变异技术综述

芮婉灵[1)] 余筱[2)] 郭翰宇[3)] 黄振斌[4)]

[1)](南京大学软件学院, 南京市, 210000)
[2)](南京大学软件学院, 南京市, 210000)
[3)](南京大学软件学院, 南京市, 210000)
[4)](南京大学软件学院, 南京市, 210000)

摘 要 变异技术作为自动化测试领域的重要手段，广泛应用于变异测试、模糊测试、基于搜索的测试等多个领域。通过引入人为变异，该技术能够有效评估测试用例的质量和覆盖率，从而显著提升软件测试的有效性。近年来，随着软件复杂性和规模的不断增长，变异技术的研究取得了显著进展，涵盖了变异算子设计、变异体生成策略及筛选机制等多个方面。本文系统总结了当前变异技术的研究动机与设计思路，深入探讨了其在模糊测试、基于搜索的测试、深度学习框架测试、操作系统内核测试等领域的实际应用。同时，本文还分析了变异技术在现有研究中的局限性与不足，并提出了未来可能的研究方向，旨在为变异技术的持续发展提供有力的参考和指导。

关键词 变异技术；自动化测试；模糊测试；基于搜索的测试；深度学习框架

A Survey of Mutation Techniques

RUI Wan-Ling[1)] YU Xiao[2)] GUO Han-Yu[3)] HUANG Zhen-Bin[4)]

[1)](School of Software, Nanjing University, Nanjing, 210000)
[2)](School of Software, Nanjing University, Nanjing, 210000)
[3)](School of Software, Nanjing University, Nanjing, 210000)
[4)](School of Software, Nanjing University, Nanjing, 210000)

Abstract

Mutation technology, as an important means in the field of automated testing, is widely applied in various domains such as mutation testing, fuzz testing, and search-based testing. By introducing artificial mutations, this technology can effectively evaluate the quality and coverage of test cases, thereby significantly enhancing the effectiveness of software testing. In recent years, with the increasing complexity and scale of software, significant progress has been made in the research of mutation technology, covering aspects such as mutation operator design, mutation generation strategies, and filtering mechanisms. This article systematically summarizes the research motivations and design concepts of current mutation technology, delves into its practical applications in areas such as fuzz testing, search-based testing, deep learning framework testing, and operating system kernel testing. Additionally, this article analyzes the limitations and shortcomings of mutation technology in existing research and proposes potential future research directions, aiming to provide strong references and guidance for the continuous development of mutation technology.

Keywords Mutation technology; Automated testing; Fuzz testing; Search-based testing; Deep learning frameworks

收稿日期： - - ；最终修改稿收到日期： - - .*投稿时不填写此项*. 本课题得到... ...基金中文完整名称(No.项目号)、... ...基金中文完整名称(No.项目号)、... ... 基金中文完整名称(No.项目号)资助.作者名1(通信作者)，性别，xxxx年生，学位(或目前学历)，职称，是/否计算机学会(CCF)会员（提供会员号),主要研究领域为*****、****.E-mail: **********.作者名2（通信作者)，性别，xxxx年生，学位(或目前学历)，职称，是/否计算机学会(CCF)会员（提供会员号),主要研究领域为*****、****.E-mail: **********. 作者名3（通信作者)，性别，xxxx年生，学位(或目前学历)，职称，是/否计算机学会(CCF)会员（提供会员号),主要研究领域为*****、****.E-mail: **********.(给出的电子邮件地址应不会因出国、毕业、更换工作单位等原因而变动。请给出所有作者的电子邮件) 第1作者手机号码(投稿时必须提供，以便紧急联系，发表时会删除): ... ..., E-mail: *此部分6号宋体*

1 引言

1.1 研究背景

1.1.1 自动化测试的重要性

随着软件系统的日益复杂化和规模的迅速扩大，软件开发中的质量保证任务变得愈发重要。在现代软件开发生命周期中，测试作为软件质量保证的关键环节，已成为不可或缺的一部分。

尽管传统的手工测试在早期开发中发挥了重要作用，但随着系统规模的不断扩大，手工测试的效率、覆盖率和准确性均无法满足现代软件开发的需求。尤其在大规模软件系统中，手工测试不仅耗时费力，还容易出现遗漏和误判，这对确保软件的稳定性和安全性构成了挑战。因此，自动化测试逐渐成为软件测试领域的核心技术，它通过工具和脚本自动执行测试用例，大大减少了人力投入并提高了测试效率。

自动化测试具有一系列显著优势。首先，它可以在无需人为干预的情况下重复执行测试，确保测试用例在不同版本的软件中持续有效。其次，自动化测试能够快速执行大量测试用例，显著提升测试的速度和覆盖范围。此外，自动化测试减少了人为错误的可能性，使测试结果更加可靠和可复现。特别是在敏捷开发模式下，频繁的代码更新和快速迭代要求进行自动化的回归测试，以确保新版本不引入新的缺陷。这些因素使得自动化测试成为确保软件质量的重要工具。

然而，尽管自动化测试能够有效提高测试效率，它仍面临一些挑战。首先，要设计高效的测试用例，以确保尽可能高的代码覆盖率和故障检测能力，其次，还需应对复杂软件系统中潜在的隐藏缺陷。这些挑战推动了包括变异技术在内的多种自动化测试技术的发展，以进一步改善测试覆盖率和有效性[1-2]。

1.1.2 变异技术在自动化测试中的应用及意义

在众多自动化测试技术中，变异技术（Mutation Testing）被认为是最有效且最具挑战性的一种。

变异技术的基本思想是通过对被测程序的代码进行故意的微小修改，生成一系列所谓的“变异体”（Mutants），并通过执行这些变异体来检验现有测试用例的有效性。如果测试用例能够识别并“杀死”这些变异体，即检测到由于代码变异而引入的错误，则表明该测试用例具有较强的错误检测能力。反之，如果某些变异体未能被测试用例检测到，则说明测试用例可能存在漏洞或覆盖不足[3]。

