Higher accuracy and lower run time

Abstract :

The abstract introduces mutation analysis as a technique for measuring the effectiveness of testing or debugging methods. It highlights the problem of computational expense, largely attributed to the execution of redundant mutants. These mutants, predictable based on the outcomes of other mutants, unnecessarily consume resources and may skew the mutation detection rate. The paper's contributions are the definition of non-redundant versions of specific mutation operators and the findings from an empirical study demonstrating the prevalence of redundant mutants and the potential for efficiency and accuracy improvements in mutation analysis.

Chapter-by-Chapter Summary:

1. Introduction :

The introduction sets the stage by discussing the origins and purpose of mutation analysis, emphasizing its computational intensity. Mutation analysis generates mutants (variations of the original program with slight syntactic changes) to test the ability of test suites to detect these changes. Despite its effectiveness, the resource-intensive nature of generating and testing a vast number of mutants makes it a challenging process. The study proposes a focus on non-redundant mutations, aiming to streamline the analysis without compromising its integrity.

2. A Detailed View of Mutation Operators :

This section outlines the theoretical foundation of mutation operators in Java, categorizing them into unary and binary operator replacements, unary operator insertion, and literal value replacement. The discussion revolves around identifying redundancies within these operators that contribute to the inefficiency of mutation analysis. By redefining these operators to exclude redundant mutations, the study aims to enhance the process's efficiency.

2.1 Definition of a Sufficient Set of Non-redundant Mutations :

The paper proposes a formal definition for identifying a sufficient set of non-redundant mutations. This set aims to cover all possible mutations that could arise from a given mutation operator without including any redundancies. The criteria for a mutation to be considered non-redundant are introduced, setting the stage for a more focused analysis.

2.2- Non-redundant COR, UOI, and ROR Mutation Operators :

These subsections delve into the specifics of conditional operator replacement (COR), unary operator insertion (UOI), and relational operator replacement (ROR) mutation operators. For each operator, the paper defines a non-redundant set of mutations and explains how these sets were determined. This rigorous analysis reveals that a significant portion of mutations generated by these operators is redundant, thus not necessary for effective mutation analysis.

3. Empirical Evaluation :

An empirical study utilizing the Major mutation framework serves to validate the theoretical propositions. Ten real-world Java programs, comprising 410,000 lines of code, are analyzed using both developer-written and automatically generated test suites. This evaluation seeks to quantify the prevalence of redundant mutants and assess the impact of their removal on the efficiency and accuracy of mutation analysis.

3.1 Methodology :

The methodology for the empirical study is outlined, detailing the selection of Java programs, the mutation operators analyzed, and the test suites employed. The study's design aims to comprehensively understand the effects of redundant mutant elimination across various program sizes and types.

3.2 Results and Analysis :

These sections present the findings from the empirical study. The analysis shows a substantial reduction in mutation analysis run time and an improvement in mutation score accuracy upon removing redundant mutants. Furthermore, the study explores the relationship between mutation coverage and traditional code

coverage metrics, finding a strong correlation that suggests mutation coverage could serve as an alternative adequacy criterion for testing.

4. Related Work :

A review of related work situates this study within the broader context of mutation testing research. Previous efforts to reduce mutation testing costs are discussed, highlighting the novelty and significance of focusing on non-redundant mutations. This section also references studies on different mutation operators and the challenges of equivalent mutant detection.

5. Conclusions and Future Work

The paper concludes that eliminating redundant mutations can significantly enhance the efficiency and accuracy of mutation analysis. The empirical study supports this conclusion, demonstrating the feasibility and benefits of the proposed approach. Potential avenues for future research include extending the analysis to other mutation operators and exploring inter-operator redundancies.