notably, to generate n MGs, the time-complexity of both MT-BART and MT-IPART is $O(n)$, which is significantly lower than that of SFIDMT-ART and MT-ART. Given that certain constants might be overlooked in time complexity calculations, empirical experiments have been conducted to assess and compare the performance of MT-BART and MT-IPART with other MG-generation algorithms (SFIDMT-ART and MT-ART). The experimental results demonstrated that both MT-BART and MT-IPART exhibit markedly superior performance to SFIDMT-ART and MT-ART in terms of test efficiency, while retaining high test effectiveness. Additionally, between the two proposed algorithms, MT-BART excels in efficiency, whereas MT-IPART excels in effectiveness.

In addition to MGs, MRs also play an important role in the successful implementation and performance of MT. With this consideration, Chapter 6 introduced a set of novel MRPs, intended to guide the identification of effective MRs tailored to various system type; MRPs serve as abstractions/templates encompassing multiple concrete MRs. This chapter has also presented a novel MT framework designed to guide the identification and application of MRPs. Furthermore, the concept of MRP trees was also introduced to categorize MRPs for enhancing user accessibility, providing users with an easy and quick way to find their target MRPs for reuse, reference, or inference. In this chapter, by collecting and classifying the proposed MRPs along with existing MRPs in the MT literature, two MRP trees were proposed. This chapter also reported on three case studies that employ the MRPs and MT framework to identify MRs for big data systems, resulting in the detection of multiple MR violations. Popular big data systems [87] have been tested and investigated, including e-commerce systems (Amazon and JD.com), map systems (Google Maps and Baidu Maps)

and machine translation systems (Google Translator, Microsoft Bing Translator and Baidu Translator).

In addition to identifying MRs and generating MGs from scratch, the performance of MT can be further augmented through the selection of appropriate MRs and MGs from existing ones. Nevertheless, despite certain studies delving into the selection of either MRs or MGs and presenting respective algorithms, only a few studies have considered both aspects simultaneously. In this context, Chapter 7 has introduced a novel metric from a black-box perspective for assessing the effectiveness of an MR-MG pair. This metric was designed following the same rationale that underlies the development of MG-generation algorithms (SFIDMT-ART, MT-BART, and MT-IPART): That is, achieving even distribution of STCs and FTCs throughout their respective input domains. Employing this metric as a foundation, this thesis has introduced a novel MR-MG pair selection algorithm, MRGS-ART. Subsequently, through empirical experimentation, this thesis has evaluated and compared its performance on public systems and MRs from previous studies related to MT, with the experimental results revealing that, in contrast to RT (the most commonly-used existing MR-MG pair selection algorithm [64]), MRGS-ART generally required fewer MGs to detect the first MR violation, while using an acceptable computational overhead to produce a group of MR-MG pairs.

Finally, Chapter 8 proposed employing MT and ME as supplementary testing, validation, and model-selection tools alongside traditional evaluation metrics for credit risk models. More specifically, the proposed method utilizes model attributes or properties based on business expectations and intuition to enable testers/researchers to predict how particular input alterations will impact model behavior or output. Subsequently, a case study was reported exploring the application of the proposed method for testing and validating popular credit risk models, including neural networks, gradient-boosting decision trees, and random forests. This chapter examined how models chosen through traditional model-fit evaluation metric (AUC) perform when evaluated with ME. Empirical experiments revealed multiple instances of MR violations. Furthermore, as the complexity of the model increases, the HMR violations become more extensive. Therefore, in the context of ML-based credit risk modeling, this chapter proposed the application of MT/ME as supplemental testing, validation, and model-selection tools, complementary to traditional model-fit measurements.

9.2 MAIN CONTRIBUTIONS

The specific contributions of this thesis are introduced as follows:

1. This thesis identified the MT-performance evaluation problem and the input-domain difference problem existing in the design, application, and evaluation of MG-generation algorithms. These two problems may negatively affect the performance of MG-generation algorithms. To improve the effectiveness and efficiency of MT, this thesis presented solutions to address them.

2. In order to improve the effectiveness and efficiency of MT, this thesis focused on the MRs and MGs. In particular, this thesis made the following achievements:
   a) This thesis presented a series of new MRPs (and MRIPs and MROPs) to guide the identification of effective MRs from scratch, as well as an MT framework designed to facilitates the identification and implementation of MRPs.
   b) By addressing the input-domain difference problem in a previously-published MG generation algorithm, this thesis presented a new MG-generation algorithm (SFIDMT-ART), which is capable of outperforming the original algorithm in terms of effectiveness and efficiency.
   c) By incorporating partition-based ART into MT, this thesis presented two new MG-generation algorithms (MT-BART and MT-IPART), which are able to perform significantly better than SFIDMT-ART in terms of efficiency while maintaining similar (or slightly worse) effectiveness.
   d) This thesis presented a new MR-MG pair selection algorithm (MRGS-ART) to automatically and adaptively select effective MR-MG pairs for execution from existing ones.
3. This thesis proposed using MT and ME as supplementary model testing, validation, and selection methodologies, in order to improve the performance of software testing (especially for ML-based credit risk models).

9.3 LIMITATION AND FUTURE WORK

This section discusses the main limitations of the research work presented in this thesis from the following three aspects, as well as the extensions to address those limitations and improve this thesis: Limitation and extensions to algorithms, limitation and extensions to MRPs and MRP trees, and limitation and extensions to the MT framework.

9.3.1 *Limitations and Extensions to Algorithms*

Although the proposed MG-generation algorithms and the MR-MG pair selection algorithm were capable of outperforming other algorithms in terms of effectiveness and/or efficiency, they still have drawbacks require research to further improve their performance: For example, SFIDMT-ART may require high computational complexity and time overhead to generate MGs; MT-BART and MT-IPART may sometimes require more MGs to detect the first MR violation compared to SFIDMT-ART; and the efficiency of MRGS-ART is still slightly higher than MT-RT, which is the most popular MR and MG selection algorithm in the literature of MT. With this consideration, future work will include exploring ways to address these limitations that may arise with the proposed algorithms, to further improve their performance. One possible solution is that, as the rationale behind these algorithms is inspired by ART, some advanced ART algorithms, or advanced methods proposed to enhance the performance of ART algorithms, may be applicable for enhancing the proposed

algorithms. Therefore, it is worth researching whether or not the performance of the proposed algorithms can be further enhanced based on the techniques from the ART-related studies, and how to incorporate those techniques into the four proposed algorithms.

9.3.2 *Limitations and Extensions to MRPs and MRP trees*

A primary limitation of the proposed MRPs and MRP trees is the lack of systematic methods for measuring the abstraction levels of MRPs. Typically, the identification of the abstraction level of MRPs still relies on human participants, such as MT experts. Considering this, this thesis will explore methods for formally and automatically measuring the abstraction levels of MRPs.

9.3.3 *Limitations and Extensions to the MT Framework*

In Chapter 6, an MT framework was proposed to facilitate the identification and application of MRs and MRPs (as well as MRIPs and MROPs). However, as shown in Section 2.2, the implementation of MT consists of many steps, each of which may have an impact on its performance. In addition, this thesis has also proposed methodologies to improve the performance of MT from different aspects, but the impact and relationships among them have not been studied yet. In this case, an overall framework can be proposed that can facilitate not only the identification and application of MRs and MRPs, but also other steps such as the generation of MGs and selection of MRs and MGs. This MT framework, for example, can include MRPs (for identifying MRs from scratch), SFIDMT-ART or MT-PART (for generating MGs from scratch), and MRGS-ART (for selecting MR-MG pairs from existing ones). Therefore, this thesis will explore the impact and relationships among these proposed algorithms and methodologies, with the aim of proposing an overall MT framework including all algorithms and methodologies proposed in this thesis to further improve the performance of MT. Once developed, this thesis will also apply it to the field of credit risk assessment to evaluate its performance and try to further improve the performance of MT as an supplementary testing tool.

BIBLIOGRAPHY

[1] ISO/IEC 25010:2011. *Systems and software engineering - systems and software quality requirements and evaluation (SQuaRE) - system and software quality models*. https://www.iso.org/standard/35733.html. [Online; accessed 15 May 2024]. 2011.

[2] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. "State-of-the-art in artificial neural network applications: A survey". In: *Heliyon* 4.11 (2018). DOI:10.1016/j.heliyon.2018.e00938, e00938.

[3] Amina Adadi and Mohammed Berrada. "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)". In: *IEEE access* 6 (2018). DOI:10.1109/ACCESS.2018.2870052, pp. 52138–52160.

[4] Wilhelmina Afua Addy, Adeola Olusola Ajayi-Nifise, Binaebi Gloria Bello, Sunday Tubokirifuruar Tula, Olubusola Odeyemi, and Titilola Falaiye. "AI in credit scoring: A comprehensive review of models and predictive analytics". In: *Global Journal of Engineering and Technology Advances* 18.02 (2024). DOI:10.30574/gjeta.2024.18.2.0029, pp. 118–129.

[5] Prathima Agrawal and Vishwani D. Agrawal. "Probabilistic analysis of random test generation method for irredundant combinational logic networks". In: *IEEE Transactions on Computers* 100.7 (1975). DOI:10.1109/T-C.1975.224289, pp. 691–695.

[6] Saurabh Agrawal, Purnima Ahirao, Saurabh Kumar, and Pinak Dere. "Credit Score Evaluation of Customer Using Machine Learning Algorithms". In: *Proceedings of the 4th International Conference on Advances in Science & Technology (ICAST'21)* (2021). DOI:10.2139/ssrn.3867420.

[7] Milam Aiken and Mina Park. "The efficacy of round-trip translation for MT evaluation". In: *Translation Journal* 14.1 (2010), pp. 1–10.

[8] Eman Alatawi, Tim Miller, and Harald Søndergaard. "Generating source inputs for metamorphic testing using dynamic symbolic execution". In: *Proceedings of the 1st International Workshop on Metamorphic Testing*. DOI:10.1145/2896971.2896980. 2016, pp. 19–25.

[9] Paul Eric Ammann and John C. Knight. "Data diversity: An approach to software fault tolerance". In: *IEEE Transactions on Computers* 37.4 (1988). DOI:10.1109/12.2185, pp. 418–425.

[10] Saswat Anand, Edmund K. Burke, Tsong Yueh Chen, John Clark, Myra B. Cohen, Wolfgang Grieskamp, Mark Harman, Mary Jean Harrold, and Phil Mcminn. "An Orchestrated Survey of Methodologies for Automated Software Test Case Generation". In: *Journal of Systems and Software* 86.8 (2013). DOI:10.1016/j.jss.2013.02.061, pp. 1978–2001.

[11] Stevão Alves de Andrade, Ítalo Santos, Claudinei Brito Junior, Misael Júnior, Simone R.S. de Souza, and Márcio E. Delamaro. "On applying metamorphic testing: An empirical study on academic search engines". In: *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)*. DOI:10.1109/MET.2019.00010. IEEE. 2019, pp. 9–16.

[12] Andrea Arcuri and Lionel Briand. "Adaptive random testing: An illusion of effectiveness?" In: *Proceedings of the 2011 International Symposium on Software Testing and Analysis*. DOI:10.1145/2001420.2001452. Association for Computing Machinery, 2011, pp. 265–275.

[13] Muhammad Ashfaq, Rubing Huang, Dave Towey, Michael Omari, Dmitry Yashunin, Patrick Kwaku Kudjo, and Tao Zhang. "SWFC-ART: A cost-effective approach for Fixed-Size-Candidate-Set Adaptive Random Testing through small world graphs". In: *Journal of Systems and Software* 180 (2021). DOI:10.1016/j.jss.2021.111008, p. 111008.

[14] Mahmuda Asrafi, Huai Liu, and Fei-Ching Kuo. "On testing effectiveness of metamorphic relations: A case study". In: *2011 Fifth International Conference on Secure Software Integration and Reliability Improvement*. DOI:10.1109/SSIRI.2011.21. IEEE. 2011, pp. 147–156.

[15] Franz Aurenhammer. "Voronoi diagrams-a survey of a fundamental geometric data structure". In: *ACM Computing Surveys (CSUR)* 23.3 (1991). DOI:10.1145/116873.116880, pp. 345–405.

[16] Bart Baesens, Rudy Setiono, Christophe Mues, and Jan Vanthienen. "Using neural network rule extraction and decision tables for credit-risk evaluation". In: *Management science* 49.3 (2003). DOI:10.1287/mnsc.49.3.312.12739, pp. 312–329.

