A Trial on Optimizing Test Sequences for LAPACK Eigenvalue Computation Routines using Machine Learning

Hiroto Kashimura† Takahiro Katagiri‡ Shuji Morisaki† Daichi Mukunoki‡ Tetsuya Hoshino‡

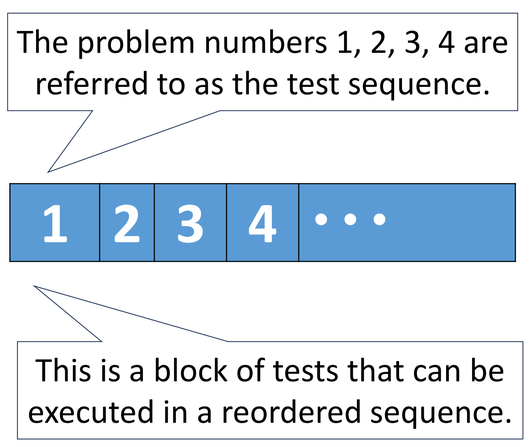
Graduate School of Informatics, Nagoya University† Information Technology Center, Nagoya University‡

Introduction

Background
Software testing is essential for quality assurance, but it carries significant execution costs.

Objective
Prioritize executing test cases prone to bugs to reduce the time required to detect bugs.

Proposal
we proposed a machine learning model to predict the fault-proneness in eigenvalue computation test cases.



LAPACK & STCollection

LAPACK
A numerical computing software library that provides routines for solving systems of linear equations, least squares problems, eigenvalue problems, and singular value problems.

STCollection[1]
A test routine that verifies procedures in LAPACK by performing eigenvalue computations on matrices with known theoretical solutions, using the QR method, the Divide and Conquer method, the Bisection/Inverse Iteration method, and the MRRR method.

Proposed Method

Inject bugs into the LAPACK library

In this study, to simulate bugs in eigenvalue computations using the divide-and-conquer method within numerical computation libraries, we injected artificial faults into the main routines of LAPACK/BLAS.

dgemm : Matrix Multiplication

$$C = \alpha * op(A) * op(B) + \beta * C$$

Set α close to 0

$$\alpha \to 0$$

This introduces uniform numerical errors in matrix operations.

dlaed3 : Eigenvector Merging

This routine calls dgemm.Injected a bug into the calculation range of dgemm.

dgemm (transa, transb, m-1, n, k, ⋯)

or

dgemm (transa, transb, m, n, k-1, ⋯)

dlaed4 : Secular Equation Solver

$$diag(D) + \rho ZZ^T$$

This routine calculates eigenvalues through iterative computation.
Increasing the tolerance reduces accuracy.

Train an AI model

1. Injected a bug into LAPACK and compute the eigenvector.
2. Calculate 11 statistical feature representing numerical stability and differences from analytical solutions from the eigenvectors.
3. Train an AI model to predict bug occurrence rates using 11 statistical features as input.

11 statistical features : Features extracted from test matrices and computed eigenvectors, including their deviations from analytical solutions. Example metrics: Frobenius norm and spectral norm.

