南京大学

研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目 基于遗传算法的变异体集合约简方法

作 者 姓 名 刘芳潇

学科、专业名称 工程硕士（软件工程领域）

研 究 方 向 软件工程

指 导 教 师 陈振宇 教授

2021 年 **5** 月 **20** 日

学　　号：**MF1932113**

论文答辩日期：**2021** 年 **05** 月 **20** 日

指 导 教 师：　　（签字）

Mutant Set Reduction Based on Genetic Algorithm

by

Fangxiao Liu

Supervised by

Professor Zhenyu Chen

A dissertation submitted to
the graduate school of Nanjing University
in partial fulfilment of the requirements for the degree of
MASTER OF ENGINEERING
in
Software Engineering

Software Institute
Nanjing University

May 20, 2021

学 位 论 文 原 创 性 声 明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不句含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名：_____

日期：　　年　　月　　日

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于遗传算法的变异体集合约简方法

工程硕士（软件工程领域）专业 2019 级工程硕士生姓名：刘芳潇

指导教师（姓名、职称）：陈振宇 教授

摘　要

变异测试通过执行变异操作以模拟典型软件缺陷，可以有效衡量测试用例集的缺陷检测能力。然而，变异操作通常会生成大量的变异体，编译、执行这些变异体会耗费大量的测试时间，导致变异测试效率低下、可用性不高。为了优化变异测试的效率，提升变异测试在实践中的可用性，测试人员通常采用变异选择方法对变异体集合进行约简。在执行变异选择方法时，需要多次执行变异测试以计算每次选择的变异体集合的质量得分，用以选择质量最优的子集，所以此类方法的计算开销依旧十分昂贵。

为了解决现有变异选择方法存在的问题，首先，本文提出了一种基于机器学习的变异体集合质量预测方法 MSQP，通过提取源程序、测试用例集和变异体的顽固特征和充分特征，预测变异体集合的质量得分，避免动态执行变异测试，以节省昂贵的计算开销。其次，本文提出一种基于遗传算法的变异体集合约简方法 GAR，本文将 MSQP 作为约简方法的适应度函数，用以评价变异体子集的优劣。应用基于遗传算法的变异体集合约简方法进行变异集合约简，可以提高变异选择的收敛速度，降低变异测试的代价。随后本文基于 React 和 Flask 框架设计并实现了一个变异体集合约简系统。该系统由变异体生成模块、质量模型构建模块、变异体集合约简模块与测试报告生成模块构成，实现了变异集合约简的功能，具有较强的可用性。

本文在先前软件测试研究中广泛使用的 9 个项目上进行了实证研究，以评估本文方法的有效性。实证研究证明：(1) 随机森林回归算法在构建预测模型时表现最好；MSQP 在有效性和效率方面都表现优秀，具有较高的性能，可作为 GAR 的适应度评价模型对变异体集合进行约简；(2) 每一类别的特征对于构建变异体集合质量预测模型都发挥了关键作用，并且缺一不可；(3) GAR 在有效性和效率方面都表现优秀，具有较高的性能，并且在确保约简后变异体子集测试效果的同时，显著减少了约简时间，达到了降低变异测试计算开销的目的。

关键词： 变异测试优化，机器学习，遗传算法

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Mutant Set Reduction Based on Genetic Algorithm
SPECIALIZATION: Software Engineering
POSTGRADUATE: Fangxiao Liu
MENTOR: Professor Zhenyu Chen

Abstract

Mutation testing can effectively detect the adequacy of the test set by performing mutation operations to simulate typical software defects. However, mutation operations usually generate a large number of mutants. Compiling and executing these mutants will consume a lot of time, resulting in inefficient and low usability of mutation testing. In order to optimize the efficiency of mutation testing and improve the usability of mutation testing in the industry, testers usually use mutation selection methods to reduce mutant set. When performing mutation selection methods, it is necessary to perform mutation testing multiple times to calculate the quality score of each selected mutant set to select the best quality subset, so the computational overhead of this type of method is still very expensive.

In order to solve the problems of the existing mutation selection methods. Firstly, this thesis proposes a quality score prediction method based on machine learning (MSQP), by extracting stubborn features and adequate features of the origin program, test case set and the mutant , predict the quality score of the mutant set, avoid dynamic execution of mutation testing, and save expensive computational overhead. Secondly, this thesis proposes a mutant set reduction method based on genetic algorithm(GAR). This thesis uses MSQP as the fitness function of the reduction method to evaluate the pros and cons of a subset of the mutant set. Applying the mutation set reduction method based on genetic algorithm to reduce the mutant set can improve the convergence speed of mutation selection and reduce the cost of mutation testing. Subsequently, this thesis designs and implements a mutant set reduction system based on the React and Flask framework. The system consists of mutant generation module, quality model building module, mutant set reduction module and test report generation module, which realizes mutation set reduction function, with strong usability.

In this thesis, an empirical study was conducted on nine items widely used in previous software testing studies to evaluate the effectiveness of this method. The experimental results show that: (1) Random forest regression algorithm performs best when constructing predictive models; MSQP performs well in terms of effectiveness and efficiency, has high performance, and can be used as a fitness evaluation model in GAR for mutant set reduction;(2) The features of each category play a key role in constructing the quality score prediction model of the mutant set, and none of them are indispensable;(3) The genetic algorithm-based mutant set reduction method performs well in terms of effectiveness and efficiency, has high performance, and while ensuring the test effect of the mutant subset reduction, it significantly reduces the reduction time , and achieve the purpose of reducing the computational cost of mutation testing.

keywords: Mutation Testing Optimization, Machine Learning, Genetic Algorithm

目　　录

图目录

表目录

第一章　绪论

1.1　课题背景和意义

软件测试是保障软件质量的重要手段，测试用例设计是软件测试中的核心步骤，缺陷检测能力是衡量测试用例集质量的重要方面 [1, 2]。变异测试通过执行变异操作以模拟典型软件缺陷，可以有效衡量测试用例集的缺陷检测能力 [3]。然而，变异操作通常会生成大量的变异体，编译、执行这些变异体会耗费大量的测试时间，导致变异测试效率低下、可用性不高 [4]。例如，Zhang 和 Hou 对 C 程序变异测试的实证研究发现，即便针对小型规模的西门子程序（513 行非注释代码）进行变异，也会产生数以万计的变异体（23,847 个）[5]。在如此大量的变异体上运行测试用例的成本很高。对于可能被杀死的每个变异体，可能会执行多次测试，直到找到一个能够杀死该变异体的测试为止。对于每个未杀死的变异体，需要执行全部的测试。因此，需要对变异测试的效率进行优化，来提高变异测试在实践中的可用性。

通过上述分析可知，影响变异测试效率的关键在于变异体集合的规模。因此为了降低变异测试成本，测试人员通常采用变异选择方法对变异体集合进行约简 [5]。此类方法使用一部分变异体来达到与传统变异测试相似的效果。也就是说，如果适合于变异体子集的测试用例集也适合于整个变异体集合，那么在评估测试用例集时，该子集被视为与整个变异体集合具有相同的有效性。即，该子集足以表示整个集合。研究人员通常采用变异体集合的质量得分衡量一个变异体子集在多大程度上代表了整个变异体集 [1]。子集具有的质量得分越高，子集越具有代表性。在执行变异选择方法时，需要多次执行变异测试以计算每次选择的变异体集合的质量得分，用以选择质量最优的子集，所以此类方法的计算开销依旧十分昂贵。为了解决这个问题，本文提出一种变异体集合质量预测方法 Mutant Set Quality Prediction（MSQP），旨在不执行变异测试的情况下获得变异体集合的质量得分，以节省变异约简的计算开销。

变异体集合质量得分预测的思路可以避免动态执行变异测试，以降低变异约简的计算开销。但是仅将此方法应用于现有的变异选择方法中，依然要执行

大量的选择操作。其原因在于现有方法选择变异体集合的次数是不可控的，可能要执行 MSQP 方法多次，所以整个执行过程存在十分耗时的风险。为了从本质上解决传统变异选择方法带来的问题，本文采用搜索算法，在可接受的时间内寻找最优解。遗传算法是最常用的搜索算法，其选择、交叉、变异等操作都是以一种概率方式进行的，增加了搜索过程的灵活性，而且能以较大概率收敛于最优解，具有较好的全局优化求解能力 [6]。所以本文提出一种基于遗传算法的变异体集合约简方法 Genetic Algorithm Reduction（GAR），运用遗传算法搜索一个最优的变异体子集，并将 MSQP 方法作为适应度函数应用到遗传算法中，以节省整个约简过程的计算开销。在此基础上，基于 GAR 本文应用 React 和 Flask 框架设计并实现了一个变异体集合约简系统。该系统由变异体生成模块、质量模型构建模块、变异体集合约简模块与测试报告生成模块构成，实现了项目上传、变异生成、变异信息查看、变异集合约简、约简结果查看、测试报告生成和测试报告查看等功能，具体较强的可用性。

1.2 国内外研究现状

本文研究基于遗传算法的变异体集合约简方法，主要涉及变异测试选择和变异测试预测两个方向，分别介绍国内外研究现状。

1.2.1 变异测试选择技术

变异选择策略旨在从给定的变异体集中选择代表性的子集。这样做的实际原因是为了约简变异体集合的规模，减少变异测试的计算成本。围绕本文中心，本节从随机变异选择和变异算子选择两个方面来介绍变异测试选择技术的研究现状。

随机选择方法尝试从生成的全部变异体中随机选择出部分变异体 [2]。Acree 等研究人员 [7] 率先提出随机变异选择技术。随后 Mathur 和 Wong[8, 9] 通过随机选择部分变异体对 Mothra[10] 中的 22 个变异算子产生的变异体进行了实证研究。实证研究表明，若随机选择 10% 的变异体，其有效性仅比使用全部变异体的变异得分低 16%。如果变异子集的取值超过全部变异集合的 10%，随机选择法是一种有效的优化方法。Papadakis 和 Malevris[11] 研究证明，随机选择 10%、20%、30%、40%、50% 和 60% 的变异体分别会导致大约 26%、16%、13%、10%、7% 和 6% 的故障损失。张路 [5] 构建的可杀死随机选择的变

异体集的测试用例集，其在所有变异体的杀伤力达 99% 以上。Gopinath 等研究人员 [12] 在大型开源程序上进行实证研究，结果表明选择少量恒定的变异体可以与使用所有变异体时获得相似的变异得分。

变异算子选择方法尝试从生成的全部变异体中通过筛选变异算子来缩小变异集合规模，以降低变异测试的计算开销 [2]。Wong 和 Mathur[8, 13] 研究了 Mothra[10] 的 22 个变异算子中的 2 个变异算子，发现用这 2 个变异算子产生的变异体可以获得与所有 22 个变异算子产生的变异体相似的结果。Offutt 等研究人员 [14] 通过实验确定了 Mothra 的 22 个变异算子中的 5 个变异算子，发现这 5 个变异算子对 10 个实验对象的有效性值在 99.0% 和 100% 之间，平均为 99.5%。他们还发现，如果没有这 5 个变异算子，则某些实验对象的有效性值将低于 99%，他们将其定义为一组充足的变异算子用于变异测试的最低要求。Wong 和 Mathur 的 2 个变异算子属于 Offutt 的 5 个变异算子。Barbosa 等研究人员 [15] 提出了六项准则来确定充足的变异算子。六个准则中的某些准则的应用需要大量变异体的编译和执行。根据六项准则，他们确定了 10 个变异算子，为了测量有效性，在 27 个实验对象上的 10 个变异算子的有效性值在 95.8% 和 100% 之间，平均为 99.6%。

当前针对变异测试选择技术的研究工作有两点不足之处：一是需要动态执行变异测试来计算变异子集的质量得分，计算开销十分昂贵；二是此类技术选择变异体子集的次数不确定，可能要执行变异测试多次，整个过程存在十分耗时的风险。因此本文首先提出一种变异体集合质量预测方法，旨在不执行变异测试的情况下获得变异体集合的质量得分，以节省变异约简的计算开销。其次提出一种基于遗传算法的变异体集合约简方法，并在遗传算法的选择过程中使用变异体集合质量预测方法评估变异体子集，以提高变异选择的收敛速度，降低变异测试的代价。

1.2.2 变异测试预测技术

机器学习已在许多领域提供了解决方案，以解决与成本相关的问题 [16]。近年来，机器学习技术广泛应用于变异测试领域，通过对实验对象进行挖掘和学习，可以提升变异测试的效率。本节主要介绍变异测试预测方面的研究。

张洁 [17] 率先提出预测性变异测试方法 PMT，该方法建立了一个分类模型，该模型预测了在不执行变异测试的情况下变异体的执行结果。该模型依赖于与变异体，测试和覆盖率度量相关的许多特征，并以相对较高的精度（超过

0.85 的精度和召回率）预测变异体的执行结果。Mao 等研究人员 [18] 在 PMT 的基础上进行了广泛的研究以评估预测变异测试的性能，他们还通过考虑更多特征和强大的深度学习模型来补充原始 PMT 工作。实验结果表明，实验项目的平均预测准确性超过 0.85，证明了 PMT 的有效性。同时，与使用 5 个线程的传统变异测试相比，平均速度也明显提高了 28.7 倍。Alireza 等研究人员 [19] 调查了未覆盖的变异体对 PMT 的研究结果的影响。具体而言，他们发现此类变异体大大降低 AUC 方面的性能。然后提出了一种基于集成学习的方法来解决该问题。实验结果表明，在 AUC，MCC 和平衡精度方面，所提出的方法优于其他机器学习技术。

本文借鉴了 PMT 的思路，从集合层面出发，在源程序、测试用例集和变异集合中提取顽固性特征和充分性特征，并采用机器学习技术预测变异体集合的质量得分，以避免动态执行变异测试，从而降低变异约简的计算开销。

1.3 本文主要工作

本文提出了基于遗传算法的变异体集合约简方法并实现了一个变异体集合约简系统，主要工作如下：

在方法上，本文将机器学习和遗传算法的思想引入变异体集合约简领域，通过机器学习模型预测变异体集合的质量得分，通过遗传算法搜索质量最优的变异体子集，以降低变异测试的计算成本。具体而言，本文首先分析源程序、变异体和测试覆盖结果，从中提取与变异体集合质量得分相关的特征，其次构建变异体集合质量预测模型预测质量得分，然后将预测模型作为的适应度函数输入至遗传算法，并进行质量最优子集的搜索。本方法通过避免动态执行变异测试和提升变异选择的收敛速度，达到了降低变异约简计算开销的目的。

在系统上，本文在理论研究的基础上，设计并实现了一个变异体集合约简系统。该系统由为变异体生成模块、质量模型构建模块、变异体集合约简模块和测试报告生成模块构成，实现了项目上传、变异生成、变异信息查看、变异集合约简、约简结果查看、测试报告生成和测试报告查看等功能。本系统基于前后端分离的架构进行实现，使用 React 作为前端框架，Flask 作为后端框架，两端统一采用 JSON 数据格式，利用符合 RESTful API 规范的接口进行通信；采用 MongoDB 和文件系统进行数据持久化管理；整体通过 Docker 容器技术进行打包，具有良好的可移植性，降低系统部署与运维方面的开销。

在实证研究上，本文首先提出三个研究问题：（1）基于机器学习的变异体集合质量预测方法在准确性和效率方面表现如何？（2）变异顽固程度特征和变异充分程度特征对基于机器学习的变异体集合质量预测模型的贡献如何？（3）基于遗传算法的变异体集合约简方法在准确性和效率方面表现如何？然后在先前的软件测试研究中广泛使用的 9 个项目上进行了实证研究，以评估本文方法的有效性。实验结果表明：（1）随机森林回归算法在构建变异体集合质量预测模型时表现最好；基于机器学习的变异体集合质量预测方法在有效性和效率方面都表现优秀，具有较高的性能，可作为遗传算法中的适应度评价模型对变异体集合进行约简；（2）每一类别的特征对于构建变异体集合质量预测模型都发挥了关键性的作用，并且缺一不可；（3）基于遗传算法的变异体集合约简方法在有效性和效率方面都表现优秀，具有较高的性能，并且在确保约简后变异体子集测试效果的同时，显著减少了约简时间，达到了降低变异测试计算开销的目的。

1.4 本文组织结构

本文详细介绍了基于遗传算法的变异体集合约简方法的设计和变异体集合约简系统的实现，其组织结构如下：

第一章，绪论。首先对本文的课题背景和意义进行了介绍，其次从变异测试选择技术、变异测试预测两个方面介绍了国内外研究现状，然后总结了本文的主要工作，最后阐述本文的组织结构。

第二章，相关概念与技术。介绍与本文相关的概念与技术，包括变异测试、机器学习算法以及遗传算法。

第三章，变异体集评价与约简方法。首先讨论评价变异体集合质量所需的特征，即高质量变异体子集所具备的特征；然后介绍构建高质量变异体子集的方法，即基于遗传算法的变异体集合约简方法。

第四章，需求分析与概要设计。首先对变异体集合约简系统进行需求分析，即通过对系统进行用例分析，确定系统的功能性需求和非功能性需求；然后根据需求分析的结果，对系统进行总体概要设计，即设计系统的整体架构图、4+1 视图和持久化存储表。

第五章，系统实现与测试。首先描述变异体集合约简系统实现，分别对为变异体生成、质量模型构建、变异体集合约简和测试报告生成模块的设计与实

现进行介绍，并通过界面截图对本系统的操作流程以及运行状态进行展示和说明。然后进行系统测试设计，包括功能测试和性能测试，并对系统测试结果进行分析。

第六章，实验与评估。首先介绍了本文的三个研究问题，其次给出了实验对象，再次详细介绍了针对研究问题所设计的三组实验，最后对实验结果进行研究分析并得出结论。

第七章，总结与展望。首先总结了方法设计、系统实现、实验验证与论文完成期间所做的工作，然后对论文方法的未来方向作了进一步展望。

第二章　相关概念与技术

本章主要介绍与本文方法设计和系统实现相关的概念与技术，包括变异测试、机器学习算法以及遗传算法。介绍变异测试时，首先简要介绍变异测试的基础知识，然后介绍变异选择技术的流程。介绍机器学习算法时，首先介绍提取特征的工具，然后详细介绍本文涉及的机器学习模型。介绍遗传算法时，首先对遗传算法的概念进行简要介绍，然后对遗传算法的流程进行描述。

2.1　变异测试

2.1.1　变异测试概述

程序变异是指自动改变程序语法以产生语义程序变体，即产生人为缺陷的过程 [3]。包含语义缺陷的程序称作变异体。变异测试是指使用程序变异量化测试用例集强度来支持测试的过程。在测试环境中，变异体构成了测试过程的目标。因此，能够将变异程序的行为与原始程序的行为区分开的测试用例可以达到测试目的。当测试用例可以区分变异体的行为时（与原始程序的行为不同），代表变异体“被杀死”或“被检测到”。在另一种情况下，代表变异体是“存活的”。

表 2-1: 典型变异算子

类型	变异算子	描述
运算符变异	MATH	对数值运算的二元算术运算符进行替换
	CONDITIONALS_BOUNDARY	对条件关系运算符”<”、”<=”、”>”、”>=” 进行替换
	NEGATE_CONDITIONALS	将程序中的条件运算符替换为相反的条件运算符
	INCREMENTS	对自增运算符”++” 或自减运算符”–” 进行替换
数值变异	INVERT_NEGS	对程序中整数类型、浮点数类型的变量取相反数
方法返回值变异	RETURN_VALS	对程序中方法的返回值进行修改
	VOID_METHOD_CALLS	删除程序中返回值类型为 void 的方法

程序变异需要在变异算子的指导下完成，变异算子用于定义如何向程序中引入语法更改。例如，运算符变异会更改算术运算符，例如，将“+”替换为“-”。表2-1给出了典型的变异算子，分别是运算符变异、数值变异以及方法返回值变异 [20]。本文使用上述变异算子对实验对象执行变异操作。

$$MS(MUT,TS)=\frac{|MUT_{killed}|}{|MUT|-|MUT_{eq}|} \tag{2-1}$$

变异测试基于选择的一组变异算子，生成一组变异体，用于执行分析。变异测试的测试目标是杀死所有的变异体。请注意，尽管源代码已更改，但可能会有一些语义与源程序相同的变异体。也就是说，测试用例集无法杀死此类变异体，因为它们的输出始终与源程序的输出相同。这些变异体称为等价变异体，它们对评估任何测试用例集没有帮助。除等价变异体之外的所有其余变异体都称为非等价变异体。在变异测试中，将被测试用例集杀死的非等价变异体与全部非等价变异体的比率定义为“变异得分”。变异得分 Mutation Score (MS) 是衡量测试用例集缺陷检测能力的重要指标 [5, 21]。公式2-1给出了变异得分的计算方法，其中 TS 表示测试用例集，$|MUT_{killed}|$ 表示被 TS 杀死的变异体数目，$|MUT|$ 表示全部的变异体数目，$|MUT_{eq}|$ 表示等价变异体数目。$MS(MUT,TS)$ 的值介于 0 与 1 之间，数值越高，表明被测试用例集杀死的变异体越多，测试用例集的缺陷检测能力越强，反之则越低。当 $MS(MUT,TS)$ 的值为 0 时，表明测试用例集没有杀死任何一个变异体；当 $MS(MUT,TS)$ 的值为 1 时，表明测试用例集杀死了所有的非等价变异体。

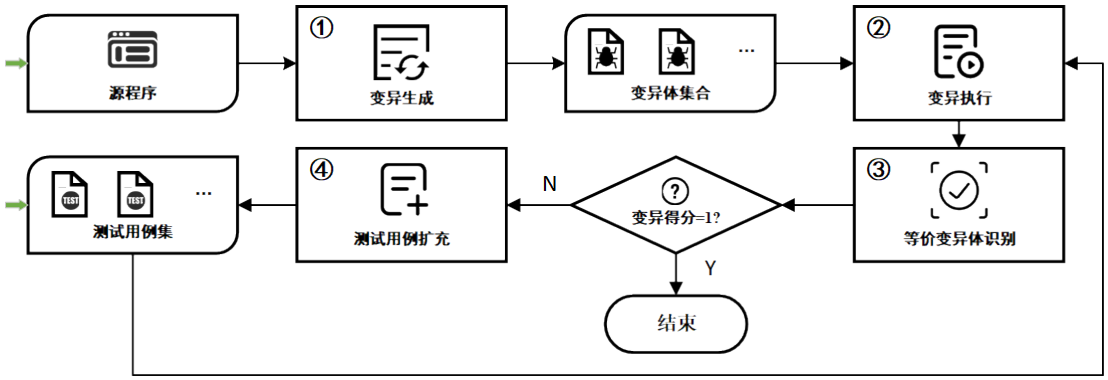


图 2-1: 变异测试基本流程图

为了更加直观地理解变异测试，图2-1给出了的变异测试基本流程图。给定源程序 PG 和测试用例集 TS，依次执行下述步骤。

步骤 1：变异生成。根据给定的变异算子生成变异体集合 MUT；

步骤 2：变异执行。在 MUT 中执行 TS 中的所有测试用例；

步骤 3：等价变异体识别。首先，根据步骤 2 的测试结果识别等价变异体，如果 MUT 中变异体的测试结果与源程序的测试结果相同，则视为该变异体为等价变异体；其次，将所有等价变异体从 MUT 中删除；然后，根据公式2-1计算变异得分，若变异得分不等于 1，说明测试用例设计不充分，执行步骤 4；若变异得分等于 1，说明测试用例已符合要求，变异测试结束；

步骤 4：测试用例扩充。首先，针对 MUT 进行测试用例扩充，直到 MUT 中的所有变异体都被杀死为止，输出扩充后的测试用例集；然后返回步骤 2 进行变异执行并计算变异得分，直至变异得分满足条件。

2.1.2 变异选择技术

变异测试具有较强的故障检测能力 [17]，可以产生较好的测试效果 [22]。然而，由于程序可能具有大量的变异体，因此即使是小型程序，在变异测试中对所有生成的变异体执行测试用例集通常也十分昂贵。为了降低这种成本，研究人员提出了选择性变异测试，以全部变异体的一个子集作为整个集合的代表，以保证该子集具有与整个集合相似的评估测试用例集的能力 [23]。研究人员通常采用变异体集合的质量得分 Quality Score (QS) 衡量一个变异体子集在多大程度上代表了整个变异体集 [1]。$QS(MUT_{sub})$ 是指在约简后的变异体子集 MUT_{sub} 上扩充的测试用例集 TS_{sub}(其中 TS_{sub} 可以尽可能杀死 MUT_{sub} 中全部变异体) 在全部变异体集合 MUT 上的变异得分。公式2-2给出了变异体子集的质量得分的计算方法, 其中 $\left|MUT_{killed}^{TS_{sub}}\right|$ 表示约简后的变异体子集 MUT_{sub} 的测试用例集 TS_{sub} 杀死 MUT 中的变异体数目，$|MUT|$ 表示 MUT 中所有的变异体数目。子集具有的质量得分越高，子集越具有代表性。下文将详细介绍已有的选择性变异测试方法：随机选择法 [1] 和变异算子选择法 [23]。

$$QS(MUT_{sub}) = \frac{\left|MUT_{killed}^{TS_{sub}}\right|}{|MUT|} \tag{2-2}$$

（1）随机选择法

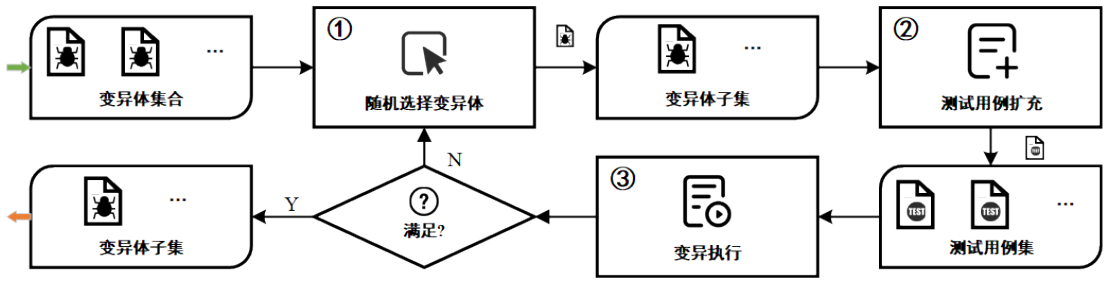


图 2-2: 随机变异选择方法流程图

随机选择方法尝试从生成的全部变异体中随机选择出部分变异体 [2]。图2-2介绍了随机选择方法的完整流程。方法的输入是全部变异体集合 MUT 和预期的变异得分 $QS_{expected}$，输出是变异体子集 MUT_{sub}。下面对方法流程进行详细描述。

步骤 1：随机选择变异体。在 MUT 中随机选择一个变异体 m，添加 m 到 MUT_{sub} 中；

步骤 2：测试用例扩充。针对 MUT_{sub} 进行测试用例扩充，一次包含一个测试用例，直到 MUT_{sub} 中的所有变异体都被杀死为止，输出扩充结果 TS_{sub}；