[17] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. "The oracle problem in software testing: A survey". In: *IEEE Transactions on Software Engineering* 41.5 (2014). DOI:10.1109/TSE.2014.2372785, pp. 507–525.

[18] Arlinta C. Barus, Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Robert Merkel, and Gregg Rothermel. "A novel linear-order algorithm for adaptive random testing of programs with non-numeric inputs". In: *Technical Report TR-UNL-CSE-2014-0004* (2014).

[19] Arlinta Christy Barus, Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, and Heinz W. Schmidt. "The impact of source test case selection on the effectiveness of metamorphic testing". In: *2016 IEEE/ACM 1st International Workshop on Metamorphic Testing (MET)*. DOI:10.1145/2896971.2896977. IEEE. 2016, pp. 5–11.

[20] Hardik Bati, Leo Giakoumakis, Steve Herbert, and Aleksandras Surna. "A genetic approach for random testing of database systems". In: *Proceedings of the 33rd international conference on Very large data bases*. 2007, pp. 1243–1251.

[21] Gagandeep Batra and Jyotsna Sengupta. "An efficient metamorphic testing technique using genetic algorithm". In: *International Conference on Information Intelligence, Systems, Technology and Management*. DOI:10.1007/978-3-642-19423-8_19. Springer. 2011, pp. 180–188.

[22] Richard Bellman. "Dynamic programming". In: *Science* 153.3731 (1966). DOI:10.1126/science.153.3731.34, pp. 34–37.

[23] Imane Rhzioual Berrada, Fatima Zohra Barramou, and Omar Bachir Alami. "A review of Artificial Intelligence approach for credit risk assessment". In: *2022 2nd International Conference on Artificial Intelligence and Signal Processing (AISP)*. DOI:10.1109/AISP53593.2022.9760655. IEEE. 2022, pp. 1–5.

[24] Peter G. Bishop. "The variation of software survival time for different operational input profiles (or why you can wait a long time for a big bug to fail)". In: *FTCS-23 The Twenty-Third International Symposium on Fault-Tolerant Computing*. DOI:10.1109/FTCS.1993.627312. IEEE. 1993, pp. 98–107.

[25] Pierre Bourque and Richard E. Fairley. "SWEBOK v3.0: Guide to the software engineering body of knowledge". In: *IEEE Computer Society, Los Alamitos, CA* (2014), pp. 1–335.

[26] Andrew P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern Recognition* 30.7 (1997). DOI:10.1016/S0031-3203(96)00142-2, pp. 1145–1159.

[27] Joseph Breeden. "A Survey of Machine Learning in Credit Risk". In: *Journal of Credit Risk* 17.3 (2021). DOI:10.13140/RG.2.2.14520.37121.

[28] Joseph L. Breeden and Nikolay Dobrinov. "Quantifying Model Selection Risk in Macroeconomic Sensitivity Models". In: *Journal of Risk Model Validation* 16.3 (2022). DOI:10.21314/JRMV.2022.021.

[29] Leo Breiman. "Bagging predictors". In: *Machine Learning* 24.2 (1996). DOI:10.1007/BF00058655, pp. 123–140.

[30] Leo Breiman. "Random forests". In: *Machine Learning* 45.1 (2001). DOI:10.1023/A:1010933404324, pp. 5–32.

[31] Iain Brown and Christophe Mues. "An experimental comparison of classification algorithms for imbalanced credit scoring data sets". In: *Expert Systems with Applications* 39.3 (2012). DOI:10.1016/j.eswa.2011.09.033, pp. 3446–3453.

[32] Joshua Brown, Zhi Quan Zhou, and Yang-Wai Chow. "Metamorphic Testing of Navigation Software: A Pilot Study with Google Maps". In: *Proceedings of the 51st Annual Hawaii International Conference on System Sciences (HICSS-51)*. DOI:10.24251/HICSS.2018.713. 2018, pp. 5687–5696.

[33] Joshua Brown, Zhi Quan Zhou, and Yang-Wai Chow. "Metamorphic Testing of Mapping Software". In: *Towards Integrated Web, Mobile, and IoT Technology*. DOI:10.1007/978-3-030-28430-5_1. Springer, 2019, pp. 1–20.

[34] Paulo M.S. Bueno, Mario Jino, and W. Eric Wong. "Diversity oriented test data generation using metaheuristic search techniques". In: *Information Sciences* 259 (2014). DOI:10.1016/j.ins.2011.01.025, pp. 490–509.

[35] Paulo M.S. Bueno, W. Eric Wong, and Mario Jino. "Improving random test sets using the diversity oriented test data generation". In: *Proceedings of the 2nd international workshop on Random testing: co-located with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007)*. DOI:10.1145/1292414.1292419. 2007, pp. 10–17.

[36] Massimo Buscema. "Back propagation neural networks". In: *Substance use & misuse* 33.2 (1998). DOI:10.3109/10826089809115863, pp. 233–270.

[37] Yuxiang Cao, Zhi Quan Zhou, and Tsong Yueh Chen. "On the correlation between the effectiveness of metamorphic relations and dissimilarities of test case executions". In: *2013 13th International Conference on Quality Software*. DOI:10.1109/QSIC.2013.43. IEEE. 2013, pp. 153–162.

[38] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. "Machine learning interpretability: A survey on methods and metrics". In: *Electronics* 8.8 (2019). DOI:10.3390/electronics8080832, p. 832.

[39] F. T. Chan, Tsong Yueh Chen, I. K. Mak, and Yuen-Tak Yu. "Proportional sampling strategy: Guidelines for software testing practitioners". In: *Information and Software Technology* 38.12 (1996). DOI:10.1016/0950-5849(96)01103-2, pp. 775–782.

[40] Kwok Ping Chan, Tsong Yueh Chen, and Dave Towey. "Restricted Random Testing". In: *Software Quality — ECSQ 2002*. DOI:10.1007/3-540-47984-8_35. Springer Berlin Heidelberg, 2002, pp. 321–330.

[41] Kwok Ping Chan, Tsong Yueh Chen, and Dave Towey. "Forgetting test cases". In: *Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06)*. Vol. 1. DOI:10.1109/COMPSAC.2006.43. IEEE. 2006, pp. 485–494.

[42] Kwok Ping Chan, Tsong Yueh Chen, and Dave Towey. "Restricted random testing: Adaptive random testing by exclusion". In: *International Journal of Software Engineering and Knowledge Engineering* 16.04 (2006). DOI:10.1142/S0218194006002926, pp. 553–584.

[43] Kwok Ping Chan, Yueh Chen Chen, and Dave Towey. "Adaptive Random Testing with Filtering: An Overhead Reduction Technique". In: *Proceedings of the 17th International Conference on Software Engineering & Knowledge Engineering (SEKE'05)*. 2005, pp. 292–299.

[44] Wing Kwong Chan, Tsong Yueh Chen, Heng Lu, T. H. Tse, and Stephen S Yau. "Integration testing of context-sensitive middleware-based applications: A metamorphic approach". In: *International Journal of Software Engineering and Knowledge Engineering* 16.05 (2006). DOI:10.1142/S0218194006002951, pp. 677–703.

[45] Wing Kwong Chan, Shing Chi Cheung, and Karl R. P. H. Leung. "A metamorphic testing approach for online testing of service-oriented software applications". In: *International Journal of Web Services Research (IJWSR)* 4.2 (2007). DOI:10.4018/jwsr.2007040103, pp. 61–81.

[46] Yung-Chia Chang, Kuei-Hu Chang, and Guan-Jhih Wu. "Application of eXtreme gradient boosting trees in the construction of credit risk assessment models for financial institutions". In: *Applied Soft Computing* 73 (2018). DOI:10.1016/j.asoc.2018.09.029, pp. 914–920.

[47] Jinfu Chen, Fei-Ching Kuo, Tsong Yueh Chen, Dave Towey, Chenfei Su, and Rubing Huang. "A similarity metric for the inputs of OO programs and its application in adaptive random testing". In: *IEEE Transactions on Reliability* 66.2 (2016). DOI:10.1109/TR.2016.2628759, pp. 373–402.

[48] Jinfu Chen, Lili Zhu, Tsong Yueh Chen, Dave Towey, Fei-Ching Kuo, Rubing Huang, and Yuchi Guo. "Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering". In: *Journal of Systems and Software* 135 (2018). DOI:10.1016/j.jss.2017.09.031, pp. 107–125.

[49] Leilei Chen, Lizhi Cai, Jiang Liu, Zhenyu Liu, Shiyan Wei, and Pan Liu. "An optimized method for generating cases of metamorphic testing". In: *2012 6th International Conference on New Trends in Information Science, Service Science and Data Mining (ISSDM2012)*. IEEE. 2012, pp. 439–443.

[50] M. H. Chen, Michael R. Lyu, and W. Eric Wong. "Effect of code coverage on software reliability measurement". In: *IEEE Transactions on reliability* 50.2 (2001). DOI:10.1109/24.963124, pp. 165–170.

[51] Mei-Hwa Chen, Michael R. Lyu, and W. Eric Wong. "An empirical study of the correlation between code coverage and reliability estimation". In: *Proceedings of the 3rd International Software Metrics Symposium*. DOI:10.1109/METRIC.1996.492450. IEEE. 1996, pp. 133–141.

[52] Tsong Yueh Chen, Shing Chi Cheung, and Siu Ming Yiu. "Metamorphic testing: A new approach for generating next test cases". In: *Department of Computer Science, Hong Kong University of Science and Technology, Tech. Rep. HKUST-CS98-01* abs/2002.12543 (1998). DOI:10.48550/arXiv.2002.12543.

[53] Tsong Yueh Chen and De Hao Huang. "Adaptive random testing by localization". In: *11th Asia-Pacific Software Engineering Conference*. DOI:10.1109/APSEC.2004.17. IEEE. 2004, pp. 292–298.

[54] Tsong Yueh Chen, De Hao Huang, and Fei-Ching Kuo. "Adaptive Random Testing by Balancing". In: *Proceedings of the 2nd International Workshop on Random Testing: Co-Located with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007)*. DOI:10.1145/1292414.1292418. ACM New York, NY, USA, 2007, pp. 2–9.

[55] Tsong Yueh Chen, De Hao Huang, T. H. Tse, and Zongyuan Yang. "An innovative approach to tackling the boundary effect in adaptive random testing". In: *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. DOI:10.1109/HICSS.2007.67. IEEE. 2007, 262a–262a.

[56] Tsong Yueh Chen, De Hao Huang, T. H. Tse, and Zhi Quan Zhou. "Case studies on the selection of useful relations in metamorphic testing". In: *Proceedings of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC 2004)*. Citeseer. 2004, pp. 569–583.

[57] Tsong Yueh Chen, De Hao Huang, and Zhi Quan Zhou. "On adaptive random testing through iterative partitioning". In: *journal of information science and engineering* 27.4 (2011). DOI:10.1007/11767077_13, pp. 1449–1472.

[58] Tsong Yueh Chen, Fei-Ching Kuo, and Huai Liu. "Distribution metric driven adaptive random testing". In: *2007 7th International Conference on Quality Software (QSIC 2007)*. DOI:10.1007/11767077_13. IEEE. 2007, pp. 274–279.

[59] Tsong Yueh Chen, Fei-Ching Kuo, and Huai Liu. "Enhancing adaptive random testing through partitioning by edge and centre". In: *2007 Australian Software Engineering Conference (ASWEC'07)*. DOI:10.1109/ASWEC.2007.20. IEEE. 2007, pp. 265–273.

[60] Tsong Yueh Chen, Fei-Ching Kuo, and Huai Liu. "On test case distributions of adaptive random testing". In: *Proceedings of the 19th International Conference on Software Engineering & Knowledge Engineering (SEKE'07)*. DOI:10.1109/TSE.2019.2942921. 2007, pp. 141–144.

[61] Tsong Yueh Chen, Fei-Ching Kuo, and Huai Liu. "Distributing test cases more evenly in adaptive random testing". In: *Journal of Systems and Software* 81.12 (2008). DOI:10.1016/j.jss.2008.03.062, pp. 2146–2162.

[62] Tsong Yueh Chen, Fei-Ching Kuo, and Huai Liu. "Adaptive random testing based on distribution metrics". In: *Journal of Systems and Software* 82.9 (2009). DOI:10.1016/j.jss.2009.05.017, pp. 1419–1433.

[63] Tsong Yueh Chen, Fei-Ching Kuo, and Huai Liu. "Application of a failure driven test profile in random testing". In: *IEEE Transactions on Reliability* 58.1 (2009). DOI:10.1109/TR.2008.2011687, pp. 179–192.

