Modern compilers apply a set of optimization passes aiming to speed up the generated code. The combined effect of individual optimizations is often unpredictable. Thus, changes to a compiler's code may hinder the performance of generated code as an unintended consequence. Due to the vast number of compilation units and applied optimizations, it is difficult to diagnose these regressions.

We propose to solve the problem of diagnosing performance regressions by capturing the compiler's optimization decisions. We do so by representing the applied optimization phases, optimization decisions, and inlining decisions in the form of trees. This thesis introduces an approach utilizing tree edit distance (TED) to detect optimization differences in a semi-automated way. Since the same source code may be inlined in different contexts and optimized differently in each, we also present an approach to compare optimization decisions in differently inlined code. We employ these techniques to pinpoint the causes of performance problems in various benchmarks of the Graal compiler.