步骤 3：变异执行。使用 TS_{sub} 对 MUT 执行测试，并计算 TS_{sub} 的变异得分 $MS(MUT, TS_{sub})$（即变异子集的质量得分 $QS(MUT_{sub})$）；如果 $QS(MUT_{sub}) < QS_{expected}(MUT_{sub})$，则返回步骤 1 继续执行变异体选择；否则，完成了变异体子集的构建工作，输出可以达到指定的质量得分的变异体子集 MUT_{sub}。

（2）变异算子选择法

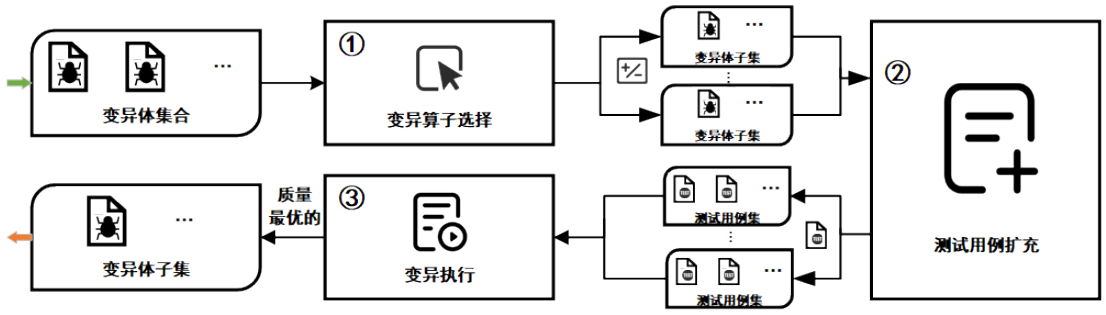


图 2-3: 变异算子选择方法流程图

变异算子选择法通过减少变异算子的规模来约简变异体集合 [2]。图2-3介绍了变异算子选择方法的流程。方法的输入是全部变异体集合 MUT 和全部变异算子，输出是质量最优的变异体子集 MUT_{sub}^{best}。下面对方法流程进行描述。

步骤 1：变异算子选择。依据变异算子类别对 MUT 中的变异体进行分类，并构建各自对应的变异体子集 MUT^i_{sub}（例如，MUT 中共存在 n 个变异算子，那么由第 1 个变异算子生成的变异体构成变异体子集 MUT^1_{sub}）；

步骤 2：测试用例扩充。对于每个 MUT^i_{sub} 进行测试用例扩充，一次包含一个测试用例，直到 MUT^i_{sub} 中所有变异体都被杀死为止，输出扩充结果 TS^i_{sub}；

步骤 3：变异执行。针对每个 MUT^i_{sub}，使用由 MUT^i_{sub} 扩充的测试用例集 TS^i_{sub} 对全部变异体 MUT 执行测试，并计算 MUT^i_{sub} 的质量得分 $QS(MUT^i_{sub})$；比较由 n 个变异体子集的 $QS(MUT_{sub})$，选择并输出质量得分最高的变异体子集 MUT^{best}_{sub}。

2.2 机器学习算法

本文的核心方法之一就是基于机器学习的变异体集合质量预测方法，该方法通过特征提取工具提取源程序、测试用例集和变异体的顽固特性和充分特征，采用机器学习模型预测变异体集合的质量得分，避免动态执行变异测试，以节省昂贵的计算开销。接下来就针对本文使用到的特征提取工具和机器学习模型进行详细介绍。

2.2.1 特征提取

基于机器学习的变异体集合质量预测方法通过提取源程序、测试用例集和变异体集合的顽固特征和充分特征预测变异体集合的质量得分。在本节将介绍特征提取所依赖的工具：Pitest[①]用于变异体生成，生成的变异体集合用于充分特征提取；Cobertura[②]用于顽固特征中的可达度特征提取；JHawk[③]用于顽固特征中的复杂度特征提取。

（1）Pitest

Pitest 是一款面向 Java 语言的变异测试工具，由 Henry Coles 等开发人员负责开发和维护。它非常流行并且被广泛使用。总而言之，Pitest 具备以下特点：

* Pitest 已完全集成到各种构建工具（Maven、Gradle、Ant），IDE（Eclipse、IntelliJ）和静态代码分析工具（SonarQube）中。要将 Pitest 与 Ant 或 Maven

[①]Pitest. http://pitest.org/
[②]Cobertura. https://github.com/cobertura/cobertura
[③]JHawk. http://www.virtualmachinery.com/jhawkprod.htm

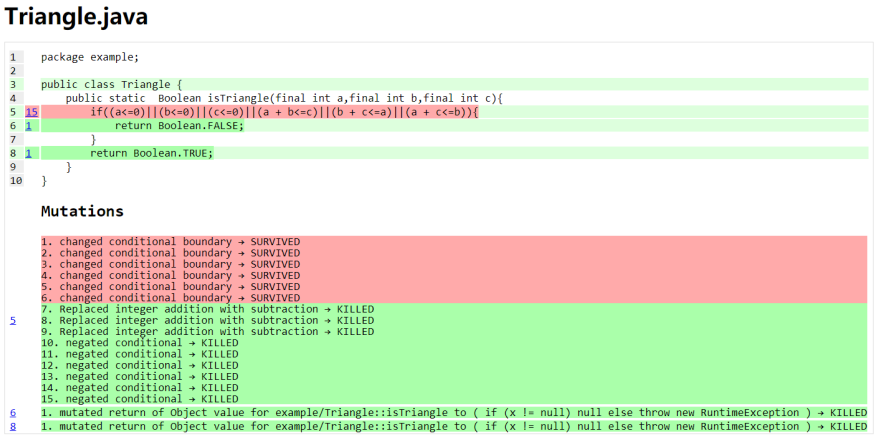


图 2-4: Pitest 的输出示例图

一起使用，测试人员需要向其构建文件中添加一个任务（或插件），以便配置 Pitest 的行为。Pitest 的配置非常简单，并且可以使测试人员指定要测试的类。测试人员还可以配置测试报告的输出目录和所使用的变异算子。

* Pitest 通过字节码操作生成变异体。直接采用字节码操作可省去编译程序的环节，从而降低变异生成成本，因此 Pitest 具备显著的性能优势。除此之外，Pitest 并不会将变异体保存至磁盘中，而是将其字节码保留至内存中，从而省去了输入输出操作，故内存开销较低。同时，为了降低内存开销，该工具每次仅在内存中保留一个变异体。
* Pitest 清楚地报告测试执行情况。Pitest 生成的 HTML 报告提供行覆盖率和变异得分并使用不同的颜色标注程序中被测试覆盖的代码行和未被测试覆盖的代码行。图2-4给出了示例程序 Triangle 的测试报告，此程序的功能是判别三条边是否可以构成有效三角形。如图所示 Triangle 由 10 行代码构成，其中浅绿色覆盖的行代表该代码行被测试覆盖，深绿色覆盖的行代表由该代码行生成的变异体被测试杀死，深粉色覆盖的行代表由该代码行生成的部分变异体未被测试杀死，目前仍处于存活状态。变异体的详细信息在 Mutations 中均有说明，例如第 5 行代码一共生成了 15 个变异体，其中有 6 个变异体处于存活状态，有 9 个已被测试杀死。此报告可以帮助测试人员开发完备的测试用例集。

（2）Cobertura

Cobertura 是一款面向 Java 语言的代码覆盖率报告工具，由 Jon Chambers 等开发人员负责开发和维护。它非常流行并且已被广泛使用。总而言之，Cobertura 具有以下主要特点：

* Cobertura 已完全集成到各种构建工具（Maven、Gradle、Ant）和 IDE（Eclipse、IntelliJ）中。同时，也支持在命令行中直接使用。要将 Cobertura 与 Ant 或 Maven 一起使用，测试人员需要向其构建文件中添加一个任务（或插件），以便配置 Cobertura 的行为。Cobertura 的配置非常简单，并且可以限于测试人员过滤特定的路径与类别。测试人员还可以配置测试报告的类别。
* Cobertura 清楚地报告测试执行情况、分支覆盖率、语句覆盖率以及程序的复杂度。Cobertura 标记在执行测试用例集时被测试用例覆盖的行，未被测试用例覆盖到的行，以及每一行被测试用例执行的次数并生成一个可视化的测试报告。通过查看测试报告，方便测试人员对测试用例进行扩充以及发现程序中隐藏的缺陷。

（3）JHawk

JHawk 是一个面向 Java 的静态代码分析工具。它已被大量用于 Java 度量的学术研究中 [24–26]。总而言之，JHawk 具有以下主要特点：

* JHawk 已完全集成到 Eclipse IDE 中。同时，也支持在命令行中直接使用。JHawk 的配置非常简单，并且可以限于测试人员配置代码度量级别、测试报告的类别以及测试报告的输出目录。
* JHawk 清楚地报告程序包级别、类级别、方法级别的代码度量指标。它是独立的代码分析器，分析项目的源代码并计算诸如各种级别的数量，复杂性，嵌套层数等多方面的代码度量指标，并导出 HTML，CSV 和 XML 多种格式的分析报告，帮助开发人员评估与确保代码的质量。

2.2.2 机器学习模型

机器学习是从数据中检测富含知识的模式并在没有任何明确指导的情况下应用这些模式进行再学习的过程 [27]。机器学习可以设计出复杂的模型以进行预测，并已广泛用于解决软件工程问题 [28, 29]。本文采用常用的回归方法（随机森林回归、岭回归、支持向量机回归和最近邻回归方法）构建变异体集合质量预测模型。

（1）随机森林回归

随机森林回归方法是将大量回归树结合在一起的集成学习方法 [30]。回归树表示一组条件或限制，这些条件或限制是按层次结构组织的，并从树的根到叶依次应用。随机森林从许多引导样本开始，这些样本是从原始训练数据集中

随机抽取而来的, 回归树适用于每个引导程序样本。对于每棵树的每个节点，将从集合中随机选择的一小组输入变量用于二进制分区。回归树拆分标准基于选择具有最小平方误差的输入变量，平方误差表示为 $\sum_{x_i \in R_m}(y_i - f(x_i))^2$，其中 $f(x_i)$ 代表每个划分单元的预测值，即该单元内每个样本点值的均值。观测值的预测值是通过对所有树木进行平均计算得出的。

（2）岭回归

岭回归 [31] 是一种改进的最小二乘法，它摒弃了最小二乘法的无偏性，以损失部分信息为代价，查找效果稍差但回归系数更符合实际情况的模型方程。岭回归用来估计 $\widehat{\beta^R}$ 以最小化 $\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$。对于 $\lambda \geq 0$ 是调整参数。$\lambda > 0$ 和 $\lambda = 0$ 之间的区别在于，当 $\lambda = 0$ 时，调整参数消失，并与最小二乘法具有相同的方程式和模型。重要的是，该算法能够通过最小二乘模型（$\lambda = 0$）的交叉验证来检查具有最佳 λ 的岭回归模型，以查看哪个模型最适合数据并且将具有最佳性能。岭回归和最小二乘的总体目标是相同的，以找到最小化 RSS 并最适合数据的系数估计。

（3）支持向量机回归

支持向量机（SVM）是一种二分类模型，其基本模型定义为特征空间中间隔最大的线性分类器 [32]。支持向量机回归（SVR）[33] 是 SVM 在回归问题上的一种应用。给定一组 n 维的训练数据 $[x_i, y_i], i = 1, 2, \cdots n, x \epsilon R^k, y \epsilon R$，SVR 通常通过非线性将输入数据的原始空间映射到更高维的特征空间高斯核函数 $\Phi(x_i, x_j) = exp(-\left\|x_i, x_j\right\|^2 / 2\delta^2)$。在特征空间中，问题就变成了适合数据的最佳线性曲面 $f(x) = \mu + \omega^T \Phi(x)$ 的构造，其中 μ 为基数或偏差，而 $\omega \in R^k$ 为权重。因此，最平坦的函数 $f(x)$ 是通过将向量范数 ω^2 最小化而寻求的，它受到一组约束：每个训练数据的预测值的误差最多应等于 ϵ。为了允许离群值，使用所谓的 ϵ 敏感损失函数对预测误差大于 ϵ 的数据进行惩罚。然后，通过对每个约束使用拉格朗日乘数，将约束的优化问题重新构造为对偶问题形式。

（4）最近邻回归

最近邻回归 [34] 是基于实例的惰性学习算法，即寻找一个样本的 K 个最近邻居，并将 K 个样本属性的平均值作为本样本的属性值。具体来说，假设一个最接近的邻居数 K 和一个预测点 x_o 的值。最近邻回归算法标识最接近预测点 x_o 的训练观测值 N_o，使用 N_o 中所有样本属性 y 的平均值估算 $f(x_o)$，即 $\widehat{f}(x_o) = \frac{1}{K}\sum_{x_i \epsilon N_o} y_i$。$K$ 值的选取决定了最近邻回归算法的最终结果。较小的 K 和较少的相邻数据将限制回归变量的区域，并导致较小的偏差但变化较大。另

一方面，较大的 K 包含更多的相邻数据，从而允许回归变量在较宽的区域内变化，从而导致较大的偏差和较小的变化。因此，在实际应用中，通常先取一个较小的 K 值，然后采用交叉验证的方法来选择 K 的最优值 [32]。

2.3 遗传算法

2.3.1 遗传算法概述

遗传算法通过选择、交叉和变异等操作，为搜索和优化问题提供了高质量的解决方案。遗传算法旨在有效解决大空间、多峰值、非线性、全局化等高复杂度问题 [35]。与其他传统搜索或优化技术 (例如爬山算法) 仅依靠本地信息来决定下一步的最佳移动方向相比, 遗传算法使用全局信息, 执行并行搜索并且不需要本地梯度信息, 这使它能够找到全局最优或接近全局最优的解决方案。以下列出了遗传算法与其他常规方法不同的四点主要特征：

（1）直接操作编码

遗传算法在字符串级别上操纵决策或控制变量表示，以利用高性能字符串之间的相似性。其他方法通常直接处理函数及其控制变量。

遗传算法处理的是有限长度的参数，这些参数使用有限的字母编码，而不是直接自己操纵参数。这意味着搜索既不受所研究功能的连续性限制，也不受衍生功能的存在限制。此外，通过探索编码的相似性，与许多其他过程相比，遗传算法可以有效地处理更大范围的功能。

（2）从种群中搜索，而不是单点搜索

通过从总体中进行搜索，遗传算法可以通过保持一组适应性强的采样点，降低达到假峰的可能性。

搜索从许多点开始，而不是仅从一个点开始。这种并行性意味着搜索不会陷入局部极大值中。

（3）具有普适性

通过利用任何搜索问题中可用的信息，遗传算法在基础编码中处理相似性，并根据结构在当前环境中的生存能力对信息进行排名。通过利用这些广泛可用的信息，遗传算法可以应用于几乎任何问题。

（4）采用概率性规则进行搜索

遗传算法使用随机运算符而不是确定性规则进行搜索。遗传算法使用随机选择来指导高度并发的搜索。

2.3.2 遗传算法流程

本节主要介绍遗传算法的具体流程。首先对生物进化概念与遗传算法进行类比，然后介绍遗传算法中的三种核心操作，最后描述遗传算法的整体流程。

为了更好地理解遗传算法，表2-2给出了生物进化概念与遗传算法的对应关系。其中个体代表所求解问题的可行解；种群代表一组可行解，即多个个体所构成的集合；染色体为对应可行解的编码，一般使用二进制形式表示；基因代表编码的元素，例如 0 或 1，即可视为染色体上的基因；环境表示适应度函数，也是遗传算法中最为核心的概念。

表 2-2: 生物进化概念与遗传算法的对应关系表

生物进化	遗传算法
个体	问题的可行解
种群	一组可行解
染色体	对应可行解的编码
基因	编码的元素
环境	适应度函数

遗传算法对其种群应用以下三种操作：

（1）选择

选择以一定的概率从种群中选择若干个个体以产生新的种群，从而开始新的一代。选择操作是建立在种群中个体的适应度评估基础上的，即根据个体的适应度从种群中选择个体进行交配。适应性是指个体在环境中生存和繁殖的特征和能力。通过适当的评价来确定当代染色体的适应值。因此，适应度值被用来从群体中为下一代选择一组更好的染色体 [36, 37]。

轮盘赌选择法是最简单、最常用的选择方法，在该方法中，个体的累积概率和其适应度值成比例，适应度越大，被选中概率越大。轮盘赌选择个体的流程如下所示：

步骤 1：计算适应度比例，即每个个体的选择概率 $p(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$，其中 x_i 代表个体，$f(x_i)$ 代表个体的适应度值。

步骤 2：计算每个个体的累积概率 $q(x_i) = \sum_{j=1}^{i} f(x_j)$。

步骤 3：随机生成一个元素取值范围在 0 和 1 之间的数组 m，并将其按从小到大的方式进行排序。若累积概率 $q(x_i)$ 大于数组中的元素 $m[i]$，则个体 x_i 被选中，若小于 $m[i]$，则比较下一个个体 x_{i+1} 直至选出符合条件的个体为止。

（2）交叉

在选择过程之后，对从种群中选择的染色体进行交叉操作。交叉涉及两个个体之间交换字符串中的位或基因序列 [38]。这种交换过程在不同的父本个体中进行，每次都重复进行，直到下一代有了最佳个体。

一般设计交叉算子的思路 [39] 如下：给定两个父解 P_1 和 P_2，首先从 [0,1] 中随机选择一个值 α，并将 P_1 分为两个测试集 P_1^{α} 和 $P_1^{1-\alpha}$，其中分别包含前 $\alpha|P_1|$ 个和后 $(1-\alpha)|P_1|$ 个测试用例。类似地，P_2 被分为 P_2^{α} 和 $P_2^{1-\alpha}$。然后，第一子代 O_1 由 P_1^{α} 和 $P_2^{1-\alpha}$ 结合产生，第二子代由 $P_1^{1-\alpha}$ 和 P_2^{α} 结合产生。

（3）变异

在交叉过程之后，将变异操作应用于随机选择的种群子集。变异通过导致染色体的改变以引入优良性状。变异的主要目的是带来种群的多样性 [36]。

遗传变异操作是以一定的概率随机改变染色体中一个或多个基因值，由于染色体已进行二进制编码，那么此时改变基因的值就是对二进制进行取反操作。一般的变异概率为 [0.0001, 0.1] 之间，当变异概率较小时，解的稳定性较高，但是容易陷入局部最优解，并且难以跳出局部最优解的区间；如果变异概率较大，可以使得解空间具有多样性，从局部最优解跳出来，最终找到全局最优解，所以选取变异概率是决定遗传算法性能优劣的关键因素。

综合上述三种操作，遗传算法的具体操作步骤如下：

步骤 1：确定适应度函数的取值范围，确立精度及染色体编码长度。

步骤 2：进行染色体编码并确立种群数量、交叉和变异概率。

步骤 3：初始化种群：随机生成初代种群。

步骤 4：利用适应度函数对种群进行评价，判断是否满足停止条件。如果是，它将停止并输出最优解；否则，它将继续运行。

步骤 5：对当前种群依次进行选择、交叉和变异操作，得到下一代种群，返回步骤 4。

2.4 本章小结

本章对本文方法设计和系统实现相关的概念与技术进行了介绍。介绍变异测试相关技术时，首先介绍变异测试的基础知识，说明了程序变异的定义并描述了典型的变异算子和变异测试流程，然后介绍变异选择技术，给出了变异体集合质量得分的定义并描述了变异选择方法的基本流程。介绍机器学习算法的相关概念时，主要从特征提取工具和机器学习模型两个方面来介绍机器学习算法的基础知识。介绍遗传算法的相关概念时，主要从基本概念和算法流程两个方面来介绍遗传算法的基础知识。

第三章　变异体集评价与约简方法

变异体集合约简技术是降低变异测试代价的有效手段，此类技术的核心是确保约简后变异体子集的测试效果。高质量的变异体子集可以在约简变异体集合规模的同时，保证变异体子集的测试效果，提升测试用例集的完备性。本章首先讨论评价变异体集合质量所需的特征，即高质量变异体子集所具备的特征；然后介绍构建高质量变异体子集的方法，即基于遗传算法的变异体集合约简方法。

3.1　高质量变异体集合具备的特征

在变异测试中，顽固变异体指的是一种很难杀死的变异体 [40]。杀死顽固变异体的测试数据将是重要的，因为它可以检测到其他测试数据难以检测到的故障。具体地说，如果变异体集合中存在顽固变异体，则说明测试用例集中的测试用例难以检测出由顽固变异体注入的缺陷，从而反映出测试用例集完备性低。基于该变异体集合，可以针对未检测到的缺陷进行测试用例集扩充，从而提升测试用例集的完备性。故基于变异顽固程度设置度量指标。即如果变异体集合中顽固变异体占比高，本文认为该变异体集合质量高，反之亦然。

除了考虑变异的顽固指标，本文还考虑了变异的充分程度。变异测试是一种用于评估测试用例集测试充分性的测试技术 [1]。如果变异体集合中的变异体类别多样化且在源程序中分布广泛，那么说明故障注入就越充分，则能越有效地评估测试用例集的故障检测能力，进而反映测试用例集是否完备。即如果变异体集合的变异充分性越高，本文认为变异体集合质量越高，反之亦然。

综上所述，高质量的变异体集合所具备高强度的变异顽固程度和变异充分程度。因此，本文从变异顽固程度和变异充分程度两个角度来评估变异体集合质量的优劣。

3.1.1 变异顽固程度

本文采用以下指标评估变异体集合的顽固程度。

复杂度：程序越复杂，被感染的程序状态可能不会传播，表明变异体被测试用例集杀死的可能性越小。因此，可以通过变异体集合的复杂度反应变异体集合的顽固程度。复杂度越高，变异体集合的顽固程度越高，反之亦然。

本文使用软件指标表征变异体集合的复杂度。软件指标用来度量可量化的软件系统特性。软件指标在许多应用中都很重要，包括测量软件复杂性 [41]，评估软件可维护性 [42]，测量软件质量 [43] 等。

本文使用现有研究 [44–46] 中广泛使用的 JHawk 工具[①]来提取方法，类和包级别软件指标作为程序的复杂度特征。如表 3-2所示，总共使用了 89 个软件指标。所有度量均为标量。有关指标的完整定义，请见第四章。本文使用变异体集合中每项指标的平均值来表征此变异体集合的复杂度特征。公式 3-1给出变异体集合复杂度的计算公式。其中，MUT 代表变异体集合，mut_i 代表变异体集合中第 i 个变异体，$ComplexityFeature(mut_i)$ 代表 mut_i 所对应原始程序的复杂度特征。

$$ComplexityFeature(MUT) = \frac{\sum_i^{|MUT|} ComplexityFeature(mut_i)}{|MUT|} \tag{3-1}$$

可达性程度：Visser 建议，变异体的可达性可用于识别难以杀死的变异体 [47]。如果测试未能到达变异语句，那么故障难以被测试检测出来，对应的顽固度较高。因此，可以通过变异体集合可达性程度反映变异体集合的顽固程度。可达性程度越高，变异体集合的顽固程度越低，反之亦然。

$$numTestExecuted(MUT) = \frac{\sum_i^{|MUT|} numTestExecuted(mut_i)}{|MUT|} \tag{3-2}$$

本文使用 numTestExecuted、numTestCovered、numMutantAssertion 和 numClassAssertion 表征变异体集合的可达性程度。numTestExecuted，该值是指整个测试用例集执行了变异体集合所对应原始程序中的变异语句多少次，变异语句被执行次数越多，说明该语句的可达性程度有可能越高。numTestCovered，该值是指整个测试用例集中有多少测试覆盖了变异体集合所对应原始程序中

[①] JHawk. http://www.virtualmachinery.com/jhawkprod.htm

的变异语句，变异语句被覆盖次数越多，说明该语句的可达性程度有可能越高。numMutantAssertion，该值是指变异体集合所对应测试方法的断言数目。numClassAssertion，该值是指变异体集合所对应测试类的测试断言数目。断言用于观察测试的执行结果，变异语句的断言覆盖率越高，说明该语句的可达性程度有可能越高。

$$numTestCovered(MUT) = \frac{\sum_{i}^{|MUT|} numTestCovered(mut_i)}{|MUT|} \tag{3-3}$$

$$numMutantAssertion(MUT) = \frac{\sum_{i}^{|MUT|} numMutantAssertion(mut_i)}{|MUT|} \tag{3-4}$$

$$numClassAssertion(MUT) = \frac{\sum_{i}^{|MUT|} numClassAssertion(mut_i)}{|MUT|} \tag{3-5}$$

本文使用变异体集合中每项可达度特征的平均值来表征此变异体集合的可达度特征。公式3-2 - 3-5给出变异体集合可达度的计算公式。其中，MUT 代表变异体集合，mut_i 代表变异体集合中第 i 个变异体，$numTestExecuted(mut_i)$ 代表 mut_i 所对应的原始程序中变异语句的测试执行数目，$numTestCovered(mut_i)$ 代表 mut_i 所对应的原始程序中变异语句的测试覆盖数目，$numMutantAssertion(mut_i)$ 代表覆盖 mut_i 所对应的原始程序中变异语句的测试方法中的断言数目，$numClassAssertion(mut_i)$ 代表 mut_i 所对应的测试类中的测试断言数目。

3.1.2 变异充分程度

本文采用以下指标评估变异体集合的充分程度。

变异算子充分性：变异的有效性直接取决于所使用的变异算子 [48]，如果变异体集合中涉及的变异算子种类越充分，本文认为该集合的故障注入越充分。因此，可以通过变异算子的充分性反应变异体集合的充分程度。变异体集合的变异算子充分性越大，本文认为该集合的充分程度越高。

$$opAdequacy(MUT) = \frac{OpTypeNum(MUT)}{|OpType|} \tag{3-6}$$

公式3-6给出变异体集合的变异算子充分性的计算公式。其中 MUT 代表变异体集合，$|OpType|$ 代表源程序 PG 中的变异算子类别总数，$OpTypeNum(MUT)$ 代表 MUT 中涉及的变异算子类别总数。

变异位置充分性: 本文认为如果变异语句在源程序中越分散，则故障注入的分布性越广泛。即变异体集合的变异位置充分性越大，该集合的充分程度越高。本文分别使用变异体集合涉及的类总数、方法总数、方法类别数来表征变异位置的充分性。

$$posMdTypeAdequacy(MUT) = \frac{MdTypeNum(MUT)}{|MdType|} \tag{3-7}$$

$$posMdAdequacy(MUT) = \frac{MdNum(MUT)}{|Md|} \tag{3-8}$$

$$posClassAdequacy(MUT) = \frac{ClassNum(MUT)}{|Class|} \tag{3-9}$$

公式 3-7 - 3-9给出变异体集合的变异算子充分性的计算公式。其中 MUT 代表变异体集合，$|MdType|$ 代表源程序 PG 中的方法类别总数，$MdTypeNum(MUT)$ 代表 MUT 中涉及的方法类别总数；$|Md|$ 代表 PG 中的方法总数，$MdNum(MUT)$ 代表 MUT 中涉及的方法总数，$|Class|$ 代表 PG 中的类总数，$ClassNum(MUT)$ 代表 MUT 中涉及的类总数。