变异技术的应用具有多重意义。首先，它能够有效评估测试用例的质量。由于变异技术引入的变异体通常模拟真实世界中的编程错误，因此能够帮助开发者识别测试用例在捕捉潜在缺陷方面的能力[1]。此外，变异技术还帮助提高测试覆盖率，通过系统生成多种变异体，可以模拟大量潜在错误场景，确保测试用例对各种情况维持良好的覆盖。最后，通过分析未能“杀死”的变异体，开发者可以识别软件中的潜在风险和隐含缺陷。

尽管变异技术具有显著优点，但在实际应用中依然面临诸多挑战。例如，生成和执行大量变异体可能带来较高的计算成本，尤其在处理复杂系统时，这种计算开销更为突出[2]。同时，等价变异体的存在也是变异技术面临的主要问题。这些等价变异体虽经变异操作，其行为却与原始程序一致，无法被测试用例识别，导致计算资源的浪费。

因此，如何高效生成有意义的变异体、减少等价变异体数量，以及加速变异测试的执行，已成为当前研究中的热点。

综上所述，变异技术作为自动化测试中的重要手段，不仅能够增强测试用例的覆盖范围和有效性，还为软件质量的保证提供了强有力的支持。其在自动化测试中的广泛应用，不仅推动了测试技术的发展，也为软件开发实践中的质量提升提供了保障。

1.2 研究目的

本论文的研究目的是系统性地总结和分析变异技术在自动化测试中的应用现状、研究进展及其未来发展方向。具体而言，本文将围绕以下几个关键问题展开研究：

1.2.1 变异技术的现状与发展趋势

变异技术作为自动化测试的核心技术之一，近年来在研究和实践中取得了显著进展。当前，变异技术已被广泛应用于软件测试的多个领域，包括单元测试、集成测试及回归测试等。通过对相关研究文献的梳理，可以发现变异技术的发展历程从早期的基本变异算子到如今的复杂变异策略，经历了多个重要阶段。

目前，研究者们不仅在变异体生成和筛选方面进行了深入探讨，还在变异技术与其他测试方法的结合应用上展现出新的活力。

未来，变异技术有望在智能化测试、持续集成及交付等领域发挥更大作用，推动软件测试的自动化和智能化进程。

1.2.2 变异技术的设计思路与策略

变异测试的效果在很大程度上依赖于变异算子的设计、变异体生成策略以及变异体筛选机制。现有研究中，针对变异算子的设计思路呈现多样化趋

势，研究者们通过不同的变异策略提升测试用例的有效性和覆盖率。例如，某些研究侧重于等价变异体的设计，以减少冗余测试，而另一些则关注在复杂环境下的高效生成与筛选。

本文将分析这些设计思路和策略在不同测试场景中的适用性与有效性，探讨如何根据具体需求选择合适的变异方法。

1.2.3 变异技术的局限性与挑战

尽管变异技术在自动化测试中发挥了重要作用，但在应用过程中依然存在诸多不足之处。首先，等价变异体问题导致了大量冗余变异体的生成，从而降低了测试效率。其次，变异测试的计算开销相对较大，特别是在大规模系统中，可能会影响测试的可行性。此外，测试用例的适用性问题使得某些变异体难以被有效覆盖。

针对这些挑战，本文将详细分析现存的局限性，并提出改进方向和策略，以期提升变异技术的实际应用价值。

1.2.4 未来研究方向

基于对现有文献的系统分析，本文展望变异技术未来的研究方向，提出一些新的可能性。例如，将人工智能技术与变异技术相结合，以实现更加智能化的变异体生成与筛选，提升测试的自动化程度。

此外，探索变异技术在新的应用领域中的潜力，如深度学习、云计算和物联网等，将为变异技术的发展提供新的动力。未来，变异技术将不断演化，以应对日益复杂的软件测试需求，推动软件质量的提升。

1.3 论文结构

为实现上述研究目标，本文结构安排如下：

（1）第一章引言：本章首先介绍了自动化测试的重要性，重点阐述了变异技术在自动化测试中的应用及其在提高测试质量中的重要作用。其次，本章明确了论文的研究目的，并概述了本文的结构安排。

（2）第二章变异技术概述：本章将详细介绍变异技术的基本概念和原理，讨论其分类、核心流程及关键技术。同时，对变异算子的设计与变异体生成策略进行系统分析。

（3）第三章变异技术的研究进展与现状：本章将全面梳理变异技术在不同测试领域中的应用进展，涵盖模糊测试、基于搜索的测试、随机测试、深度学习框架测试、操作系统内核测试等多个领域的研究成果。

（4）第四章变异技术的局限性与挑战：本章将重点讨论变异技术在实际应用中的局限性，包括等价变异体的处理、计算成本的优化问题以及测试用例有效性的提升。

（5）第五章未来的研究方向：本章将在前述分析的基础上提出未来的研究方向，探索新的变异算子设计思路、智能化变异体生成策略及跨领域应用的可能性。

（6）第六章结论：本章将总结全文的主要研究成果，并对变异技术的未来发展提出进一步的建议。

2 变异技术概述

变异技术在自动化测试领域中占据重要地位，能够有效提升测试用例的质量与覆盖率。通过改变程序代码以创建变异体，这些变异体模拟潜在的缺陷，从而帮助测试人员评估现有测试用例的有效性。