[64] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. "Metamorphic testing: A review of challenges and opportunities". In: *ACM Computing Surveys (CSUR)* 51.1 (2018). DOI:10.1145/3143561, pp. 1–27.

[65] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, and W. Eric Wong. "Does adaptive random testing deliver a higher confidence than random testing?" In: *2008 8th International Conference on Quality Software*. DOI:10.1109/QSIC.2008.23. IEEE. 2008, pp. 145–154.

[66] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, and W. Eric Wong. "Code coverage of adaptive random testing". In: *IEEE Transactions on Reliability* 62.1 (2013). DOI:10.1109/TR.2013.2240898, pp. 226–237.

[67] Tsong Yueh Chen, Fei-Ching Kuo, Ying Liu, and Antony Tang. "Metamorphic Testing and Testing with Special Values". In: *Proceedings of the 5th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'04)*. 2004, pp. 128–134.

[68] Tsong Yueh Chen, Fei-Ching Kuo, Wenjuan Ma, Willy Susilo, Dave Towey, Jeffrey Voas, and Zhi Quan Zhou. "Metamorphic Testing for Cybersecurity". In: *Computer* 49.6 (2016), pp. 48–55. DOI: 10.1109/MC.2016.176.

[69] Tsong Yueh Chen, Fei-Ching Kuo, and Robert Merkel. "On the Statistical Properties of the F-Measure". In: *Fourth International Conference onQuality Software, 2004. QSIC 2004. Proceedings.* DOI:10.1109/QSIC.2004.1357955. IEEE Computer Society, Los Alamitos, CA, 2004, pp. 146–153.

[70] Tsong Yueh Chen, Fei-Ching Kuo, and Robert Merkel. "On the Statistical Properties of Testing Effectiveness Measures". In: *Journal of Systems and Software* 79.5 (2006). DOI:10.1016/j.jss.2005.05.029, pp. 591–601.

[71] Tsong Yueh Chen, Fei-Ching Kuo, Robert G. Merkel, and Sebastian P. Ng. "Mirror Adaptive Random Testing". In: *Information & Software Technology* (2003). DOI:10.1109/QSIC.2003.1319079, pp. 4–11.

[72] Tsong Yueh Chen, Fei-Ching Kuo, Robert G. Merkel, and Sebastian P. Ng. "Mirror adaptive random testing". In: *Information and Software Technology* 46.15 (2004), pp. 1001–1010.

[73] Tsong Yueh Chen, Fei-Ching Kuo, Robert G. Merkel, and T. H. Tse. "Adaptive random testing: The art of test case diversity". In: *Journal of Systems and Software* 83.1 (2010). DOI:10.1016/j.jss.2009.02.022, pp. 60–66.

[74] Tsong Yueh Chen, Fei-Ching Kuo, Dave Towey, and Zhi Quan Zhou. "A Revisit of Three Studies Related to Random Testing". In: *Science China Information Sciences* 58 (2015). DOI:10.1007/s11432-015-5314-x, pp. 1–9.

[75] Tsong Yueh Chen, Fei-Ching Kuo, and Zhi Quan Zhou. "On the Relationships between the Distribution of Failure-Causing Inputs and Effectiveness of Adaptive Random Testing". In: *Proceedings of the 17th International Conference on Software Engineering & Knowledge Engineering (SEKE'05)*. 2005, pp. 306–311.

[76] Tsong Yueh Chen, Fei-Ching Kuo, and Zhi Quan Zhou. "On favourable conditions for adaptive random testing". In: *International Journal of Software Engineering and Knowledge Engineering* 17.06 (2007). DOI:10.1142/S0218194007003501, pp. 805–825.

[77] Tsong Yueh Chen, Hing Leung, and I. K. Mak. "Adaptive random testing". In: *Proceedings of the 9th Asian Computing Science Conference (ASIAN'04)*. Vol. 3321. DOI:10.1007/978-3-540-30502-6_23. Lecture Notes in Computer Science, 2004, pp. 320–329.

[78] Tsong Yueh Chen, R. Merkel, P. K. Wong, and G. Eddy. "Adaptive Random Testing Through Dynamic Partitioning". In: *Proceedings of the Quality Software, Fourth International Conference*. DOI:10.1109/QSIC.2004.1357947. IEEE Computer Society, Los Alamitos, CA, 2004, pp. 79–86.

[79] Tsong Yueh Chen and Robert Merkel. "Efficient and Effective Random Testing Using the Voronoi Diagram". In: *Proceedings of the Australian Software Engineering Conference*. DOI:10.1109/ASWEC.2006.25. IEEE Computer Society, Los Alamitos, CA, 2006, pp. 300–299.

[80] Tsong Yueh Chen and Robert Merkel. "Quasi-random testing". In: *IEEE Transactions on Reliability* 56.3 (2007). DOI:10.1145/1101908.1101957, pp. 562–568.

[81] Tsong Yueh Chen and Robert Merkel. "An upper bound on software testing effectiveness". In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 17.3 (2008). DOI:10.1145/1363102.1363107, pp. 1–27.

[82] Tsong Yueh Chen, Pak-Lok Poon, and Xiaoyuan Xie. "METRIC: METamorphic Relation Identification based on the Category-choice framework". In: *Journal of Systems and Software* 116 (2016). DOI:10.1016/j.jss.2015.07.037, pp. 177–190.

[83] Tsong Yueh Chen, T. H. Tse, and Yuen-Tak Yu. "Proportional sampling strategy: A compendium and some insights". In: *Journal of Systems and Software* 58.1 (2001). DOI:10.1016/S0164-1212(01)00028-0, pp. 65–81.

[84] Tsong Yueh Chen, T. H. Tse, and Zhi Quan Zhou. "Fault-based testing without the need of oracles". In: *Information and Software Technology* 45.1 (2003). DOI:10.1016/S0950-5849(02)00129-5, pp. 1–9.

[85] Tsong Yueh Chen, T. H. Tse, and Zhi Quan Zhou. "Semi-proving: An integrated method for program proving, testing, and debugging". In: *IEEE Transactions on Software Engineering* 37.1 (2010). DOI:10.1109/TSE.2010.23, pp. 109–125.

[86] Tsong Yueh Chen, Zhi Quan Zhou, and De Hao Huang. "Adaptive random testing through iterative partitioning". In: *International Conference on Reliable Software Technologies*. DOI:10.1007/11767077_13. Springer. 2006, pp. 155–166.

[87] Xue-Qi Cheng, Xiao Long Jin, Yuanzhuo Wang, Jiafeng Guo, Tieying Zhang, and Guojie Li. "Survey on big data system and analytic technology". In: *Journal of software* 25.9 (2014). DOI:10.13328/j.cnki.jos.004674, pp. 1889–1908.

[88] Cliff Chow, Tsong Yueh Chen, and T. H. Tse. "The art of divide and conquer: An innovative approach to improving the efficiency of adaptive random testing". In: *2013 13th International Conference on Quality Software*. DOI:10.1109/QSIC.2013.19. IEEE. 2013, pp. 268–275.

[89] Ilinca Ciupa, Andreas Leitner, Manuel Oriol, and Bertrand Meyer. "ARTOO: adaptive random testing for object-oriented software". In: *Proceedings of the 30th international conference on Software engineering (ICSE)*. DOI:10.1145/1368088.1368099. 2008, pp. 71–80.

[90] Jacob Cohen. *Statistical power analysis for the behavioral sciences (2nd ed.)* Hillsdale, MI, USA: Hillsdale, 1988.

[91] Andrew T. Collins, John M. Rose, and Stephane Hess. "Interactive stated choice surveys: A study of air travel behaviour". In: *Transportation* 39.1 (2012). DOI:10.1007/s11116-011-9327-z, pp. 55–79.

[92] Kristof Coussement and Dries F. Benoit. "Interpretable data science for decision making". In: *Decision Support Systems* 150 (2021). DOI:10.1016/j.dss.2021.113664, p. 113664.

[93] Randall Davis, Andrew W. Lo, Sudhanshu Mishra, Arash Nourian, Manish Singh, Nicholas Wu, and Ruixun Zhang. "Explainable Machine Learning Models of Consumer Credit Risk". In: *SSRN Electronic Journal* (2022). DOI:10.2139/ssrn.4006840.

[94] Rober Hunter Davis, D. B. Edelman, and A. J. Gammerman. "Machine-learning algorithms for credit-card applications". In: *IMA Journal of Management Mathematics* 4.1 (1992). DOI:10.1093/imaman/4.1.43, pp. 43–51.

[95] Alexandru M. Degeratu, Arvind Rangaswamy, and Jianan Wu. "Consumer choice behavior in online and traditional supermarkets: The effects of brand name, price, and other search attributes". In: *International Journal of research in Marketing* 17.1 (2000). DOI:10.1016/S0167-8116(00)00005-7, pp. 55–78.

[96] Deloitte. *Global Powers of Retailing 2022*. `https://www.deloitte.com/cbc/en/Industries/consumer/analysis/global-powers-of-retailing-2022.html`. [Online; accessed 15 May 2024]. 2022.

[97] Vijay S. Desai, Jonathan N. Crook, and George A. Overstreet Jr. "A comparison of neural networks and linear scoring models in the credit union environment". In: *European Journal of Operational Research* 95.1 (1996). DOI:10.1016/0377-2217(95)00246-4, pp. 24–37.

[98] Edsger W. Dijkstra. "The Humble Programmer". In: *Communications of The ACM - CACM* 15.10 (1972). DOI:10.1145/355604.361591, pp. 859–866.

[99] Junhua Ding, Tong Wu, Jun Lu, and Xin-Hua Hu. "Self-checked metamorphic testing of an image processing program". In: *2010 Fourth International Conference on Secure Software Integration and Reliability Improvement*. DOI:10.1109/SSIRI.2010.25. IEEE. 2010, pp. 190–197.

[100] Guowei Dong. "Metamorphic testing techniques for error detection efficiency". PhD thesis. Nanjing, China: School of Computer Science and Engineering, Southeast University, 2009.

[101] Guowei Dong, Tao Guo, and Puhan Zhang. "Security assurance with program path analysis and metamorphic testing". In: *2013 IEEE 4th International Conference on Software Engineering and Service Science (ICSESS)*. DOI:10.1109/ICSESS.2013.6615286. IEEE. 2013, pp. 193–197.

[102] Guowei Dong, Changhai Nie, Baowen Xu, and Lulu Wang. "An effective iterative metamorphic testing algorithm based on program path analysis". In: *2007 7th International Conference on Quality Software (QSIC 2007)*. DOI:10.1109/QSIC.2007.4385510. IEEE. 2007, pp. 292–297.

[103] Guowei Dong, Baowen Xu, Lin Chen, Changhai Nie, and Lulu Wang. "Case studies on testing with compositional metamorphic relations". In: *Journal of Southeast University (English Edition)* 24.4 (2008), pp. 437–443.

[104] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. "A survey on ensemble learning". In: *Frontiers of Computer Science* 14 (2020). DOI:10.1007/s11704-019-8208-z, pp. 241–258.

[105] A. D. Dongare, R. R. Kharde, and Amit D. Kachare. "Introduction to artificial neural network". In: *International Journal of Engineering and Innovative Technology (IJEIT)* 2.1 (2012), pp. 189–194.

[106] Robert Dorfman. "A formula for the Gini coefficient". In: *The review of economics and statistics* (1979). DOI:10.2307/1924845, pp. 146–149.

[107] Bernard Dushimimana, Yvonne Wambui, Timothy Lubega, and Patrick E. McSharry. "Use of Machine Learning Techniques to Create a Credit Score Model for Airtime Loans". In: *Journal of Risk and Financial Management* 13.8 (2020). DOI:10.3390/jrfm13080180, p. 180.

[108] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8 (2006). DOI:10.1016/j.patrec.2005.10.010, pp. 861–874.

[109] FICO. *Machine learning and FICO scores*. https://www.fico.com/en/resource-access/download/6559. [Online; accessed 15 May 2024]. 2018.

[110] Roy Thomas Fielding. "Architectural styles and the design of network-based software architectures". PhD thesis. University of California, Irvine, 2000.

[111] George B. Finelli. "NASA software failure characterization experiments". In: *Reliability Engineering & System Safety* 32.1-2 (1991). DOI:10.1016/0951-8320(91)90052-9, pp. 155–169.