表3-1对以上特征进行了总结。预测特征分为两类，共 97 种：变异顽固程度和变异充分程度。其中，变异体集合复杂度（详见表3-2）和变异体集合可达程度两类特征用来表征变异体集合的顽固程度；变异算子充分性和变异位置充分性两类特征用来表征变异体集合的充分程度。

表 3-1: 预测特征

类别	名称	描述
变异顽固特征	复杂度特征	表 3-2 的特征
	可达度特征	numTestExecuted: 变异体集合的测试执行数目
		numTestCovered: 变异体集合的测试覆盖数目
		numMutantAssertion: 变异体集合所对应测试方法的断言数目
		numClassAssertion: 变异体集合所对应测试类的测试断言数目
变异充分特征	变异算子充分性	opAdequacy: 变异体集合中变异算子的充分性
	变异位置充分性	posMdTypeAdequacy: 变异位置的方法类别充分性
		posMdAdequacy: 变异位置的方法充分性
		posClassAdequacy: 变异位置的类充分性

表 3-2: 复杂度特征

级别	指标列表
package	Number of Classes,Number of Statements,Average Cyclomatic Complexity,Halstead Cumulative Bugs,Halstead Effort,Halstead Cumulative Length,Halstead Cumulative Volume,Maintainability Index,Cumulative Number of Comment Lines,Lines Of Code,Review Factor,Total Complexity Volume,Cumulative Number of Comments,Instability,Distance,FanIn,Number of Methods,Maintainability Index (NC),Abstractness,Max Complexity,FanOut
class	LCOM,Average Cyclomatic Complexity ,Number of Statements ,Halstead Cumulative Bugs ,Halstead Effort,UnWeighted Class Size ,Instance Variables ,Imported Packages,Response For class,CBO,Maintainability Index,Cumulative Number of Comment Lines,Lines Of Code,Review Factor,Fan In ,DIT,Maintainability Index (NC),Specialization ratio,Reuse Ratio,COH,Local Method Calls,LCOM2,Max Complexity,Halstead Cumulative Volume,Hierarchy Method Calls,Number of Querie,Fan Out,SIXs,External Method Calls,Superclasses,Total,Complexity,Subclasses,Message Passing Coupling ,Number of Commands,Interfaces,Cumulative Number of Comments,Halstead Cumulative Length,Modifiers,Number of Methods
method	Cyclomatic Complexity ,Number of Comment Lines ,Number of Statements ,Halstead Length ,Halstead Vocabulary ,Halstead Effort ,Halstead Bugs,Classes Referenced,External Methods Called ,Local Methods Called ,Lines Of Code ,Number of Comments ,Arguments ,Modifiers,Halstead Difficulty ,Variable Declarations,Exceptions Thrown,Exceptions Referenced,Number of casts,Total Depth of Nesting,Halstead Volume,Number of Operands,Variable References,Number of Operators,Max. depth of nesting,Number of Expressions,Number of Loops

3.2 基于遗传算法的变异体集合约简方法

为解决现有变异选择方法计算开销昂贵的问题，本文提出了一种基于遗传算法的变异体集合约简方法。该方法的流程如图3-1所示，分为两部分。第一部分是变异体集合质量预测方法，该方法采用机器学习算法从源程序、测试用例集和变异体集合中提取上节所述的特征构建变异体集合质量预测模型，该模型作为遗传算法中的适应度评价模型评估变异体子集质量的优劣, 以节省评估变异体子集的计算开销。第二部分是变异体集合约简方法，该方法接收变异体集合质量预测模型并应用遗传算法搜索质量最优的变异体子集，以提高变异选择的收敛速度，降低变异选择的计算开销。下面将详细介绍这两种方法。

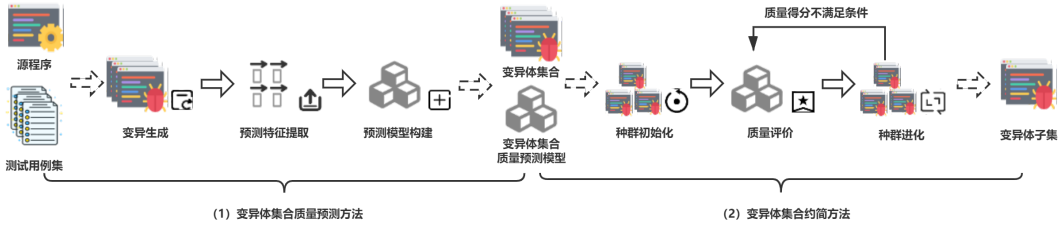


图 3-1: 基于遗传算法的变异体集合约简方法流程图

3.2.1 变异体集合质量预测方法

现有研究通过计算变异体集合质量得分去评估约简后的变异体子集的质量。由于需要通过执行多次变异测试计算变异体子集的质量得分，所以评估过程的计算开销依旧高昂。为缓解此问题，本文提出基于机器学习的变异体集合质量预测方法（MSQP），通过提取源程序、测试用例集和变异体的顽固特征和充分特征，预测变异体集合的质量得分，而无需动态执行变异测试，从而避免高昂的计算开销。

在本节中，将从方法假设和方法概述两个方面介绍 MSQP：

（1）方法假设

在介绍方法框架之前，首先列出方法的假设。方法假设对于使用同一变异体集合所生成的各个最小的测试用例集（即满足其变异得分接近 1 的最小集合），它们的缺陷检测能力是相近的。由于变异体集合是固定的，在为变异体集合生成测试用例集时，每个测试用例旨在于检测出变异体集合的缺陷，故针

对同一变异体集合，多次生成的测试用例集目标相同，缺陷检测能力也是相近的。所以该假设是合理的。

（2）方法概述

本文使用由变异体子集 MUT_{sub} 扩充后的测试用例集 TS_{sub} 在全部变异体集合 MUT 上的变异得分 $MS(MUT, TS_{sub})$（即变异体子集的质量得分 $QS(MUT_{sub})$）作为预测模型的目标值。由于目标值为数值，因此本文将变异体集合质量得分预测问题简化为有监督的回归问题。

图 3-1(1) 介绍了 MSQP 的完整流程。方法的输入是源程序 PG 和测试用例集 TS，输出是全部变异体集合 MUT 和变异体集合质量预测模型 $MSQPModel$。下面对该方法流程进行详细描述。

步骤 1：变异生成。对源程序 PG 进行变异，生成全部变异体集合 MUT；将 PG、MUT 和测试用例集 TS 输入到特征提取阶段。

步骤 2：预测特征提取。从 MUT 中随机选择 n 个变异体子集 MUT_{sub}，并从 PG、TS 和 n 个 MUT_{sub} 中提取相关预测特征 $Feature$（即变异顽固特征和变异充分特征）和目标值 $label$（即 $QS(MUT_{sub})$），并生成训练集 $TrainingSet$。

步骤 3：预测模型构建。将训练集输入到机器学习算法中进行调参和训练，生成变异体集合质量预测模型 $MSQPModel$。$MSQPModel$ 可以根据输入的变异体子集 MUT_{sub} 预测该子集的质量得分 $QS(MUT_{sub})$。最终方法输出 MUT 和 $MSQPModel$ 以用于变异体集合约简。

在本文中，采用基于树型回归的随机森林算法作为默认的回归技术，进行预测模型的构建。因为随机森林通过保持基于树的学习器的功能、灵活性和可解释性大大提高了回归模型的鲁棒性 [49]。在训练预测模型时，输入训练集 $TrainingSet = \{(x^1, y^1), (x^2, y^2), \cdots, (x^m, y^m)\}$，其中 m 代表样本大小，$x^i = (x_1^i; x_2^i; \cdots; x_n^i)$ 代表样本的 n 维特征向量（即预测特征，具体细节详见 3.1 节），$y^i \epsilon \mathbb{R}$ 是样本标记（即变异体集合的质量得分，具体公式详见 2.1.2 节）。在测试预测模型时，输入测试集 $TestSet = \{x^1, x^2, \cdots, x^k\}$，输出目标向量 $y = (y^1; y^2; \cdots; y^k)$。

尽管本文使用随机森林作为默认方法，但 MSQP 并非特定于随机森林，在实证研究中（详见第 6 章），本文还研究了其他回归方法的选择，例如岭回归，支持向量机回归和最近邻回归方法。

3.2.2 变异体集合约简方法

使用机器学习算法预测变异体集合的质量得分可以大幅度降低计算开销，但是从大量的变异体子集中选取一个最优的变异体子集的过程依旧十分耗时。举例来说，源程序 *PG* 生成了 *m* 个变异体，若采取一一遍历的方式，则一共需要遍历 $\sum_{k=1}^{m} C_m^k$ 次（其中 *k* 代表变异体子集的大小），这是一个极为耗时的工作。因此需要一种搜索方法，在可接受的时间内寻找最优解。遗传算法是最常用的搜索方法，其选择、交叉、变异等运算都是以一种概率方式进行的，增加了搜索过程的灵活性，而且能以较大概率收敛于最优解，具有较好的全局优化求解能力 [6]。所以本文运用遗传算法搜索一个最优的变异体子集。并将 MSQP 作为适应度函数应用到遗传算法中，以节省整个约简过程的计算开销。

图 3-1(2) 介绍了变异体集合约简方法的完整流程。方法的输入是全部变异体集合 *MUT* 和变异体集合质量预测模型 *MSQPModel*，输出是约简后的变异体子集 MUT_{sub}。下面对该方法进行详细描述。

步骤 1：种群初始化。首先进行个体表示和种群初始化。在本方法中，种群个体代表一个变异体子集 MUT_{sub}，变异体子集编码成二进制位数组的形式，即基因用 0 或 1 表示，在 *n* bit 的二进制位中 0 表示未选择相应变异体，1 表示选择了相应变异体，其中 *n* 代表由全部变异算子生成的变异体总数。在 *MUT* 中随机选择 *m* 个变异体子集构成初始种群。

步骤 2：质量评估。首先配置迭代次数 *r*；然后采用 *MSQPModel* 对当前种群进行适应度质量评价，获取质量评价结果后，保存当前种群中适应度最高的个体至结果集合 *Res* 并判断是否达到给定的迭代次数 *r*。如果达到要求，则输出 *Res* 中适应度最高的个体，即高质量变异体子集 MUT_{sub}，否则执行步骤 3。

步骤 3：种群演化。首先从当前种群中采用轮盘赌方法选择 *x* 对个体，其中每对个体分别为父代和母代，对其依次进行交叉和变异生成新个体；其次对新个体进行适应度计算；再次根据适应度值对新个体和原有种群的个体进行排序，保留适应度值高的前 *m* 个个体组成新一代种群。最后将新一代种群输入至步骤 2 进行迭代，直至满足终止条件。对于上述的交叉操作，其具体步骤是将父代个体和母代个体上的基因进行重新组合分配。对于上述的变异操作，其具体步骤是对种群个体上的基因作变动。即随机选择 1bit, 将 0 和 1 替换，将未选择的相应变异体替换成选择的相应变异体或者将选择的相应变异体替换成未选择的相应变异体。

3.3 本章小结

本章介绍了基于遗传算法的变异体集合约简技术的研究工作。首先从变异顽固程度和变异充分程度两个角度来介绍高质量集合所具备的特征，并使用上述特征评价变异体集合的质量。然后详细介绍构建高质量变异体子集的方法，即基于遗传算法的变异体集合约简方法，为下一章节系统的需求分析与总体设计奠定基础。

第四章　需求分析与概要设计

本章介绍变异体集合约简系统的需求分析与概要设计，首先对系统进行需求分析，即通过对系统进行用例分析，确定系统的功能性需求和非功能性需求；然后根据需求分析的结果，对系统进行总体概要设计，即设计系统的整体架构图、架构视图和持久化存储表。

4.1　需求分析

4.1.1　用例分析

回归测试是指修改了程序后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。回归测试在整个软件测试过程中占有很大的比重，软件开发的各个阶段都可能执行多次回归测试 [50]。在执行回归测试的过程中，软件测试人员需要衡量测试用例集的缺陷检测能力。变异测试可以有效地衡量测试用例集的缺陷检测能力，但是变异操作通常会生成大量的变异体，编译、执行这些变异体会耗费大量的测试时间，导致变异测试效率低下、可用性不高。为了解决上述问题，本文结合上一章节所提的基于遗传算法的变异体集合约简方法设计并实现了一个面向回归测试测试用例评估场景的变异体集合约简系统，使用由程序最初版本的变异体集合构建的变异体集合质量预测模型约简由后续迭代版本产生的变异体集合，即多次回归测试可重复使用一个模型进行约简。根据上述分析，使用本系统可以有效降低变异约简的计算开销，高效地衡量测试用例集的缺陷检测能力。

图4-1展示了系统整体用例图，本系统作为一个变异体集合约简系统，用户一般为软件测试人员，供其在回归测试场景下衡量测试用例集的缺陷检测能力。主要可以分成变异体生成、变异体集合约简和测试报告生成三个用例。在变异体生成阶段，测试人员输入待约简的项目路径，系统对该项目依次进行合规性检测和 Pitest 变异生成，最后向测试人员展示变异体信息。在变异体集合约简阶段，测试人员根据上一阶段提供的变异体信息输入约简范围（若默认则

系统筛选最优约简比例），系统首先根据测试人员上传的项目和 Pitest 生成的变异体集合提取预测特征并构建变异体集合质量预测模型，然后根据测试人员输入的约简比例和质量预测模型进行变异体集合约简，随后测试人员可以在前端查看约简后的变异体信息。在测试报告生成阶段，系统会调用 Pitest 插件在约简后的变异体集合中运行测试用例集，当变异体子集测试完毕后，系统根据测试结果生成变异测试报告。系统会调 Cobertura 插件对项目进行覆盖测试并根据测试结果生成覆盖测试报告。测试报告可供测试人员导出。下面详细描述上述系统整体用例图给出的三个用例。

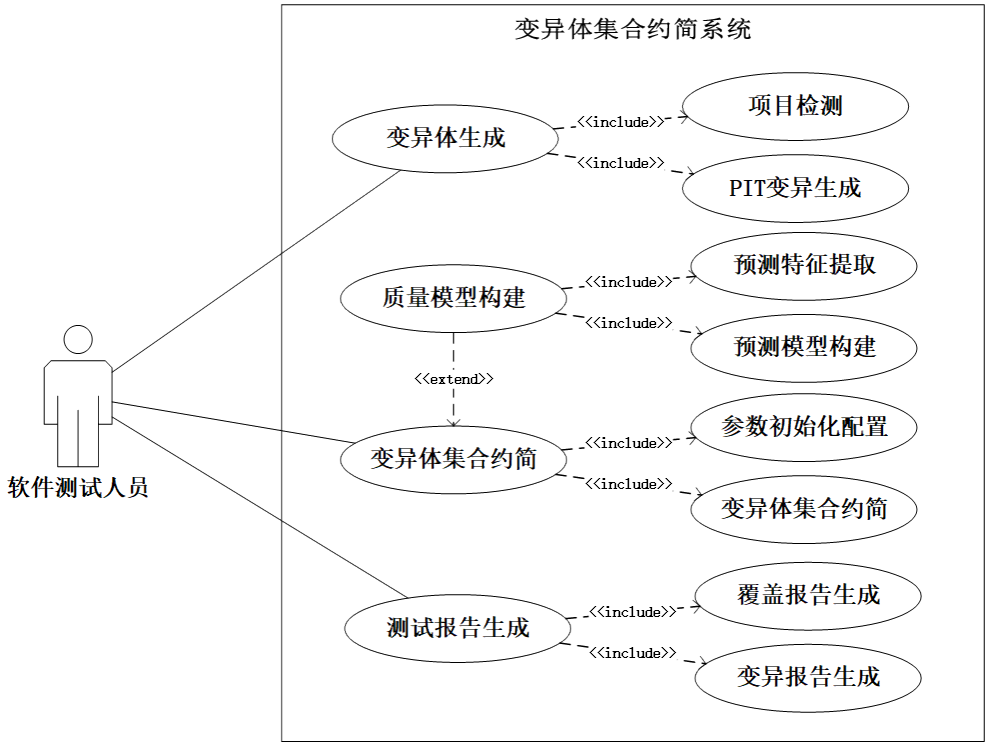


图 4-1: 系统用例构图

表 4-1给出了变异体生成的用例描述。变异体生成主要包括项目检测和变异生成两个子用例。项目检测用于验证上传的项目类型是否正确、项目中的测试用例是否全部通过以及项目中是否存在相关工具的配置信息。变异生成是将生成的变异体用于特征提取以及方便测试人员查看变异信息，确定约简比例。

表 4-1: 变异体生成用例描述

ID	UC1
名称	变异体生成
参与者	软件测试人员
触发条件	软件测试人员切换到上传项目页面
前置条件	软件开发人员已完成项目开发和软件测试人员已完成对应测试用例集的开发
后置条件	无
正常流程	1. 测试人员上传本地的待测项目 2. 测试人员点击 [变异生成] 按钮，系统对该项目依次进行合规性检查和变异生成 3. 变异体生成后，系统获取并展示变异体信息
扩展流程	1a. 上传的项目类型有误 1. 系统提示项目类型有误，请上传正确项目 2a. 项目中测试用例有报错 1. 系统提示测试用例无法全部通过 2b. 项目中插件配置信息有误 1. 项目中不包含 Pitest 插件或者 Cobertura 插件

表 4-2: 变异体集合约简用例描述

ID	UC2
名称	变异体集合约简
参与者	软件测试人员
触发条件	软件测试人员切换到变异体集合约简页面
前置条件	软件测试人员已完成项目检测和变异生成
后置条件	无
正常流程	1. 测试人员确定约简范围 2. 测试人员点击 [变异体集合约简] 按钮 3. 系统生成变异体子集并跳转到变异体展示页面
扩展流程	无

表4-2给出了变异体集合约简的用例描述。变异体集合约简用例分为参数初始化配置和变异体集合约简两个子用例，并扩展了质量模型构建用例，其中质量模型构建用例又包含预测特征提取和预测模型构建两个子用例。变异体生成后，系统在变异体集合约简页面向测试人员展示变异体信息，供测试人员确定变异体集合约简范围（若选择默认，系统筛选最优约简比例），测试人员点

击 [变异体集合约简] 按钮，系统首先根据源程序、测试用例集和变异体集合提取预测特征，然后根据预测特征训练变异体集合质量预测模型，最后将训练完成的预测模型和测试人员选定的约简范围应用到变异体集合约简算法中，生成约简后的变异体子集。随后，系统会自动跳转到变异体子集展示界面。在该界面中，测试人员可以查看变异体子集的信息。

表 4-3: 测试报告生成用例描述

ID	UC3
名称	测试报告生成
参与者	软件测试人员
触发条件	软件测试人员切换到测试报告生成界面
前置条件	软件测试人员已完成变异体集合约简步骤
后置条件	无
正常流程	1. 测试人员选择执行的变异体集合（约简后） 2. 测试人员选择生成测试覆盖报告 3. 测试人员选择生成变异报告 4. 测试人员点击 [开始运行] 按钮 5. 系统开始运行变异测试和覆盖测试 6. 系统运行完毕，展示测试结果
扩展流程	无

表4-3给出了测试报告生成的用例描述。测试报告生成用例分为覆盖报告生成和变异报告生成两个子用例。在测试报告生成阶段，测试人员首先切换到测试报告生成界面，接着选择执行的变异体集合，最后点击 [开始运行] 按钮，系统即可开始运行覆盖测试和变异测试。测试完成后，系统根据测试用例的执行结果生成覆盖测试报告和变异测试报告，显示覆盖信息、变异信息，便于测试人员分析测试用例集的完备性。

4.1.2 功能性需求

根据用例分析，本系统可以分为变异体生成、质量模型构建、变异体集合约简和测试报告生成四个模块。

变异体生成。软件测试人员向系统输入本地 Java 项目的路径，系统能够自行获取项目中类文件、测试文件以及插件配置文件。在变异测试前，系统需要自行验证上传的项目类型是否正确、项目是否包含插件配置信息且初始化项目

并进行覆盖测试检测。若项目检测出现错误，系统应向软件测试人员展示相应的提示信息。在测试用例全部通过后，系统进行变异生成。

质量模型构建。系统能够从测试人员传入的项目和测试用例集以及变异体生成模块生成的变异体集合中提取变异顽固特征和变异充分特征，并构建变异体集合质量预测模型，随后系统将生成的变异体集合质量预测模型传入到变异体集合约简模块以进行变异集合约简。

变异体集合约简。变异体生成后，系统向测试人员展示变异体信息，供测试人员确定变异体集合约简范围（若选择默认，系统筛选最优约简比例），测试人员点击 [变异体集合约简] 按钮，系统依次进行质量模型获取和变异体集合约简，生成约简后的变异体子集。随后，系统会自动跳转到变异体子集展示界面。在该界面中，测试人员可以查看变异体子集的信息。

测试报告生成。测试报告生成模块包含了测试用例运行的完整流程。系统支持测试人员选择要测试的变异体集合和生成的测试报告类型。系统根据测试人员的选择执行相应的测试，并根据测试结果生成一份详细的测试报告。报告应包涵覆盖测试结果以及变异测试结果。

4.1.3 非功能性需求

本系统目的是帮助软件测试人员评估测试用例集的缺陷检测能力，从而提高软件的质量。为了方便使用，本系统需要满足以下非功能性需求。

性能。系统前端各个界面的响应时间不应高于 1 秒，系统执行变异体集合约简的时间对于小型项目不应超过 5 分钟，对于大型项目不应超过 30 分钟。

可靠性。变异体集合约简的效果与测试用例集缺陷检测能力的准确度直接相关，因此对系统可靠性有较大需求。系统可靠性还要求当测试人员输入异常数据或系统出现运行故障时有相应的提示信息。

易用性。在变异体集合约简阶段，由于预测特征提取和预测模型构建的过程耗时较长，为了提升测试人员体验，系统应在变异体集合约简阶段给出整体运行进度。

可扩展性。由于使用的特征提取插件在不断的迭代更新，为了兼容变更的插件版本，系统需要保证良好的可扩展性。

4.2 总体设计

本节以上述需求分析总结出的功能性需求和非功能性需求为基础，对系统进行总体设计。首先，从宏观角度给出系统的架构设计，整体把握系统的层次。其次，从系统的逻辑视图、进程视图、开发视图、物理视图多个角度对系统的设计进行描述，然后对系统的持久化设计进行描述。

4.2.1 系统架构

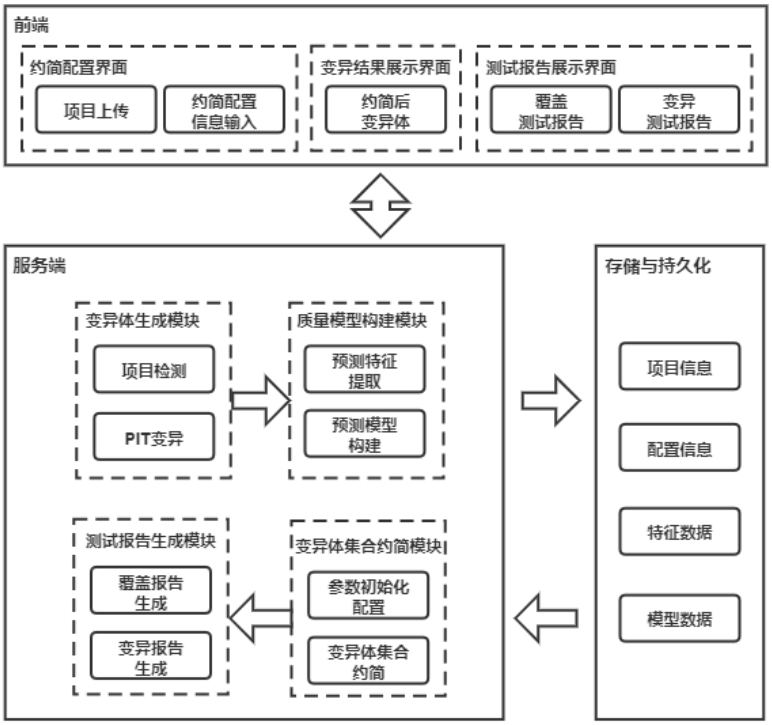


图 4-2: 系统整体架构图

本系统是一个面向回归测试测试用例评估场景的变异体集合约简系统，其架构如图4-2所示。基于对系统需求以及实现技术的分析与预测，可以将系统整体分为三部分：前端、服务端和数据存储层。从而确定本系统基于 C/S（Client/Server）架构，设计模式为 MVC（Model View Controller）。前端使用业界比较流行的 React-Mobx 技术进行开发，使 Ant Design 进行 UI 组件的设计，使用 Webpack 进行前端资源打包，并通过 Axios 与服务端 api 交互。前端部分采用前端界面的 UI 控件实现与用户交互，并应用 JavaScript 脚本语言编写

事件处理函数用以向服务端传送请求以及接收响应。服务端采用 Python 语言支持的 Flask 框架进行设计，便于预测变异体集合质量和处理集成测试结果数据。前端由约简配置界面、变异结果展示界面和测试报告展示界面构成，其中约简配置界面的项目上传功能负责与服务端中的变异体生成模块进行交互，传输项目信息；约简配置界面的约简配置信息输入功能负责与服务端中的变异体集合约简模块进行交互，传输变异体集合的约简比例；变异结果展示界面与服务端中的变异体集合约简模块进行交互，展示变异体集合约简结果；测试报告展示界面与服务端中的测试报告生成模块进行交互，分别展示覆盖测试报告和变异测试报告。服务端由变异体生成模块、质量模型构建模块、变异体集合约简模块和测试报告生成模块构成，其中变异体生成模块主要负责实现项目检测和 Pitest 变异生成功能；质量模型构建模块主要负责实现预测特征提取和预测模型构建功能；变异体集合约简模块主要负责实现参数初始化配置和变异体集合约简功能；测试报告生成模块主要负责实现覆盖测试报告生成和变异测试报告生成功能。所有的业务服务均采用传统的分层设计，可以较大程度减少业务耦合且具有良好的可拓展性。服务端使用 MongoDB 数据库存储项目信息、配置信息、特征数据和模型数据。为提高系统的可移植性和降低运维开销，系统整体采用 Docker 容器引擎进行打包部署。

4.2.2 架构视图

本节采用 Philippe Kruchten 的“4+1”视图 [51] 介绍本系统的总体设计。分别从逻辑视角、进程视角、开发视角以及物理部署视角描述系统的架构。