本章将对变异技术进行全面而系统的概述，包括其定义、分类和基本原理。

2.1 变异技术的定义

变异技术、是自动化测试中的一种核心方法，其主要目标在于评估和提升测试用例的质量。变异技术通过对被测系统（通常是源代码或模型）引入微小的、可控的变更，生成所谓的“变异体”。这些变异体通过一定的变异算子（Mutation Operators）生成，用于模拟真实世界中可能出现的编程错误。执行变异体后，测试用例是否能够检测到变异体的行为差异是评估其有效性的关键。如果测试用例能够成功“杀死”变异体，即准确地检测到因变异引入的错误，则认为该测试用例具备较高的故障检测能力[6]。

变异技术不仅可以帮助开发人员识别测试用例的缺陷和潜在的错误覆盖区域，还能促使测试用例的改进，以提高整体软件系统的质量。

在现代软件测试中，变异技术被广泛应用于各种测试场景，包括变异测试、模糊测试以及基于搜索的测试等[7]。

2.2 变异技术的分类

变异技术可根据其在不同测试领域中的具体应用需求分为几大类。

变异测试是变异技术的基础应用，通过对源代码应用特定的变异算子（如修改变量、操作符替换等）生成变异体，用于评估测试用例的覆盖能力，最早用于编程错误检测，重点在于分析代码逻辑的变异性[8]。

模糊测试（Fuzz Testing）通常生成大量随机或半随机输入，以使系统崩溃或出现异常，从而识别安全漏洞和隐藏缺陷。在模糊测试中，变异主要集

中于输入数据层面的变化，通常对输入格式和协议进行轻微的调整，模拟可能的输入异常[9]。

基于搜索的软件测试（Search-Based Software Testing, SBST）则通过优化算法生成变异体，以寻找能够最大化覆盖的测试路径或测试用例，常用的变异算子包括变异种群、遗传算法和模拟退火等，侧重于寻找符合特定目标的变异体生成策略，以提高测试效率[10]。

随机测试（Random Testing）以非确定性方式生成变异体，并通过统计结果评估测试覆盖范围，此方法中的变异算子相对简单，主要旨在迅速生成大量多样化的变异体，以增加测试的广度[11]。

随着深度学习技术的广泛应用，深度学习框架的测试需求也日益增加，这类测试中的变异技术侧重于模型结构、数据流和训练过程中的变异操作，模拟网络权重、激活函数或输入数据的细微变化，以检测模型在不同数据和结构变异下的稳定性[12]。

在操作系统内核测试中，变异技术主要集中于系统调用、内存管理和文件系统等关键模块，通过变异系统调用的参数或程序流程，模拟潜在系统错误，以确保内核的鲁棒性和安全性[13]。

最后，协议测试中的变异通过改变通信协议的特定字段或帧结构，测试协议实现的健壮性，这种方法多应用于网络协议和通信协议，帮助识别协议在非预期输入或通信中断情况下的表现[14]。

不同领域的变异测试应用根据目标和环境的差异，对变异算子的设计和变异体生成策略等有所不同，这些差异确保了变异技术在不同测试场景中能有效地提供故障检测和提升覆盖率。

2.3 变异技术的基本原理

变异技术的核心在于变异体生成和变异算子的设计，其基本原理包括以下几个方面：

2.3.1 变异体生成

变异体生成是变异技术的基础，通过对原始程序或系统引入变异操作生成多个版本。每一个变异体都含有特定类型的“错误”或变化，用于模拟现实中可能出现的错误类型。生成策略通常包括全局变异、局部变异和条件变异，分别对代码中的所有可能元素、局部代码块和特定逻辑条件进行更改[15]。

2.3.2 变异算子的设计

变异算子定义了生成变异体的具体方法，是变异技术中至关重要的设计环节。常用的变异算子类型包括：

（1）算术变异算子：如加减乘除操作的替换，主要用于模拟算法逻辑中的错误。

（2）逻辑变异算子：如条件判断操作的修改，帮助模拟条件判断中的常见错误。

（3）变量变异算子：如变量值的增加、减少或替换，常用于检查数据处理中的异常。

（4）输入变异算子：特别应用于模糊测试，通常对输入参数的类型和长度进行更改，以测试输入的极限。

（5）系统行为变异算子：如修改系统调用的顺序、延迟等，主要应用于内核或协议测试中[16]。

2.3.3 变异体筛选与优化

在生成大量变异体后，筛选与优化是确保变异测试效率的关键步骤。筛选通常包括去除等价变异体和减少冗余变异体，确保生成的变异体能够提供足够的信息增益，同时避免不必要的计算成本[17]。常用的筛选方法包括基于覆盖率和故障检测率的筛选。

此外，近年来，基于机器学习的变异体优化方法也取得了一定进展，进一步提升了变异技术的智能化和有效性。

2.3.4 等价变异体的识别

等价变异体是变异技术面临的主要挑战之一，这些变异体尽管经过变异操作，但其行为与原始程序一致，无法被测试用例检测到。因此，如何有效识别和处理等价变异体成为变异技术的研究重点，通常采用程序静态分析、符号执行等技术手段来识别这些等价变异体[18]。

变异技术作为自动化测试的重要组成部分，在不同测试领域中有着广泛的应用。通过细致的变异体生成和算子设计，变异技术不仅提高了测试的覆盖率和故障检测能力，还为复杂系统中的潜在风险和缺陷识别提供了有力支持。然而，变异技术在等价变异体处理、计算资源优化等方面仍然面临诸多挑战。

3 变异技术的研究进展与现状

3.1 变异测试中的进展

变异测试技术在近年的研究与应用中取得了显著进展，其核心理念在于通过生成和检测变异体来评估测试用例的覆盖性和有效性。研究者在变异算子的设计、等价变异体的识别以及测试效率的提升等方面进行了广泛探索。传统变异算子，如变量替换和逻辑运算符修改，虽然在早期应用广泛，但生成的变异体往往数量庞大且可能包含无效或等价的变异体。

