ML-Driven Compiler Testing: Optimization Methods and Technical Approaches

Xuechen Shao

Shanghai Baoshan District Shanghai University Baoshan Campus 201900

Abstract: ***Within the domain of system software, compilers play a critically important role. If a compiler contains defects or malfunctions, the resulting executable files can be severely compromised, introducing errors that propagate throughout the software stack. In compiler quality management, rigorous testing through appropriate methodologies is therefore essential. Recent technological advances have facilitated the evolution of automated compiler testing techniques. However, most contemporary testing approaches rely on test case generation tools such as Csmith, implementing functional testing by generating large volumes of test cases during the testing process. Moreover, as compilers constitute large-scale software systems, the associated testing processes also perform stress testing through the execution of extensive test case suites. Nevertheless, this approach suffers from significant efficiency limitations.***

Keywords: Machine learning; Compiler testing; Method optimization.

1. INTRODUCTION

In today's computer world, compilers play a crucial role in software development. As a result, compiler testing has become a hot topic in computer research. In software development, software testing is an essential step and a key to software quality, and compilers help designers to identify and correct defects in software. However, in practice, there are oracle problems in testing. In testing the compiler, the relevant researchers use random differences, Differences and deformation testing techniques have to some extent solved the superficial effects of Oracle problems in testing, played a great role in the analysis of test results and helped researchers to trace the problems back to their roots. However, in existing technologies, the practice cycle of testing compilers is long, and the number of errors discovered by customers through testing is very low, and the compiler itself is large during testing, which creates a more serious efficiency problem. In order to speed up the quality and efficiency of compiler testing, it is necessary to optimize the current test methodology. Ding et al. [1] proposed a multi-scale adaptive clustering and local consistency learning method for unsupervised clothing-changing person re-identification in 2026, while Pinyoanuntapong et al. [2] introduced a self-aligned domain adaptation approach for mmWave gait recognition in 2023. For pose estimation, Peng, Zheng, and Chen [3] developed a dual-augmentor framework for domain generalization in 3D human pose estimation in 2024. In environmental health, Narouei et al. [4] examined the effects of germicidal far-UVC on ozone and particulate matter in a conference room in 2025. For biomedical applications, Xiao et al. [5] designed an ultrasmall Fe_3O_4-decorated polydopamine hybrid nanozyme for wound disinfection in 2022. In human-computer interaction and service design, Shan, Xu, Xia, and Lin [6] rethought wine tasting for Chinese consumers using multimodal personalization in 2025; Xia, Xu, and Shan [7] developed KOA-Monitor for knee osteoarthritis patients in 2025; and Xu et al. [8] conducted a systematic review of game-based learning for environmental resilience in 2026. For natural language processing, Deng et al. [9] proposed LLM-guided multi-view reasoning distillation for sarcasm detection in 2026. In person re-identification, Wang et al. [10] introduced a quality-aware query-adaptive convolution for clothes-changing ReID in 2026. For video analysis, Li et al. [11] presented a memory-aware fine-tuning of SAM2 for long-sequence video object segmentation in 2026. In multi-agent systems, Yan et al. [12] designed PRISM for root-cause investigation via specialized multi-agents in 2026; Yuan et al. [13] developed TA-Mem for tool-augmented memory retrieval in long-term conversational QA in 2026; and Yang et al. [14] proposed a recursive multi-agent trading system under geopolitical uncertainty in 2026. For business and trade, Li, Yang, and Zhang [15] presented a data-driven study on international sales strategies and performance in 2026; Zhou [16] explored digital precision distribution for social media content in the automotive industry in 2025; Wensi [17] investigated AI-enabled data visualization marketing for automated production lines in 2026; and Yang, Zheng, and Lu [18] constructed a multi-dimensional credit risk map using graph neural networks in 2025. For fraud detection, Shen et al. [19] applied the whale optimization algorithm to financial payment fraud detection in 2025. In photonics, Tang et al. [20] designed and optimized a shallow-angle grating coupler for vertical emission from indium phosphide devices in 2020. Finally, Sun [21] addressed accessibility challenges and solutions in digital product interface design in 2025.