逻辑视图。图 4-3展示了变异体集合约简系统的逻辑视图。逻辑视图根据上文的功能性需求描述系统，通常按照面向对象的思路，将系统抽象为类或对象进行描述。本系统主要涉及的实体可以抽象为源程序（Origin Program）、变异体集合（MutantSet）、测试用例集（TestSet）、特征（Feature）、模型（Model）、变异体子集（MutantSubSet）和测试报告（Test Report），系统中主要服务可以抽象为用户服务（UserService）、变异体生成服务（MutantGenService）、项目检测服务（ProgramDefectService）、变异体生成子服务（MutantSetGenService）、测试运行服务（TestRunService）、质量模型构建服务（ModelGenService）、特征提取服务（TestNetService）、质量模型构建子服务（QuaModelService）、变异体集合约简服务（MSReduceService）、参数初始化服务（ParamConfigService）、变异体集合约简子服务（MutantSe-

tRedService）、测试报告生成服务（TestReportGenService）和数据存储服务（DataStorageService）。其中变异体生成服务由项目检测服务、变异体生成服务和测试运行服务构成，提供验证输入的项目路径是否正确、项目是否配置插件、项目的测试用例是否正确执行和变异生成的功能；质量模型构建服务由特征提取服务和质量模型构建子服务构成，提供预测特征提取和预测模型构建的服务；变异体集合约简服务由参数初始化服务和变异体集合约简子服务构成，提供变异体集合约简的功能；测试报告生成服务提供测试报告生成的功能；数据存储服务提供数据存储的功能。

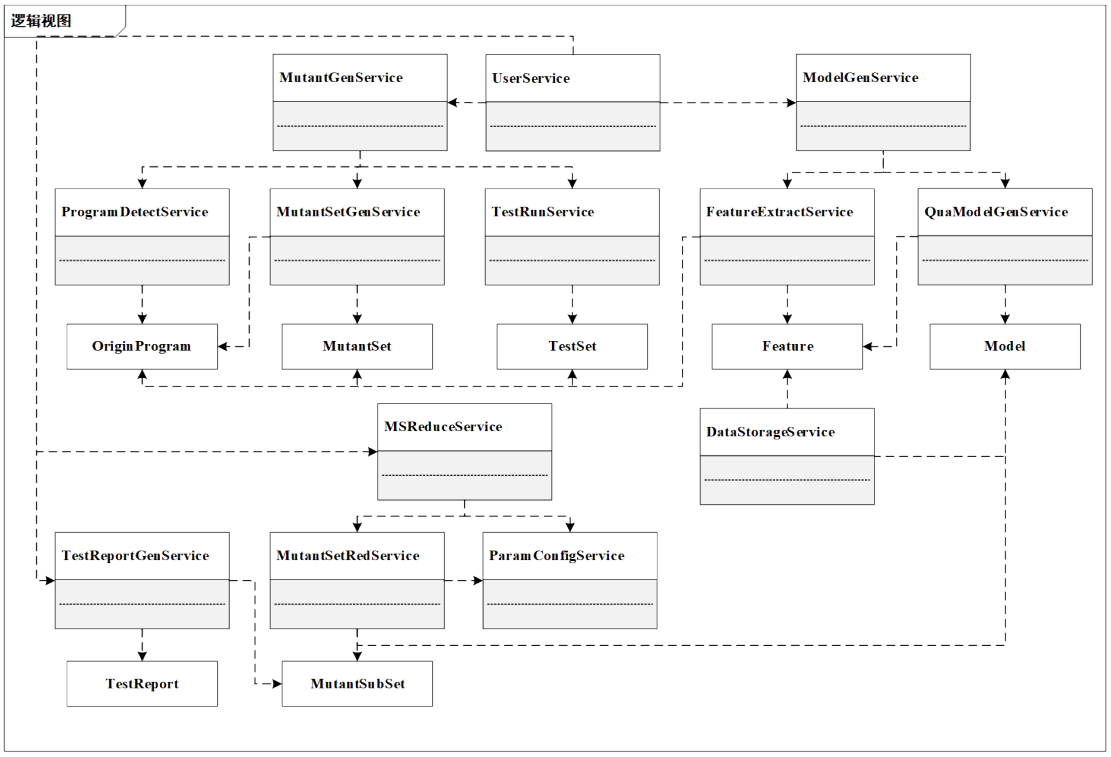


图 4-3: 逻辑视图

进程视图。图 4-4是本系统的进程视图。进程视图用于描述进程、线程、对象等运行时概念以及他们相关的并发、同步、通信问题。本系统中，当用户配置好源项目之后，项目路径以 JSON 的格式传递给变异体生成进程，开始对项目的合规性进行验证。验证通过后依次进行覆盖测试和变异生成。变异生成完毕后将全部变异体信息返回至主线程，以供展示。用户根据变异体信息选择约简比例并选择应用的模型场景，前端将用户输入的参数传递至变异体集合约简进程。变异体集合约简进程将调用质量模型构建进程进行特征提取以及预测模型构建，待获取预测模型后该进程再进行具体的变异体集合约简操作，生成

的变异体子集以供前端展示和系统后续运行。测试进程首先初始化本地测试环境，然后在测试环境中执行变异测试和覆盖测试，并将测试结果传递至测试报告生成进程。测试报告进程分析测试结果并返回测试报告至主进程。

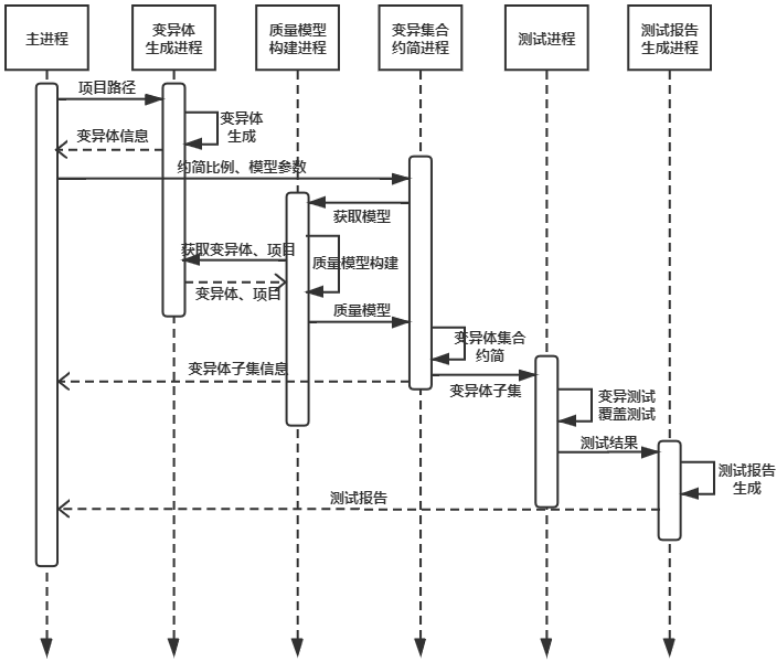


图 4-4: 进程视图

开发视图。图 4-5是本系统的开发视图。开发视图从系统的模块构成和开发过程的角度来描述系统，展示了软件包的层级结构。顶层是展示给用户的 UI，包含了 JavaScript、CSS、HTML 以及交互组件。服务层 ServiceLogic 根据系统功能和模块划分，主要包含 Controller、MutantGenService、ModelGenService、MSReduceService、TRGenService 和 DSService 六个包。Controller 包作为控制层，主要处理前端传回的外部请求，调用相应的系统服务。MutantGenService 包负责变异体生成，主要包含项目检测（ProgramDefect）、变异生成（MutantGen）和测试运行（TestRun）三个部分。ModelGenService 包负责质量模型构建，主要包含预测特征提取（FeatureExtract）和预测模型训练（ModelTrain）两个部分的代码。MSReduceService 包负责变异体集合约简，主要包含参数初始化配置（ParamConfig）和变异体集合约简（MutantSetRed）两个部分的代码。TRGenService 负责测试报告生成。DSService 负责数据存储。技术服务层 TechService 为开发本系统会用到的第三方组件和库，第三方库提供大量功能强大的通用方法，帮助开发人员从繁琐的重复逻辑中抽离出来，

从而让开发者专注业务逻辑。本系统用到多种第三方组件和库，变异体生成运行用到了 Pitest 插件；覆盖测试运用到了 Cobertura 插件；静态特征提取运用到了 JHawk 插件；机器学习模型训练运用到了 Sklearn 库；数据存储运用到了 MongoDB 数据库。为了提高系统的可靠性与可移植性，采用 Docker 和 GitHub 对系统进行部署和管理，Dockerfile 文件负责构建镜像使项目运行在容器中，.git 文件夹记录着 GitHub 的配置和版本信息。

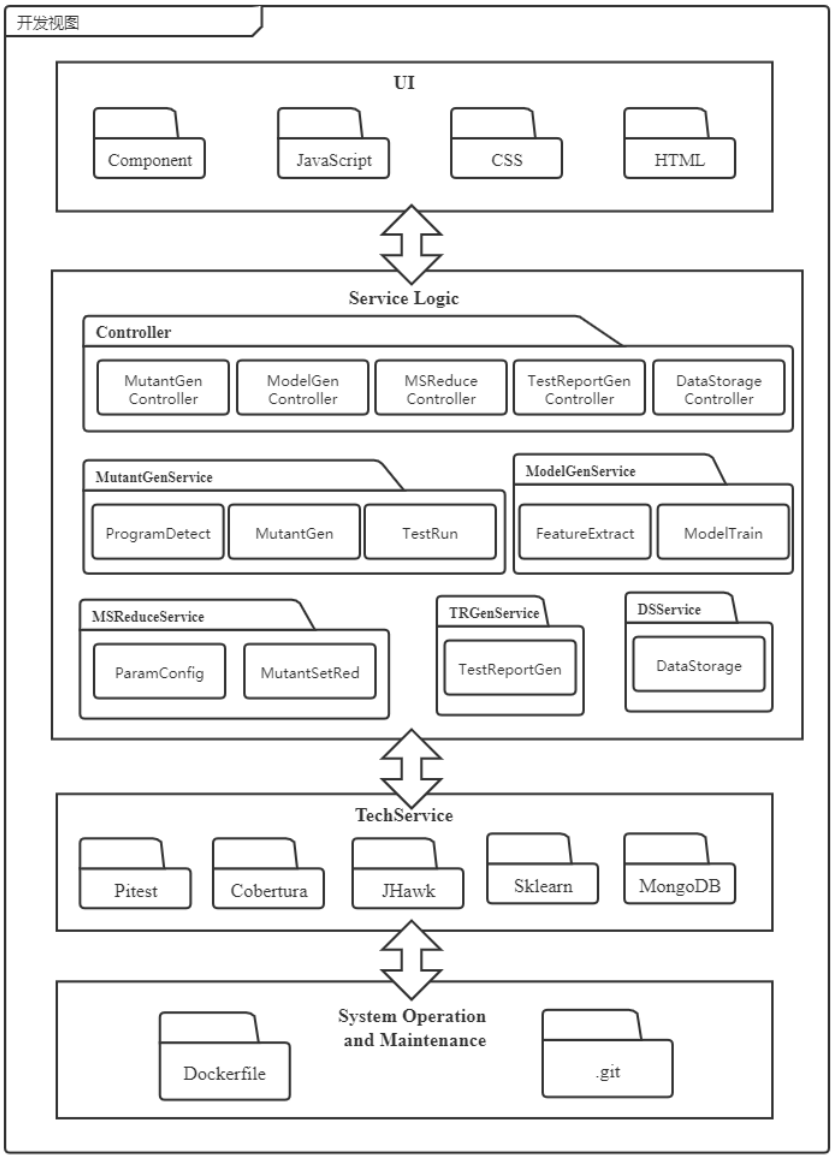


图 4-5: 开发视图

物理视图。图 4-6是本系统的物理视图。物理视图从系统运维人员的角度出发，对系统各个物理节点的服务部署以及节点间的通信机制进行说明。用户使用 PC 浏览器能够访问前端网页，请求经过 Nginx 转发到服务端服务器。服务端节点通过 Restful 接口与变异体集合约简服务 MSRServer 通信，实现变异体集合约简功能；通过 Restful 接口与测试报告生成服务 TRGServer 通信，实现测试报告生成功能。质量模型构建服务 QMGServer 和变异体集合约简 MSRServer 通过 Restful 接口与变异生成服务 MGServer 通信，获取项目的变异体集合。数据存储服务 DBServer 负责与 MSRSever 和 QMFServer 通信，用于存储预测特征和预测模型。

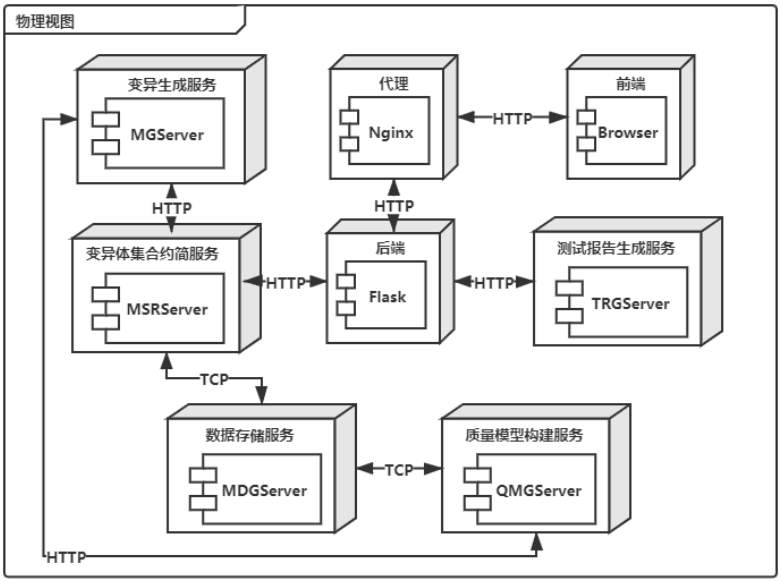


图 4-6: 物理视图

4.2.3 持久化设计

本文提出的面向回归测试测试用例评估场景的变异体集合约简系统主要涉及变异体信息、预测特征、覆盖测试报告和变异测试报告四个持久化对象。按照需求分析以及后续实验需求，以上四个持久化对象设计如下：

表4-4展示了变异体信息的持久化设计，每一条数据都对应一个变异体。该数据记录了生成变异体的详细信息，主要包含唯一识别 ID、变异行号、变异类名、变异方法名、变异算子和变异内容。所有信息在变异体生成步骤获取并传至前端和特征提取模块。

表 4-4: 变异体信息设计

字段名	类型	描述
id	string	变异体唯一识别 ID
Mutant RowNumber	int	变异行号
Mutant ClassName	string	变异类名
MutantMethod Name	string	变异方法名
Mutant Operator	string	变异算子
Mutant Detail	String	变异内容

表 4-5: 预测特征信息设计-方法级别

字段名	类型	描述
id	string	预测特征唯一识别 ID
CC	float	圈复杂度
NMofCommentLines	int	注释行数
Number of Statements	int	语句数
Halstead Length	int	霍尔斯特德长度
Halstead Vocabulary	int	霍尔斯特德词汇数
Halstead Effort	float	霍尔斯特德精力
Halstead Bugs	float	霍尔斯特德漏洞数
Classes Referenced	int	引用类数
ExternalMd Called	int	外部方法调用数
Local Methods Called	int	本地方法调用数
Lines Of Code	int	行数
Number of Comments	int	注释数
Arguments	int	方法参数数目
Modifiers	int	修饰词数
Halstead Difficulty	float	霍尔斯特德难度
Variable Declarations	int	变量声明数
Exceptions Thrown	int	抛出异常数
Exceptions Referenced	int	引用异常数
Number of casts	int	casts 数目
Total Depth of Nesting	int	嵌套深度
Halstead Volume	float	霍尔斯特德容量
Number of Operands	int	操作符数目
Variable References	int	变量引用数目
Number of Operators	int	运算符数目
Max. depth of nesting	int	最大嵌套深度
NMofExpressions	int	表达式数目
Number of Loops	int	循环数目

表 4-6: 预测特征信息设计-类级别

字段名	类型	描述
LCOM	float	缺乏内聚性指标
Average CC	int	平均圈复杂度
Number of Statements	int	语句数
Halstead C Bugs	float	霍尔斯特德漏洞数
Halstead Effort	float	霍尔斯特德精力
UnWeightedCls Size	int	未加权类大小
Instance Variables	int	实例变量数
Imported Packages	int	导入包数目
Response For class	int	类响应数
CBO	int	对象耦合数
Maintainability Index	float	可维护性指数
Cumulative Number of Comment Lines	int	注释行的累计数目
Lines Of Code	int	类行数
Review Factor	float	审查因数
Fan In	int	扇入
DIT	int	继承树深度
MIndex (NC)	float	可维护性指数（NC）
Specialization ratio	float	专业化比率
Reuse Ratio	float	重用比率
COH	flaot	COH
Local Method Calls	int	本地方法调用数目
LCOM2	float	LCOM2
Max Complexity	int	最大复杂度
Halstead Cumulative Volume	float	霍尔斯特德累计容量
Hierarchy Method Calls	int	最大嵌套深度
Number of Queries	int	查询数目
Fan Out	int	扇出
SIXs	int	SIX
External Method Calls	int	外部方法调用数
Superclasses	int	超类数目
Total Complexity	int	复杂度
Subclasses	int	基类数
Message Passing Coupling	int	信息传递耦合数
Number of Commands	int	类注释数
Interfaces	int	接口数
Cumulative Number of Comments	int	类累计注释数
Halstead Cumulative Length	int	霍尔斯特德累计长度
Modifiers	int	类修饰符
Number of Methods	int	类方法数目

表 4-7: 预测特征信息设计-包级别

字段名	类型	描述
Number of Classes	int	包中类数
Number of Statements	int	包中语句数目
Average Cyclomatic Complexity	float	平均圈复杂度
Halstead Cumulative Bugs	float	霍尔斯特德漏洞数
Halstead Effort	float	霍尔斯特德精力
Halstead Cumulative Length	int	霍尔斯特德累计长度
Halstead Cumulative Volume	float	霍尔斯特德累计容量
Maintainability Index	float	可维护性指数
Cumulative Number of Comment Lines	int	累计注释行数
Lines Of Code	int	行数
Review Factor	int	审查因素
Total Complexity Volume	int	复杂度
Cumulative Number of Comments	int	累计注释数
Instability	float	不稳定性
Distance	float	距离
FanIn	int	扇入
Number of Methods	int	方法数
Maintainability Index (NC)	int	可维护性指数（NC）
Abstractness	float	抽象性
Max Complexity	int	最大复杂度
FanOut	int	扇出

表 4-8: 预测特征信息设计-可达度、充分性

字段名	类型	描述
numTestExecuted	int	变异体集合中变异语句被整个测试用例集执行的次数
numTestCovered	int	变异体集合中变异语句被整个测试用例集覆盖的测试数
numMutantAssertion	int	覆盖变异体体集合中变异语句的测试方法中的断言总数
numClassAssertion	int	变异体体集合中变异类中的测试断言数
opAdequacy	float	变异算子充分性
posMdTypeAdequacy	float	变异位置的方法类别充分性
posMdeAdequacy	float	变异位置的方法充分性
posClasseAdequacy	float	变异位置的类充分性

表4-5-4-8展示了预测特征信息的持久化设计，其中包括复杂度特征、可达度特征和充分性特征的字段名、类型和描述。由于复杂度特征较多且涉及方

法、类和包三个类别，所以分成三张表进行描述；可达度特征和充分性特征合成一张表描述。预测特征的类型全部为数值型，有利于预测模型的处理。

表4-9展示了覆盖测试报告的持久化设计。该数据记录了项目测试用例覆盖率的详细信息，主要包含项目的覆盖测试唯一识别ID、包名、类数目、代码行数、语句覆盖率、分支覆盖率和包复杂度。在覆盖测试报告生成步骤获取并传至前端供用户查看。表4-10展示变异测试报告的持久化设计，该数据记录了测试用例集在变异体子集上变异执行结果的详细信息，主要包含变异测试唯一识别ID、变异体数目、变异得分和测试用例数目。在变异测试报告生成步骤获取并传至前端供用户查看。

表 4-9: 覆盖测试报告设计

字段名	类型	描述
id	string	项目覆盖测试唯一识别 ID
Package Name	string	包名
Class Number	int	类数目
Line Number	int	代码行数
Line Coverage	float	语句覆盖率
Branch Coverage	float	分支覆盖率
Complexity	float	包复杂度

表 4-10: 变异测试报告设计

字段名	类型	描述
id	string	项目变异测试唯一识别 ID
Mutant Number	int	变异体数目
Mutant Score	float	变异得分
TestCase Number	int	测试用例数目

4.3 本章小结

本章首先对变异体集合约简系统进行用例分析，根据用例分析的结果，将系统划分为变异体生成、质量模型构建、变异体集合约简、测试报告生成四个模块；随后从功能性需求和非功能性需求两个方面对系统进行需求分析；然后介绍系统总体设计，介绍了系统整体架构和持久化设计并从逻辑、进程、开发和物理四个角度对系统进行详细描述。

第五章　系统实现与测试

依据系统需求分析与概要设计，本章介绍变异体集合约简系统的实现与测试。首先描述系统设计与实现，分别对变异体生成模块、质量模型构建模块、变异体集合约简模块和测试报告生成模块的设计与实现进行介绍；并通过界面截图对本系统的操作流程以及运行状态进行展示和说明。然后进行系统测试设计，包括功能测试和性能测试，并对系统测试结果进行分析。

5.1　系统设计与实现

5.1.1　变异体生成模块设计与实现

变异体生成模块主要负责变异体约简前的准备工作，包括项目检测和变异体生成两个功能。项目检测主要负责验证用户上传的项目是否符合标准，即是否是 Java Maven 项目，验证项目中的测试用例集是否可以成功执行，且是否包含 Pitest 插件和 Cobertura 插件的配置信息。用户首先在前端提供的文件上传框中上传本地 Java Maven 项目压缩包，接着服务端获取前端返回的项目压缩包，并解压项目保存至服务器端，并在服务端对项目执行覆盖测试。测试成功后进入该项目目录下检测是否包含主文件夹 main、测试文件夹 test 以及项目对象模型文件 pom.xml。如果包含上述文件或文件夹，则初步判断项目为 Maven 构建。然后进入 pom.xml 文件检测是否包含 Pitest 插件和 Cobertura 插件的配置信息。如果包含上述配置信息，则判断项目可以进入变异体生成阶段。

变异体生成主要负责调用 Pitest 插件生成源程序的变异体。Pitest 是一种面向 Java 的基于快速字节码的变异测试工具，可用于生成变异体、计算变异得分和检测试单元测试的有效性。它可作为 Maven 的插件与整个项目集成为一体。系统首先调用” mvn org.PIT:PIT-maven:mutationCoverage” 命令生成变异体，变异结果保存在 mutations.xml 文件中。然后判断该文件的大小，若文件不为空，说明变异体已成功生成，则可以进入到特征提取子模块并将变异体信息反馈给用户，以用于约简比例的选择。

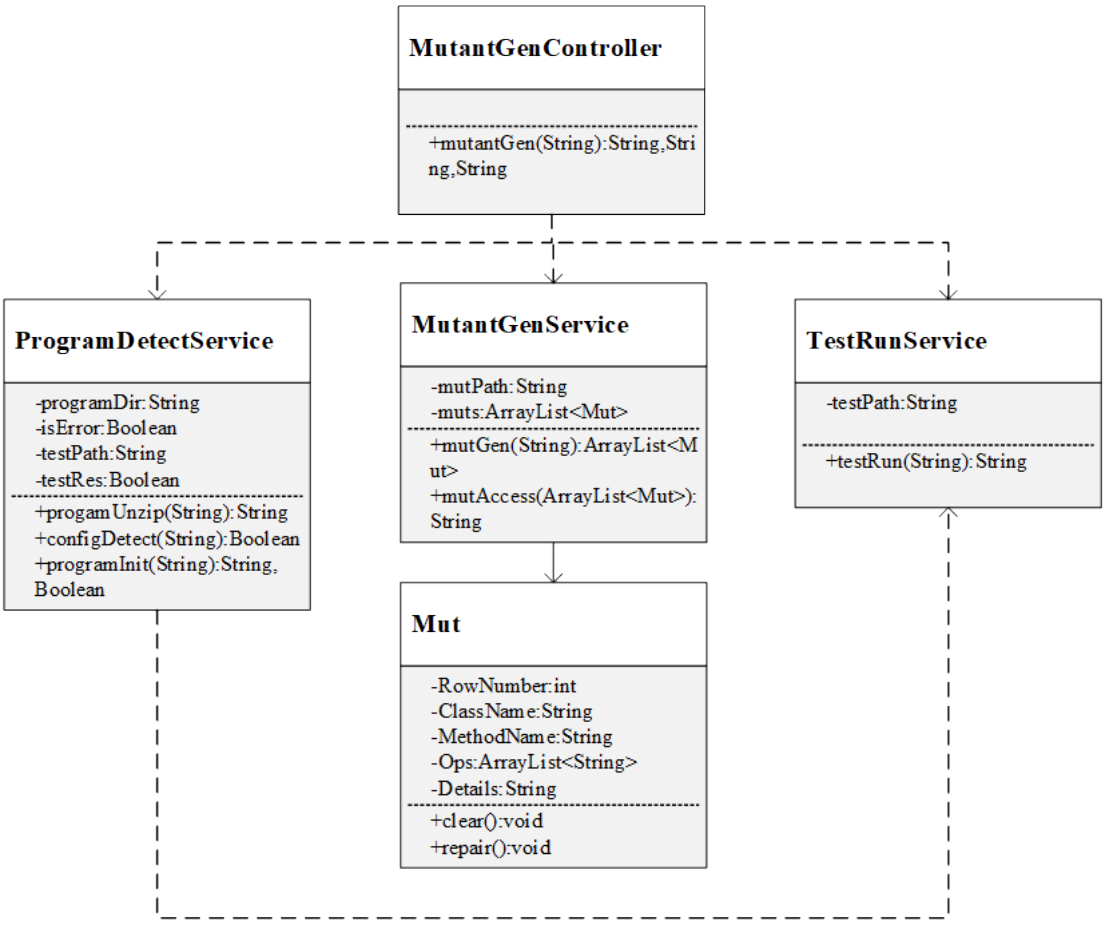


图 5-1: 变异体生成模块类图

变异体生成模块的核心类图如图 5-1所示。其中 MutantGenController 主要负责调用 ProgramDetectService 进行项目检测、调用 TestRunService 进行覆盖测试以及调用 MuantGenService 进行变异生成。ProgramDetectService 主要接受前端上传的项目，并负责解压缩项目、检测项目的插件配置信息以及初始化项目，初始化项目时，调用 TestRunService 对源程序进行覆盖测试，检查项目中的测试用例是否可以正确执行。MuantGenService 负责调用 Pitest 插件生成源程序的变异体，并分别将变异体信息返回至前端和质量模型构建模块。