因此，近年来研究者们开发了基于程序语义的高级变异算子，能够更精准地生成可能引发错误的变异体。例如，有研究通过语义分析设计“情境

敏感”算子，根据代码上下文生成更具意义的变异体，提高了检测潜在缺陷的能力[6]。

等价变异体是变异测试中的主要挑战之一。这些变异体尽管在结构上有所变化，但其行为与原始程序一致，使得它们无法被测试用例检测到。近年来，研究人员采用静态分析、符号执行等技术来识别和过滤等价变异体。此外，基于机器学习的预测模型在等价变异体识别中的应用也有效降低了计算成本。

在实际应用方面，许多案例展示了变异测试的有效性。例如，在Android应用测试中，研究人员通过定制适合移动应用的变异算子，成功提升了应用测试的覆盖率和错误检测能力。另一个实例是数据库系统的变异测试，通过生成变异体模拟复杂查询中的潜在错误，帮助优化数据库引擎的鲁棒性。

3.2 模糊测试中的变异技术

模糊测试作为一种基于随机输入的测试方法，通过变异输入数据来识别程序中的潜在漏洞和错误。将变异技术引入模糊测试显著增强了测试输入的多样性，从而提高了测试的广度和深度。模糊测试本身具备生成速度快、应用灵活的优点，尤其适用于输入数据格式不固定或接口复杂的系统。然而，其随机性高的特点可能导致测试覆盖率低或生成无效数据。

为此，基于结构的模糊测试（如AFL和LibFuzzer）通过对输入格式的解析和变异，生成符合特定格式的高质量测试数据，提高了测试效率[17]。

大量的研究和实验也证实了模糊测试的有效性。例如，针对网络协议的模糊测试实验表明，通过特定变异算子生成协议头字段的轻微变化，能够触发协议处理中的边界情况和异常行为。将模糊测试与变异技术结合，研究者们在测试深度和覆盖率上取得了显著提升，特别是在工业控制系统的研究中，未被检测的安全漏洞的发现显著提高了系统安全性。

3.3 基于搜索的软件测试

基于搜索的软件测试（Search-Based Software Testing, SBST）利用优化算法在广泛的搜索空间中自动生成和选择测试用例，以覆盖程序的复杂路径或提高测试效率。这种方法引入变异技术，特别是针对代码特征和路径生成的变异操作，可以进一步提升测试的覆盖率和质量。

以Evosuite为例，这一基于搜索的测试工具广泛应用于Java单元测试生成，其主要通过遗传算法生成高覆盖率的测试用例，并在进化过程中引入变异技术，确保用户生成的测试用例涵盖更多的代码逻辑[18]。

研究表明，Evosuite在生成的测试用例质量上具有明显优势，特别是在复杂路径的覆盖和边界测试方面。

此外，基于搜索的变异技术也应用于其他领域，如网络服务测试中，通过变异生成不同通信状态的协议字段，显著提高测试输入的多样性和程序覆盖率，尤其在高复杂度和高变异性的系统中，变异技术的效果尤为突显[19]。

3.4 深度学习框架与变异技术

随着深度学习的广泛应用，针对深度学习框架的变异测试技术也取得了显著进展。

深度学习系统的复杂性使得传统测试技术难以直接应用，而变异技术则通过引入微小的变异来模拟潜在错误场景，检测模型和框架的健壮性。例如，深度学习框架测试中的变异技术主要针对模型结构、权重参数和输入数据等方面进行，通过细微变更网络结构（如删除或增加神经元），可以生成变异模型进行鲁棒性验证[20]。

此外，研究表明，深度学习中的变异技术可帮助识别模型对数据扰动的敏感性，如在自动驾驶系统中，研究人员通过模糊或对抗性变异测试对模型反应进行评估，从而识别系统的潜在漏洞。这些研究成果也为深度学习在关键任务中的应用提供了保障。

3.5 操作系统与协议测试中的变异

变异技术在操作系统和协议测试中的应用主要集中于对关键模块（如系统调用、协议字段等）的变异，以模拟系统在边界条件或非预期输入下的表现。通过系统性变异，可以有效提高系统的稳定性和安全性。

在操作系统内核测试中，研究人员通过变异系统调用的参数或者程序状态生成测试用例，发现多项潜在漏洞，确保内核在极端条件下的鲁棒性[21]。

而在协议测试中，通过变异通信协议特定字段或帧结构，评估协议实现的健壮性，多数情况下有助于识别在非预期输入和通信中断情况下的表现，从而提升总体安全性[22]。

3.6 本章小结

第三章通过不同应用领域中的变异技术实例，展现了变异测试、模糊测试、基于搜索的软件测试、深度学习框架测试及操作系统与协议测试中的变异技术的最新研究进展。

这些研究不仅丰富了变异技术的应用场景，也推动了变异算子设计和等价变异体处理等关键问题的解决。不同领域的变异技术应用各有侧重，提供

了丰富的参考以促进未来变异技术的跨领域应用和进一步优化[23-24]。

4 变异技术的局限性与挑战

4.1 当前变异测试技术的不足

4.1.1 等价变异体问题

等价变异体是指在变异过程中生成的变异体与原始程序在功能上等效，不会引发不同的输出行为，导致测试用例无法检测到这些变异体的存在。这种等价变异体的存在不仅增加了测试成本，还会对测试效果产生误导。

等价变异体的识别是一个计算复杂的问题，传统的静态分析或手工检查方法难以全面处理。当前研究在等价变异体检测方面虽有所进展，但大多数解决方案仍具有较高的计算开销，难以适用于复杂的系统。

（1）生成和执行大量变异体会带来巨大的计算成本，尤其在大型系统或复杂程序中，这一问题尤为显著。变异体的生成、存储、执行以及测试结果的收集都需要大量的计算资源。尤其在对深度学习模型、操作系统内核等复杂系统进行变异测试时，庞大的变异体数量会使得整个测试过程异常耗时，并带来高昂的硬件资源消耗。这一问题限制了变异测试的实际应用和推广。