[112] Steven Finlay. *Credit scoring, response modeling, and insurance rating: A practical guide to forecasting consumer behavior*. Springer, 2012.

[113] Justin E. Forrester and Barton P. Miller. "An Empirical Study of the Robustness of Windows NT Applications Using Random Testing". In: *Proceedings of the 4th Conference on USENIX Windows Systems Symposium - Volume 4*. Vol. 4. USENIX Association, 2000, pp. 59–68.

[114] Jerome H. Friedman. "Greedy function approximation: A gradient boosting machine". In: *Annals of Statistics* 29.5 (2001). DOI:10.1214/AOS/1013203451, pp. 1189–1232.

[115] Halina Frydman, Edward I. Altman, and Duen-Li Kao. "Introducing Recursive Partitioning for Financial Classification: The Case of Financial Distress". In: *The Journal of Finance* 40.1 (1985). DOI:10.1111/j.1540-6261.1985.tb04949.x, pp. 269–291.

[116] Jixin Geng and Jiongmin Zhang. "A new method to solve the "boundary effect" of adaptive random testing". In: *2010 International Conference on Educational and Information Technology*. Vol. 1. DOI:10.1109/ICEIT.2010.5607704. IEEE. 2010, pp. V1–298.

[117] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering, 2nd ed*. Pearson, 2002.

[118] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. "Explaining explanations: An overview of interpretability of machine learning". In: *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. DOI:10.1109/DSAA.2018.00018. IEEE. 2018, pp. 80–89.

[119] Anthony T. C. Goh. "Back-propagation neural networks for modeling complex systems". In: *Artificial intelligence in engineering* 9.3 (1995). DOI:10.1016/0954-1810(94)00011-S, pp. 143–151.

[120] Arnaud Gotlieb and Bernard Botella. "Automated metamorphic testing". In: *Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC'03)*. DOI:10.1109/CMPSAC.2003.1245319. IEEE Computer Society, Los Alamitos, CA. 2003, pp. 34–40.

[121] Jean-Christophe Goulet-Pelletier and Denis Cousineau. "A review of effect sizes and their confidence intervals, Part I: The Cohen's d family". In: *The Quantitative Methods for Psychology* 14.4 (2018). DOI:10.20982/tqmp.14.4.p242, pp. 242–265.

[122] Brandon M. Greenwell. "pdp: An R package for constructing partial dependence plots". In: *The R Journal* 9.1 (2017), p. 421.

[123] Richard Hamlet. "Random testing". In: *Encyclopedia of software Engineering* (2002). DOI:10.1002/0471028959.sof268.

[124] James A. Hanley and Barbara J. McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve". In: *Radiology* 143.1 (1982). DOI:10.1148/radiology.143.1.7063747, pp. 29–36.

[125] Mark Harman and Phil McMinn. "A theoretical and empirical study of search-based testing: Local, global, and hybrid search". In: *IEEE Transactions on Software Engineering* 36.2 (2009). DOI:10.1109/TSE.2009.71, pp. 226–247.

[126] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "Boosting and additive trees". In: *The elements of statistical learning*. DOI:10.1007/978-0-387-84858-7_10. Springer, 2009, pp. 337–387.

[127] Douglas M. Hawkins. "The problem of overfitting". In: *Journal of chemical information and computer sciences* 44.1 (2004). DOI:10.1021/ci0342472, pp. 1–12.

[128] Jane Huffman Hayes and A. Jefferson Offutt. "Increased software reliability through input validation analysis and testing". In: *Proceedings 10th International Symposium on Software Reliability Engineering (Cat. No. PR00443)*. DOI:10.1109/ISSRE.1999.809325. IEEE. 1999, pp. 199–209.

[129] Pinjia He, Clara Meister, and Zhendong Su. "Structure-invariant testing for machine translation". In: *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. DOI:10.1145/3377811.3380339. IEEE. 2020, pp. 961–973.

[130] Hadi Hemmati, Andrea Arcuri, and Lionel Briand. "Reducing the cost of model-based testing through test case diversity". In: *Testing Software and Systems*. DOI:10.1007/978-3-642-16573-3_6. Springer Berlin Heidelberg, 2010, pp. 63–78.

[131] Hadi Hemmati, Andrea Arcuri, and Lionel Briand. "Empirical investigation of the effects of test suite properties on similarity-based test case selection". In: *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*. DOI:10.1109/ICST.2011.12. IEEE Computer Society, 2011, pp. 327–336.

[132] Hadi Hemmati, Andrea Arcuri, and Lionel Briand. "Achieving scalable model-based testing through test case diversity". In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 22.1 (2013). DOI:10.1109/ICST.2011.12, pp. 1–42.

[133] Akhil Bandhu Hens and Manoj Kumar Tiwari. "Computational time reduction for credit scoring: An integrated approach based on support vector machine and stratified sampling method". In: *Expert Systems with Applications* 39.8 (2012). DOI:10.1016/j.eswa.2011.12.057, pp. 6774–6781.

[134] M. S. Hossain. "Challenges of software quality assurance and testing". In: *International Journal of Software Engineering and Computer Systems* 4.1 (2018). DOI:10.15282/ijsecs.4.1.2018.11.0044, pp. 133–144.

[135] J. C. Huang. "An approach to program testing". In: *ACM Computing Surveys (CSUR)* 7.3 (1975). DOI:10.1145/356651.356652, pp. 113–128.

[136] Rubing Huang, Haibo Chen, Weifeng Sun, and Dave Towey. "Candidate test set reduction for adaptive random testing: An overheads reduction technique". In: *Science of Computer Programming* 214 (2022). DOI:10.1016/j.scico.2021.102730, p. 102730.

[137] Rubing Huang, Chenhui Cui, Dave Towey, Weifeng Sun, and Junlong Lian. "VPP-ART: An Efficient Implementation of Fixed-Size-Candidate-Set Adaptive Random Testing Using Vantage Point Partitioning". In: *IEEE Transactions on Reliability* 72.4 (2023). DOI:10.1109/TR.2022.3218602, pp. 1632–1647.

[138] Rubing Huang, Huai Liu, Xiaodong Xie, and Jinfu Chen. "Enhancing mirror adaptive random testing through dynamic partitioning". In: *Information and Software Technology* 67 (2015). DOI:10.1016/j.infsof.2015.06.003, pp. 13–29.

[139] Rubing Huang, Weifeng Sun, Haibo Chen, Chenhui Cui, and Ning Yang. "A nearest-neighbor divide-and-conquer approach for adaptive random testing". In: *Science of Computer Programming* 215 (2022). DOI:10.1016/j.scico.2021.102743, p. 102743.

[140] Rubing Huang, Weifeng Sun, Yinyin Xu, Haibo Chen, Dave Towey, and Xin Xia. "A survey on adaptive random testing". In: *IEEE Transactions on Software Engineering* 47.10 (2021). DOI:10.1109/TSE.2019.2942921, pp. 2052–2083.

[141] Rubing Huang, Xiaodong Xie, Jinfu Chen, and Yansheng Lu. "Failure-detection capability analysis of implementing parallelism in adaptive random testing algorithms". In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. DOI:10.1145/2480362.2480562. 2013, pp. 1049–1054.

[142] Zhanwei Hui and Song Huang. "A formal model for metamorphic relation decomposition". In: *2013 Fourth World Congress on Software Engineering*. DOI:10.1109/WCSE.2013.14. IEEE. 2013, pp. 64–68.

[143] Zhanwei Hui and Song Huang. "MD-ART: A Test Case Generation Method without Test Oracle Problem". In: *Proceedings of the 1st International Workshop on Specification, Comprehension, Testing, and Debugging of Concurrent Programs*. DOI:10.1145/2975954.2975959. ACM New York, NY, USA, 2016, pp. 27–34.

[144] Zhanwei Hui, Xiaojuan Wang, Song Huang, and Sen Yang. "MT-ART: A Test Case Generation Method Based on Adaptive Random Testing and Metamorphic Relation". In: *IEEE Transactions on Reliability* 70.4 (2021). DOI:10.1109/TR.2021.3106389, pp. 1397–1421.

[145] Muhammad Zohaib Iqbal, Andrea Arcuri, and Lionel Briand. "Combining search-based and adaptive random testing strategies for environment model-based testing of real-time embedded systems". In: *International Symposium on Search Based Software Engineering*. DOI:10.1007/978-3-642-33119-0_11. Springer. 2012, pp. 136–151.

[146] "ISO/IEC/IEEE International Standard - Systems and software engineering – Vocabulary". In: *ISO/IEC/IEEE 24765:2010(E)* (2010), pp. 1–418. DOI: 10.1109/IEEESTD.2010.5733835.

[147] Joxan Jaffar and Jean-Louis Lassez. "Constraint logic programming". In: *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. DOI:10.1145/41625.41635. 1987, pp. 111–119.

[148] Herbert L. Jensen. "Using neural networks for credit scoring". In: *Managerial Finance* (1992). DOI:10.1108/eb013696.

[149] Yue Jia and Mark Harman. "An analysis and survey of the development of mutation testing". In: *IEEE Transactions on Software Engineering* 37.5 (2011). DOI:10.1109/TSE.2010.62, pp. 649–678.

[150] Bo Jiang, Zhenyu Zhang, Wing Kwong Chan, and T. H. Tse. "Adaptive random test case prioritization". In: *2009 IEEE/ACM International Conference on Automated Software Engineering*. DOI:10.1109/ASE.2009.77. 2009, pp. 233–244.

[151] Mingyue Jiang, Tsong Yueh Chen, Fei-Ching Kuo, and Zuohua Ding. "Testing central processing unit scheduling algorithms using metamorphic testing". In: *2013 IEEE 4th International Conference on Software Engineering and Service Science (ICSESS)*. DOI:10.1109/ICSESS.2013.6615365. IEEE. 2013, pp. 530–536.

[152] Hao Jin, Yanyan Jiang, Na Liu, Chang Xu, Xiaoxing Ma, and Jian Lu. "Concolic Metamorphic Debugging". In: *2015 IEEE 39th Annual Computer Software and Applications Conference*. Vol. 2. IEEE. 2015, pp. 232–241. DOI: 10.1109/COMPSAC.2015.79.

[153] René Just and Franz Schweiggert. "Automating software tests with partial oracles in integrated environments". In: *Proceedings of the 5th Workshop on Automation of Software Test*. DOI:10.1145/1808266.1808280. 2010, pp. 91–94.

[154] René Just and Franz Schweiggert. "Automating unit and integration testing with partial oracles". In: *Software Quality Journal* 19.4 (2011). DOI:10.1007/s11219-011-9151-x, p. 753.

[155] Upulee Kanewala. "Techniques for automatic detection of metamorphic relations". In: *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*. DOI:10.1109/ICSTW.2014.62. IEEE. 2014, pp. 237–238.

[156] Upulee Kanewala and James M. Bieman. "Using machine learning techniques to detect metamorphic relations for programs without test oracles". In: *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*. DOI:10.1109/ISSRE.2013.6698899. IEEE. 2013, pp. 1–10.

[157] Roger E. Kirk. "Practical significance: A concept whose time has come". In: *Educational and psychological measurement* 56.5 (1996). DOI:10.1177/0013164496056005002, pp. 746–759.

[158] Philipp Koehn and Christof Monz. "Manual and automatic evaluation of machine translation between European languages". In: *Proceedings on the Workshop on Statistical Machine Translation*. DOI:10.3115/1654650.1654666. 2006, pp. 102–121.

[159] Ron Kohavi and Roger Longbotham. "Online experiments: Lessons learned". In: *Computer* 40.9 (2007). DOI:10.1109/MC.2007.328, pp. 103–105.

[160] Harsh Kukreja, N. Bharath, C. S. Siddesh, and S. Kuldeep. "An introduction to artificial neural network". In: *International Journal of Advance Research and Innovative Ideas in Education* 1.5 (2016). DOI:10.1016/j.patrec.2005.10.010, pp. 27–30.

[161] Vaibhav Kumar and M. L. Garg. "Predictive analytics: A review of trends and techniques". In: *International Journal of Computer Applications* 182.1 (2018). DOI:10.5120/ijca2018917434, pp. 31–37.

[162] Fei-Ching Kuo. "An indepth study of mirror adaptive random testing". In: *2009 9th International Conference on Quality Software*. DOI:10.1109/QSIC.2009.15. IEEE. 2009, pp. 51–58.

[163] Fei-Ching Kuo, Tsong Yueh Chen, Huai Liu, and W. K. Chan. "Enhancing adaptive random testing in high dimensional input domains". In: *Proceedings of the 2007 ACM symposium on Applied computing*. DOI:10.1145/1244002.1244316. 2007, pp. 1467–1472.