变异体生成模块的顺序图如图 5-2所示。用户上传待测项目，系统首先解压缩项目并验证是否包含 Pitest 插件和 Cobertura 插件，若验证成功则开始初始化项目，否则返回错误信息。初始化项目时将项目路径传递至测试运行服务执行覆盖测试，用于检测项目中的测试用例集是否成功执行。若测试成功则将项目路径传递至变异生成服务，否则返回错误信息。变异生成服务调用 Pitest 插件生成变异体集合，并将其分别传递至前端和质量模型构建进程。

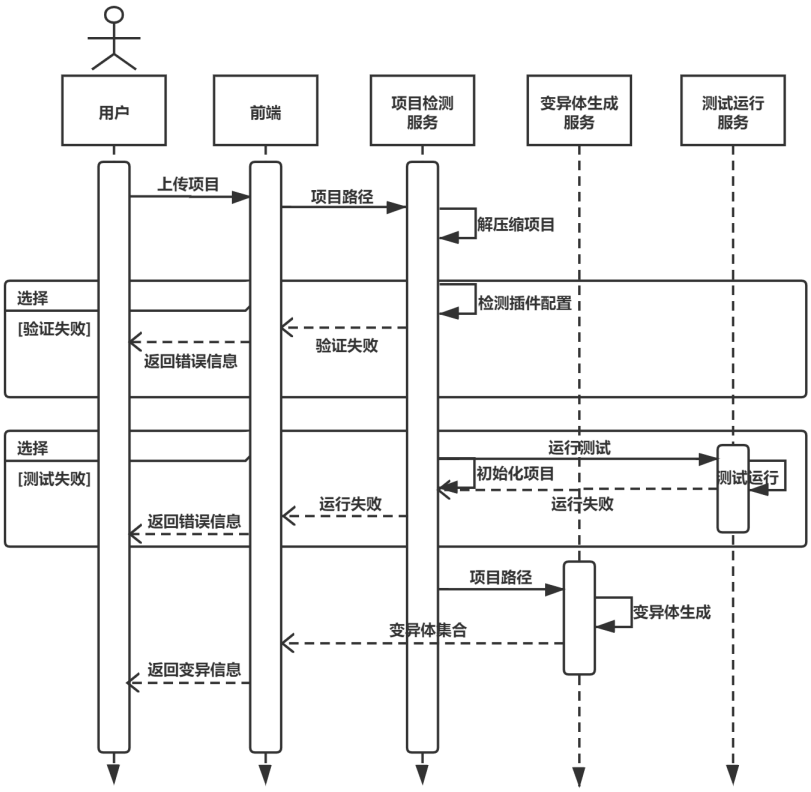


图 5-2: 变异体生成模块顺序图

```
def mutantGen(self,orignprogramPath):
pds=ProgramDetectService()
# 解压缩文件
programdir=pds.programUnzip(orignprogramPath)
# 配置信息检查
isTrue=pds.configDetect(programdir)
# 初始化错误信息、测试用例集路径和变异体集合路径
error,mutPath,testPath=None
if(isTrue):
# 项目初始化并执行覆盖测试
testPath,testRes=pds.programInit(programdir)
if(testRes):
# 变异信息生成
mgs=MutantGenService()
mut=mgs.mutGen(programdir)
mutPath=mgs.MutAccess(mut)
```

```
17 # 返回错误信息
18 if(!isTrue):
19 error='项目配置有误'
20 elif(!testRes):
21 error='项目运行失败'
22 return error,testPath,mutPath
```

根据上述设计，本文对变异体生成模块进行了实现。上述代码展示了变异体生成模块的关键代码。mutantGen 主要负责检测项目和生成变异体集合。该方法接收压缩项目文件路径作为参数，用于传递此参数至 ProgramDetectService 类中的 programUnzip() 方法进行项目解压缩。代码第 4-11 行的作用是先解压缩项目，再调用 configDetect() 方法来验证项目是否存在 Pitest 插件和 Cobertura 插件的信息，验证通过后则调用 programInit() 方法进行项目初始化并执行覆盖测试。代码第 12-16 行的作用是生成变异体集合。如果验证通过且执行无误则顺序调用 MutantGenService 类中的 mutGen() 方法和 mutAccess() 方法进行变异信息生成。如果验证失败则在第 18-21 行记录错误信息。最后返回错误信息、测试用例集路径和变异体集合路径。

5.1.2 质量模型构建模块设计与实现

质量模型构建模块主要负责变异体集合质量预测模型的构建，包括预测特征提取和预测模型构建两个功能。预测特征提取主要负责提取质量得分的预测特征。首先采用 JHawk 工具对项目进行静态分析, 采用 Cobertura 对项目进行覆盖测试，以提取变异顽固特征。JHawk 是静态代码分析工具，它获取项目的源代码并根据代码的数量，复杂性，类与包之间的关系以及类与包内的关系来计算指标。Cobertura 是一种报告工具，可以计算 Java 项目的测试覆盖率。然后对上一阶段生成的变异体进行分析，提取变异充分特征。特征提取完成后将预测特征以 csv 文件格式存储在文件系统中，用于变异体集合质量预测模型构建。

预测模型构建主要用于训练变异体集合质量预测模型进行变异体集合质量得分的预测。首先处理上一阶段生成的预测特征构建训练集，然后运用训练集训练变异体集合质量预测模型，最后将变异体集合质量预测模型保存在数据库。该模型将作为质量评估函数输入至变异体集合约简模块。

图 5–3展示了质量模型构建模块的核心类图。其中 ModelGenController 主要负责调用特征提取服务 FeatureExtractService 进行预测特征提取以及调用质

量模型构建服务 QuaModelGenService 进行预测模型构建。FeatureExtractService 主要接受变异体集合约简服务传来的指令进行预测特征提取，首先调用变异体生成服务获取测试用例集和变异体集合，然后对其进行分析并提取顽固特征和充分特征，最后将两类特征合并以 csv 文件格式存储在服务端的文件系统中，并将预测特征的文件路径传递至质量模型构建服务 QuaModelGenService。QuaModelGenService 负责构建训练集和预测模型。首先对预测特征进行处理生成训练集，然后采用训练集构建预测模型，最后将预测模型传入至数据库和变异体集合约简模块。

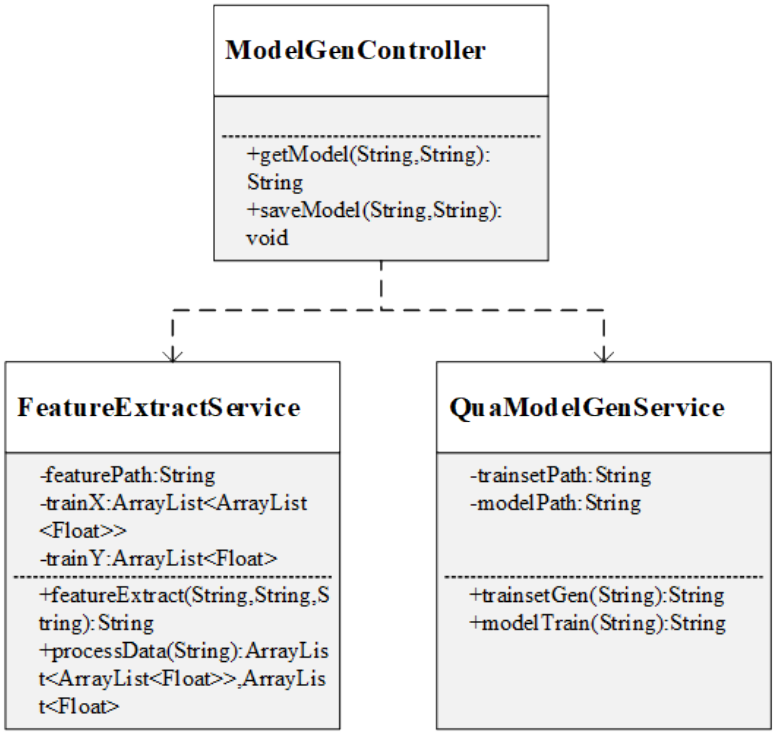


图 5-3: 质量模型构建模块类图

质量模型构建模块的顺序图如图5-4所示。首先，系统的集合约简进程在进行质量评估函数构建时，调用预测特征提取服务获取预测模型。然后，预测特征提取服务调用变异体生成进程获取测试用例集和变异体集合，并对其进行分析提取预测特征。最后，将预测特征传输至质量模型构建服务进行质量模型构建，构建成功后分别将预测模型传入至数据库和集合约简进程。

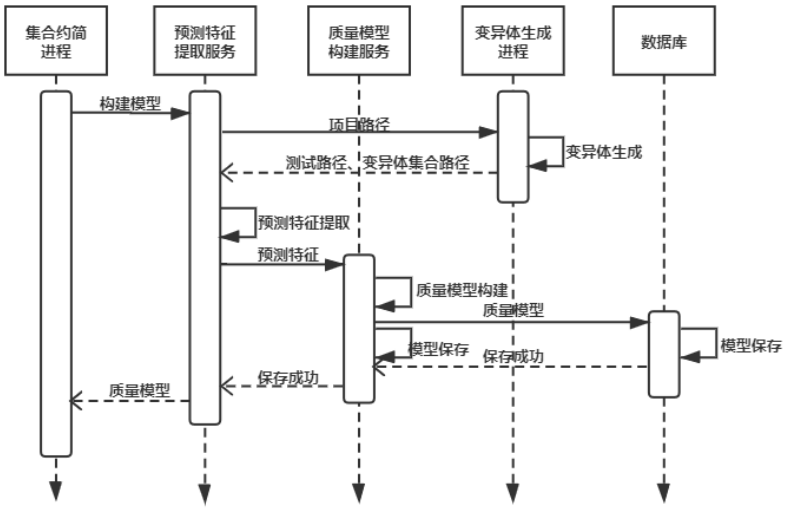


图 5-4: 质量模型构建模块顺序图

```
def modelGen(self,programPath):
mtgc=MutantGenController()
# 初始化特征路径和模型路径
featurePath,modelPath=None
# 根据项目获取测试用例集和变异体集合
error,testPath,mutPath=mtgc.mutantGen(programPath)
if(error):
return error,modelPath
else:
# 提取预测特征
fes=FeatureExtractService()
featurePath=fes.featureExtract(programPath,testPath,mutPath)
# 构建训练集
qmgs=QuaModelGenService()
trainsetPath=qmgs.trainsetGen(featurePath)
# 构建预测模型
modelPath=qmgs.modelTrain(trainsetPath)
return error,featurePath,modelPath
```

根据上述设计，本文对质量模型构建模块进行了实现。上述代码展示了质量模型构建模块的关键代码。modelGen 主要负责提取预测特征以及构建预测模型。该方法接收项目路径作为参数，用于传递此参数至 MutantGenController 类中的 mutantGen() 方法获取测试用例集的路径 testPath 和变异体集合

的路径。如果 mutantGen() 方法存在错误则在第 7-8 行返回错误信息以及空的特征路径和模型路径。否则在第 9-17 行依次执行 FeatureExtractService 类的 featureExtract() 方法进行预测特征提取，再调用 QuaModelGenService 类的 trainsetGen() 和 modelTrain() 方法构建预测模型。最后返回错误信息、特征路径和模型路径。

```
1  def modelTrain(self,featurePath):
2  # 获取训练集
3  X,y=self.processData(featurePath)
4  # 网格搜索法搜索最优参数
5  parameters={'n_estimators':[10,20,30,40,50],
6  'max_features': [1,2,3,4,5]}
7  forestReg=RandomForestRegressor()
8  gridSearch=GridSearchCV(forestReg, parameters, cv=5,scoring='r2')
9  gridSearch.fit(X, y)
10 # 生成最佳训练模型
11 bestParms=gridSearch.best_params_
12 bestModel = RandomForestRegressor(n_estimators=bestParms.n_estimators,
13 max_features=bestParms.max_features)
14 bestModel.fit(X, y)
15 # 保存训练模型
16 parentpath=os.path.abspath(os.path.dirname(featurePath)
17 +os.path.sep+".")
18 modelpath=os.path.join(parentpath,'quaModel.pkl')
19 with open(modelpath, "wb") as f:
20 pickle.dump(bestModel, f)
21 return modelpath
```

变异体集合质量预测模型作为变异集合约简方法的适应度方法用来评价变异集合质量的优劣，质量预测模型训练是本文方法实现的关键步骤，在此详细介绍质量预测模型训练的具体实现。上述代码展示了 QuaModelGenService 类中 modelTrain() 方法的代码，modelTrain 主要负责训练质量预测模型。该方法接收特征路径 featurePath 作为参数，在第 3 行传递此参数至 QuaModelGenService 类的 processData() 方法获取处理后的训练特征 X 和训练标签 y。代码 5-9 行的作用是运用网格搜索法搜索模型的最佳参数，首先配置参数 parameters 和机器学习模型 forestReg, 然后将其传入 GridSearchCV 算法中搜索最优参数 bestParams。代码 11-14 行的作用是根据 bestParams 生成最佳训练模型，首先将

bestParams 中的参数依次传入机器学习模型中，然后传入训练数据拟合最佳模型 bestModel。代码 16-20 行的作用是保存最佳预测模型，首先根据 featurePath 获取输出父路径 parentPath, 然后根据 parentPath 构建模型输出路径 modelPath, 最后将 bestModel 保存至 modelPath 中，并返回模型路径。

5.1.3 变异体集合约简模块设计与实现

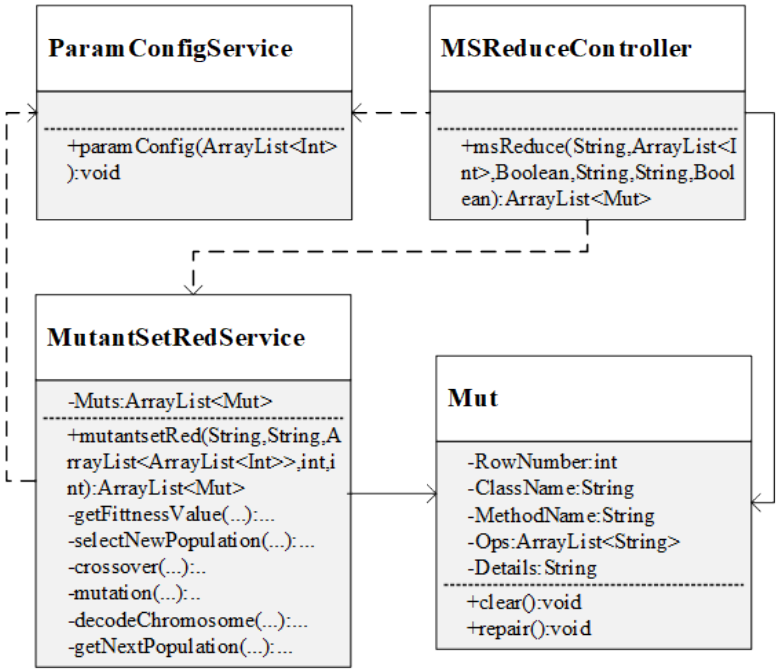


图 5-5: 变异体集合约简模块类图

变异体集合约简模块主要负责的是根据用户输入的约简比例进行变异体约简，包括参数初始化配置和变异体集合约简两个功能。参数初始化配置功能主要负责种群初始化和预测模型获取。首先，用户在前端选择新建模型还是使用已有模型，如果选择新建模型并再次选择是否保存模型，服务端使用 request 对象获取前端返回的参数 hasModel、isSave、username 和 programPath, 用以判断是否需要重新训练变异体集合质量预测模型。如果 hasModel 为真，系统则根据 username 和 programPath 从数据库中获取已有模型；否则系统将 isSave 传递至质量模型构建服务新建预测模型。如果 isSave 为真，质量模型构建服务将预测模型保存在数据库。然后，用户在前端提供的滑动输入条中选择约简比例，若不选择，则系统默认进行最优化约简，即约简后的变异体子集的质量得分

最高。接着服务端使用 request 对象获取前端返回的参数 reduceRatio，并根据 reduceRatio 初始化种群。

变异体集合约简主要运用遗传算法进行变异体集合的约简。首先，系统根据上一阶段初始化的种群和获取的预测模型调用遗传算法，依次进行质量评估、种群迭代，最终生成变异体子集。然后，将约简后的变异体子集返回至前端页面，以供用户参考。

变异体集合约简模块的核心类图如图5-5所示。其中 MSReduceController 主要负责调用 ParamConfigService 进行遗传算法的参数初始配置、调用 MuantSetRedService 进行变异体集合约简。ParamConfigService 负责接收约简比例进行种群初始化，负责接收模型参数获取变异体集合质量预测模型。MuantSetRedService 负责调用遗传算法生成变异体子集，并将约简后的变异体子集返回至前端，以供用户参考。

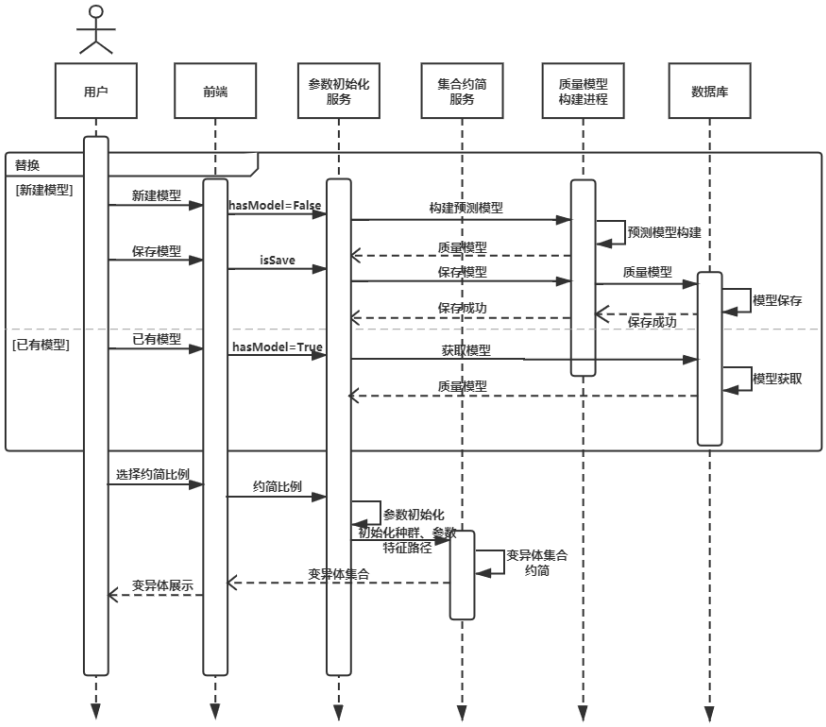


图 5-6: 变异体集合约简模块顺序图

变异体集合约简模块的顺序图如图5-6所示。首先用户选择合适的约简应用场景，如果选择新建模型，前端传递”hasModel=False” 给服务端，服务端调用质量模型构建进程新建预测模型，并将预测模型传递至参数初始化服务。如果用户触发保存模型按钮，前端传递 isSave 参数给服务端，服务端将预测模型

保存在数据库。如果选择已有模型，前端传递”hasModel=True” 给服务端，服务端从数据库获取已有模型。然后用户选择约简比例，前端将约简比例传递至服务端，服务端使用约简比例初始化种群，并将种群和预测模型作为参数传递至遗传算法进行变异体集合的约简。最后将约简后的变异体子集返回至前端页面，以供用户参考。

```
1  def msReduce(self,reduceRatio,hasmodel,username,programPath,isSave=False):
2  mgc=ModelGenController()
3  pcs=ParamConfigService()
4  # 初始化错误信息和变异体子集
5  error,mutantset=None
6  # 如果不存在模型则进行模型训练
7  if not hasmodel:
8  # 新建模型
9  error,featurePath,modelPath=mgc.modelGen(programPath)
10 if error:
11 return error,mutantset
12 # 保存模型
13 if isSave:
14 mgc.modelSave(username,programName,modelPath)
15 # 若存在模型则根据用户名和项目名从数据库中获取模型
16 else:
17 modelPath=mgc.modelAccess(username,programPath)
18 # 参数初始化配置
19 population,modelPath=pcs.paramConfig(mutationSize,popSize,
20 reduceRatio,modelPath)
21 # 变异体集合约简
22 mrs = MutantRedService()
23 mutantSet=mrs.mutantsetRed(featurePath,modelPath,population,
24 maxIter,popSize)
25 return error,mutantSet
```

根据上述设计，本文对变异体集合约简模块进行了实现。上述代码展示了变异体集合约简模块的关键代码。MSReduce 主要负责配置遗传算法参数和约简变异体集合。该方法接收约简比例 reduceRation、模型存在标志 hasModel、用户名 usename、项目名 programPath、模型保存标志 isSave 作为参数，用于变异体集合约简的一系列操作。代码第 7-20 行的作用是获取变异体集合质量预

测模型和初始化种群。首先系统根据 hasModel 判断是否存在预测模型，如果不存在，则将 programPath 传递至 ModelGenController 类中的 modelGen() 方法进行预测模型构建。如果 modelGen() 方法存在错误则在第 7-9 行返回错误信息以及空的变异体子集，否则获取特征路径 featurePath 和模型路径 modelPath。再根据 isSave 判断是否需要调用 ModelGenController 类中的 modelSave() 方法保存新建模型，默认不保存。如果存在模型，则将 username 和 programPath 传递至 ModelGenController 类中的 modelAccess() 方法获取已有模型。最后开发人员配置变异体集合大小 mutationSize 和种群大小 popSize，并将以上两个参数连同 featurePath 和 modelPath 传递至 ParamConfigService 类中的 paramConfig() 方法进行种群初始化。代码 22-24 行的作用是约简变异体集合，系统将开发人员配置的参数迭代次数 maxIter 连同 featurePath、modelPath、population 和 popSize 传递至 MutantRedService 类中的 mutantsetRed() 方法进行变异体集合约简。最后系统返回错误信息和变异体子集。

```
1  def mutantsetRed(self,featurePath,modelPath,population,maxIter,popSize):
2  # 每次迭代得到的最优解
3  optimalSolutions=[],optimalValues=[]
4  # 获取变异体集合的特征向量
5  df=pd.read_csv(featurePath,delimiter=",",
6  quoting=csv.QUOTE_NONE,encoding='utf-8')
7  # 获取预测变异体集合质量的预测模型
8  with open(modelpath,"rb") as f:
9  model=pickle.load(f)
10 for iteration in range(max_iter):
11 # 得到个体适应度值和个体的累积概率
12 fitnessvalues,cum_proba=self.getFittnessValue(model,
13 indexList,df)
14 # 选择新的种群
15 newpopulations=self.selectNewPopulation(population,cum_proba)
16 # 进行交叉操作
17 crossoverpopulation=self.crossover(newpopulations)
18 # 进行变异操作
19 mutationpopulation=self.mutation(crossoverpopulation)
20 # 构建新种群
21 totalpopulation=np.vstack((population,mutationpopulation))
```

```
22 nextpopulation=self.getNextPopulation(totalpopulation,
23 model,popSize,df)
24 # 将新种群解码，得到每轮迭代最终的种群的indexList
25 newIndexList=self.decodedChromosome(nextpopulation)
26 # 适应度评价
27 fitnessvalues,cum_proba=self.getFittnessValue(model,
28 newIndexList,df)
29 # 搜索每次迭代的最优解，以及最优解对应的目标函数的取值
30 optimalValues.append(np.max(list(fitnessvalues)))
31 index=np.where(fitnessvalues==max(list(fitnessvalues)))
32 optimalSolutions.append(nextpopulation[index[0][0],:])
33 population=nextpopulation
34 # 搜索最优解
35 optimalValue=np.max(optimalValues)
36 optimalIndex=np.where(optimalValues==optimalValue)
37 optimalSolution=optimalSolutions[optimalIndex[0][0]]
38 return optimalSolution
```

本系统的核心功能是由基于遗传算法的变异体集合约简方法实现的，在此详细介绍基于遗传算法的变异体集合约简方法的具体实现。上述代码展示了MutantRedService 类中 mutantsetRed() 方法的代码，mutantsetRed 主要负责约简变异体集合。该方法接收特征路径 featurePath、模型路径 modelPath、初始化种群 population、迭代次数 maxIter 和种群大小 popSize 作为参数，用于变异集合约简的一系列操作。代码 3-9 行的作用是获取种群迭代所需的参数，首先建立列表 optimalSolutions 和 optimalValue 以保存每次迭代的最优解和最优值，其次根据 featurePath 获取变异集合的特征向量 df, 然后根据 modelPath 获取质量预测模型。代码 10-33 行的作用是进行种群迭代，以搜索质量最优的变异集合。在每次迭代过程中，首先采用质量预测模型进行个体适应度值的计算，再依次进行种群选择、种群交叉和种群变异，然后对原有种群 population 和变异后的种群 mutationpopulation 进行适应度值的升序排序，选择前 popSize 个个体构成新种群 nextpopulation，最后对 nextpopulation 进行适应度评价，将每次迭代的最优解存放至 optimalSolutions。代码 35-37 行的作用是搜索最优解，即搜索 optimalSolutions 中适应度值最高的个体。最后返回质量最优的变异体子集。

5.1.4 测试报告生成模块设计与实现

测试报告生成模块主要负责生成测试报告，包括覆盖报告生成功能和变异报告生成功能。覆盖报告生成功能主要负责生成覆盖测试报告。系统首先调用Cobertura插件对项目进行覆盖测试，Cobertura是一种报告工具，可以计算Java项目的测试覆盖率。接着解析覆盖测试结果并将结果数据通过api接口传递至前端，前端以表格的形式向用户展示覆盖测试报告，以供用户参考。

变异报告生成功能主要负责生成变异测试报告。系统首先调用Pitest插件计算测试用例集在约简后的变异体集合上的变异得分，Pitest是一种面向Java的基于快速字节码的变异测试工具，可用于生成变异体、计算变异得分和检测试单元测试的有效性。然后将约简后的变异体数目、测试用例数目和变异得分通过api接口传递给前端，前端以表格的形式向用户展示变异测试报告，以供用户评估测试用例集的有效性。