2. COMPILER TESTING TECHNIQUES

Compilers play an important role in software engineering. In order to ensure the running quality of compilers, lots of testing techniques have been proposed. In addition, some researchers have conducted experimental studies on current test techniques to compare test performance of different compilers. Here is a brief description of the three mainstream compiler testing techniques.

2.1 Random variance test

This test technique is widely used in compiler testing. In specific use, it is by comparing each other among multiple compilers to achieve the effect of detecting errors in their functions. If the test input conditions are the same and no error occurs, the output results will be the same. However, if the output results differ, it can be determined that there is a problem with the compiler used, either one or several. When more than two compilers are involved in a random difference test, the wrong compiler can be identified by voting when the test results differ.

2.2 Differential Optimization Level Testing

This test method is a variant of random differential testing, the difference between the two is that differential optimization testing can be achieved with the same inputs. Using different optimization level parameters to compile the input program, if the resulting executable is the same in the test, the compiler is proved to have no problem. Otherwise, there is a problem with the compiler. Its basic methodology is as shown in Figure 1 (b). In this $L_1, L_2, \cdots, L_n$ represents the compiler optimization level of compiler C, $O_1, O_2, \cdots, O_n$ the output of the input I representing the input condition P and P.

2.3 Equator input method

This is also a proven method of compiler detection. In conducting a test, a substitute is first generated based on an existing program, which is used as an input use case for the compiler test. After the test is completed, it is used as a comparison to the input use case test results by comparing it with the original test. Judging whether the compiler is working properly is based on comparing the test results obtained with the raw test data and the equivalent test results generated. The overall flow diagram is shown in Figure 1 (c). Of which $V_1, V_2, \cdots, V_n$ represents the equivalent variant of the original test program input, $O_1, O_2, \cdots, O_n$ represents the output of the test input, O p Table represents the output when testing the input program P case.

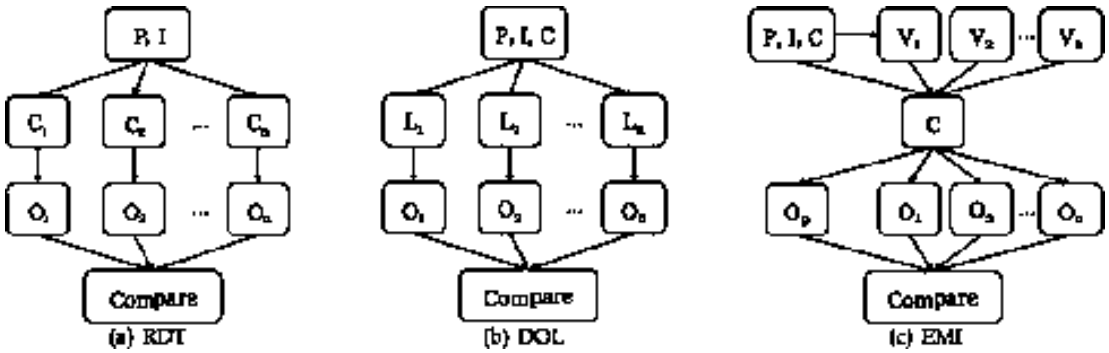

Figure 1: Flowchart of the current three major compilers

3. COMPILER TEST ACCELERATION

Most of the above testing techniques rely heavily on test case automatic generation tools, which generate test programs by way of a tool to complete the testing of the compiler. However, there is a very serious efficiency problem with this test method, which is evident at the present time. In recent years of research on compiler optimization methods, a number of methods for improving test efficiency have been proposed and gradually applied. Among the more widely used are TB-G and LET methods. Among them, the TB-G detection method is to test the text program for TB-G. In specific use, the test program is treated as text, extracting snippets that may be in error during detection and eventually transforming them into text vectors. Among them, the characters associated with the error include operational characters, statement characters, and type characters. After obtaining the

corresponding text vectors, these contents are normalized by TB-G method. Its main processes are shown in Figure 2.