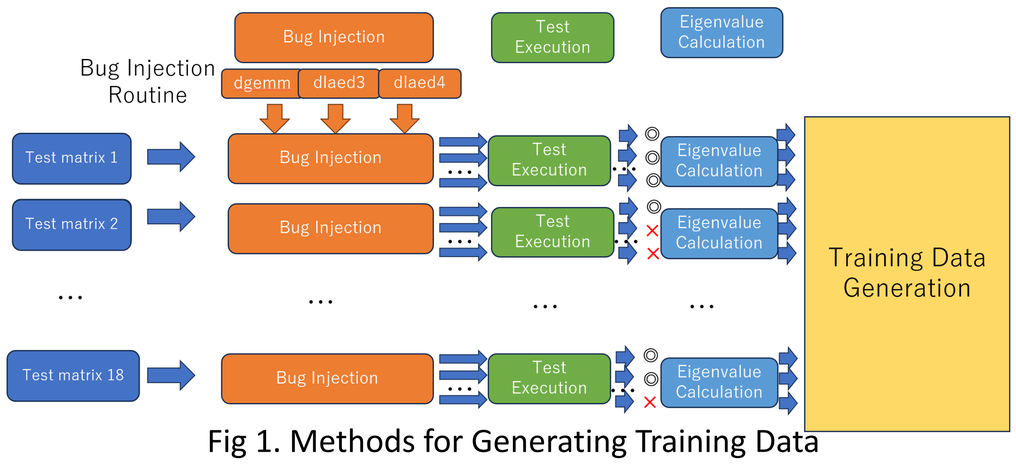


Fig 1. Methods for Generating Training Data

Optimize Test Sequence

1. For each test case in an unknown test sequence, calculate the eigenvector and statistical features, then perform data preprocessing.
2. Use the preprocessed data to predict the bug occurrence rate.
3. To improve the APFD score, the test sequence is reordered so that test cases with higher predicted fault-proneness are executed first.

APFD : Efficiency score of test sequence indicating how quickly bugs were found[2]

$$APFD = 1 - \frac{Tf1 + Tf2 + \cdots + Tfm}{(m * n)} + \frac{1}{2n}$$

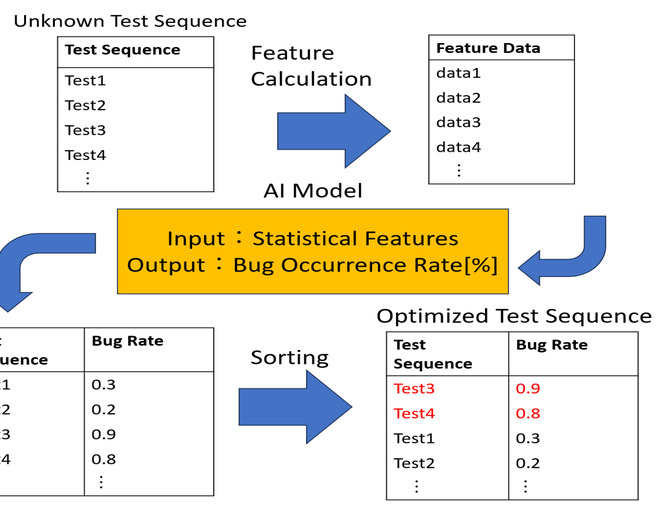


Fig 2. Methods for Optimizing Test Sequences

Evaluation Result

The proposed model achieved a high F1-score and significantly improved the APFD compared to random test sequences by optimizing the execution order.

Metric	Result
Accuracy	0.9335
Precision	0.9411
Recall	0.9658
F1 Score	0.9533

Table 1. Performance of the Bug Prediction Model

	APFD
random	0.9751
optimized	0.9988

Table 2. APFD of the optimized test sequence

Conclusion

- We injected bugs into the LAPACK routine performing eigenvalue calculations and computed the eigenvectors.
- We extracted 11 features from the eigenvectors and created an AI model to predict bug occurrence rates.
- Using the AI model, we predicted the fault-proneness for the test sequence and performed sorting.
- The proposed model achieved high prediction performance, enabling effective prioritization of bug-prone test cases.

Acknowledgements
This work was supported by the Joint Usage/Research Center for Interdisciplinary Large- scale Information Infrastructures (JHPCN) and the High-Performance Computing Infrastructure (HPCI) under project number jh250015. In addition, this work was funded by JSPS KAKENHI Grants JP23K11126 and JP24K02945.

References
[1] OSNI, A. M., CHRISTOF, V., JAMES, W. D., BERESFORD, N. P.: A Testing Infrastructure for Symmetric Tridiagonal Eigensolvers, ACM Transactions on Mathematical Software, Vol. 35, No. 1, Article 8（2008）.
[2] Rothermel, G., Untch, R. H., Chu, C. and Harrold, M. J.: Prioritizing Test Cases For Regression Testing, IEEE Transactions on Software Engineering, Vol. 27, No. 10, pp. 929–948 (Oct. 2001).