[164] Fei-Ching Kuo, Tsong Yueh Chen, Huai Liu, and Wing Kwong Chan. "Enhancing adaptive random testing for programs with high dimensional input domains or failure-unrelated parameters". In: *Software Quality Journal* 16.3 (2008). DOI:10.1007/s11219-008-9047-6, pp. 303–327.

[165] Fei-Ching Kuo, Tsong Yueh Chen, and Wing K. Tam. "Testing embedded software by metamorphic testing: A wireless metering system case study". In: *2011 IEEE 36th Conference on Local Computer Networks*. DOI:10.1109/LCN.2011.6115306. IEEE. 2011, pp. 291–294.

[166] Fei-Ching Kuo, Kwan Yong Sim, Chang-Ai Sun, S.-F. Tang, and Zhi Quan Zhou. "Enhanced random testing for programs with high dimensional input domains". In: *Software Quality Journal* 16 (2007). DOI:10.1109/QSIC.2006.8, pp. 135–140.

[167] Fei-Ching Kuo, Zhi Quan Zhou, Jun Ma, and Guang Quan Zhang. "Metamorphic testing of decision support systems: A case study". In: *IET software* 4.4 (2010). DOI:10.1049/iet-sen.2009.0084, pp. 294–301.

[168] Janusz W. Laski and Bogdan Korel. "A data flow oriented program testing strategy". In: *IEEE Transactions on Software Engineering* 3 (1983). DOI:10.1109/TSE.1983.236871, pp. 347–354.

[169] Vu Le, Mehrdad Afshari, and Zhendong Su. "Compiler validation via equivalence modulo inputs". In: *ACM SIGPLAN Notices* 49.6 (2014). DOI:10.1145/2666356.2594334, pp. 216–226.

[170] Dickson T. S. Lee, Zhi Quan Zhou, and T. H. Tse. "Metamorphic Robustness Testing of Google Translate". In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20)*. DOI:10.1145/3387940.3391484. ACM New York, NY, USA, 2020, pp. 388–395.

[171] Stefan Lessmann, Bart Baesens, Hsin-Vonn Seow, and Lyn C. Thomas. "Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research". In: *European Journal of Operational Research* 247.1 (2015). DOI:10.1016/j.ejor.2015.05.030, pp. 124–136.

[172] Jing Li, Ji hang Cheng, Jing yuan Shi, and Fei Huang. "Brief introduction of Back Propagation (BP) neural network algorithm and its improvement". In: *Advances in Computer Science and Information Engineering*. DOI:10.1007/978-3-642-30223-7_87. Springer. 2012, pp. 553–558.

[173] Yu Li. "Credit risk prediction based on machine learning methods". In: *2019 14th International Conference on Computer Science & Education (ICCSE)*. DOI:10.1109/ICCSE.2019.8845444. IEEE. 2019, pp. 1011–1013.

[174] Song Lin, Zhiguo Gao, and Ke Xu. "Web 2.0 Traffic Measurement: Analysis on Online Map Applications". In: *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. DOI:10.1145/1542245.1542248. ACM New York, NY, USA, 2009, pp. 7–12.

[175] Yu Lin, Xucheng Tang, Yuting Chen, and Jianjun Zhao. "A divergence-oriented approach to adaptive random testing of Java programs". In: *2009 IEEE/ACM International Conference on Automated Software Engineering*. DOI:10.1109/ASE.2009.13. IEEE. 2009, pp. 221–232.

[176] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. "Explainable AI: A review of machine learning interpretability methods". In: *Entropy* 23.1 (2020). DOI:10.3390/e23010018, p. 18.

[177] Mikael Lindvall, Dharmalingam Ganesan, Sigurthor Bjorgvinsson, Kristjan Jonsson, Haukur Steinn Logason, Frederik Dietrich, and Robert E Wiegand. "Agile metamorphic model-based testing". In: *2016 IEEE/ACM 1st International Workshop on Metamorphic Testing (MET)*. DOI:10.1145/2896971.2896979. Association for Computing Machinery, 2016, pp. 26–32.

[178] Steve Lipner. "The trustworthy computing security development lifecycle". In: *20th Annual Computer Security Applications Conference*. DOI:10.1109/CSAC.2004.41. IEEE. 2004, pp. 2–13.

[179] Huai Liu and Tsong Yueh Chen. "An innovative approach to randomising quasi-random sequences and its application into software testing". In: *2009 9th International Conference on Quality Software*. DOI:10.1109/QSIC.2009.16. IEEE. 2009, pp. 59–64.

[180] Huai Liu and Tsong Yueh Chen. "Randomized quasi-random testing". In: *IEEE Transactions on Computers* 65.6 (2015). DOI:10.1109/TC.2015.2455981, pp. 1896–1909.

[181] Huai Liu, Fei-Ching Kuo, Dave Towey, and Tsong Yueh Chen. "How effectively does metamorphic testing alleviate the oracle problem?" In: *IEEE Transactions on Software Engineering* 40.1 (2013). DOI:10.1109/TSE.2013.46, pp. 4–22.

[182] Huai Liu, Xuan Liu, and Tsong Yueh Chen. "A new method for constructing metamorphic relations". In: *2012 12th International Conference on Quality Software*. DOI:10.1109/QSIC.2012.10. IEEE. 2012, pp. 59–68.

[183] Huai Liu, Xiaodong Xie, Jing Yang, Yansheng Lu, and Tsong Yueh Chen. "Adaptive random testing by exclusion through test profile". In: *2010 10th International Conference on Quality Software*. DOI:10.1109/QSIC.2010.61. IEEE. 2010, pp. 92–101.

[184] Huai Liu, Xiaodong Xie, Jing Yang, Yansheng Lu, and Tsong Yueh Chen. "Adaptive random testing through test profiles". In: *Software: Practice and Experience* 41.10 (2011). DOI:10.1002/spe.1067, pp. 1131–1154.

[185] Hui Liu and Hee Beng Kuan Tan. "Covering code behavior on input validation in functional testing". In: *Information and Software Technology* 51.2 (2009). DOI:10.1016/j.infsof.2008.07.001, pp. 546–553.

[186] Zhifang Liu, Xiaopeng Gao, and Xiang Long. "Adaptive random testing of mobile application". In: *2010 2nd International Conference on Computer Engineering and Technology*. Vol. 2. DOI:10.1109/ICCET.2010.5485442. IEEE. 2010, pp. V2–297.

[187] Wei-Yin Loh. "Classification and regression trees". In: *Wiley interdisciplinary reviews: Data mining and knowledge discovery* 1.1 (2011). DOI:10.1038/nmeth.4370, pp. 14–23.

[188] Freddie Mac. *Single-Family Loan-Level Dataset General User Guide*. https://www.freddiemac.com/fmac-resources/research/pdf/user_guide.pdf. [Online; accessed 15 May 2024]. 2023.

[189] Milad Malekipirbazari and Vural Aksakalli. "Risk assessment in social lending via random forests". In: *Expert Systems with Applications* 42.10 (2015). DOI:10.1016/j.eswa.2015.02.001, pp. 4621–4631.

[190] Rashmi Malhotra and Davinder K. Malhotra. "Evaluating consumer loans using neural networks". In: *Omega* 31.2 (2003). DOI:10.1016/S0305-0483(03)00016-1, pp. 83–96.

[191] Rajib Mall. *Fundamentals of software engineering*. PHI Learning Pvt. Ltd., 2018.

[192] Chengying Mao, Tsong Yueh Chen, and Fei-Ching Kuo. "Out of sight, out of mind: A distance-aware forgetting strategy for adaptive random testing". In: *Science China Information Sciences* 60.9 (2017). DOI:10.1007/s11432-016-0087-0, p. 092106.

[193] Chengying Mao, Xuzheng Zhan, T. H. Tse, and Tsong Yueh Chen. "KDFC-ART: A KD-tree approach to enhancing Fixed-size-Candidate-set Adaptive Random Testing". In: *IEEE Transactions on Reliability* 68.4 (2019). DOI:10.1109/TR.2019.2892230, pp. 1444–1469.

[194] David Martens, Jan Vanthienen, Wouter Verbeke, and Bart Baesens. "Performance of classification models from a user perspective". In: *Decision Support Systems* 51.4 (2011). DOI:10.1016/j.dss.2011.01.013, pp. 782–793.

[195] Reza Matinnejad, Shiva Nejati, Lionel C. Briand, and Thomas Bruckmann. "Effective test suites for mixed discrete-continuous stateflow controllers". In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. DOI:10.1145/2786805.2786818. Association for Computing Machinery, 2015, pp. 84–95.

[196] Johannes Mayer. "Adaptive random testing by bisection and localization". In: *International Workshop on Formal Approaches to Software Testing*. DOI:10.1007/11759744_6. Springer. 2005, pp. 72–86.

[197] Johannes Mayer. "Adaptive random testing by bisection with restriction". In: *International Conference on Formal Engineering Methods*. DOI:10.1007/11576280_18. Springer. 2005, pp. 251–263.

[198] Johannes Mayer. "Adaptive random testing with randomly translated failure region". In: *Proceedings of the 1st international workshop on Random testing*. DOI:10.1145/1145735.1145746. 2006, pp. 70–77.

[199] Johannes Mayer. "Efficient and Effective Random Testing based on Partitioning and Neighborhood". In: *Proceedings of the 18th International Conference on Software Engineering & Knowledge Engineering (SEKE'06)*. 2006, pp. 479–484.

[200] Johannes Mayer. "Towards effective adaptive random testing for higher-dimensional input domains". In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. DOI:10.1145/1143997.1144323. 2006, pp. 1955–1956.

[201] Johannes Mayer and Ralph Guderlei. "An empirical study on the selection of good metamorphic relations". In: *Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06)*. Vol. 1. DOI:10.1109/COMPSAC.2006.24. IEEE. 2006, pp. 475–484.

[202] Johannes Mayer and Christoph Schneckenburger. "Adaptive random testing with enlarged input domain". In: *2006 6th International Conference on Quality Software (QSIC'06)*. DOI:10.1109/QSIC.2006.8. IEEE. 2006, pp. 251–258.

[203] Johannes Mayer and Christoph Schneckenburger. "An empirical analysis and comparison of random testing techniques". In: *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. DOI:10.1145/1159733.1159751. 2006, pp. 105–114.

[204] Johannes Mayer and Christoph Schneckenburger. "Statistical Analysis and Enhancement of Random Testing Methods also under Constrained Resources." In: *Software Engineering Research and Practice*. Citeseer. 2006, pp. 16–23.

[205] Michael McBurnett, Peter Maynard, and John Power. *Comparing scores and reason codes in credit scoring systems*. https://www.equifax.com/resource/-/asset/white-paper/comparing-scores-and-reason-codes-in-credit-scoring-systems/. [Online; accessed 15 May 2024]. 2020.

[206] Michael McBurnett, Peter Maynard, and John Power. *Putting Neural Network Models to the Test*. https://www.equifax.com/resource/-/asset/white-paper/putting-neural-network-models-test/. [Online; accessed 15 May 2024]. 2020.

[207] Phil McMinn. "Search-based software test data generation: A survey". In: *Software testing, Verification and reliability* 14.2 (2004). DOI:10.23919/CSMS.2022.0027, pp. 105–156.

[208] Nishchol Mishra and Sanjay Silakari. "Predictive analytics: A survey, trends, applications, oppurtunities & challenges". In: *International Journal of Computer Science and Information Technologies* 3.3 (2012), pp. 4434–4438.

[209] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. Lulu.com, 2020.

[210] Sandro Morasca and Stefano Serra-Capizzano. "On the analytical comparison of testing techniques". In: *Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis*. DOI:10.1145/1007512.1007533. ACM New York, NY, USA, 2004, pp. 154–164.

[211] Larry J. Morell. "A theory of fault-based testing". In: *IEEE Transactions on Software Engineering* 16.8 (1990), pp. 844–857.

[212] Vincenzo Moscato, Antonio Picariello, and Giancarlo Sperlí. "A benchmark of machine learning approaches for credit score prediction". In: *Expert Systems with Applications* 165 (2021). DOI:10.1016/j.eswa.2020.113986, p. 113986.

[213] Woramet Muangsiri and Shingo Takada. "Random GUI testing of android application using behavioral model". In: *International Journal of Software Engineering and Knowledge Engineering* 27.09n10 (2017). DOI:10.1142/S0218194017400149, pp. 1603–1612.