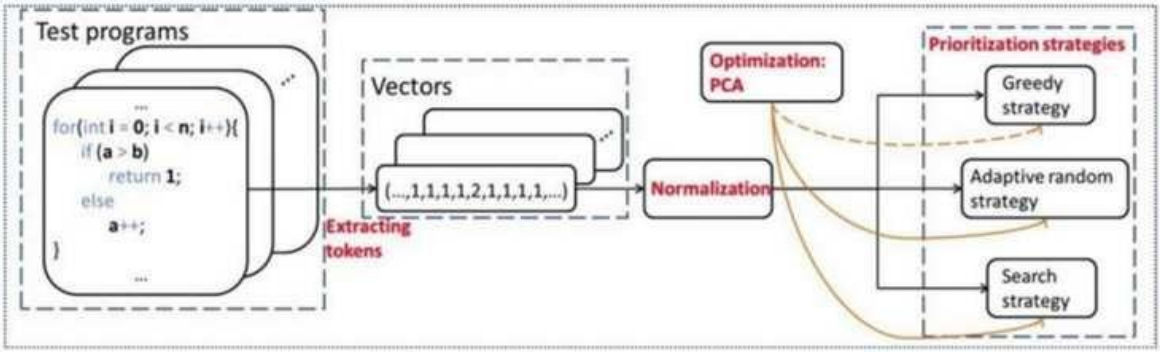

Figure 2: TB-G Basic Flow Diagram

In contrast, the LET method is more effective in accelerating the compiler test. In this detection method, the process of learning scheduling is included. During the learning phase, the method defines the relevant features of the error program, then pre-processes the data through those features, and then generates the training model data. In LET training, each program is evaluated for error detection, the probability of error detection is predicted, and the program execution time is calculated. During the scheduling phase, the detection method predicts the probability of detection and the execution time of each test program through a corresponding model.And the results are calculated, and the test program is sorted in descending order according to these two dimensions at last, so as to determine the final program determination sequence.

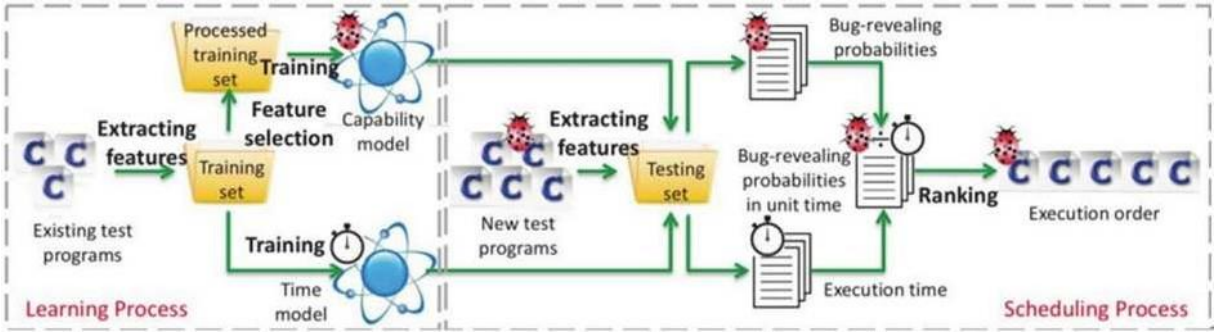

Figure 3: LET basic flow diagram

In the use of traditional sequencing methods, there are obvious inefficiencies that make it difficult to achieve the goal of improving the efficiency of detection. But in the LET detection method, this method can achieve the effect of adaptive sorting. In the selection of test cases, the use cases with the greatest difference will be chosen, generally measured by distance. In relevant test statistics, the method of adaptive random sequencing was found to be inefficient in practical use, and it cost a lot in sequencing in specific use.

4. COMPILER TEST ACCELERATION METHOD

In software engineering, compilers are very important basic software. Testing techniques are particularly critical in compiler quality management. However, most current test techniques require a large number of test cases as input to achieve test effects on compilers. But this approach is bound to consume a lot of practice and presents serious inefficiencies. In recent years, in order to further optimize the testing capabilities of compilers and improve the efficiency of their testing, relevant researchers have proposed a priority approach, which is to test the programs most likely to expose errors, thus achieving an improved efficiency of compiler testing. However, in practice, these testing methods have a serious problem. Different test programs may have the potential to trigger the same errors in the compiler during compilation, which seriously affects the efficiency of compiler testing. Here, by testing using coverage, it can effectively achieve the effect of distinguishing between test programs that repeatedly expose errors. However, the collection of coverage information consumes a lot of resources. Because the coverage information generation process is dynamic.Based on the coverage information, the researchers cluster the test

programs to achieve the effect of improving efficiency, and call it COP, which is divided into three parts to discuss the method [3].