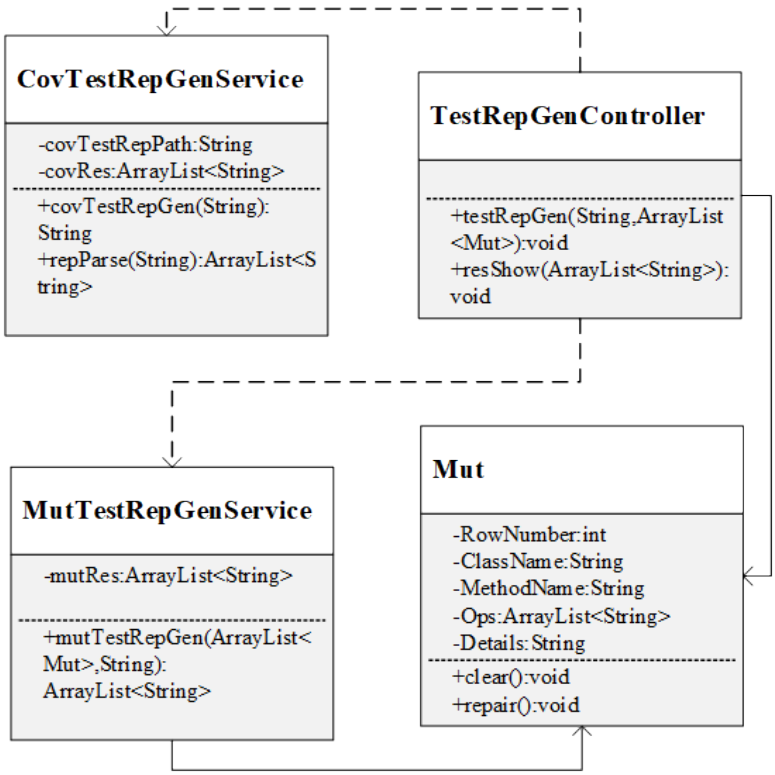


图 5-7: 测试报告生成模块类图

测试报告生成模块的核心类图如图5-7所示。其中TestRepGenController主要负责调用CovTestRepGenService生成覆盖测试报告、调用MutTestRepGenService生成变异测试报告。CovTestRepGenService负责调用Cobertura插件执行覆

盖测试并解析覆盖测试报告，将测试结果返回至前端供用户参考。MutTestRepGenService 负责调用 Pitest 插件计算测试用例集在约简后的变异体集合上的变异得分，将变异得分、约简后变异体数目和测试用例集的测试用例数目返回至前端供用户参考。

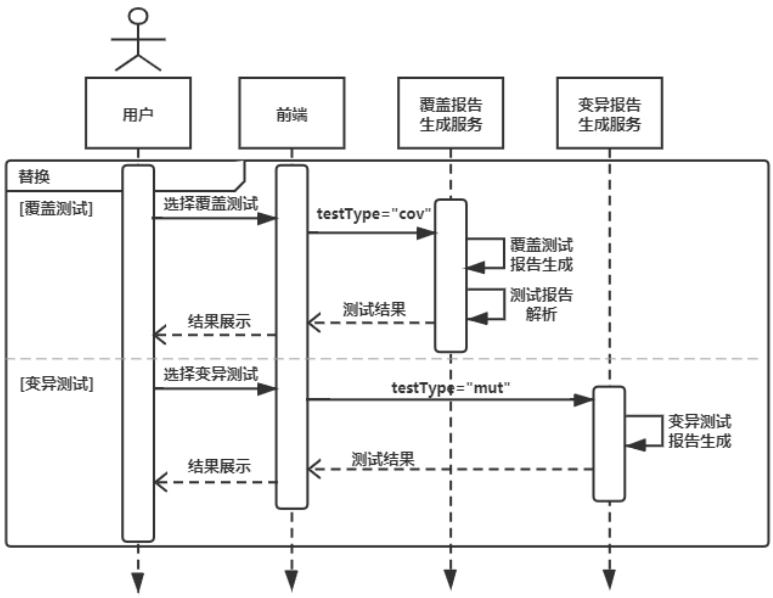


图 5-8: 测试报告生成模块顺序图

测试报告生成模块的顺序图如图5-8所示。首先用户选择测试类型，如果选择覆盖测试，前端传递”testType=cov” 给服务端，服务端调用 Cobertura 插件执行覆盖测试，并解析覆盖测试报告将测试结果返回至前端供用户参考。如果选择变异测试，前端传递”testType=mut” 给服务端，服务端调用 Pitest 插件计算测试用例集在约简后的变异体集合上的变异得分，将变异得分、约简后变异体的数目和测试用例数目返回至前端供用户参考。

```
def testRepGen(self,programPath,muts,testType):
# 根据类型判断进行哪一种测试
if testType==”cov”:
ctrgs=CovTestRepGenService()
# 执行覆盖测试
covRepPath=ctrgs.covTestRepGen(programPath)
# 解析测试报告
covRes=ctrgs.repParse(covRepPath)
# 输出结果
```

```
10 resShow(covRes)
11 else:
12 mtrgs=MutTestRepGenService()
13 # 执行变异测试
14 mutRes=mtrgs.mutTestRepGen(muts,programPath)
15 # 输出结果
16 resShow(mutRes)
```

根据上述设计，本文对测试报告生成模块进行了实现。上述代码展示了测试报告生成模块的关键代码。testRepGen 主要负责根据用户选择的测试类型，生成相应的测试报告。该方法接收项目路径 programPath、约简后的变异体集合 muts 和测试类型作为参数，用于测试报告生成的一系列操作。如果用户选择的测试类型是覆盖测试，系统执行代码第 3-10 行进行覆盖测试报告生成。首先，系统将 programPath 传递至 CovTestRepGenService 类的 covTestReGen() 方法执行覆盖测试，并返回测试报告路径 covRepPath。接着，将 covRepPath 传递至 CovTestRepGenService 类的 repParse() 方法进行覆盖报告解析。最后，调用 resShow() 方法将覆盖测试结果返回至前端。如果用户选择的测试类型是变异测试，系统执行代码第 11-16 行进行变异测试报告生成。首先，系统将 programPath 传递至 MutTestRepGenService 类的 mutTestReGen() 方法计算测试用例集在约简后的变异体集合上的变异得分。接着，调用 resShow() 方法将变异得分、约简后变异体的数目和测试用例数目返回至前端，以供用户参考。

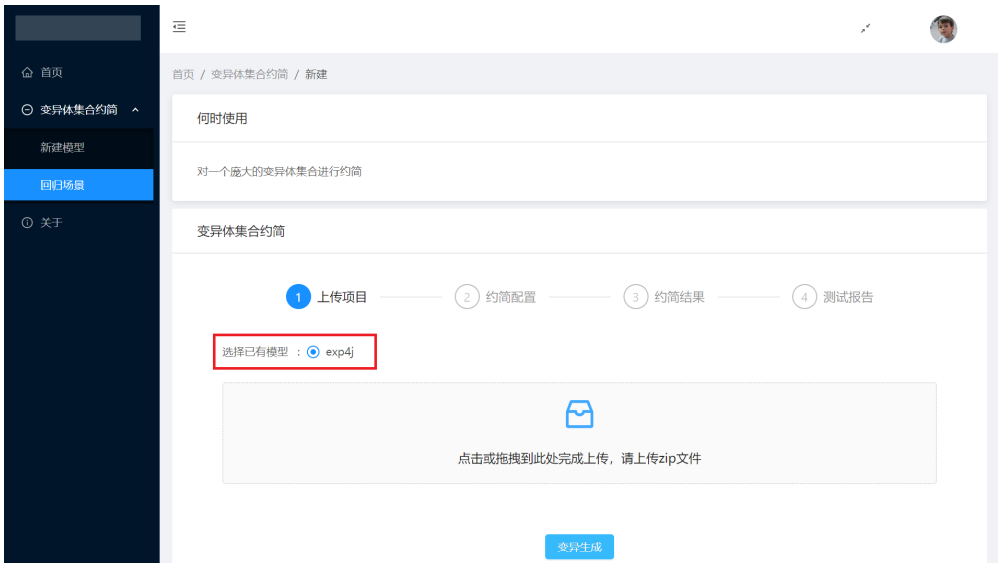
5.1.5 系统运行展示

本文实现的变异体集合约简系统包含八个页面，分别是新建模型场景下的项目上传页面、约简配置页面、结果展示页面和测试报告展示页面和回归测试场景下的项目上传页面、约简配置页面、结果展示页面和测试报告展示页面。由于新建模型和回归测试场景下只有项目上传页面有差别，故本节以截图的形式展示系统不同的五个页面，并作简要说明。

项目上传（Program Upload）页面如图5-9所示。该页面主要负责的功能为项目上传和变异生成。使用时首先在页面上的文件上传框中上传项目压缩包，然后点击 [变异生成] 按钮即可进行变异生成。若项目包含 Pitest 插件和 Cobertura 插件，系统会自动调用插件生成变异体和提取预测特征，并成功跳转变异体集合约简页面。若项目缺少相关插件，系统会弹出错误提示框，提示用

户项目缺少相关插件，需要重新上传项目。新建模型与回归测试模型唯一不同点是，用户可以选择之前保存过的模型进行变异体集合约简。如图5-9(b)所示（图中采用红色方框标出），页面展示已有模型供用户选择。

(a) 项目上传页面-新建模型



(b) 项目上传页面-回归场景

图 5-9: 项目上传页面展示图

约简配置（Reduce Config）页面如图5-10所示。该页面主要负责的功能为约简配置信息输入以及变异体集合约简。用户首先可以查看生成的变异体集合的信息，并可以根据自己的需求选择约简比例。若不选择，则系统默认进行最优化约简，即约简后的变异体子集的质量得分最高。然后点击 [提交] 按钮即可执行变异体集合约简操作，待变异体集合约简完成便成功跳转结果展示页面。

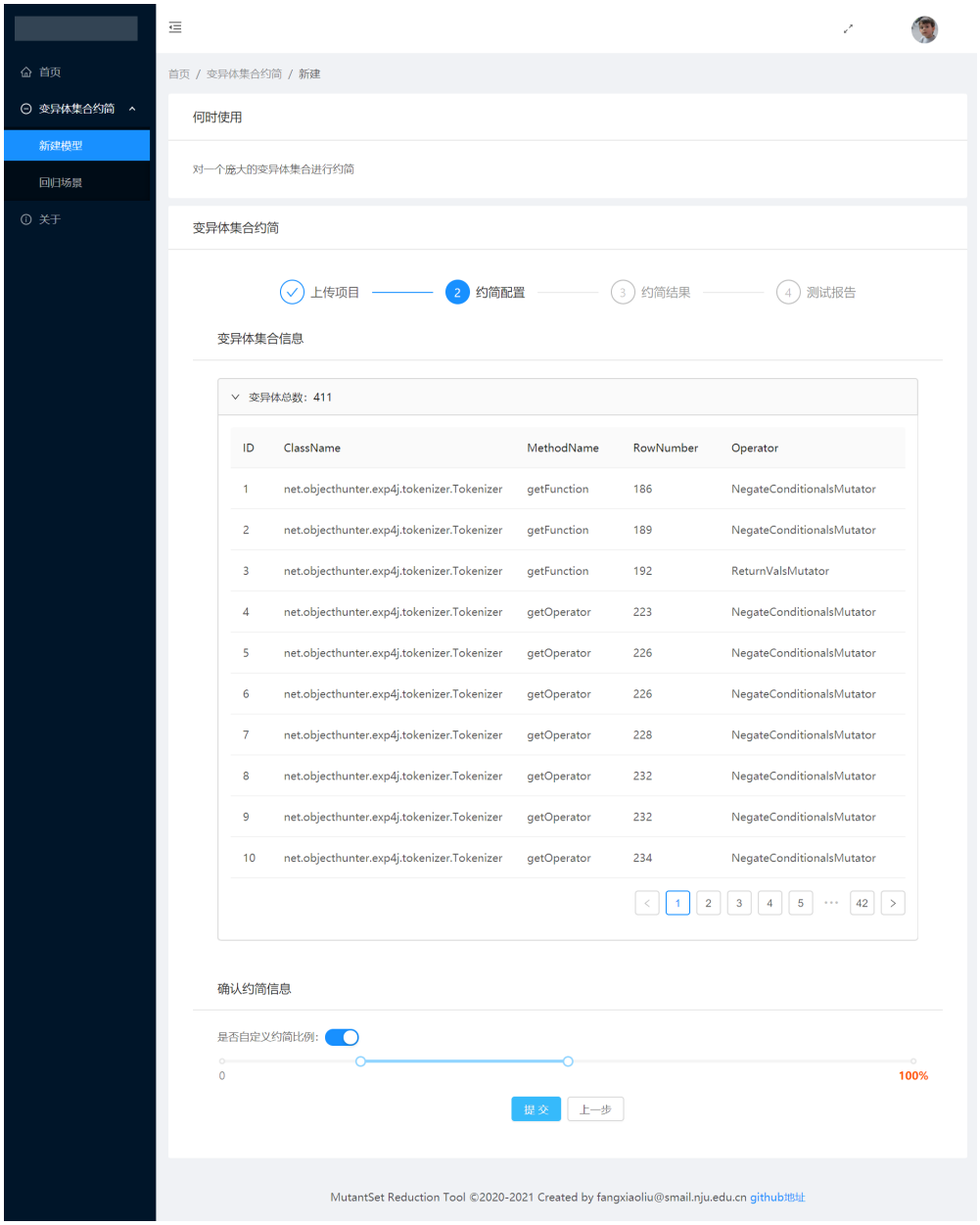


图 5-10: 约简配置页面展示图

结果展示（Results Show）页面如图5-11所示。该页面主要负责的功能为变异体集合的约简结果展示、预测模型保存以及变异体集合的约简结果下载。用户首先可以查看约简后的变异体集合的信息，并可以根据自己的需求选择执行后续操作。若点击 [查看测试报告] 按钮，系统自动跳转测试报告展示页面。若点击 [保存模型] 按钮，系统执行保存模型操作将模型存入数据库中，以供下次使用。若点击 [下载结果] 按钮，系统发送请求给服务端，服务端返回约简后的变异体子集，并导出给用户。

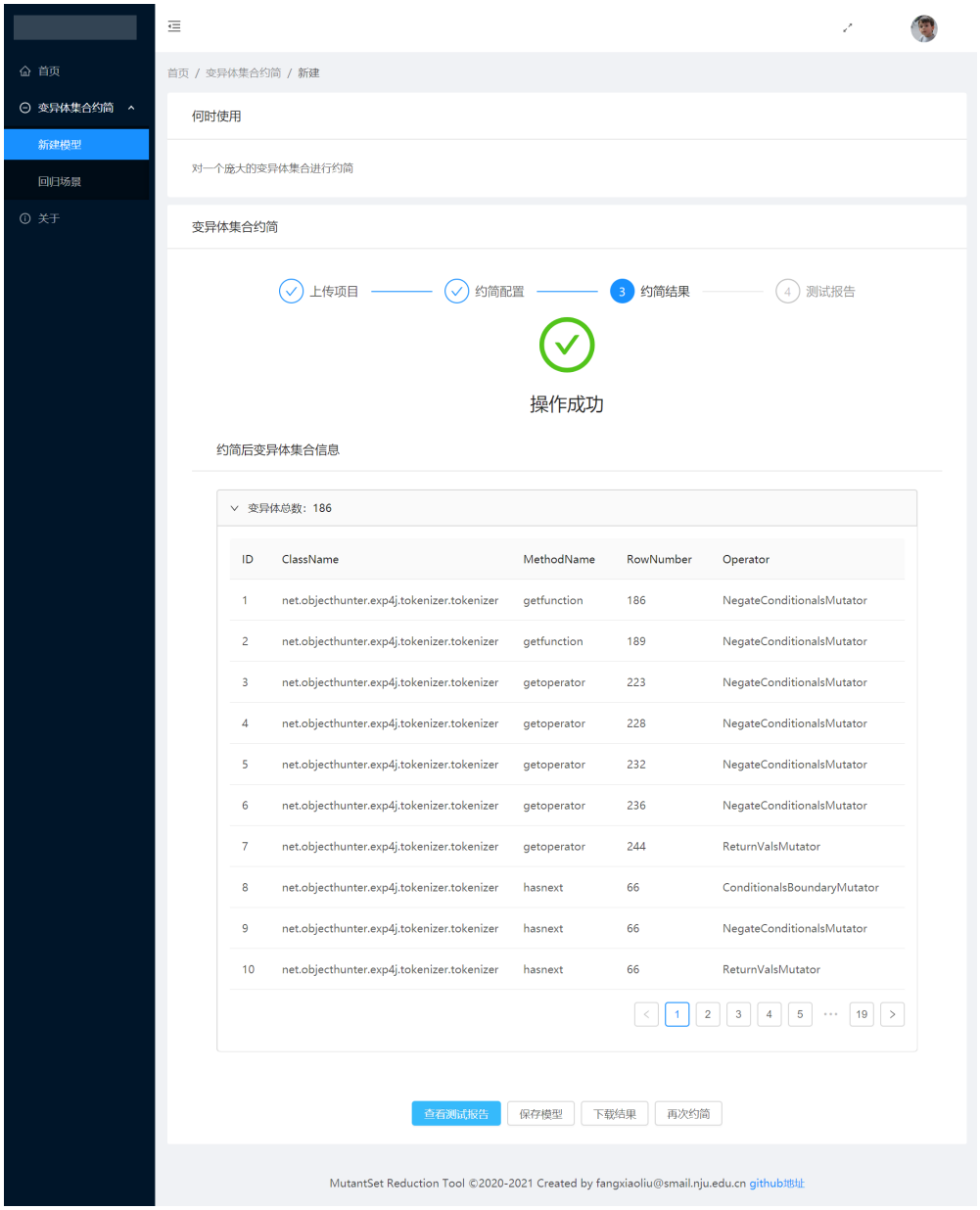
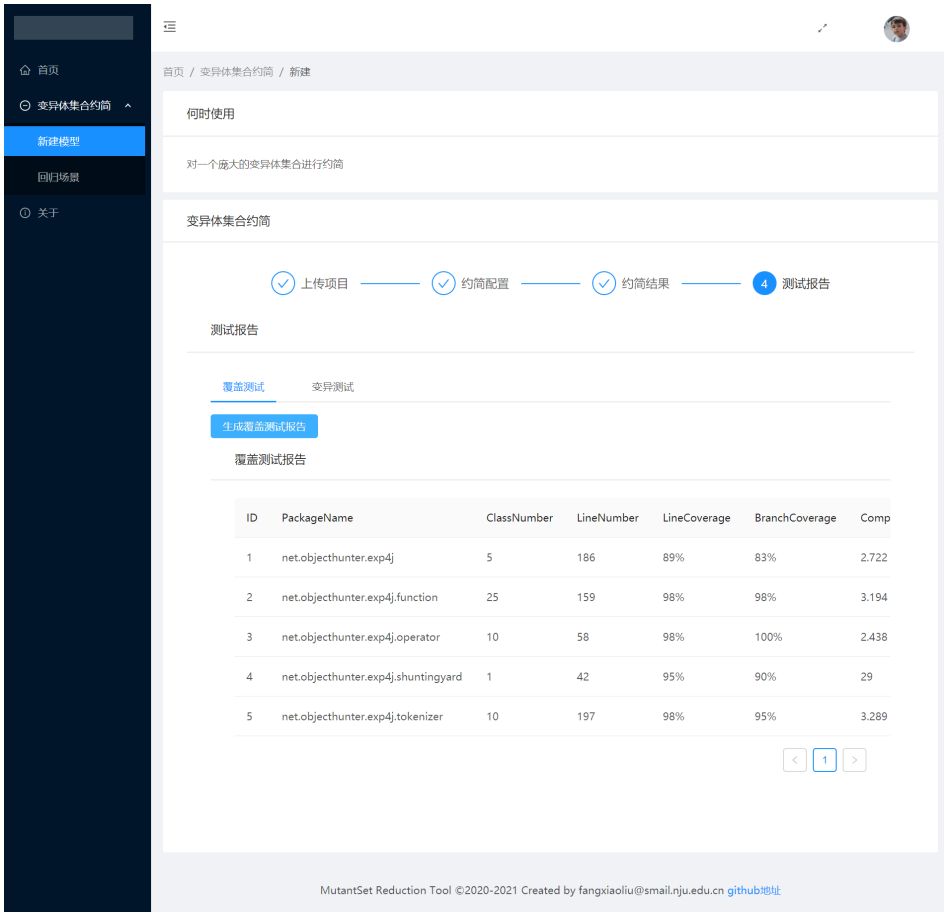
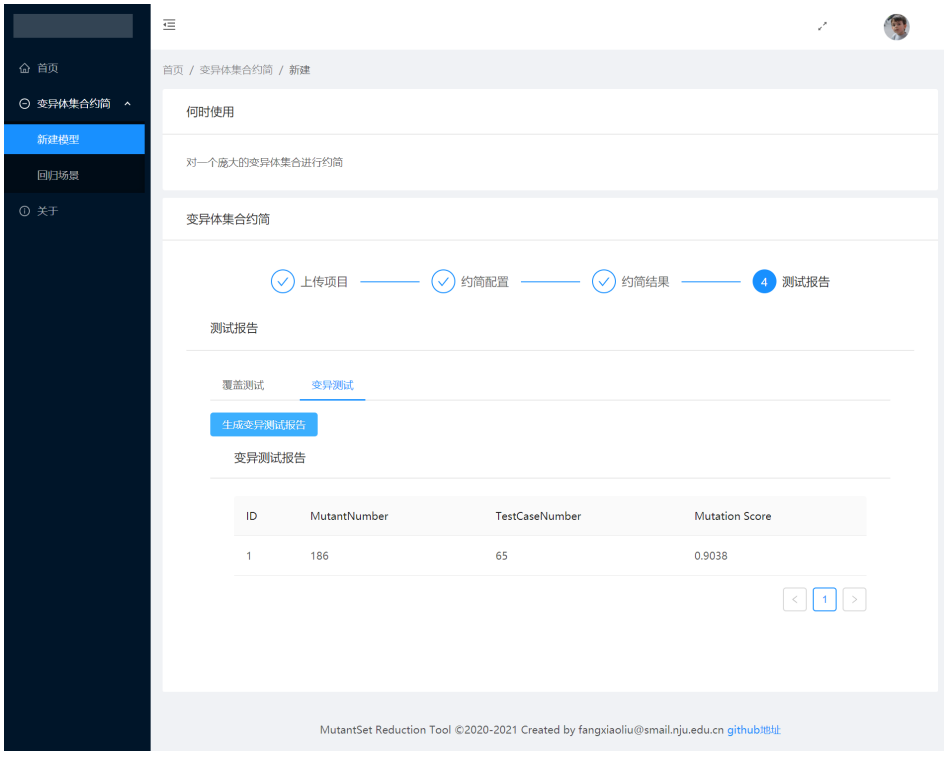


图 5-11: 结果展示页面展示图

测试报告展示（Test Report Show）页面如图 5-12所示。该页面主要负责的功能为测试报告生成。生成的测试报告以表格的格式展示给用户。用户点击覆盖测试标签页中的 [生成覆盖测试报告] 按钮, 如图 5-12(a)所示, 系统自动运行覆盖测试并将覆盖测试结果展示给用户。用户点击变异测试标签页中的 [生成变异测试报告] 按钮, 如图 5-12(b)所示, 系统自动运行变异测试并将变异测试结果展示给用户。



(a) 测试报告展示页面-覆盖测试报告



(b) 测试报告展示页面-变异测试报告

图 5-12: 测试报告展示页面展示图

5.2 系统测试

5.2.1 测试环境

表 5-1: 测试环境配置表

设备或软件	配置或版本信息
React	16.4.1
Webpack	3.8.1
Mobx	5.0.3
AntDesign	3.6.2
Axios	0.18.0
Python	3.6.5
Flask	1.0.2
Scikit-learn	0.19.1
JDK	1.8.0
Pitest	1.3.2
Cobertura	2.5.1
MongoDB	4.4
Docker	1.13.1
浏览器	Chrome87、Firefox71、IE9
硬件设备	操作系统 Windows 10
系统内存	16G
网络带宽	100M

针对系统的具体分析与设计，表5-1列举了系统部署和运行的所需的关键软件工具与硬件设备。在此基础上，对其开展功能测试、性能测试以及实证研究。系统的 API 接口使用 Python Flask 进行构建，为此需要提供 Python 运行环境；系统的前端项目使用 React 进行构建，通过 Webpack 进行打包。软硬件工具均使用官方提供的稳定版本。

5.2.2 系统测试设计

(1) 功能测试设计

功能测试也叫黑盒测试，从软件产品的界面、架构出发，按照第四章所述的需求编写测试用例，以验证系统各项功能。本系统的功能测试主要从用户注册登录、项目上传、约简配置、约简结果展示及测试报告展示五个方面展开。

表 5-2: 用户注册登录测试用例

测试 ID	TC1
测试名称	用户登录注册
测试功能	进行用户的注册和登录操作
测试步骤	1. 进入用户注册页面，输入新用户 ID、密码及确认密码进行注册； 2. 进入用户注册页面，输入已注册用户 ID、密码及确认密码进行注册； 3. 进入用户登录页面，输入已注册用户 ID、密码及验证码进行登录； 4. 进入用户登录页面，输入已注册用户 ID、错误密码及验证码进行登录； 5. 进入用户登录页面，输入新用户 ID、密码及验证码进行登录。
预期结果	1. 用户注册接口正常访问并返回注册成功标识，弹窗显示注册成功； 2. 用户注册接口正常访问并返回用户名已注册标识，弹窗显示已注册； 3. 用户登录接口正常访问并返回登录成功标识，弹窗显示登录成功； 4. 用户登录接口正常访问并返回密码错误标识，弹窗显示密码错误； 5. 用户登录接口正常访问并返回用户名不存在标识，弹窗显示用户不存在。

表 5-2展示了用户注册登录测试用例。用例分别模拟了不同情况下的用户注册和登录操作。在预期情况下，系统注册接口能正常访问并根据数据库对用户 ID 的查询结果返回相应的标识，前端跳出弹窗显示标识信息；系统登录接口能正常访问并根据数据库对用户 ID 及密码的查询结果返回相应的标识，前端跳出弹窗显示标识信息；

表 5-3展示了项目上传测试用例。用例分别模拟了在新建场景和回归场景下不同情况的项目上传操作。在预期情况下，文件上传接口能正常访问并根据上传的文件格式返回相应的标识；如果文件上传成功，特征提取接口能正常访问并根据项目解析结果返回相应的标识；如果文件上传失败，系统不会访问特征提取接口，前端直接跳出弹窗显示上传失败的提示信息。

表 5-3: 项目上传测试用例

测试 ID	TC2
测试名称	项目上传
测试功能	上传项目并进行变异体生成操作
测试步骤	1. 进入项目上传-新建页面，上传项目压缩包；等待项目上传成功后，点击 [变异生成] 按钮，进行变异生成； 2. 进入项目上传-回归页面，选择已有模型并上传项目压缩包；等待项目上传成功后，点击 [变异生成] 按钮，进行变异生成； 3. 进入项目上传-新建/回归页面，上传非压缩包文件；等待项目上传成功后，点击 [变异生成] 按钮，进行变异生成； 4. 进入项目上传-新建/回归页面，上传非 Java 项目的压缩包；等待项目上传成功后，点击 [变异生成] 按钮，进行变异生成； 5. 进入项目上传-新建/回归页面，未上传项目，直接点击 [变异生成] 按钮，进行变异生成。
预期结果	1. 文件上传接口能正常访问并返回上传成功标识; 特征提取接口正常访问, 返回变异生成成功标识，跳转变异体集合约简页面； 2. 约简模型查询接口能正常访问并返回模型列表；文件上传接口能正常访问并返回上传成功标识; 特征提取接口正常访问, 返回变异生成成功标识，跳转变异体集合约简页面； 3. 文件上传接口能正常访问并返回非压缩包格式标识； 4. 文件上传接口能正常访问并返回上传成功标识; 特征提取接口正常访问, 并返回非 Java 项目标识； 5. 弹窗提示：未上传项目。