（2）变异测试的效果在很大程度上依赖于测试用例的质量。若测试用例设计不合理，或覆盖率不足，则即便生成了大量变异体，仍可能导致重要缺陷未被发现。此外，某些自动生成的测试用例尽管能杀死变异体，但无法有效解释或验证故障原因，导致测试结果缺乏实际指导意义。这种情况在复杂领域（如深度学习、操作系统等）中尤为突出，因为这些领域的测试用例通常需要考虑多维度的逻辑覆盖。

（3）当前的变异测试技术多应用于传统的代码级测试，对于高层次的系统（如分布式系统、嵌入式系统等）和异构系统中的应用适用性较低。在这些系统中，不同模块和协议之间的复杂交互增加了变异测试的难度。变异技术在这些场景下的扩展性和适用性问题尚未得到有效解决，需要进一步研究适用于特定应用场景的变异算子和变异策略。

（4）尽管自动化测试在一定程度上减轻了人工负担，但变异测试过程中仍需要人工干预，尤其在等价变异体的识别、测试用例的生成和适配、结果分析等环节。人工干预会增加测试的成本和时间，同时增加了人为因素的干扰。此外，对于等价变异体的处理也可能需要人工识别和调整，这在实际应用中会极大地影响变异测试的自动化程度和应用效率。

4.2 提升空间与改进建议

针对上述变异测试技术的局限性，当前研究者已提出多种改进策略，以下是针对主要问题的优化建议

（1）为了解决等价变异体的识别问题，可以考虑结合程序分析和机器学习技术。基于静态分析和符号执行的方法可以帮助自动识别等价变异体，减少不必要的计算负担；同时，使用机器学习算法构建预测模型，根据变异体的特征预测其是否为等价变异体，从而进一步提升识别的准确性。此外，结合程序分支覆盖信息，对无差异行为的变异体进行快速过滤，也是减少等价变异体影响的有效手段。

（2）针对高计算成本问题，可采取分层生成变异体的策略，通过选择性执行关键路径或高优先级变异体，降低计算量。分层生成方法可以基于变异算子的影响范围，对影响较大的变异体优先生成并执行，确保有限资源下的测试效果。同时，可以采用增量变异测试的方法，在每次代码更改后仅生成和测试与更改相关的变异体，减少重复计算和测试成本。

（3）为提高测试用例的有效性，研究者可以借助自动化生成技术与人工干预相结合的方法优化测试用例。基于路径覆盖、错误注入等技术生成更加多样化的测试用例，确保测试用例在逻辑分支上的覆盖。此外，可以引入自动化的结果分析工具，通过对比变异体和原始程序的行为差异来评估测试用例的有效性，保证生成的测试用例在实际测试中具有较高的故障检测能力。

（4）针对异构系统和复杂交互系统的适用性问题，可以开发专用的变异算子和变异策略。例如，在分布式系统中，可设计特定的网络通信变异算子，模拟不同节点间的通信异常；在嵌入式系统中，则可以开发硬件交互相关的变异算子。这种场景化的变异设计将提高变异测试在特殊领域的适用性，使其更具针对性和实用性。

（5）针对变异测试中的人工干预需求，人工智能和深度学习技术可以在以下方面提供支持：首先，通过构建测试用例生成模型，基于历史数据自动生成适应特定变异体的测试用例；其次，在变异体结果分析中应用深度学习模型，自动识别和标记等价变异体，减少人工干预。此外，借助AI技术可以在测试过程中动态优化变异体生成和筛选策略，实现更高效的自动化变异测试。

（6）云计算与分布式测试平台变异测试的计算成本问题可以通过分布式测试平台或云计算资源来部分解决。将变异体的生成和测试任务分布到多个节点进行并行执行，利用云端的计算资源和存储

资源，可以显著缩短测试时间。此外，分布式测试平台可以通过任务调度和资源分配优化，提高整个变异测试过程的效率和覆盖率。

4.3 本章小结

第四章探讨了当前变异测试技术的局限性和挑战，分别从等价变异体、计算成本、测试用例质量、适用性和人工干预等方面进行了分析。

针对这些问题，本章提出了多项改进建议，包括等价变异体识别优化、分层生成与选择性执行、测试用例质量提升、场景化变异技术开发、人工智能辅助自动化和云计算支持的分布式测试平台等。这些策略不仅能够在不同层面上提高变异测试的效率和适用性，也为未来变异技术的进一步发展提供了方向和参考。

5 未来的研究方向

5.1 研究框架构建

为了推动变异技术的发展并解决当前技术的局限性，未来的研究首先应明确关键的研究问题和目标，构建系统性的研究框架。研究框架的构建可以围绕以下核心问题展开：

5.1.1 关键研究问题

（1）如何进一步提升变异体生成的效率并减少等价变异体的生成：通过优化变异算子和引入智能化技术，减少不必要的计算资源浪费，提高变异体的生成效率。

（2）如何优化测试用例的自动化生成与筛选，以实现更高的测试覆盖率和错误检测能力：利用机器学习和搜索优化算法，自动生成高质量的测试用例，并设计高效的筛选机制，确保测试用例的有效性。

（3）如何增强变异技术在复杂系统（如深度学习模型、分布式系统、嵌入式系统等）中的适用性：针对不同系统和应用场景，开发特定的变异策略和测试框架，提高变异技术在复杂系统中的应用效果。

（4）如何引入智能化技术提升变异测试的自动化水平，以减少人工干预需求：通过人工智能和深度学习技术，实现变异体生成、测试用例筛选和结果分析的自动化，降低人工干预的成本和时间。