[214] Christian Murphy, Gail E. Kaiser, and Lifeng Hu. "Properties of machine learning applications for use in metamorphic testing". In: *Proceedings of the 20th International Conference on Software Engineering & Knowledge Engineering (SEKE'08)* (2008). DOI:10.1007/978-3-642-19423-8_19, pp. 867–872.

[215] John D. Musa. "Software reliability-engineered testing". In: *Computer* 29.11 (1996). DOI:10.1109/2.544239, pp. 61–68.

[216] Glenford J. Myers, Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.

[217] Anthony J. Myles, Robert N. Feudale, Yang Liu, Nathaniel A. Woody, and Steven D. Brown. "An introduction to decision tree modeling". In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 18.6 (2004). DOI:10.1002/cem.873, pp. 275–285.

[218] Shin Nakajima and Hai Ngoc Bui. "Dataset coverage for testing machine learning computer programs". In: *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. DOI:10.1109/APSEC.2016.049. IEEE. 2016, pp. 297–304.

[219] Changhai Nie and Hareton Leung. "A survey of combinatorial testing". In: *ACM Computing Surveys (CSUR)* 43.2 (2011). DOI:10.1145/1883612.1883618, pp. 1–29.

[220] Janping Nie, Yueying Qian, and Nan Cui. "Enhanced Mirror Adaptive Random Testing Based on I/O Relation Analysis". In: *Software Engineering and Knowledge Engineering: Theory and Practice*. DOI:10.1007/978-3-642-03718-4_5. Springer, 2012, pp. 33–47.

[221] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. "Using of Jaccard coefficient for keywords similarity". In: *Proceedings of the international multiconference of engineers and computer scientists*. Vol. 1. 2013, pp. 380–384.

[222] Alessandro Orso and Gregg Rothermel. "Software Testing: A Research Travelogue (2000-2014)". In: *Future of Software Engineering Proceedings*. DOI:10.1145/2593882.2593885. ACM New York, NY, USA, 2014, pp. 117–132.

[223] Thomas J. Ostrand and Marc J. Balcer. "The category-partition method for specifying and generating fuctional tests". In: *Communications of the ACM* 31.6 (1988). DOI:10.1145/62959.62964, pp. 676–686.

[224] Carlos Pacheco, Shuvendu K. Lahiri, Michael D. Ernst, and Thomas Ball. "Feedback-Directed Random Test Generation". In: *Proceedings of the 29th International Conference on Software Engineering (ICSE)*. DOI:10.1109/ICSE.2007.37. IEEE Computer Society, Los Alamitos, CA, 2007, pp. 75–84.

[225] Trilok Nath Pandey, Alok Kumar Jagadev, Suman Kumar Mohapatra, and Satchidananda Dehuri. "Credit risk analysis using machine learning classifiers". In: *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. IEEE. 2017, pp. 1850–1854. DOI: 10.1109/ICECDS.2017.8389769.

[226] Daniel Pesu, Zhi Quan Zhou, Jingfeng Zhen, and Dave Towey. "A Monte Carlo method for metamorphic testing of machine translation services". In: *2018 IEEE/ACM 3rd International Workshop on Metamorphic Testing (MET)*. DOI:10.1145/3193977.3193980. Association for Computing Machinery, 2018, pp. 38–45.

[227] Rob Pooley and Perdita Stevens. *Component-Based Software Testing with UML*. Addison-Wesley, Novermber 1998.

[228] Pak-Lok Poon, Fei-Ching Kuo, Huai Liu, and Tsong Yueh Chen. "How can non-technical end users effectively test their spreadsheets?" In: *Information Technology & People* 27.4 (2014). DOI:10.1108/ITP-01-2013-0004, pp. 440–462.

[229] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes 2nd edition*. Cambridge university press, 1992.

[230] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[231] I. Putu Edy Suardiyana Putra and Petrus Mursanto. "Centroid Based Adaptive Random Testing for object oriented program". In: *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. DOI:10.1109/ICACSIS.2013.6761550. IEEE. 2013, pp. 39–45.

[232] Kun Qiu, Zheng Zheng, Tsong Yueh Chen, and Pak-Lok Poon. "Theoretical and empirical analyses of the effectiveness of metamorphic relation composition". In: *IEEE Transactions on software engineering* 48.3 (2020). DOI:10.1109/TSE.2020.3009698, pp. 1001–1017.

[233] Peter Quell, Anthony Graham Bellotti, Joseph L. Breeden, and Javier Calvo Martin. *Machine Learning and Model Risk Management*. Tech. rep. Model Risk Managers' International Association (MRMIA), 2021.

[234] Quora. *Why does Google only return 50 pages of 10 results when it claims that there are 560,000 results?* https://www.quora.com/. [Online; accessed 15 May 2024]. 2012.

[235] John Regehr. "Random testing of interrupt-driven software". In: *Proceedings of the 5th ACM international conference on Embedded software*. DOI:10.1145/1086228.1086282. 2005, pp. 290–298.

[236] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should I trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. DOI:10.1145/2939672.2939778. 2016, pp. 1135–1144.

[237] Robert Rosenthal, Harris Cooper, and Larry Hedges. "Parametric measures of effect size". In: *The handbook of research synthesis* 621.2 (1994), pp. 231–244.

[238] Prashanta Saha and Upulee Kanewala. "Fault detection effectiveness of source test case generation strategies for metamorphic testing". In: *2018 IEEE/ACM 3rd International Workshop on Metamorphic Testing (MET)*. DOI:10.1145/3193977.3193982. 2018, pp. 2–9.

[239] Shlomo S. Sawilowsky. "New effect size rules of thumb". In: *Journal of modern applied statistical methods* 8.2 (2009). DOI:10.56801/10.56801/v8.i.452, pp. 467–474.

[240] Christoph Schneckenburger and Johannes Mayer. "Towards the determination of typical failure patterns". In: *Fourth international workshop on Software quality assurance: in conjunction with the 6th ESEC/FSE joint meeting*. DOI:10.1145/1295074.1295091. 2007, pp. 90–93.

[241] Christoph Schneckenburger and Franz Schweiggert. "Investigating the dimensionality problem of Adaptive Random Testing incorporating a local search technique". In: *2008 IEEE International Conference on Software Testing Verification and Validation Workshop*. DOI:10.1109/ICSTW.2008.24. IEEE. 2008, pp. 241–250.

[242] Cedric Seger. *An investigation of categorical variable encoding techniques in machine learning: Binary versus one-hot and feature hashing*. https://api.semanticscholar.org/CorpusID:250534659. [Online; accessed 15 May 2024]. 2018.

[243] Sergio Segura, Juan C. Alonso, Alberto Martin-Lopez, Amador Durán, Javier Troya, and Antonio Ruiz-Cortés. "Automated generation of metamorphic relations for query-based systems". In: *Proceedings of the 7th International Workshop on Metamorphic Testing*. DOI:10.1145/3524846.3527338. 2022, pp. 48–55.

[244] Sergio Segura, Amador Durán, Javier Troya, and Antonio Ruiz-Cortés. "A template-based approach to describing metamorphic relations". In: *2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)*. DOI:10.1109/MET.2017.3. IEEE. 2017, pp. 3–9.

[245] Sergio Segura, Amador Durán, Javier Troya, and Antonio Ruiz-Cortés. "Metamorphic relation patterns for query-based systems". In: *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)*. DOI:10.1109/MET.2019.00012. IEEE. 2019, pp. 24–31.

[246] Sergio Segura, Gordon Fraser, Ana B. Sanchez, and Antonio Ruiz-Cortés. "A survey on metamorphic testing". In: *IEEE Transactions on software engineering* 42.9 (2016). DOI:10.1109/TSE.2016.2532875, pp. 805–824.

[247] Sergio Segura, José A Parejo, Javier Troya, and Antonio Ruiz-Cortés. "Metamorphic testing of RESTful web APIs". In: *IEEE Transactions on Software Engineering* 44.11 (2017). DOI:10.1145/3180155.3182528, pp. 1083–1099.

[248] Elmin Selay, Zhi Quan Zhou, and Jingjie Zou. "Adaptive random testing for image comparison in regression web testing". In: *2014 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. DOI:10.1109/DICTA.2014.7008093. IEEE. 2014, pp. 1–7.

[249] Ali Shahbazi, Andrew F. Tappenden, and James Miller. "Centroidal Voronoi Tessellations - A New Approach to Random Testing". In: *IEEE Transactions on Software Engineering* 39.2 (2013). DOI:10.1109/TSE.2012.18, pp. 163–183.

[250] Lloyd S. Shapley. *Notes on the n-person game—ii: The value of an n-person game*. DOI:10.7249/RM0670. Rand Corporation, 1951.

[251] Rohan Sharma, Milos Gligoric, Andrea Arcuri, Gordon Fraser, and Darko Marinov. "Testing container classes: Random or systematic?" In: *International Conference on Fundamental Approaches to Software Engineering*. DOI:10.1007/978-3-642-19811-3_19. Springer. 2011, pp. 262–277.

[252] Alessio Signorini and Tomasz Imielinski. "If You Ask Nicely, I Will Answer: Semantic Search and Today's Search Engines". In: *Proceedings of the 2009 IEEE International Conference on Semantic Computing*. DOI:10.1109/ICSC.2009.31. IEEE Computer Society, Los Alamitos, CA, 2009, pp. 184–191.

[253] Gurdeepak Singh. "An automated metamorphic testing technique for designing effective metamorphic relations". In: *Contemporary Computing: 5th International Conference, IC3 2012, Noida, India, August 6-8, 2012. Proceedings 5*. DOI:10.1007/978-3-642-32129-0_20. Springer. 2012, pp. 152–163.

[254] Yan-Yan Song and Lu Ying. "Decision tree methods: Applications for classification and prediction". In: *Shanghai archives of psychiatry* 27.2 (2015). DOI:10.11919/j.issn.1002-0829.215044, p. 130.

[255] Yu Song, Yuyan Wang, Xin Ye, Dujuan Wang, Yunqiang Yin, and Yanzhang Wang. "Multi-view ensemble learning based on distance-to-model and adaptive clustering for imbalanced credit risk assessment in P2P lending". In: *Information Sciences* 525 (2020). DOI:10.1016/j.ins.2020.03.027, pp. 182–204.

[256] Helge Spieker and Arnaud Gotlieb. "Adaptive metamorphic testing with contextual bandits". In: *Journal of Systems and Software* 165 (2020). DOI:10.1016/j.jss.2020.110574, p. 110574.

[257] Felix Stahlberg. "Neural machine translation: A review". In: *Journal of Artificial Intelligence Research* 69 (2020). DOI:10.1613/jair.1.12007, pp. 343–418.

[258] Kwanho Suk, Jiheon Lee, and Donald R. Lichtenstein. "The influence of price presentation order on consumer choice". In: *Journal of Marketing Research* 49.5 (2012). DOI:10.1509/jmr.11.0309, pp. 708–717.

[259] Bo Sun, Yunwei Dong, and Hong Ye. "On enhancing adaptive random testing for AADL model". In: *2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*. DOI:10.1109/UIC-ATC.2012.77. IEEE. IEEE Computer Society, 2012, pp. 455–461.

[260] Chang-Ai Sun, Hepeng Dai, Huai Liu, and Tsong Yueh Chen. "Feedback-Directed Metamorphic Testing". In: *ACM Transactions on Software Engineering and Methodology* 32.1 (2023). DOI:10.1145/3533314.

[261] Chang-Ai Sun, An Fu, Pak-Lok Poon, Xiaoyuan Xie, Huai Liu, and Tsong Yueh Chen. "METRIC+: A metamorphic relation identification technique based on input plus output domains". In: *IEEE Transactions on Software Engineering* (2019). DOI:10.1109/TSE.2019.2934848.

[262] Chang-Ai Sun, Baoli Liu, An Fu, Yiqiang Liu, and Huai Liu. "Path-directed source test case generation and prioritization in metamorphic testing". In: *Journal of Systems and Software* 183 (2022). DOI:10.1016/j.jss.2021.111091, p. 111091.

[263] Chang-Ai Sun, Guan Wang, Baohong Mu, Huai Liu, Zhaoshun Wang, and Tsong Yueh Chen. "Metamorphic testing for web services: Framework and a case study". In: *2011 IEEE International Conference on Web Services*. DOI:10.1109/ICWS.2011.65. IEEE. 2011, pp. 283–290.