4.1 Prediction of Coverage Information

When a test program is mutated by the encoder, the foottip information generated in the process is covered. Information such as whether a function is executed or not. This is the coverage information in compilation. In predicting it, you can think of this process as a multi-output regression problem, labeled to represent various elements of the component that are concerned, such as whether the code is executed, etc. This method is very similar to the application of traditional machine learning algorithms, in making predictions of coverage information, it is necessary to first define the features, make labels, and make predictions.

4.2 Clustering of predictive programs

After successfully obtaining the coverage information in the program test, COP organizes the information into different categories, which have a high probability of triggering different errors in the test. The method is to achieve clustering by clustering algorithm, the purpose of choosing the X-means clustering algorithm is that the way you do not need to treat the number of categories confirmed in advance.

4.3 Sequence of test procedures

Based on the results of the test program classification, COP can classify the test program. Different classes often lead to different errors in the compiler to be tested. In this, the COP will first calculate the time consumed by the test program and the probability of error detection through the way of LET. The program is then selected, the program with the greatest probability of error per unit of time is selected for testing in each category, and the sequencing is sequenced according to that result. In addition, test thresholds are selected, and when the use cases with a high probability of error per unit time have been selected, the use examples with a low probability per unit time are further selected as test programs.

5. CONCLUSIONS

In summary, there are some serious efficiency problems in current testing of compilers. In this context, many testing methods have been proposed. However, these testing methods are based primarily on a few criteria, a reordering of program use cases, and a priority execution approach to selecting the program that may have the highest starting error for testing. While this approach can achieve some speed improvements, it ignores the fact that the same errors can occur in different test programs. This situation has, to a large extent, a superficial impact on existing test methods. The greater the difference in the information covered by a program during a program's testing, the greater the likelihood that the program will penalize different errors. However, the current program testing process is generated in a dynamic environment, which creates a lot of difficulty in information gathering. Through the COP method to screen the validity of the coverage information, it is very helpful to improve the overall test efficiency.

REFERENCES

[1] Y. Ding, Z. Ye, I. Xu, S. Lyu and L. Zhang, "Multi-Scale Adaptive Clustering and Local Consistency Learning for Unsupervised Clothing-Changing Person Re-Identification," in IEEE Transactions on Information Forensics and Security, vol. 21, pp. 2889-2904, 2026, doi: 10.1109/TIFS.2026.3671089.

[2] Pinyoanuntapong, Ekkasit, et al. "Gaitsada: Self-aligned domain adaptation for mmwave gait recognition." 2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS). IEEE, 2023.

[3] Peng, Qucheng, Ce Zheng, and Chen Chen. "A Dual-Augmentor Framework for Domain Generalization in 3D Human Pose Estimation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.

[4] Narouei, F. H., Tang, Z., Wang, S. I., Hashmi, R. H., Welch, D., Sethuraman, S., ... & McNeill, V. F. (2025). Effects of germicidal far-UVC on ozone and particulate matter in a conference room. Plos one, 20(8), e0328224.