表 5-4: 变异体集合约简测试用例

测试 ID	TC3
测试名称	变异体集合约简
测试功能	查看已生成的变异体并输入约简比例进行变异体集合约简操作
测试步骤	1. 变异体成功生成后自动跳转变异体集合约简页面，查看变异体信息，选择约简比例，点击 [提交] 按钮, 进行变异体集合约简； 2. 变异体成功生成后自动跳转变异体集合约简页面，查看变异体信息点击 [提交] 按钮, 进行变异体集合约简。
预期结果	1. 变异体结果展示接口能正常访问, 并返回全部变异体信息; 约简配置接口能正常访问，服务端按照前端传入的约简比例进行约简，并返回约简成功标识; 2. 变异体结果展示接口能正常访问，并返回变异体全部信息; 约简配置接口能正常访问，服务端按照最优的约简比例进行约简，并返回约简成功标识。

表 5-4展示了变异体集合约简测试用例。用例分别模拟了不同情况下的变异体集合约简操作。在预期情况下，变异体集合约简接口能正常访问并根据用户选择的约简比例进行相应的约简操作。

表 5-5: 约简结果展示测试用例

测试 ID	TC4
测试名称	约简结果展示
测试功能	查看约简后的变异体信息，进行约简模型保存和约简结果下载操作
测试步骤	1. 变异体集合约简后自动跳转约简结果展示页面，查看约简后变异体； 2. 变异体集合约简后自动跳转约简结果展示页面，点击 [模型保存] 按钮, 进行约简模型保存操作； 3. 变异体集合约简后自动跳转约简结果展示页面，点击 [结果下载] 按钮, 进行约简结果下载操作； 4. 变异体集合约简后自动跳转约简结果展示页面，点击 [查看测试报告] 按钮, 查看测试报告。
预期结果	1. 变异体结果展示接口能正常访问，并返回约简后变异体信息; 2. 约简模型保存接口能正常访问，并返回保存成功标识; 3. 约简结果下载接口能正常访问，并返回约简结果的 csv 文件; 4. 跳转测试报告展示页面。

表 5-5展示了约简结果展示测试用例。用例分别模拟了约简结果展示操作、约简模型保存和约简结果下载操作。在预期情况下，变异体结果展示接口、约简模型保存接口及约简结果下载接口均能正常访问并返回相应的结果。

表 5-6: 测试报告展示测试用例

测试 ID	TC5
测试名称	测试报告展示
测试功能	查看覆盖测试报告和变异测试报告
测试步骤	1. 点击 [查看测试报告] 按钮，自动跳转测试报告展示页面，点击 [生成覆盖测试报告] 按钮，查看覆盖测试报告； 2. 点击 [查看测试报告] 按钮，自动跳转测试报告展示页面，点击 [生成变异测试报告] 按钮，查看变异测试报告。
预期结果	1. 测试报告展示接口能正常访问，并返回覆盖测试结果； 2. 测试报告展示接口能正常访问，并返回变异测试结果。

表 5-6展示了测试报告展示测试用例。用例分别模拟了覆盖测试报告生成操作和变异报告生成操作。在预期情况下，测试报告展示接口能正常访问并返回相应的结果。

(2) 性能测试设计

本文性能测试包括页面性能测试及接口性能测试。页面性能测试采用 Chrome Devtools[①] Web 开发者工具完成测试。接口性能测试通过 Apache JMeter[②]性能测试工具测试。

1）页面性能测试设计

页面性能测试是一种针对页面性能优化的性能测试。通过对页面的性能测试，可以发现页面存在的性能问题，并根据性能测试结果进行页面优化，提高页面的加载性能，从而提高系统的整体性能。本文使用 Chrome DevTools 进行页面性能测试，其性能面板可以记录和分析网站在运行时的所有活动，其中包括在整个页面加载过程中网络通信和 HTML 解析时间、JavaScript 的执行时间、渲染时间、绘制时间、系统消耗时间和空闲时间。本文记录用户打开变异体集合约简页面后直至操作完成的性能数据，以查看页面性能测试结果。

2）接口性能测试设计

表 5-7: 业务层接口

接口名称	接口作用
用户注册	根据用户 ID 及密码对用户进行注册操作
用户登录	根据用户 ID 及密码对用户进行登录操作
文件上传	接收前端传入的项目并对项目进行保存操作
特征提取	根据项目 ID 解析项目并对项目进行变异生成和特征提取操作
变异体集合约简	根据项目 ID 对项目进行变异体集合约简操作
约简模型保存	根据项目 ID 和用户 ID 对约简模型进行保存操作
约简模型查询	根据用户 ID 对用户已保存的约简模型进行查询操作
变异结果展示	根据项目 ID 获取项目的变异约简结果
测试报告展示	根据项目 ID 获取项目的测试报告
约简结果下载	根据项目 ID 获取项目的变异约简结果并以 csv 文件格式保存

接口性能测试指通过模拟生产运行的业务压力或用户使用场景来测试系统的性能指标是否满足性能需求要求的测试活动。本文使用 Apache JMeter 进行接口性能测试，其支持用户创建线程访问接口地址，并查看系统接口响应时间。首先创建线程组，它的每一个线程都可以看作虚拟用户。接着在线程组中添加 HTTP 请求，地址为服务器提供的接口地址。然后在线程组中设置 100 个

[①]Chrome Devtools. https://developers.google.com/web/tools/chrome-devtools
[②]Apache JMeter. https://jmeter.apache.org/

用户，在 2 秒内向系统同时发送请求，并循环请求 2 次。最后添加测试结果的聚合报告。表 5-7给出了测试的业务层接口，该表记录了变异体集合约简系统业务层接口的详细信息，包括接口名称和接口作用。

5.2.3 测试结果与分析

(1) 功能测试结果与分析

表 5-8: 功能测试用例执行结果

测试用例 ID	用例描述	测试结果
TC1	用户登录注册	通过
TC2	项目上传	通过
TC3	变异体集合约简	通过
TC4	约简结果展示	通过
TC5	测试报告展示	通过

表 5-8给出了功能测试的执行结果。根据功能测试设计阶段设计的测试用例，本文严格按照测试用例描述步骤，依次执行所有测试用例。根据表中的数据可以发现，所有用例的执行结果全部通过。系统完成了需求分析中提出的系统功能需求，表明系统完全符合产品功能设计。

(2) 性能测试结果与分析

1）页面性能测试结果

图 5-13为用户打开变异体集合约简页面后直至操作完成的性能数据，在整个页面加载过程中，JavaScript 的执行时间为 120 秒，浏览器渲染时间为 39 秒，绘制时间为 4 秒。页面从开始约简到展示测试报告，整个流程耗时 463 秒，符合正常预期。整个操作过程，页面的每帧绘制得比较流畅，用户不会感觉到明显的卡顿。

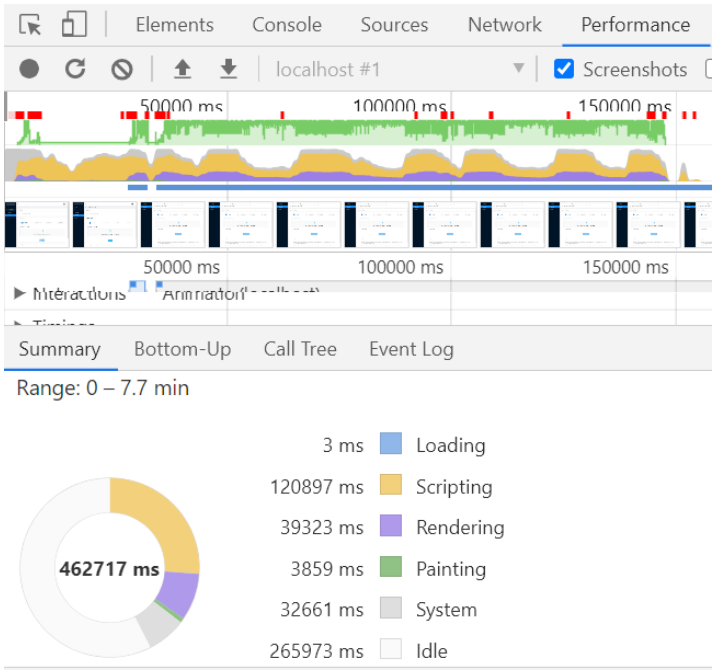


图 5-13: 页面性能测试结果

2）接口性能测试结果

表 5-9: Apache JMeter 测试结果

接口名称	样本	平均值	中位数	异常%	吞吐量
用户注册	200	3,202ms	3,502ms	0.00%	28.8/sec
用户登录	200	5,124ms	5,222ms	0.00%	15.4/sec
文件上传	200	773ms	503ms	0.00%	22.6/sec
特征提取	200	95,146ms	116,679ms	0.00%	0.03/sec
变异体集合约简	200	41,063ms	48,654ms	0.00%	2.3/sec
约简模型保存	200	3,461ms	3,880ms	0.00%	27.6/sec
约简模型查询	200	5,190ms	5,254ms	0.00%	12.7/sec
变异结果展示	200	81ms	84ms	0.00%	94.1/sec
测试报告展示	200	96ms	87ms	0.00%	100.3/sec
约简结果下载	200	327ms	362ms	0.00%	82.9/sec

Apache JMeter 测试结果如表 5-9所示。由表可知，系统接口正确处理了 200 次请求，由于每个接口负责的功能不同，所以接口的响应时间相差巨大，其中特征提取接口的响应时间最长，因为特征提取需要调用 Pitest 插件、Cobertura 插件和 JHawk 工具进行相应特征的提取，所以耗时过高。为了提升用户体验，系统使用 loading 图标提示用户需要耐心等待。系统所有接口功能正常，在高并发情况下响应时间较短。

5.3 本章小结

本章介绍了变异体集合约简系统的实现细节和系统测试。本章将系统分为变异体生成、质量模型构建、变异体集合约简和测试报告生成四个模块。首先针对每个模块给出了模块的设计与实现，并展示系统主要页面的运行截图。然后进行了系统测试设计并对测试结果进行了分析。

第六章　实验与评估

本章节主要评估本文所提出的基于机器学习的变异体集合质量预测方法及基于遗传算法的变异体集合约简方法的有效性。首先介绍研究问题、实验对象以及评估指标，然后针对三个研究问题设计了三个实验，分别给出了详细的实验步骤并对实验结果进行分析，最后讨论效度分析。

6.1　研究问题

本节旨在回答以下研究问题：

- **RQ1：基于机器学习的变异体集合质量预测方法在准确性和效率方面表现如何？** 高质量的变异体集合用于充分衡量测试用例集的完备性，变异体集合质量评估是变异体集合约简的重要环节，其准确性对于变异体集合约简方法至关重要。本实验通过计算预测模型的均方根误差、平均绝对误差、拟合优度和运算时间，来验证 MSQP 是否能够准确高效地预测质量得分。
- **RQ2：变异顽固程度特征和变异充分程度特征对基于机器学习的变异体集合质量预测模型的贡献如何？** MSQP 是基于一组特征构建的回归预测模型，这些特征可能对回归模型有不同的贡献。本实验将研究这些特征的作用，以便更好地了解 MSQP。即通过比较仅使用某一类别特征（变异顽固特征或者变异充分特征）构建的预测模型的拟合优度，用来评价变异顽固程度和变异充分程度对变异体集合质量预测模型的贡献。
- **RQ3：基于遗传算法的变异体集合约简方法在准确性和效率方面表现如何？** 基于遗传算法的变异体集合约简方法的主要目的是在减少变异体数量的同时, 保证变异体子集的测试效果。本实验通过对比随机选择方法和本方法的变异体约简比例、测试用例减少比例、约简后变异体子集的质量得分及运算时间，来验证本文所提的变异体集合约简方法是否能够准确高效地进行变异体集合约简。

6.2 实验对象

表 6-1: 实验对象

名称	版本号	规模	测试用例	语句覆盖率	非等价变异体
joda	v2.10.8	14,956	4,238	90.37%	7,753
lafj	v0.4.0	2,943	245	63%	1,527
lang	v2.1.1	12,372	2,362	97.16%	8,748
msg	v0.6.4	4,720	973	69%	1,666
exp4j	v0.4.8	642	311	95%	341
text	v1.0	3,097	476	96%	1,880
io	v2.5	5,182	1,032	88.50%	2,433
linq4j	v0.4	6,133	221	48%	1,265
collections	v4.0	11,555	13,677	85%	2,392

本文以先前的软件测试研究中广泛使用的 9 个项目 [17, 52] 作为实证研究的实验对象。

- Joda-Time(joda) 是 Java SE 8 之前的 Java 日期和时间类的广泛替代品。
- Linear Algebra for (la4j) 是一个提供了线性代数原语和算法的开源库。
- Apache Commons Lang(lang)，是 Java 实用程序类的软件包。
- MessagePack(msg) 是一种有效的二进制序列化格式, 就像 JSON 轻量快速。
- Mathematical Expression Evaluat(exp4j) 是 Java 的数学表达式评估器。
- Apache Commons Text(text) 是一个专注于处理字符串的算法的库。
- Apache Commons IO(io) 是一个包含实用程序类，流实现，文件过滤器，文件比较器，字节序转换类等开源库。
- A port of LINQ to Java(linq4j) 是一个基于 Java 的语言集成查询窗口。
- Apache Commons Collections(collections) 是一个包含扩展和增强 Java Collections Framework 类型的软件包。

表 6-1展示了 9 个实验对象的基本信息，第一列列出每个实验对象的名称，第二列列出每个实验对象的版本号，第三列列出每个实验对象删除空白语句和注释后的可执行代码行数，第四列列出每个实验对象包含的测试用例数，第五列列出每个实验对象中测试用例集的语句覆盖率，第六列列出每个实验对象的非等价变异体总数。

6.3 评价指标

本节介绍实验中使用到的评价指标。针对 RQ1 变异体集合质量得分的准确性和高效性，本文使用均方根误差、平均绝对误差、拟合优度和预测时间作为评价指标。针对 RQ2 不同类别特征对于预测模型的贡献，本文使用拟合优度作为评价指标。针对 RQ3 变异体集合约简的准确性和高效性，本文使用变异体约简比例、测试用例减少比例、约简后变异体子集的质量得分和约简方法运行时间作为评价指标。评价指标细节如下：

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^*)^2} \tag{6-1}$$

均方根误差（RMSE）[53] 用于测量预测值与实际值之间的偏差，可以很好地反映测量的精度。均方根误差越小，预测的准确性越高，反之亦然。公式 6-1给出了均方根误差的计算方法。其中，y_i 为第 i 个变异体集合的真实质量得分，y_i^* 为第 i 个变异体集合的预测质量得分，$\bar{y}$ 为变异体集合的平均质量得分，n 为预测样本的个数。

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - y_i^*\right| \tag{6-2}$$

平均绝对误差（MAE）[53] 表示预测值和观察值之间的绝对误差的平均值。平均绝对误差越小，预测的准确性越高，反之亦然。公式 6-2给出了平均绝对误差的计算方法。

$$R2 = 1 - \frac{\sum_i (y_i^* - y_i)^2}{\sum_i (\bar{y} - y_i)^2} \tag{6-3}$$

拟合优度（R2）[53] 表示预测值和观察值之间的拟合优度，来判断模型的拟合效果，其取值范围为 [0,1]。拟合优度越接近 1，模型拟合效果越好，反之亦然。公式 6-3给出了拟合优度的计算方法。

预测时间 [17] 是指变异体集合质量得分预测的时间，单位为秒。该指标可以衡量变异体集合质量预测模型的计算开销。预测时间的越长，其计算开销越大，效率越低，反之亦然。

$$MRR = 1 - \frac{|MUT_{sub}|}{|MUT_{ne}|} \tag{6-4}$$

变异体约简比例（MRR）[14] 是指被约简的变异体数量（$|MUT_{ne}| - |MUT_{sub}|$）占所有非等价变异体数量（$|MUT_{ne}|$）的比例。变异体约简比例越大，表明约简的变异体数量越多，约简方法效果越显著，反之亦然。公式6-4给出了变异体约简比例的计算公式。

$$TRR = 1 - \frac{\left|TS_{complete}^{MUT_{sub}}\right|}{\left|TS_{complete}\right|} \tag{6-5}$$

测试用例减少比例（TRR）是指对于约简后变异体子集的测试用例集相对于测试用例池减少的数量（$\left|TS_{complete}\right| - \left|TS_{complete}^{MUT_{sub}}\right|$）占测试用例池数量（$\left|TS_{complete}\right|$）的比例。值越大，减少的测试用例数量越多，约简方法效果越显著，反之亦然。公式6-5给出了测试用例减少比例的计算公式。

$$QS(MUT_{sub}) = \frac{\left|MUT_{killed}^{TS_{sub}}\right|}{|MUT_{ne}|} \tag{6-6}$$

约简后变异体子集的质量得分（$QS(MUT_{sub})$）[14] 是指在约简后的变异体子集上扩充的测试用例集 TS_{sub} 在全部非等价变异体集合 MUT_{ne} 上的变异得分 $MS(MUT_{ne}, TS_{sub})$。值越大，变异体子集的测试效果越好，约简方法效果越显著，反之亦然。公式6-6给出了变异体子集的质量得分的计算公式。其中，$\left|MUT_{killed}^{TS_{sub}}\right|$ 表示由变异体子集扩充的测试用例集杀死的变异体数量，$|MUT_{ne}|$ 为所有非等价变异体数量。

运算时间 [54] 是指变异体集合约简的 CPU 运行时间，单位为秒。该指标可以衡量变异体集合约简的计算开销。运行时间越长，其计算开销越大，效率越低，反之亦然。

6.4 实验设计

本节针对上述三个研究问题进行实验设计，具体设计如下：对于每一个实验对象，依次执行下列步骤。

（1） **测试用例池构建**。本文采用测试用例自动生成工具 Evosuite[①]扩充所测试实验对象的测试用例集，扩充目标是满足尽可能高的语句覆盖率、分支覆盖率以及变异得分。若 Evosuite 无法达成扩充目标，则进行人工扩充, 最终生成测试用例池 $TS_{complete}$。

（2） **获取非等价变异体**。首先，本文使用 Pitest 的全部 18 个变异算子来生成整个变异体集。然后，对全部变异体运行所有测试用例，并获得一组可以被这些测试用例杀死的变异体。参考先前工作 [5, 55, 56] 识别等价变异体的步骤，在实验中，将不能被任何测试用例杀死的变异体作为等价变异体。也就是说，将可以被测试用例杀死的变异体视为非等价变异体 MUT_{ne}。

（3） **训练集构建**。本文参考变异约简论文 [3] 中的约简比例构造训练集。首先，设置约简比例数组 $ReduceRatios = [10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%]$。其次，依次从 $ReduceRatios$ 中取值，设为 x, 从全部非等价变异体 MUT_{ne} 中随机选取 x 个变异体构建变异体子集。为保证训练集分布均匀，执行上一步骤 n 次，使具有不同约简比例子集的训练样本数量相同，从而构建了 $n * |ReduceRatios|$ 个变异体子集, 其中，为了确保训练模型的准确性，将训练集的大小设置成非等价变异体数量大小相同，即 n 为实验对象的非等价变异体数量与约简比例数组的商。然后，遍历每个变异体子集 MUT^i_{sub}，从 MUT^i_{sub} 中提取第 3 章提到的相关特征构成特征向量，并从 $TS_{complete}$ 中选择 MUT^i_{sub} 的测试用例集，即所选的测试用例集在变异子集上的变异得分接近 1。再计算所选测试用例集在全部变异体中的变异得分构成标记值。最终获得 $n * |ReduceRatios|$ 个样本，构成训练集 $TrainingSet$。

（4） **模型训练与评估**。本步骤主要用来回答研究问题一，通过计算变异体集合质量预测模型的均方根误差、平均绝对误差、拟合优度和预测时间来评估 MSQP 的准确性和效率。首先，将 $TrainingSet$ 依次输入到给定的预测算法中，使用网格搜索法 [57] 自动找到预测模型的最佳参数，并选择最高分数的参数作为预测模型的最终参数，训练预测模型 $MSQPModel$。然后，采用 5 折交叉验证法 [17] 评估所有预测模型的有效性，并选择最优的预测模型。

[①]Evosuite. https://www.evosuite.org/

其中将验证指标设置成均方根误差、平均绝对误差和拟合优度。最后，计算在 5 次运行中验证指标的平均值，以评估 $MSQPModel$ 的准确性；计算预测模型的预测时间，以评估 $MSQPModel$ 的效率。

（5） **特征贡献评价**。本步骤主要用来回答研究问题二，通过比较仅使用某一类别特征（即变异顽固程度和变异充分程度）构建的预测模型的拟合优度，用来评价变异顽固程度和变异充分程度对变异体集合质量预测模型的贡献。首先，选取所有变异顽固程度的特征构建特征向量，并依次执行步骤 4，记录使用变异顽固程度特征构建的 $MSQPModel$ 的拟合优度 $R2_{stubborn}$（拟合优度用于评价模型的拟合效果，值越接近 1，拟合效果越好，因此使用拟合优度来评价特征的贡献）。然后，选取所有变异充分程度的特征构建特征向量，并依次执行步骤 4，记录使用变异顽固程度特征构建的 $MSQPModel$ 的拟合优度 $R2_{abundant}$。最后，比较 $R2_{stubborn}$ 和 $R2_{abundant}$，以评价变异顽固程度和变异充分程度对 $MSQPModel$ 的贡献。

（6） **基于遗传算法的变异体集合约简**。本文首先执行基于遗传算法的变异体集合约简方法，并使用网格搜索法 [57] 自动寻找最佳的迭代次数。其次，收集约简前后的变异体集合数量，计算变异约简比例 MRR_{GAR}。再次，在步骤 1 生成的 $TS_{complete}$ 中选择测试用例构建约简后的变异体子集的测试用例集 $TS_{complete}^{MUT_{sub}}$。然后计算 $TS_{complete}^{MUT_{sub}}$ 在全部变异体集合上的变异得分 QS_{GAR} 和测试用例减少比例 TRR_{GAR}。最后，记录执行约简方法所消耗的时间 $Time_{GAR}$。

（7） **基于随机选择的变异体集合约简**。张路研究表明，随机变异选择方法优于变异算子选择方法 [5]；而且在随机变异选择中，每个变异体的选择概率均等，而在变异算子选择中，变异体是基于变异算子选择的，因此使用随机方法选择没有偏见。综上本文采用随机变异选择方法作为对照方法。首先，根据 MRR_{GAR} 执行基于随机选择的变异体集合约简方法 [1]，记录变异约简比例 MRR_{RMR}。其次，在步骤 1 生成的 $TS_{complete}$ 中选择测试用例构建约简后的变异体子集的测试用例集 $TS_{complete}^{MUT_{sub}}$。然后计算 $TS_{complete}^{MUT_{sub}}$ 在全部变异体集合上的变异得分 QS_{RMR} 和测试用例减少比例 TRR_{RMR}。最后，记录执行约简方法所消耗的时间 $Time_{RMR}$。

（8） **评价指标比较**。本步骤主要用来回答研究问题三，通过对比随机选择方法和本方法的变异体约简比例、约简后变异体子集的质量得分及运算时间，来评估本文所提约简方法的准确性。首先，为了减少意外结果发生，分别

执行 m 次步骤 6 和步骤 7，收集 m 次遗传算法和随机选择方法的变异体约简比例、测试用例减少比例、约简后变异体子集的质量得分及运算时间的平均值。本文参考 [5], 将 m 的值设置为 50，该值足以避免意外结果。然后比较收集到的平均指标，以评估本文所提约简方法的准确性和效率。

6.5 实验结果分析

（1）RQ1：基于机器学习的变异体集合质量预测方法在准确性和效率方面表现如何？

在本节将对 9 个实验对象依次采用岭回归算法、支持向量机回归算法、最近邻回归算法以及随机森林算法进行变异体集合质量得分模型的构建，选择最优预测模型并评估最优预测模型的性能。本文通过计算最优预测模型的均方根误差、平均绝对误差、拟合优度和运算时间，评价 MSQP 的有效性和效率。

表 6-2: RQ1 机器学习模型对比结果

名称	岭回归	支持向量机回归	最近邻回归	随机森林
joda	0.9896	-0.083	0.5185	**0.9987**
lafj	0.8937	-0.9640	0.3406	**0.9642**
lang	0.9625	-0.1627	0.4025	**0.9983**
msg	0.9303	-0.8553	0.2936	**0.9722**
exp4j	0.9451	-0.956	0.4030	**0.9591**
text	0.9365	-0.2635	0.4962	**0.9878**
io	0.9641	-0.1877	0.3762	**0.9946**
linq4j	0.9111	-0.6001	0.5891	**0.9744**
collections	0.9893	-0.0441	0.4648	**0.9975**

表6-2中总结了分别采用岭回归、支持向量机回归、最近邻回归以及随机森林四种机器学习回归算法在不同实验对象上构建的预测模型的拟合优度。比较这四种模型的实验结果可以看出，采用随机森林算法构建的预测模型的拟合程度要优于其余三种回归算法构建的模型，原因在于随机森林属于集成学习算法，其本身不是单独的机器学习算法。相反，它构建并组合了多个机器学习器以完成学习任务，因此随机森林的学习能力更强。支持向量机模型表现最差，甚至出现了负值，说明拟合结果比平均值还差，因此表明支持向量机回归模型

不适合预测集合的质量得分。综上所述，本文采用随机森林构建变异体集合质量预测模型。

表 6-3: RQ1 随机森林预测模型的实验结果

名称	拟合优度	均方根误差	平均绝对误差	耗时
joda	0.9987	0.0050	0.0033	1,371s
lafj	0.9642	0.0116	0.0072	205s
lang	0.9983	0.0049	0.0032	810s
msg	0.9722	0.0119	0.0067	488s
exp4j	0.9591	0.0277	0.0187	92s
text	0.9878	0.0106	0.0067	343s
io	0.9946	0.0091	0.0064	1,210s
linq4j	0.9744	0.0135	0.0081	415s
collections	0.9975	0.0086	0.0064	1,083s

表6-3中总结了由不同实验对象构建的最优预测模型的实验结果，即在测试阶段，预测模型的拟合优度、均方根误差、平均绝对误差和构建预测模型所消耗的时间。从表中的数据可以得出，预测模型的拟合优度均在 0.95 以上并且均方根误差和平均绝对误差较低，其中，joda 项目、lang 项目、io 项目以及 collections 项目的拟合优度超过了 0.99。上述观察表明，MSQP 具有较好的稳定性并且准确率高。通过分析最后一列发现，每个项目的耗时相差巨大，其原因在于该项指标包含特征收集时间以及模型构建时间。由于每个项目的规模、复杂度不同，所以解析每个项目所消耗的时间也有所不同。例如 joda 项目共有 14,956 行非注释代码，exp4j 项目共有 642 行非注释代码，两者相差巨大，所以解析这两个项目的耗时也会相差巨大。由于质量预测模型构建一次，可重复使用多次，即使存在 23 分钟的执行时间，也在可接收范围内。由此分析可知，MSQP 在效率方面表现较优。