5.1.2 研究框架设计

基于上述研究问题，可以构建一个分层次的研究框架，包含以下模块：

（1）变异算子设计与优化：针对不同系统和应用场景，探索特定场景下的变异算子设计方法，并开发更加高效和智能化的算子生成机制。

（2）自动化测试用例生成与筛选：利用机器学习、搜索优化等算法，开发基于目标覆盖率的测试用例生成策略，同时设计自动筛选机制来过滤无效和低覆盖的用例。

（3）智能化等价变异体识别：基于深度学习、符号执行和静态分析等技术，构建等价变异体的智能识别系统，进一步减少不必要的计算资源浪费。

（4）跨领域的变异技术应用探索：研究变异技术在深度学习框架、操作系统内核、嵌入式系统、协议测试等新兴领域中的潜力和应用模式，并针对这些领域开发特定的变异策略和测试框架。

5.1.3 实验验证与评估体系

未来研究框架中还应包括一套标准化的实验验证和评估体系。该体系可以通过以下方面实现：

（1）覆盖率与检测率评估：针对生成的变异体，设计一套基于代码覆盖率和错误检测率的评估指标，以量化测试效果。

（2）性能与资源效率评估：通过计算资源使用、执行时间、测试效率等指标，评估变异技术在不同应用场景中的效率表现。

（3）对比实验：通过不同变异技术的对比实验，验证新方法相较于传统方法的有效性和优越性，为未来的应用提供可靠的数据支持。

5.2 新的研究方向

未来变异技术的发展将朝着更加智能化、高效化和跨领域的方向迈进，以下是一些具有潜力的新研究方向：

5.2.1 智能化变异体生成与筛选

未来的变异技术可以借助机器学习和深度学习模型来智能生成变异体和筛选测试用例。通过对已有测试数据进行深度学习模型训练，自动识别不同场景中可能导致故障的高风险代码位置，并优先生成这些位置的变异体。

此外，通过深度学习预测和筛选算法，可自动剔除无效变异体并聚焦高优先级的变异体，提升测试效率。

5.2.2 变异技术与强化学习的结合

将变异技术与强化学习相结合，利用强化学习算法在动态测试过程中逐步优化变异算子的选择和变异体的生成。强化学习能够根据测试过程中的反馈不断调整变异策略，以获得更高的代码覆盖率和错误检测率。例如，可以将测试过程视作一个多步决策过程，通过强化学习算法自适应调整变异体生成策略，以在特定领域中取得最佳测试效果。

5.2.3 跨领域与多场景变异测试应用

随着深度学习、物联网、自动驾驶等新兴技术的发展，变异技术在新领域中的应用潜力巨大。未

来研究可以探索变异技术在这些领域的应用模式和挑战：

（1）深度学习中的变异：进一步研究变异技术在深度学习模型测试中的应用，包括对网络结构、数据输入和训练过程的变异，以检测模型在不同场景下的鲁棒性。

（2）自动驾驶系统的变异测试：自动驾驶系统依赖大量传感器数据和复杂的控制算法，通过变异技术模拟传感器数据异常、交通场景变化等情况，可以评估系统在实际环境中的稳定性和安全性。

（3）物联网协议与嵌入式系统：在物联网设备和嵌入式系统中，变异技术可以模拟协议交互中的异常情况，识别潜在的安全漏洞。此外，针对硬件限制开发轻量化的变异算子，以提高嵌入式系统的变异测试效率。

5.2.4 变异技术与自然语言处理的结合

随着自然语言处理（NLP）在软件文档、代码生成等领域的应用不断增多，未来研究可以探索NLP与变异技术的结合。例如，使用NLP生成与系统需求相匹配的测试用例或生成多样化的变异描述以提升变异算子的多样性。这种结合能够将测试用例的生成与系统需求更好地匹配，提升测试的针对性。

5.2.5 基于云计算的分布式变异测试平台

面向大型软件系统的变异测试，未来研究可以借助云计算和分布式架构设计高效的测试平台。通过将变异测试任务分布到多个节点并行执行，充分利用云计算的弹性资源，显著提升变异测试的执行效率。

此外，可以基于云端数据共享和协作特性，构建集中管理的变异库和测试用例库，方便不同测试团队进行资源共享和数据对比，从而提高变异技术的复用性和应用效率。

5.2.6 基于量子计算的变异优化

量子计算以其超强的计算能力为变异技术优化带来了新的可能性。未来研究可以探讨在量子计算环境下实现变异体生成和筛选的并行加速。

量子算法有望提升变异体生成的速度，尤其在复杂系统的测试中，通过量子计算可以有效减少测试时间。此外，量子计算还可以用于加速等价变异体的检测和识别，解决当前计算资源的限制问题。

5.3 本章小结

第五章围绕变异技术未来研究的可能方向，提出了系统化的研究框架和新兴领域的探索路径。未来的变异技术研究不仅需要在智能化生成、筛选等方面进行优化，也需要拓展其在深度学习、自动驾驶、物联网等新兴领域中的应用模式。

同时，云计算、强化学习和量子计算等新技术的结合为变异测试提供了新的动力。

通过探索这些方向，变异技术将更加广泛地应用于复杂系统的测试中，进一步推动软件测试领域的革新和发展。

6 结论

6.1 研究总结

本论文系统性地探讨了变异技术在自动化测试领域的应用、研究进展、局限性与挑战，并针对这些问题提出了改进策略和未来研究方向。在当前的技术发展背景下，变异技术已被广泛应用于变异测试、模糊测试、基于搜索的测试、深度学习框架测试、操作系统内核测试等多个领域。通过对比和分析不同测试场景下的变异技术应用，本文提出了以下主要贡献：

（1）对变异技术现状的综合分析：本文详细总结了变异技术在模糊测试、基于搜索的测试、深度学习框架测试、内核和协议测试中的应用情况，阐明了变异技术在不同应用场景下的独特作用与效果。