[5] Xiao, J., Hai, L., Li, Y., Li, H., Gong, M., Wang, Z., ... & He, D. (2022). An Ultrasmall Fe3O4 ‐ Decorated Polydopamine Hybrid Nanozyme Enables Continuous Conversion of Oxygen into Toxic Hydroxyl Radical via GSH ‐ Depleted Cascade Redox Reactions for Intensive Wound Disinfection. Small, 18(9), 2105465.
[6] Shan, X., Xu, Y., Xia, T., & Lin, Y. S. (2025, October). Rethinking Wine Tasting for Chinese Consumers: A Service Design Approach Enhanced by Multimodal Personalization. In 2025 International Conference on Content-Based Multimedia Indexing (CBMI) (pp. 1-5). IEEE.
[7] Xia, T., Xu, Y., & Shan, X. (2025, May). KOA-Monitor: A Digital Intervention and Functional Assessment System for Knee Osteoarthritis Patients. In International Conference on Human-Computer Interaction (pp. 388-405). Cham: Springer Nature Switzerland.
[8] Xu, Y., Sun, Z., Zhen, C., Lin, Y.-S., Sarker, T. H., Thorogood, M., Lasserre, P., & Dulic, A. (2026). From Engagement to Resilience: A Systematic Review of Game-Based Learning for Environmental Resilience. Sustainability, 18(5), 2305. https://doi.org/10.3390/su18052305
[9] Deng, X., Yang, Y., Wang, B., Zhang, X., Huang, S., Zhang, Y., & Lu, Q. (2026). LLM-MVR: LLM-Guided Multi-View Reasoning Distillation for Sarcasm Detection. Available at SSRN 6795979.
[10] Wang, Y., Jiang, K., Zhang, T., Tian, K., & Jiang, G. (2026). QA-ReID: Quality-Aware Query-Adaptive Convolution Leveraging Fused Global and Structural Cues for Clothes-Changing ReID. arXiv preprint arXiv:2601.19133.
[11] Li, G., Yuan, H., Chen, S., Hu, Q., Wang, J., & Jiang, K. (2026). MFT: Memory-Aware Fine-Tuning of SAM2 for Efficient Long-Sequence Video Object Segmentation. IEEE Signal Processing Letters.
[12] Yan, W., Wu, Y., Liang, P., Yuan, M., Liu, J., Yang, J., & Li, X. (2026, March). PRISM: Pipeline for Root-cause Investigation via Specialized Multi-agents. In 2026 International Conference on Generative Artificial Intelligence and Information Security (GAIIS) (pp. 709-712). IEEE.
[13] Yuan, M., Liu, J., Yang, J., Li, X., Yan, W., Wu, Y., & Liang, P. (2026, March). TA-Mem: Tool-Augmented Autonomous Memory Retrieval for LLM in Long-Term Conversational QA. In 2026 9th International Conference on Advanced Algorithms and Control Engineering (ICAACE) (pp. 2684-2688). IEEE.
[14] Yang, J., Wu, Y., Liu, J., Liang, P., Yuan, M., Li, X., & Yan, W. (2026). Recursive Multi-Agent Trading System: Iterative Optimized Portfolio Strategy Under Geopolitical Uncertainty. arXiv preprint arXiv:2605.25311.
[15] Li, X., Yang, X., & Zhang, G. (2026, January). A Data-Driven Study on the Correlation between International Sales Strategies and Performance: An Empirical Model Based on the Theoretical Framework of International Trade. In Proceedings of the 2nd International Conference on Digital Society, Information Science and Risk Management (pp. 396-401).
[16] Zhou, Z. (2025, November). Digital precision distribution strategy for social media content on private domain platforms in the automotive industry: a collaborative filtering model based on user behavior. In Proceedings of the 2025 International Conference on Digital Society and Intelligent Computing (pp. 516-521).
[17] Wensi, L. (2026). AI-Enabled Data Visualization Marketing for Automated Production Lines: Building Customer Trust and Improving Lead-to-Order Conversion. Academic Journal of Natural Science, 3(1), 8-13.
[18] Yang, X., Zheng, X., & Lu, Q. (2025, October). Construction and early warning of multi-dimensional network credit-related transaction risk maps by integrating graph neural network (GNN). In Proceedings of the 2025 2nd International Conference on Digital Economy and Computer Science (pp. 919-923).
[19] Shen, Zepeng, et al. "Research on Application of Whale Optimization Algorithm in Financial Payment Fraud Detection." 2025 4th International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID). IEEE, 2025.
[20] Tang, Yingheng, et al. "Design and Optimization of Shallow-Angle Grating Coupler for Vertical Emission from Indium Phosphide Devices." (2020).
[21] Sun, Lingxin. "Designing Inclusive Interfaces: Accessibility Challenges and Solutions in Digital Products." Proceedings of the 2025 International Conference on Artificial Intelligence and Sustainable Development. 2025.