[264] Liqun Sun and Zhi Quan Zhou. "Metamorphic testing for machine translations: MT4MT". In: *2018 25th Australasian Software Engineering Conference (ASWEC'18)*. DOI:10.1109/ASWEC.2018.00021. IEEE. 2018, pp. 96–100.

[265] Andrew F. Tappenden and James Miller. "A novel evolutionary approach for adaptive random testing". In: *IEEE Transactions on Reliability* 58.4 (2009). DOI:10.1109/TR.2009.2034288, pp. 619–633.

[266] Krishnaiyan Thulasiraman and Madisetti N. S. Swamy. *Graphs: Theory and algorithms*. John Wiley & Sons, 1992, p. 118.

[267] Dave Towey, James Walker, and Ricky Ng. "Traditional higher education engineering versus vocational and professional education and training: What can we learn from each other?" In: *Proceedings of the 2018 International Conference on Open and Innovative Education (ICOIE'18)*. 2018, pp. 474–486.

[268] Dave Towey, James Walker, and Ricky Ng. "Embracing ambiguity: Agile insights for sustainability in engineering in traditional higher education and in technical and vocational education and training". In: *Interactive Technology and Smart Education* 16.2 (2019). DOI:10.1108/ITSE-10-2018-0088, pp. 143–158.

[269] Dave Towey, Sen Yang, Zhihao Ying, Zhi Quan Zhou, and Tsong Yueh Chen. "Learning by doing: Developing the next generation of software quality assurance professionals". In: *Proceedings of the 2019 International Conference on Open and Innovative Education (ICOIE'19)*. The Open University of Hong Kong, 2019, pp. 347–355.

[270] Shrawan Kumar Trivedi. "A study on credit scoring modeling with different feature selection and machine learning approaches". In: *Technology in Society* 63 (2020). DOI:10.1016/j.techsoc.2020.101413, p. 101413.

[271] Kanewala Upulee, Bieman James M., and Ben-Hur Asa. "Predicting metamorphic relations for testing scientific software: A machine learning approach using graph kernels". In: *Software testing, verification and reliability* 26.3 (2016). DOI:10.1002/stvr.1594, pp. 245–269.

[272] Apostol Vassilev and Christopher Celi. "Avoiding Cyberspace Catastrophes through Smarter Testing". In: *Computer* 47.10 (2014). DOI:10.1109/MC.2014.273, pp. 102–106.

[273] Alfredo Vellido, Paulo JG. Lisboa, and J. Vaughan. "Neural networks in business: A survey of applications (1992–1998)". In: *Expert Systems with applications* 17.1 (1999). DOI:10.1016/S0957-4174(99)00016-0, pp. 51–70.

[274] Thomas Verbraken, Cristián Bravo, Richard Weber, and Bart Baesens. "Development and application of consumer credit scoring models using profit-based classification measures". In: *European Journal of Operational Research* 238.2 (2014). DOI:10.1016/j.ejor.2014.04.001, pp. 505–513.

[275] Willem Visser, Corina S. Pasareanu, and Radek Pelánek. "Test input generation for Java containers using state matching". In: *Proceedings of the 2006 International Symposium on Software Testing and Analysis*. DOI:10.1145/1146238.1146243. Association for Computing Machinery, 2006, pp. 37–48.

[276] Neil Walkinshaw and Gordon Fraser. "Uncertainty-driven black-box test data generation". In: *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. DOI:10.1109/ICST.2017.30. IEEE. 2017, pp. 253–263.

[277] Yuyan Wang, Dujuan Wang, Xin Ye, Yanzhang Wang, Yunqiang Yin, and Yaochu Jin. "A tree ensemble-based two-stage model for advanced-stage colorectal cancer survival prediction". In: *Information Sciences* 474 (2019). DOI:10.1016/j.ins.2018.09.046, pp. 106–124.

[278] David West. "Neural network credit scoring models". In: *Computers & Operations Research* 27.11-12 (2000). DOI:10.1016/S0305-0548(99)00149-5, pp. 1131–1152.

[279] Lee J. White and Edward I. Cohen. "A domain strategy for computer program testing". In: *IEEE Transactions on Software Engineering* SE-6.3 (1980). DOI:10.1109/TSE.1980.234486, pp. 247–257.

[280] Chaohua Wu, Liqun Sun, and Zhi Quan Zhou. "The impact of a dot: Case studies of a noise metamorphic relation pattern". In: *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)*. DOI:10.1109/MET.2019.00011. IEEE. 2019, pp. 17–23.

[281] Xiaoyuan Xie, Joshua W. K. Ho, Christian Murphy, Gail Kaiser, Baowen Xu, and Tsong Yueh Chen. "Testing and validating machine learning classifiers by metamorphic testing". In: *Journal of Systems and Software* 84.4 (2011). DOI:10.1016/j.jss.2010.11.920, pp. 544–558.

[282] Xiaoyuan Xie, Jingxuan Tu, Tsong Yueh Chen, and Baowen Xu. "Bottom-up integration testing with the technique of metamorphic testing". In: *2014 14th International Conference on Quality Software*. DOI:10.1109/QSIC.2014.29. IEEE. 2014, pp. 73–78.

[283] Xue Ying. "An overview of overfitting and its solutions". In: *Journal of physics: Conference series*. Vol. 1168. DOI:10.1088/1742-6596/1168/2/022022. IOP Publishing. 2019, p. 022022.

[284] Zhihao Ying, Anthony Bellotti, Dave Towey, Tsong Yueh Chen, and Zhi Quan Zhou. "Using Metamorphic Relation Violation Regions to Support a Simulation Framework for the Process of Metamorphic Testing". In: *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC'22)*. DOI:10.1109/COMPSAC54236.2022.00274. IEEE. 2022, pp. 1722–1727.

[285] Zhihao Ying, Dave Towey, Anthony Bellotti, Zhi Quan Zhou, and Tsong Yueh Chen. "Preparing SQA Professionals: Metamorphic Relation Patterns, Exploration, and Testing for Big Data". In: *Proceedings of the 2021 International Conference on Open and Innovative Education (ICOIE'21)*. The Open University of Hong Kong, 2021, pp. 22–30.

[286] Shin Yoo and Mark Harman. "Regression testing minimization, selection and prioritization: A survey". In: *Software testing, verification and reliability* 22.2 (2012). DOI:10.1002/stv.430, pp. 67–120.

[287] Jie Zhang, Junjie Chen, Dan Hao, Yingfei Xiong, Bing Xie, Lu Zhang, and Hong Mei. "Search-based inference of polynomial metamorphic relations". In: *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*. DOI:10.1145/2642937.2642994. 2014, pp. 701–712.

[288] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. "A survey on neural network interpretability". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 5.5 (2021). DOI:10.1109/TETCI.2021.3100641, pp. 726–742.

[289] Zhirui Zhang, Dave Towey, Zhihao Ying, Yifan Zhang, and Zhi Quan Zhou. "MT4NS: Metamorphic Testing for Network Scanning". In: *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*. DOI:10.1109/MET52542.2021.00010. IEEE. 2021, pp. 17–23.

[290] Bo Zhou, Hiroyuki Okamura, and Tadashi Dohi. "Enhancing performance of random testing through Markov chain Monte Carlo methods". In: *IEEE Transactions on Computers* 62.1 (2011). DOI:10.1109/TC.2011.208, pp. 186–192.

[291] Zenghui Zhou, Zheng Zheng, Tsong Yueh Chen, Jinyi Zhou, and Kun Qiu. "Follow-up Test Cases are Better Than Source Test Cases in Metamorphic Testing: A Preliminary Study". In: *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*. DOI:10.1109/MET52542.2021.00018. IEEE. 2021, pp. 69–74.

[292] Zhi Quan Zhou, Liqun Sun, Tsong Yueh Chen, and Dave Towey. "Metamorphic Relations for Enhancing System Understanding and Use". In: *IEEE Transactions on Software Engineering* 46.10 (2020). DOI:10.1109/TSE.2018.2876433, pp. 1120–1154.

[293] Zhi Quan Zhou, Dave Towey, Pak Poon, and T. H. Tse. "Introduction to the special issue on test oracles". In: *Journal of Systems and Software, Editorial* 136 (2018). DOI:10.1016/j.jss.2017.08.031, pp. 187–187.

[294] Zhi Quan Zhou, T. H. Tse, Fei-Ching Kuo, and Tsong Yueh Chen. "Automated Functional Testing of Web Search Engines in the Absence of an Oracle". In: *Department of Computer Science, The University of Hong Kong, Tech. Rep. TR-2007-06* (2007).

[295] Zhi Quan Zhou, T. H. Tse, and Matt Witheridge. "Metamorphic robustness testing: Exposing hidden defects in citation statistics and journal impact factors". In: *IEEE Transactions on Software Engineering* 47.6 (2021). DOI:10.1109/TSE.2019.2915065, pp. 1164–1183.

[296] Zhi Quan Zhou, Shaowen Xiang, and Tsong Yueh Chen. "Metamorphic testing for software quality assessment: A study of search engines". In: *IEEE Transactions on Software Engineering* 42.3 (2016). DOI:10.1109/TSE.2015.2478001, pp. 264–284.

[297] Zhi Quan Zhou, Junting Zhu, Tsong Yueh Chen, and Dave Towey. "In-place metamorphic testing and exploration". In: *Proceedings of the 7th International Workshop on Metamorphic Testing*. DOI:10.1145/3524846.3527334. Association for Computing Machinery, 2023, 1–6.

[298] Hong Zhu. "JFuzz: A tool for automated Java unit testing based on data mutation and metamorphic testing methods". In: *Proceedings of the 2015 Second International Conference on Trustworthy Systems and Their Applications*. DOI:10.1109/TSA.2015.13. IEEE Computer Society, Los Alamitos, CA, 2015, pp. 8–15.

APPENDIX 1

In the domain of Sin Function
assuming that

- The input contains one parameter: x.
- The output consists of one parameter: $S(x)$.

the following metamorphic relation(s) should hold

- MR_{Sin1}:

if $x_2 = -x_1$,

then $S(x_1) = -S(x_2)$.

- MR_{Sin2}:

if $x_2 = \pi - x_1$,

then $S(x_1) = S(x_2)$.

- MR_{Sin3}:

if $x_2 = x_1 + 2 * \pi$,

then $S(x_1) = S(x_2)$.

- MR_{Sin4}:

if $x_2 = 2 * \pi - x_1$,

then $S(x_1) = -S(x_2)$.

- MR_{Sin5}:

if $x_2 = x_1$ * 3, $x_3 = 5 * x_1$,

then $16 * S(x_1)^5 - 10 * S(x_1) = S(x_3) - 5 * S(x_2)$.

- MR_{Sin6}:

if $x_2 = x_1 * 3$,

then $3 * S(x_1) - 4 * S^3(x_1) = S(x_2)$.

- MR_{Sin7}:

if $x_2 = x_1 + \pi$,

then $S(x_1) = -S(x_2)$.

- MR_{Sin8}:

if $x_2 = \pi/2 - x_1$,

then $S(x_1) * S(x_1) = 1 - S(x_2) * S(x_2)$.

- MR_{Sin9}:

if $x_2 = x_1 * 3$, $x_3 = -x_1$,

then $3 * S(x_1) = S(x_2) - 4 * S^3(x_3)$.

- MR_{Sin10}:

if $x_2 = x_1 * 3$, $x_3 = \pi * 2 - x_1$,

then $4 * S^3(x_1) = -S(x_2) - 3 * S(x_3)$.

- MR_{Sin11}:

if $x_2 = x_1$ * 3, $x_3 = 5 * x_1$, $x_4 = -x_1$,

then $10 * S(x_1) = 5 * S(x_2) - S(x_3) - 16 * S(x_4)$.

- MR_{Sin12}:

if $x_2 = x_1$ * 3, $x_3 = 5 * x_1$, $x_4 = \pi - x_1$,

then $10 * S(x_1) = 5 * S(x_2) - S(x_3) + 16 * S(x_4)$.

- MR_{Sin13}:

if $x_2 = x_1/3$,

then $S(x_1) = 3 * S(x_2) - 4 * S^3(x_2)$.

In the domain of *tanh* Function
assuming that

- The input contains one parameter: x.
- The output consists of one parameter: $t(x)$.

the following metamorphic relation(s) should hold

- MR_{tanh} :

if $x_2 = x_1 * 3$,
then $t(x_2) = (t^3(x_1) + 3 * t(x_1))/(1 + 3 * t^2(x_1))$.