（2）变异技术局限性的深入探讨：本文指出当前变异测试技术面临的主要问题，包括等价变异体的难以识别、高计算成本、测试用例质量限制和应用适用性问题。

通过这些分析，为后续研究者明确了变异技术中的关键瓶颈和优化目标。

（3）针对性的改进策略：基于对变异技术局限性的研究，本文提出了一系列改进建议，如智能化等价变异体识别、分层生成与选择性执行、跨领域的变异算子设计和云计算支持的分布式测试平台。

这些改进策略为提升变异测试的效率、适用性和自动化程度提供了具体思路。

（4）未来研究框架和方向的展望：本文构建了未来变异技术研究的框架，提出智能化变异体生成、跨领域应用拓展、强化学习和量子计算在变异技术中的应用等前沿方向。

这一框架为未来的变异技术研究提供了系统化的路径，指明了跨领域扩展和新兴技术结合的潜力。

综上所述，本研究通过对变异技术的全面分析和改进建议，为变异测试的理论基础和应用实践提供了支持，进一步推动了自动化测试领域的研究进展。

6.2 未来研究的建议

变异技术的未来发展具有极大的潜力，特别是在智能化和跨领域应用方面。针对当前研究中的不足和技术趋势，本文提出以下建议以进一步推动变异技术的发展：

（1）推进智能化和自动化随着人工智能技术的不断进步，未来研究可重点探索深度学习、机器学习在变异体生成、筛选和等价变异体识别中的应用。通过智能化手段实现测试过程中的自适应和动态调整，将有助于减少人工干预，提高测试效率和精准度。

此外，强化学习等方法可以为变异技术带来更高的自动化水平，降低资源消耗并提高覆盖率。

（2）开发跨领域的变异测试工具变异技术在新兴技术领域（如自动驾驶、物联网、深度学习）中的应用尚处于起步阶段。未来可以针对这些领域的特殊需求，开发适配性强、灵活性高的变异测试工具，为不同领域的系统提供定制化的测试支持。

此外，跨领域的变异测试工具应考虑异构系统和分布式环境的特性，确保变异技术能够广泛应用于多样化的系统结构。

（3）结合量子计算以提升计算效率变异测试的计算复杂度在大型系统中尤为突出，而量子计算在并行处理和优化算法方面的优势有望解决这一问题。未来研究可以探索在量子计算环境中实现变异体的并行生成与筛选，以缩短测试执行时间，并在复杂系统的测试场景中拓展变异技术的应用边界。

（4）加强测试结果的可解释性随着测试系统复杂性的增加，测试结果的可解释性显得尤为重要。未来研究应注重构建更透明的变异测试评估体系，并采用可视化技术帮助测试人员理解变异体生成和筛选过程中的关键决策。

这将提高测试结果的可信度，并为开发者提供更多关于测试缺陷和潜在风险的信息。

（5）探索基于云计算的分布式测试平台面向资源密集型的变异测试任务，未来研究可以依托云计算的分布式架构设计高效的变异测试平台。借助云端的计算资源和分布式任务管理，可以提升变异测试的资源利用率和执行效率。

同时，构建一个集中管理的测试资源库和变异库，支持变异测试在不同团队和项目中的复用。

6.3 本章小结

本章总结了论文的主要研究成果，并提出了未来研究的若干建议。随着变异技术在自动化测试领域应用的不断拓展，未来研究不仅需要在智能化、效率优化等方面进行改进，还需要针对不同行业和领域的需求进行定制化创新。

通过结合人工智能、量子计算、云计算等前沿技术，变异技术有望在更广泛的应用领域内实现突破，推动软件质量保障体系的进一步完善和升级。

参 考 文 献

[1] Delgado-Pérez, P., Medina-Bulo, I., Merayo, M. G. Using evolutionary computation to improve mutation testing. Proceedings of the 14th International Work-Conference on Artificial Neural Networks (IWANN), 2017.

[2] Zhang, J., Wang, Z., Zhang, L., et al. Predictive mutation testing. IEEE Transactions on Software Engineering, 2019.

[3] Sun, C., Jia, J., Liu, H., et al. A lightweight program dependence based approach to concurrent mutation analysis. Proceedings of the 42nd Annual Conference on Computer Software and Applications (COMPSAC), 2018.

[4] Mirshokraie, S., Mesbah, A., Pattabiraman, K. Guided mutation testing for JavaScript web applications. IEEE Transactions on Software Engineering, 2015.

[5] Sun, C., Guo, X., Zhang, X., Chen, Z. A data flow analysis based redundant mutants identification technique. Computer Journal, 2019.

[6] Sun, C., Fu, X., Guo, X., Chen, T. ReMuSSE: A redundant mutant identification technique based on selective symbolic execution. IEEE Transactions on Reliability, 2022.

[7] Zhang, L., Marinov, D., Khurshid, S. Faster mutation testing inspired by test prioritization and reduction. Proceedings of the 7th International Symposium on Software Testing and Analysis (ISSTA), 2013.

[8] Just, R., Ernst, M. D., Fraser, G. Efficient mutation analysis by propagating and partitioning infected execution states. Proceedings of the 8th International Symposium on Software Testing and Analysis (ISSTA), 2014.

[9] Wang, B., Xiong, Y., Shi, Y., et al. Faster mutation analysis via equivalence modulo states. Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), 2017.

[10] Mateo, P. R., Usaola, M. P. Parallel mutation testing. Software Testing, Verification & Reliability, 2013.

[11] Sun, C., Zhao, Y., Pan, L., Liu, H., Chen, T. Y. Automated testing of WS-BPEL service compositions: A scenario-oriented approach. IEEE Transactions on Services Computing, 2018.

[12] Sun, C., Liu, Y., Wang, Z., et al. A data mutation directed metamorphic relation acquisition methodology. Proceedings of MET 2016, collocated with ICSE, 2016.

