Qeios

Peer Review

Review of: "Finding Missed Code Size Optimizations in Compilers using LLMs"

Sen Fang[1]

1. North Carolina State University, United States

The paper presents a novel approach to identifying missed code size optimization opportunities in compilers using LLMs. Unlike traditional compiler testing methods that require complex program generators (like CSmith with 40,000+ lines of code), this approach is remarkably simple, requiring fewer than 150 lines of code to implement.

The key aspects of their methodology are:

1. Starting with a simple seed program that gets iteratively mutated by an LLM using predefined mutation instructions.
2. Using four differential testing strategies to detect potential optimization issues:
    1. Dead code differential testing
    2. Optimization pipeline differential testing
    3. Single-compiler differential testing
    4. Multi-compiler differential testing
3. Employing validation techniques to filter out false positives

The researchers tested their approach using two configurations of the Llama 3.1 model (70B and 405B parameters). While the 405B model was more effective at generating valid mutations, the faster 70B model produced more bugs overall due to higher throughput.

The results were significant:

1. Found 24 confirmed bugs in production compilers
2. Successfully extended to other languages (Rust and Swift) with minimal modifications
3. Discovered 4 bugs in Rust and 1 bug in Swift during just one hour of testing each

doi.org/10.32388/I7XVDB

This work demonstrates that LLMs can be effectively used to simplify compiler testing while maintaining high bug-finding capability, potentially making compiler testing more accessible for newer programming languages that lack established testing infrastructures.

Declarations

Potential competing interests: No potential competing interests to declare.