综上所述表明：MSQP 在有效性和效率方面都表现优秀，具有较高的性能，可作为遗传算法中的适应度评价模型对变异体集合进行约简。

（2）RQ2：变异顽固程度特征和变异充分程度特征对基于机器学习的变异体集合质量预测模型的贡献如何？

在本节将比较变异顽固程度特征和变异充分程度特征对构建基于机器学习的变异体集合质量预测模型的贡献。为了调查每个类别特征的贡献，本文比较

了仅使用某一类别特征（变异顽固特征或者变异充分特征）构建的预测模型的拟合优度（有关这两类特征的更多详细信息，请参见第 3.1 节）。

如图 6-1为在不同实验对象下每个类别对预测模型的贡献结果。x 轴为实验对象名称，y 轴为拟合优度，蓝色图标代表仅用变异顽固程度特征进行预测时的拟合优度值，黄色代表仅用变异充分程度特征进行预测时的拟合优度值，灰色代表使用全部特征（变异顽固特征或变异充分特征）进行预测时的拟合优度值。从图中可以看出，综合 9 个实验对象的实验结果，仅使用变异顽固程度特征和仅使用变异充分程度特征构建回归预测模型，模型的拟合程度相差无几且都具有较高的拟合优度，但是会略低于使用全部特征构建的模型的拟合优度。然而也会出现噪音数据，例如，在构建 joda 项目的预测模型时，变异顽固程度特征的贡献值明显低于变异充分程度的贡献值，原因在于特征通常与实验对象本身密切相关，可能此类特征不适合 joda 项目。

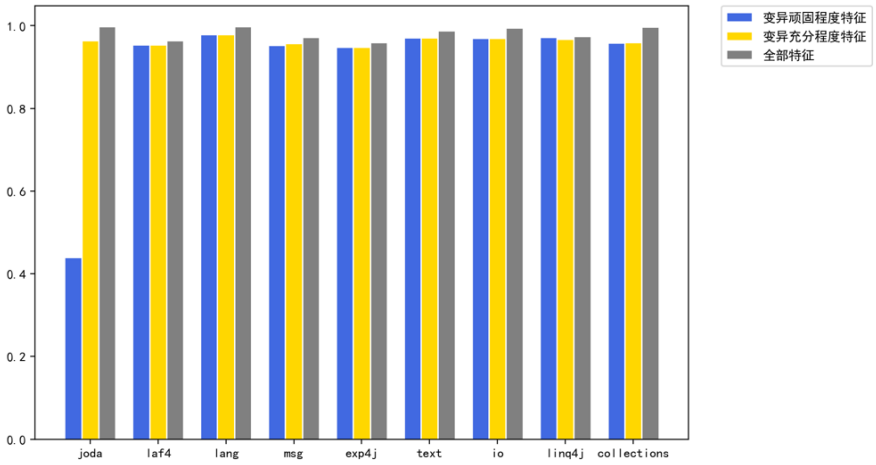


图 6-1: RQ2 比较两类特征的贡献

上述观察表明：每一类别的特征对于构建变异体集合质量预测模型都发挥了关键性的作用，并且缺一不可。

（3）RQ3：基于遗传算法的变异体集合约简方法在准确性和效率方面表现如何？

在本节将对 9 个实验对象分别采用随机选择方法 *RMR* 和遗传算法 *GAR* 进行变异体集合约简，并评估基于遗传算法的变异体集合约简方法的性能。本文通过比较 *RMR* 和 *GAR* 的变异体约简比例、测试用例减少比例、约简后变异体子集的质量得分及运算时间，从有效性和效率方面评价 *GAR* 的性能。

表 6-4: RQ3 实验结果-GAR

名称	变异体约简比例	测试用例减少比例	质量得分	约简时间
joda	39.82%	20.60%	0.9313	104s
lafj	40.76%	9.93%	0.9902	60s
lang	40.36%	16.90%	0.9675	123s
msg	41.86%	9.09%	0.9814	61s
exp4j	39.87%	20.88%	0.9359	35s
text	39.44%	10.94%	0.9761	51s
io	40.30%	18.03%	0.9499	58s
linq4j	39.79%	8.88%	0.9850	41s
collections	40.19%	20.44%	0.9193	49s

表 6-5: RQ3 实验结果-RMR

名称	变异体约简比例	测试用例减少比例	质量得分	约简时间
joda	39.82%	19.86%	0.9292	2,409s
lafj	40.76%	4.26%	0.9802	483s
lang	40.36%	10.59%	0.9646	2,532s
msg	41.86%	8.02%	0.9722	3,219s
exp4j	39.87%	16.48%	0.8910	220s
text	39.44%	7.55%	0.9695	758s
io	40.30%	12.64%	0.9462	1,757s
linq4j	39.79%	7.70%	0.9720	596s
collections	40.19%	19.56%	0.8982	961s

表6-4和表6-5分别总结了在不同实验对象下随机选择方法和遗传算法的实验结果，即变异体约简比例、测试用例减少比例、约简后变异体集合的质量得分和约简时间。对比两张表中的数据可以得出，*GAR* 的变异体约简比例、测试用例减少比例、质量得分和约简时间都优于 *RMR*，其中约简时间的变化最为显著。为了清晰地比较二者的差别，本文对 *GAR* 和 *RMR* 的约简时间进行单独分析。如图6-2所示，x 轴为实验对象名称，y 轴为约简时间，暗绿色图标代表 *RMR* 所消耗的约简时间，嫩绿色图标代表 *GAR* 所消耗的约简时间。从图中可以看出，项目的规模越大，约简时间的变化越显著，说明本方法可以有效降低大型项目的约简时间。综合 9 个实验对象的实验结果，*RMR* 是一个极为耗时的

方法，*RMR* 消耗的时间平均约为 *GAR* 的 21.13 倍，由此可见 *GAR* 的效率要明显优于 *RMR*，可以进行高效的变异体集合约简操作。

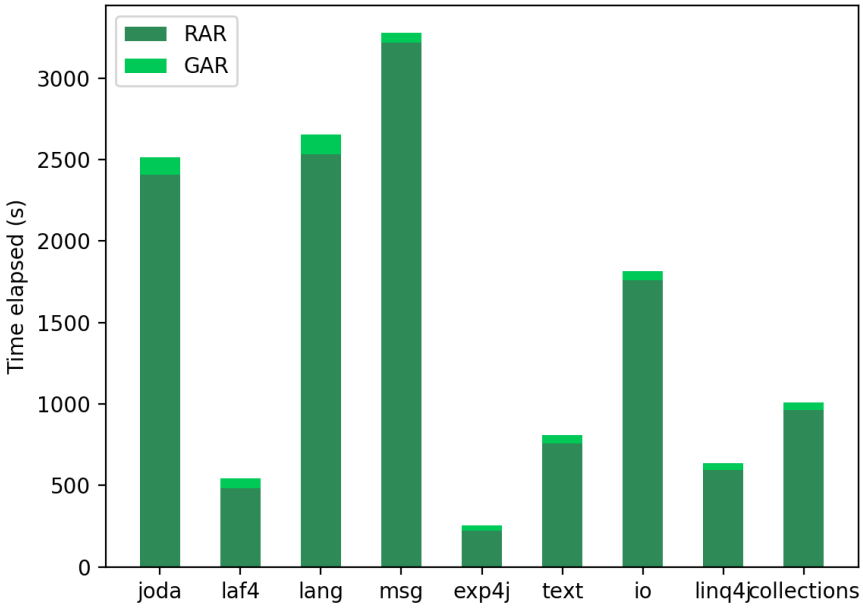


图 6-2: RQ3 比较两种方法的耗时

上述观察表明：*GAR* 在有效性和效率方面都表现优秀，具有较高的性能，并且在确保约简后变异体子集测试效果的同时，显著减少了约简时间，达到了降低变异测试计算开销的目的。

6.6 效度分析

内部效度分析。内部效度指因变量和自变量之间关系的确定性程度，代表实验结论的真实性。本文实验存在一种内部效度威胁，即本文方法是否可以正确执行。为了降低此类威胁，本文重用了广泛的库和框架，例如 JHawk、Pitest、ASM、Cobertura、Sklearn 来帮助实现本文方法。

外部效度分析。外部效度是指研究结果、变量条件、时间和背景的代表性和普遍适用性。本文采用先前的软件测试研究中广泛使用的 9 个项目 [17, 52] 作为实证研究的实验对象，这些项目涉及不同领域，来保证项目的可变性。

构建效度分析。构建效度指的是一个测验能测量理论的概念或特质的程度。本实验存在两种构建效度威胁。一是用于衡量变异体集合质量预测模型的指标，为了减少这种威胁，本文对变异体集合质量模型应用了各种广泛使用的

度量标准（例如，均方根误差、平均绝对误差和拟合优度）。二是等价变异体对变异体集合约简方法有效性的影响，在变异体集合约简方法中，本文并没有考虑等价变异体对变异集合质量得分带来的影响，为了消除这种影响，本文在实证研究中参考先前工作 [5, 55, 56] 识别等价变异体，以确保基于遗传算法的变异体集合约简方法的准确性。

6.7 本章小结

本章主要介绍基于遗传算法的变异体集合约简方法的实验部分。首先提出三个研究问题；然后介绍本文的实验对象，并针对三个研究问题，分别设计了三个实验并给出详细的实验步骤和结果分析；最后对从内部效度、外部效度和构建效度三方面对实验进行效度分析。通过实验一可以得出结论：随机森林回归算法在构建质量预测模型时表现最好；MSQP 在有效性和效率方面都表现优秀，具有较高的性能，可作为 GAR 中的适应度评价模型对变异体集合进行约简。通过实验二可以得出结论：每一类别的特征对于构建质量预测模型都发挥了关键性的作用，并且缺一不可。通过实验三可以得出结论：GAR 在有效性和效率方面都表现优秀，具有较高的性能，并且在确保约简后变异体子集测试效果的同时，显著减少了约简时间，达到了降低变异测试计算开销的目的。

第七章　总结与展望

7.1　总结

当前针对变异测试选择技术研究工作的一个主要不足之处在于需要多次动态执行变异测试来计算每次选择的变异体子集质量得分，计算开销十分昂贵。为了解决已有变异选择方法的不足，本文将机器学习和遗传算法的思想引入变异体集合约简领域，通过构建机器学习模型预测变异体集合的质量得分，通过应用遗传算法搜索质量最优的变异体子集，以降低变异测试的计算成本。具体而言，本文首先分析源程序、变异体和测试覆盖结果，从中提取与变异体集合质量得分相关的特征，其次构建变异体集合质量预测模型预测质量得分，然后将预测模型作为的适应度函数输入至遗传算法，并进行质量最优子集的搜索。本方法通过避免动态执行变异测试和提升变异选择的收敛速度，达到了降低变异测试计算开销的目的。

在基于遗传算法的变异体集合约简方法的基础上，本文设计并实现了一个变异体集合约简系统。该系统采用前后端分离的模式进行构建，前端使用业界比较流行的 React-Mobx 技术进行开发。后端采用 Python 语言支持的 Flask 框架为前端提供 Restful API。该系统由变异体生成模块、质量模型构建模块、变异体集合约简模块与测试报告生成模块构成，实现了项目上传、变异生成、变异信息查看、变异集合约简、约简结果查看、测试报告生成和测试报告查看等功能。整体采用 Docker 容器技术进行打包，并以微服务的形式进行集成并向外提供服务，具有良好的可移植性，降低系统部署与运维方面的开销。此外，该系统还提供了一组简洁易用的图形用户界面，这些特点使该系统成为了测试人员的一个理想变异约简工具。该系统已经在 GitHub 上开源，任何人都可以从https://github.com/Provence613/MSRTool获取其源码并使用。

为了评估本文方法的有效性，本文在先前的软件测试研究中广泛使用的 9 个项目上进行了实证研究。实证研究表明：（1）随机森林回归算法在构建质量预测模型时表现最好；MSQP 在有效性和效率方面都表现优秀，具有较高的性能，可作为 GAR 中的适应度评价模型对变异体集合进行约简；（2）每一类

别的特征对于构建变异体集合质量预测模型都发挥了关键性的作用，并且缺一不可；（3）GAR 在有效性和效率方面都表现优秀，具有较高的性能，并且在确保约简后变异体子集测试效果的同时，显著减少了约简时间，达到了降低变异测试计算开销的目的。

7.2 展望

本文方法是将遗传算法应用于变异体集合约简领域的一次成功探索，同时实现了遗传算法与机器学习相结合，大幅提升了变异测试的效率，然而在未来，本文方法在以下方面还有较大的改进和提高空间：

(1) 针对 MSQP 的假设是合理的，但未经实验验证，后续将进行实证研究，通过实验结果验证假设的合理性。
(2) 本文的实证研究是针对项目内的，未来可将预测模型扩展到项目间（跨项目类别）进行实验，以验证本文方法的有效性。
(3) 由于时间原因，本文中实验对象仅为 9 个流行的 Java 项目，代表性有限。在后续的实验中，可以扩大实验规模，从而提高了实验的说服力。
(4) 未来将通过设计更强大的特征来提高 MSQP 的预测质量和效率，并尝试使用其他语言来评估 MSQP。

参考文献

[1] ZHANG J, ZHU M, HAO D, et al. An empirical study on the scalability of selective mutation testing[C] // 2014 IEEE 25th International Symposium on Software Reliability Engineering. 2014 : 277 – 287.

[2] 陈翔, 顾庆. 变异测试: 原理, 优化和应用 [J]. 计算机科学与探索, 2012, 6(12) : 1057 – 1075.

[3] PAPADAKIS M, KINTIS M, ZHANG J, et al. Mutation testing advances: an analysis and survey[G] // Advances in Computers : Vol 112. [S.l.] : Elsevier, 2019 : 275 – 378.

[4] JIA Y, HARMAN M. An analysis and survey of the development of mutation testing[J]. IEEE Transactions on Software Engineering, 2010, 37(5) : 649 – 678.

[5] ZHANG L, HOU S-S, HU J-J, et al. Is operator-based mutant selection superior to random mutant selection?[C] // Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1. 2010 : 435 – 444.

[6] BAJAJ A, SANGWAN O P. A systematic literature review of test case prioritization using genetic algorithms[J]. IEEE Access, 2019, 7 : 126355 – 126375.

[7] ACREE A T, BUDD T A, DEMILLO R A, et al. Mutation analysis[R]. [S.l.] : Georgia Inst of Tech Atlanta School of Information and Computer Science, 1979.

[8] WONG W E, MATHUR A P. Reducing the cost of mutation testing: An empirical study[J]. Journal of Systems and Software, 1995, 31(3) : 185 – 196.

[9] WONG W E, MATHUR A P, MALDONADO J C. Mutation versus all-uses: An empirical evaluation of cost, strength and effectiveness[G] // Software Quality and Productivity. [S.l.] : Springer, 1995 : 258 – 265.

[10] JONES J A, HARROLD M J. Empirical evaluation of the tarantula automatic fault-localization technique[C] // Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. 2005 : 273 – 282.

[11] PAPADAKIS M, MALEVRIS N. An empirical evaluation of the first and second order mutation testing strategies[C] // 2010 Third International Conference on Software Testing, Verification, and Validation Workshops. 2010 : 90 – 99.

[12] GOPINATH R, ALIPOUR A, AHMED I, et al. How hard does mutation analysis have to be, anyway?[C] // 2015 IEEE 26th International Symposium on Software Reliability Engineering. 2015 : 216 – 227.

[13] MATHUR A P, WONG W E. A theoretical comparison between mutation and data flow based test adequacy criteria[C] // Proceedings of the ACM 22rd Annual Computer Science Conference on Scaling up: Meeting the Challenge of Complexity in Real-World Computing Applications. 1994 : 38 – 45.

[14] OFFUTT A J, LEE A, ROTHERMEL G, et al. An experimental determination of sufficient mutant operators[J]. ACM Transactions on Software Engineering and Methodology, 1996, 5(2) : 99 – 118.

[15] BARBOSA E F, MALDONADO J C, VINCENZI A M R. Toward the determination of sufficient mutant operators for C[J]. Software Testing, Verification and Reliability, 2001, 11(2) : 113 – 136.

[16] NAEEM M R, LIN T, NAEEM H, et al. Scalable mutation testing using predictive analysis of deep learning model[J]. IEEE Access, 2019, 7 : 158264 – 158283.

[17] ZHANG J, ZHANG L, HARMAN M, et al. Predictive mutation testing[J]. IEEE Transactions on Software Engineering, 2018, 45(9) : 898 – 918.

[18] MAO D, CHEN L, ZHANG L. An extensive study on cross-project predictive mutation testing[C] // 2019 12th IEEE Conference on Software Testing, Validation and Verification. 2019 : 160 – 171.

[19] AGHAMOHAMMADI A, MIRIAN-HOSSEINABADI S-H. The threat to the validity of predictive mutation testing: The impact of uncovered mutants[J]. arXiv preprint arXiv:2005.11532, 2020.

[20] COLES H, LAURENT T, HENARD C, et al. Pit: a practical mutation testing tool for java[C] // Proceedings of the 25th International Symposium on Software Testing and Analysis. 2016 : 449 – 452.

[21] 张功杰, 巩敦卫, 姚香娟. 基于变异分析和集合进化的测试用例生成方法 [J]. 计算机学报, 2015, 000(011) : 2318 – 2331.

[22] JUST R, JALALI D, INOZEMTSEVA L, et al. Are mutants a valid substitute for real faults in software testing?[C] // Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. 2014 : 654 – 665.

[23] MATHUR A P. Performance, effectiveness, and reliability issues in software testing[C] // 1991 The Fifteenth Annual International Computer Software & Applications Conference. 1991 : 604 – 605.

[24] GUPTA V, AGGARWAL K, SINGH Y. A fuzzy approach for integrated measure of object-oriented software testability[J]. Journal of Computer Science, 2005, 1(2) : 276 – 282.

[25] BENESTAD H C, ANDA B, ARISHOLM E. Assessing software product maintainability based on class-level structural measures[C] // International Conference on Product Focused Software Process Improvement. 2006 : 94 – 111.

[26] MAUŠA G, GRBAC T G, BAŠIĆ B D. Data collection for software defect prediction-An exploratory case study of open source software projects[C] // 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics. 2015 : 463 – 469.

[27] CHAND N, MISHRA P, KRISHNA C R, et al. A comparative analysis of SVM and its stacking with other classification algorithm for intrusion detection[C] // 2016 International Conference on Advances in Computing, Communication, Automation. 2016 : 1 – 6.

[28] BRUN Y, ERNST M D. Finding latent code errors via machine learning over program executions[C] // Proceedings. 26th International Conference on Software Engineering. 2004 : 480 – 490.

[29] ZHANG J, WANG X, HAO D, et al. A survey on bug-report analysis[J]. Science China Information Sciences, 2015, 58(2): 1–24.

[30] RODRIGUEZ-GALIANO V, MENDES M P, GARCIA-SOLDADO M J, et al. Predictive modeling of groundwater nitrate pollution using Random Forest and multisource variables related to intrinsic and specific vulnerability: A case study in an agricultural setting (Southern Spain)[J]. Science of the Total Environment, 2014, 476: 189–206.

[31] MCDONALD G C. Ridge regression[J]. Wiley Interdisciplinary Reviews: Computational Statistics, 2009, 1(1): 93–100.

[32] 李航, OTHERS. 统计学习方法 [M]. [S.l.]: 清华大学出版社, 2012.

[33] SHOKRY A, AUDINO F, VICENTE P, et al. Modeling and simulation of complex nonlinear dynamic processes using data based models: Application to photo-Fenton process[G] // Computer Aided Chemical Engineering. [S.l.]: Elsevier, 2015: 191–196.

[34] GOYAL R, CHANDRA P, SINGH Y. Suitability of KNN regression in the development of interaction based software fault prediction models[J]. Ieri Procedia, 2014, 6: 15–21.

[35] 吴昊, 李浩然, 万交龙. 对于测试用例生成的遗传算法改进 [J]. 计算机系统应用, 2016, 25(8): 200–205.

[36] LIN J-C, YEH P-L. Using genetic algorithms for test case generation in path testing[C] // Proceedings of the Ninth Asian Test Symposium. 2000: 241–246.

[37] BARESEL A, STHAMER H, SCHMIDT M. Fitness function design to improve evolutionary structural testing[C] // Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation. 2002: 1329–1336.

[38] SOMMERVILLE I. Software engineering[M]. [S.l.]: Pearson Education India, 2004.

[39] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182 – 197.

[40] DANG X, YAO X, GONG D, et al. Efficiently generating test data to kill stubborn mutants by dynamically reducing the search domain[J]. IEEE Transactions on Reliability, 2020, 69(1): 334 – 348.

[41] HONGLEI T, WEI S, YANAN Z. The research on software metrics and software complexity metrics[C] // 2009 International Forum on Computer Science-Technology and Applications. 2009: 131 – 136.

[42] COLEMAN D, ASH D, LOWTHER B, et al. Using metrics to evaluate software system maintainability[J]. Computer, 1994, 27(8): 44 – 49.

[43] KAN S H. Metrics and models in software quality engineering[M]. [S.l.]: Addison-Wesley Professional, 2003.

[44] MURGIA A, MARCHESI M, CONCAS G, et al. Parameter-based refactoring and the relationship with fan-in/fan-out coupling[C] // 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops. 2011: 430 – 436.

[45] COUNSELL S, LIU X, ELDH S, et al. Re-visiting the maintainability index metric from an object-oriented perspective[C] // 2015 41st Euromicro Conference on Software Engineering and Advanced Applications. 2015: 84 – 87.

[46] MUBARAK A, COUNSELL S, HIERONS R M. An evolutionary study of fan-in and fan-out metrics in OSS[C] // 2010 Fourth International Conference on Research Challenges in Information Science. 2010: 473 – 482.

[47] VISSER W. What makes killing a mutant hard[C] // Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. 2016: 39 – 44.

[48] DELAMARO M E, OFFUTT J, AMMANN P. Designing deletion mutation operators[C] // 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation. 2014: 11 – 20.

[49] BREIMAN L. Random forests[J]. Machine Learning, 2001, 45(1): 5 – 32.

[50] 朱少民. 软件测试方法和技术 [M]. [S.l.]: 清华大学出版社有限公司, 2005.

[51] KRUCHTEN P B. The 4+1 view model of architecture[J]. IEEE Software, 1995, 12(6): 42 – 50.

[52] KINTIS M, PAPADAKIS M, PAPADOPOULOS A, et al. How effective are mutation testing tools? An empirical analysis of Java mutation testing tools with manual analysis and real faults[J]. Empirical Software Engineering, 2018, 23(4): 2426 – 2463.

[53] WU H, LI W. Downscaling land surface temperatures using a random forest regression model with multitype predictor variables[J]. IEEE Access, 2019, 7: 21904 – 21916.

[54] 张功杰, 巩敦卫, 姚香娟. 基于统计占优分析的变异测试 [J]. 软件学报, 2015, 26(10): 2504 – 2520.

[55] NAMIN A S, ANDREWS J, MURDOCH D. Sufficient mutation operators for measuring test effectiveness[C] // 2008 ACM/IEEE 30th International Conference on Software Engineering. 2008: 351 – 360.

[56] ZHANG L, GLIGORIC M, MARINOV D, et al. Operator-based and random mutant selection: Better together[C] // 2013 28th IEEE/ACM International Conference on Automated Software Engineering. 2013: 92 – 102.

[57] NDIAYE E, LE T, FERCOQ O, et al. Safe grid search with optimal complexity[C] // International Conference on Machine Learning. 2019: 4771 – 4780.

致　　谢

两年的研究生生涯转瞬即逝，马上就要接近尾声。在这两年的时光中，我学习到了许多，同时也成长了许多。在毕业之际我要感谢在过去的两年中帮助过我的人，有了你们，我的研究生生涯才会如此完美。

首先我要感谢我的导师陈振宇老师，陈老师有责任心，对待学术严谨认真，对待学生耐心教导，陈老师每一次开会，都能指出学生的优点与不足，每一次都会指引我进步。陈老师从论文的选题到最终完成对我给予了许多帮助与指导，在这里我由衷地感谢陈老师对我的关心、鼓励和支持。

其次，我要感谢实验室的王兴亚老师和赵源师兄，王老师总是和我们一起分析，找出问题的关键，然后想出合理的方案去解决问题；而当我们尝试使用一些较新的、没有在课程中涉及的技术时，他为我们找了相关的资料并且提了很多宝贵的建议，引导我们去学习和使用那些技术，从而弥补了我们面对新技术时的不足，并且一次次耐心地指导我的论文写作，没有你们的帮助，我的论文无法顺利完成。我还要感谢同组的李紫欣同学、王新宇同学、巫浩然同学，大家互相帮助，互相学习，一同度过了一段美好的时光。

接下来我要感谢软院的各位老师和同学们。感谢老师，你们教授的知识是我一生中最重要的财富，我将永远珍惜和心怀感谢。感谢同学们，人生中有你们的陪伴，使我受益颇多。

最后，我要感谢我的父母，是他们一直在我身后作为我经济上以及生活上的坚实后盾，他们对我的成长可以说是无私地奉献，给予我无限的关爱，教我做人，尽可能让我有最好的条件。在此，我祝他们身体健康，万事如意！向他们表示我最诚挚的感谢！

简历与科研成果

基本信息

刘芳潇，女，汉族，1997 年 6 月出生，山东省德州市人。

教育背景

2019 年 9 月 — 2021 年 6 月 南京大学软件学院 硕士

2015 年 9 月 — 2019 年 6 月 青岛大学数据科学与软件工程学院 本科

攻读工程硕士学位期间完成的学术成果

1. **F. Liu**, X. Wang, Z. Li, J. Xu and Y. Gao, ”Effective GasPrice Prediction for Carrying Out Economical Ethereum Transaction,” 2019 6th International Conference on Dependable Systems and Their Applications (DSA), Harbin, China, 2020, pp. 329-334, doi: 10.1109/DSA.2019.00050.
2. 王兴亚, **刘芳潇**, 李紫欣, 李玉莹, 黄勇, 徐剑锋. 一种基于不完全信息静态博弈的以太坊燃油定价方法. 2019.10.14, 中国, 2019109747709. (受理)
3. 王兴亚, **刘芳潇**, 徐介晖, 王新宇, 陈振宇. 一种基于 Xgboost 的以太坊交易场景下最低燃油价格预测方法. 2019.06.11, 中国, 2019104983809. (受理)