[13] Sun, C., et al. Metamorphic testing for web services: Framework and a case study. Proceedings of the 9th IEEE International Conference on Web Services (IEEE ICWS), 2011.

[14] Sun, C., Fan, C., Wang, Z., et al. A path-aware mutation analysis guided approach to regression testing. Proceedings of AST 2017, collocated with ICSE, 2017.

[15] Ma, L., et al. DeepMutation: Mutation testing of deep learning systems. 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE), 2018.

[16] Dang, X. Enhancement of mutation testing via fuzzy clustering and multi-population genetic algorithm. IEEE Transactions on Software Engineering, 2021.
[17] Lee, J., et al. Fuzzing for CPS mutation testing. 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2023.
[18] Wang, X. MuSC: A tool for mutation testing of Ethereum smart contracts. 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019.
[19] Ojdanic, M. Keeping mutation test suites consistent and relevant with long-standing mutants. ESEC/FSE, 2023.
[20] Beller, M. What it would take to use mutation testing in industry: A study at Facebook. Proceedings - International Conference on Software Engineering, 2021.
[21] Zhang, L. Faster mutation testing inspired by test prioritization and reduction. 2013 International Symposium on Software Testing and Analysis, ISSTA 2013.
[22] Le Thi My Hanh. Applying the meta-heuristic algorithms for mutation-based test data generation for Simulink models. ACM International Conference Proceeding Series, 2014.
[23] Jain, K. Contextual predictive mutation testing. ESEC/FSE, 2023.
[24] Gao, X. Fuzz testing based data augmentation to improve robustness of deep neural networks. 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE), 2020.
[25] Zhang, M. IntelliGen: Automatic driver synthesis for fuzz testing. Proceedings - International Conference on Software Engineering, 2021.
[26] Zhang, Q. BigFuzz: Efficient fuzz testing for data analytics using framework abstraction. 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2020.
[27] Guo, Q. Audee: Automated testing for deep learning frameworks. 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2020.
[28] Zhang, S. FuzzSlice: Pruning false positives in static analysis warnings through function-level fuzzing. ICSE, 2024.
[29] Sun, C., Jia, J., Liu, H., et al. Constraint-based model-driven testing of web services for behavior conformance. Proceedings of the 16th International Conference on Service-Oriented Computing (ICSOC), 2018.
[30] Ma, L., et al. DeepMutation: Mutation testing of deep learning systems. 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE), 2018.
[31] Dang, X. Enhancement of mutation testing via fuzzy clustering and multi-population genetic algorithm. IEEE Transactions on Software Engineering, 2021.
[32] Lee, J., et al. Fuzzing for CPS mutation testing. 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2023.
[33] Nguyen, H. L., et al. MoFuzz: A fuzzer suite for testing model-driven software engineering tools. 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2020.
[34] Bartocci, E., et al. Property-based mutation testing. 2023 IEEE Conference on Software Testing, Verification and Validation (ICST), 2023.
[35] Zhang, J., et al. Predictive mutation testing. IEEE Transactions on Software Engineering, 2019.
[36] Lemieux, C., et al. FairFuzz: A targeted mutation strategy for increasing greybox fuzz testing coverage. 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE), 2018.
[37] Delgado-Perez, P., et al. Search-based mutant selection for efficient test suite improvement: Evaluation and results. Information and Software Technology, 2018.
[38] Tambon, F., et al. A probabilistic framework for mutation testing in deep neural networks. Information and Software Technology, 2022.
[39] Luo, W., et al. Graph-based fuzz testing for deep learning inference engines. 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), 2020.
[40] Vikram, V., et al. Guiding greybox fuzzing with mutation testing. Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, 2023.
[41] Silva, R. A., et al. A systematic review on search based mutation testing. Information and Software Technology, 2017.
[42] Petrovic, G., et al. Practical mutation testing at scale: A view from Google. IEEE Transactions on Software Engineering, 2021.
[43] Papadakis, M., et al. Special issue on mutation testing. Information and Software Technology, 2017.

RUI Wan-Ling master.Her direction is Deep Learning and Java development.
YU Xiao master.Her direction is Deep Learning and Java development.
GUO Han-Yu master.His direction is Deep Learning and Java development.
HUANG Zhen-Bin master.His direction is Deep Learning and Java development.

Background

In the field of automated testing, mutation technology has become a key means to improve the effectiveness of software

testing with its unique advantages. By introducing human-designed mutation into the software under test (SUT), this technique simulates the possible faults, and then evaluates the quality and coverage of the existing test suite. This method can not only identify the deficiency of test cases, but also guide the optimization of test cases, so as to improve the accuracy and comprehensiveness of software testing.

With the increasing complexity and scale of software systems, the research and application of mutation technology has also ushered in rapid development. Researchers have made remarkable progress in the design of mutation operators, the generation strategy of mutants, and the screening mechanism of mutants. These research results not only deepen our understanding of mutation technology, but also provide strong support for the actual software testing practice.

This paper aims to systematically summarize the research motivation and design ideas of mutation technology, and deeply discuss its application in different testing fields, including fuzzing, search-based testing, deep learning framework testing, and operating system kernel testing. Through the analysis of these application scenarios, this paper reveals the value and potential of mutation techniques in practical software testing.

At the same time, this paper also focuses on the limitations and shortcomings of mutation technology in the existing research. These challenges include the applicability of mutation operators, the efficiency of mutant generation, and the scalability of mutation testing. In response to these challenges, we propose possible future research directions in order to drive the continuous development and improvement of mutation techniques.

In summary, the background of this paper not only provides readers with an overview of the research background and applications of mutation techniques, but also lays the foundation for the in-depth analysis and discussion in the following. Through a comprehensive review of mutation technology, this paper aims to provide valuable reference and guidance for researchers and practitioners in this field, and jointly promote the progress of software testing technology.