In the domain of Erf Function
assuming that

- The input contains one parameter: x.
- The output consists of one parameter: $E(x)$.

the following metamorphic relation(s) should hold

- MR_{Erf1}:

if $x_2 = -x_1$,

then $E(x_1) = -E(x_2)$.

- MR_{Erf2}:

if $x_2 = x_1 + 1$,

then $E(x_1) \leq E(x_2)$.

- MR_{Erf3}:

if $x_2 = x_1 + 5$,

then $E(x_1) \leq E(x_2)$.

- MR_{Erf4}:

if $x_2 = x_1 + 10$,

then $E(x_1) \leq E(x_2)$.

- MR_{Erf5}:

if $x_2 = x_1 + 100$,

then $E(x_1) \leq E(x_2)$.

- MR_{Erf6}:

if $x_2 = x_1 * 5$,

then $E(x_1) \leq E(x_2)$.

- MR_{Erf7}:

if $x_2 = x_1 * 10$,

then $E(x_1) \leq E(x_2)$.

- MR_{Erf8}:

if $x_2 = x_1 * 100$,

then $E(x_1) \leq E(x_2)$.

In the domain of sncndn Function
assuming that

- The inputs contain two parameters: x, y.
- The outputs consist of three parameters: sn, cn, and dn.

the following metamorphic relation(s) should hold

- $MR_{sncndn1}$:

if $x_2 = x_1 * 2, y_2 = y_1 * 2,$

then $(1 - y_1) * sn_1^2 + dn_1^2 = (1 - y_2) * sn_2^2 + dn_2^2.$

- $MR_{sncndn2}$:

if $x_2 = x_1 * 3, y_2 = y_1 * 3,$

then $(1 - y_1) * sn_1^2 + dn_1^2 = (1 - y_2) * sn_2^2 + dn_2^2.$

- $MR_{sncndn3}$:

if $x_2 = x_1 * 4, y_2 = y_1 * 4,$

then $(1 - y_1) * sn_1^2 + dn_1^2 = (1 - y_2) * sn_2^2 + dn_2^2.$

- $MR_{sncndn4}$:

if $x_2 = x_1 * 5, y_2 = y_1 * 5,$

then $(1 - y_1) * sn_1^2 + dn_1^2 = (1 - y_2) * sn_2^2 + dn_2^2.$

- $MR_{sncndn5}$:

if $x_2 = x_1 + 1, y_2 = y_1 + 1,$

then $(1 - y_1) * sn_1^2 + dn_1^2 = (1 - y_2) * sn_2^2 + dn_2^2.$

- $MR_{sncndn6}$:

if $x_2 = x_1 + 2, y_2 = y_1 + 2,$

then $(1 - y_1) * sn_1^2 + dn_1^2 = (1 - y_2) * sn_2^2 + dn_2^2.$

- $MR_{sncndn7}$:

if $x_2 = x_1 + 5, y_2 = y_1 + 5,$

then $(1 - y_1) * sn_1^2 + dn_1^2 = (1 - y_2) * sn_2^2 + dn_2^2.$

- $MR_{sncndn8}$:

if $x_2 = x_1 + 10, y_2 = y_1 + 10,$

then $(1 - y_1) * sn_1^2 + dn_1^2 = (1 - y_2) * sn_2^2 + dn_2^2.$

In the domain of BesselJ Function
assuming that

- The inputs contain two parameters: x, y.
- The outputs consist of one parameter: $J_y(x)$.

the following metamorphic relation(s) should hold

- $MR_{BesselJ1}$:

if $x_3 = x_2 = x_1, y_2 = y_1 + 1, y_3 = y_1 - 1,$

then $(J_{y_2}(x_2) + J_{y_3}(x_3)) = (2 * y_1 * J_{y_1}(x_1))/x_1.$

- $MR_{BesselJ2}$:

if $x_1 = 0, y_1 = 0, x_2 = x_1 + a, y_2 = y_1 + b$ $(a, b \in [1, 100])$,

then $J_{y_2}(x_2) < J_{y_1}(x_1).$

- $MR_{BesselJ3}$:

if $x_2 = x_1 = 0, y_1 = x_1 + a, y_1 = x_1 + b$ $(a, b \in [1, 100])$,

then $J_{y_2}(x_2) = J_{y_1}(x_1).$

In the domain of TriSquare Function
assuming that

- The inputs contain three parameters: x, y, z.
- The outputs consist of one parameter: $T(x, y, z)$.

the following metamorphic relation(s) should hold

- $MR_{TriSquare1}$:

if $x_2 = z_1, y_2 = x_1, z_2 = y_1,$

then $T_1 = T_2.$

- $MR_{TriSquare2}$:

if $x_2 = y_1, y_2 = x_1, z_2 = z_1,$

then $T_1 = T_2.$

- $MR_{TriSquare3}$:

if $x_2 = x_1, y_2 = z_1, z_2 = y_1,$

then $T_1 = T_2.$

- $MR_{TriSquare4}$:

if $x_2 = z_1, y_2 = y_1, z_2 = x_1,$

then $T_1 = T_2.$

- $MR_{TriSquare5}$:

if $x_2 = x_1 * 2, y_2 = y_1 * 2, z_2 = z_1 * 2,$

then $4 * T_1 = T_2.$

- $MR_{TriSquare6}$:

if $x_2 = x_1 * 3, y_2 = y_1 * 3, z_2 = z_1 * 3,$

then $9 * T_1 = T_2$.

- $MR_{TriSquare7}$:

if $x_2 = x_1 * 4, y_2 = y_1 * 4, z_2 = z_1 * 4,$

then $16 * T_1 = T_2$.

- $MR_{TriSquare8}$:

if $x_2 = x_1 * 5, y_2 = y_1 * 5, z_2 = z_1 * 5,$

then $25 * T_1 = T_2$.

- $MR_{TriSquare9}$:

if $x_2 = x_1 * 6, y_2 = y_1 * 6, z_2 = z_1 * 6,$

then $36 * T_1 = T_2$.

- $MR_{TriSquare10}$:

if $x_2 = x_1 * 7, y_2 = y_1 * 7, z_2 = z_1 * 7,$

then $49 * T_1 = T_2$.

- $MR_{TriSquare11}$:

if $x_2 = x_1 * 8, y_2 = y_1 * 8, z_2 = z_1 * 8,$

then $81 * T_1 = T_2$.

In the domain of TriSquarePlus Function
assuming that

- The inputs contain three parameters: x, y, z.
- The outputs consist of one parameter: $T(x, y, z)$.

the following metamorphic relation(s) should hold

- $MR_{TriPlus1}$:

if $x_2 = z_1, y_2 = x_1, z_2 = y_1,$

then $T_1 = T_2$.

- $MR_{TriPlus2}$:

if $x_2 = y_1, y_2 = x_1, z_2 = z_1,$

then $T_1 = T_2$.

- $MR_{TriPlus3}$:

if $x_2 = x_1, y_2 = z_1, z_2 = y_1,$

then $T_1 = T_2$.

- $MR_{TriPlus4}$:

if $x_2 = z_1, y_2 = y_1, z_2 = x_1$,

then $T_1 = T_2$.

- $MR_{TriPlus5}$:

if $x_2 = x_1 * 2, y_2 = y_1 * 2, z_2 = z_1 * 2$,

then $4 * T_1 = T_2$.

- $MR_{TriPlus6}$:

if $x_2 = x_1 * 3, y_2 = y_1 * 3, z_2 = z_1 * 3$,

then $9 * T_1 = T_2$.

- $MR_{TriPlus7}$:

if $x_2 = x_1 * 4, y_2 = y_1 * 4, z_2 = z_1 * 4$,

then $16 * T_1 = T_2$.

- $MR_{TriPlus8}$:

if $x_2 = x_1 * 5, y_2 = y_1 * 5, z_2 = z_1 * 5$,

then $25 * T_1 = T_2$.

- $MR_{TriPlus9}$:

if $x_2 = x_1 * 6, y_2 = y_1 * 6, z_2 = z_1 * 6$,

then $36 * T_1 = T_2$.

- $MR_{TriPlus10}$:

if $x_2 = x_1 * 7, y_2 = y_1 * 7, z_2 = z_1 * 7$,

then $49 * T_1 = T_2$.

- $MR_{TriPlus11}$:

if $x_2 = x_1 * 8, y_2 = y_1 * 8, z_2 = z_1 * 8$,

then $81 * T_1 = T_2$.

- $\mathrm{MR}_{TriPlus12}$:

if $x_2 = x_1/3, y_2 = y_1/3, z_2 = z_1/3$,
then $T_1 = 9 * T_2$.

In the domain of rj Function
assuming that

- The inputs contain three parameters: x, y, z, p.
- The outputs consist of one parameter: $R(x, y, z, p)$.

the following metamorphic relation(s) should hold

- MR_{rj1}:

if $x_2 = x_1 * 3, y_2 = y_1 * 3, z_2 = z_1 * 3, p_2 = p_1 * 3,$

then $R_1 \geq R_2$.

- MR_{rj2}:

if $x_2 = y_1, y_2 = x_1, z_2 = z_1, p_2 = p_1,$

then $R_1 = R_2$.

- MR_{rj3}:

if $x_2 = z_1, y_2 = y_1, z_2 = x_1, p_2 = p_1,$

then $R_1 = R_2$.

- MR_{rj4}:

if $x_2 = x_1, y_2 = z_1, z_2 = y_1, p_2 = p_1,$

then $R_1 = R_2$.

- MR_{rj5}:

if $x_2 = z_1, y_2 = x_1, z_2 = y_1, p_2 = p_1,$

then $R_1 = R_2$.

- MR_{rj6}:

if $x_2 = y_1, y_2 = z_1, z_2 = x_1, p_2 = p_1,$

then $R_1 = R_2$.

In the domain of PntLinePos Function
assuming that

- The source inputs contain six parameters: $x1_s$, $y1_s$, $x2_s$, $y2_s$, $x3_s$, $y3_s$.
- The follow-up inputs contain six parameters: $x1_f$, $y1_f$, $x2_f$, $y2_f$, $x3_f$, $y3_f$.
- The two endpoints of the line segment are represented by $(x1, y1)$ and $(x2, y2)$.
- The point is represented by $(x3, y3)$.
- The source output consists of one parameter: P_s.
- The follow-up output consists of one parameter: P_f.

the following metamorphic relation(s) should hold

- MR_{Pnt1}:

if $x1_f = x1_s * 2, y1_f = y1_s * 2, x2_f = x2_s * 2, y2_f = y2_s * 2, x3_f = x3_s * 2, y3_f = y3_s * 2,$

then $P_s = P_f$.

- MR_{Pnt2}:

if $x1_f = x1_s * 3, y1_f = y1_s * 3, x2_f = x2_s * 3, y2_f = y2_s * 3, x3_f = x3_s * 3, y3_f = y3_s * 3,$

then $P_s = P_f$.

- MR_{Pnt3}:

if $x1_f = x1_s * 4, y1_f = y1_s * 4, x2_f = x2_s * 4, y2_f = y2_s * 4, x3_f = x3_s * 4, y3_f = y3_s * 4,$

then $P_s = P_f$.

- MR_{Pnt4}:

if $x1_f = x1_s + 1, y1_f = y1_s + 1, x2_f = x2_s + 1, y2_f = y2_s + 1, x3_f = x3_s + 1, y3_f = y3_s + 1,$

then $P_s = P_f$.

- MR_{Pnt5}:

if $x1_f = x1_s - 1, y1_f = y1_s - 1, x2_f = x2_s - 1, y2_f = y2_s - 1, x3_f = x3_s - 1, y3_f = y3_s - 1,$

then $P_s = P_f$.

- MR_{Pnt6}:

if $x1_f = x1_s + 5, y1_f = y1_s + 5, x2_f = x2_s + 5, y2_f = y2_s + 5, x3_f = x3_s + 5, y3_f = y3_s + 5,$

then $P_s = P_f$.

- MR_{Pnt7}:

if $x1_f = x1_s + 10, y1_f = y1_s + 10, x2_f = x2_s + 10, y2_f = y2_s + 10, x3_f = x3_s + 10, y3_f = y3_s + 10,$

then $P_s = P_f$.

- MR_{Pnt8}:

if $x1_f = x2_s, y1_f = y2_s, x2_f = x1_s, y2_f = y1_s, x3_f = x3_s, y3_f = y3_s,$

when $P_s \in \{1, 2\}$,

then $P_s = P_f$;

when $P_s \in \{0, 3\}$,

then $P_s \neq P_f$.