Sikraken Optimisation Functional Spec 25/10/2024 Kacper Krakowiak

Brief Description:

This document serves as the functional specification for the Sikraken Optimization Tool, an intelligent assistant designed to enhance test case generation for C code. The tool aims to automate the parameter selection process for Sikraken, a test generation tool, by improving code coverage and reducing the manual effort required for test configuration.

Full Description:

"Sikraken is a tool used for generating regression tests for C code. It aims to achieve full branch coverage. Most testing tools, including Sikraken, depend on various parameters that can influence the efficiency of the search (both in its completeness, run time and memory usage). While it can be easy to optimise a tool against a particular target C code via trial and error, this process is time-consuming and not generalisable to a benchmark containing thousands of target C programs.

Sikraken has two modes, Regression and Budget. In Regression mode two values (restarts, tries). Tries will attempt to generate a number of test vectors from the current path before abandoning it, while restarts will restart from the root with a different path, the number of times specified by the user. Budget mode assigns a value in seconds for the test to run, and will cease when the value assigned is reached.

The project aims to develop a helper tool to test Sikraken against benchmarks under various combinations of arguments for regression mode and to help devise a better search algorithm to find the optimal $restarts$tries combination. The tool needs to be fully automated, and due to its high computing requirements, when running multiple test suites at once, will need to incorporate Multi-threading to speed up the processing.

Research:

Research done for Sikraken optimisation includes:

- Linux operating system and configuration
- Installation of Sikraken and other necessary packages/coding languages in Linux
- Detailed understanding on various optimisation algorithms such as the Genetic Algorith, Hessian Matrix, Simulated Annealing or The Hill Climbing Algorith
- Benchmarking tools such those used in Testcomp
- Consistent Runtime using external tools such as BenchExec
- Coverage Measuring using TestCov
- Generating graphs/charts from benchmark results
- Parallelization using python

Non Functional Requirements:

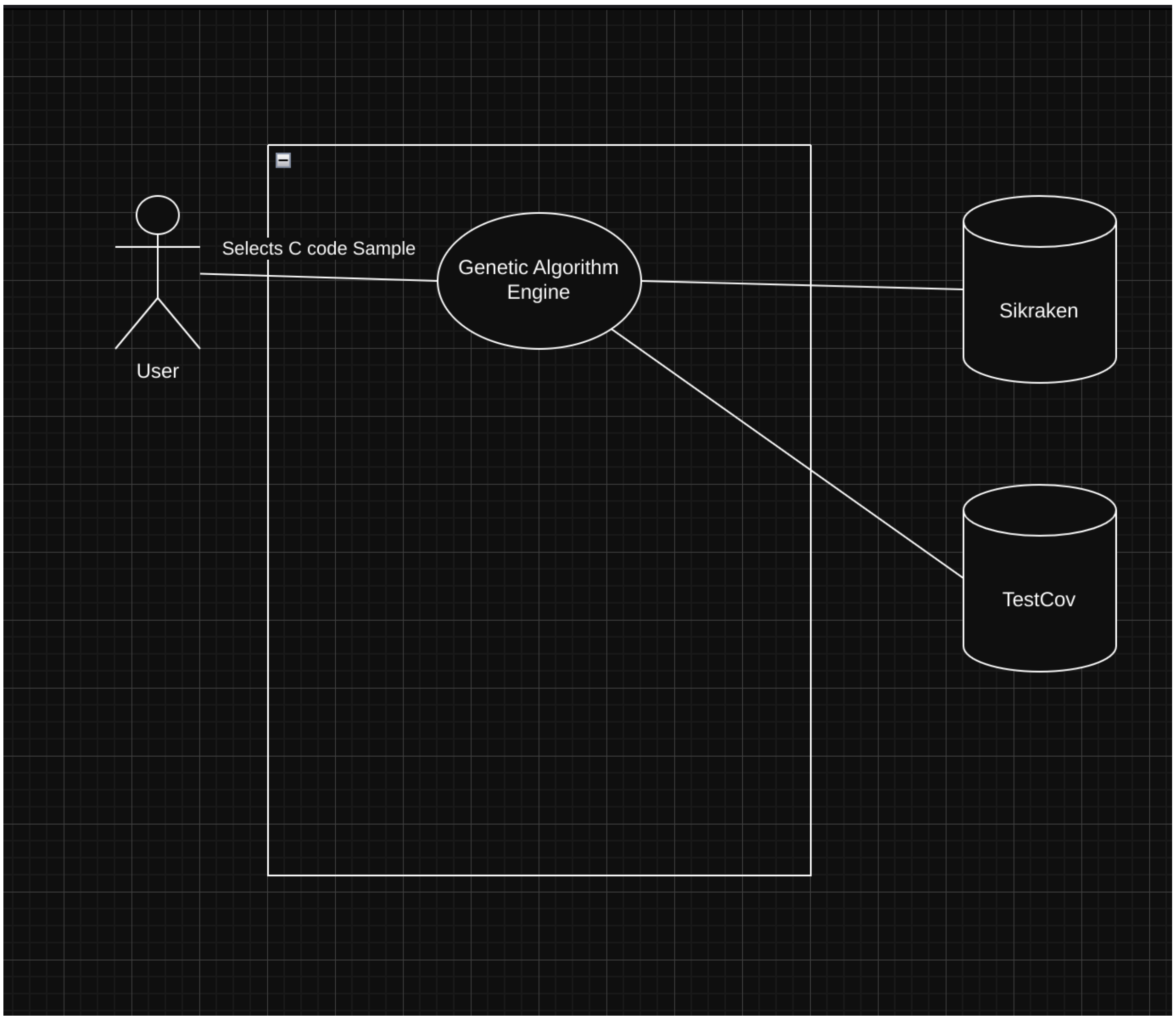
1. Visualization
    - Visual representation of genetic algorithm progress
    - Interactive parameter selection
    - Result visualization

2. Extended Parameter Optimization
    - Support for additional Sikraken parameters beyond `$restarts` and `$tries`
    - Multi-objective optimization considering both coverage and execution time

Functional Requirements:

Utilizing the Genetic Algorithm the tool needs to be able to find the optimal combination of $restarts$tries for regression mode. Currently the user has to blindly guess these two integers and run regression mode a few times and see the results in the separate log file. Sikraken User Assist tool needs to be able to find the optimal combination of these 2 values for a given C code sample utilizing machine learning.

Sikraken Optimizer Use Case Diagram:



Research Foundation

The project builds on the research in evolutionary computation, test case generation, and parameter optimization:

1. Evolutionary Computation
   - Genetic algorithms for parameter optimization
   - Selection mechanisms for fitness improvement
   - Crossover and mutation strategies for search space exploration

2. Testing Theory
   - Code coverage metrics and their significance
   - Automated test case generation approaches
   - Parameter sensitivity in test generation

3. Optimization Techniques
   - Parallel processing for efficiency gain
   - Multi-objective optimization considerations
   - Handling of resource constraints

The theoretical foundation combined with practical implementation creates a powerful tool for enhancing Sikraken's test generation capabilities through automated parameter optimization.

Key Features:

- Parameter Optimization - Utilises the genetic algorithm to find optimal parameter combinations for Regression mode ($restarts,$tries) without the user having to guess the combinations of the two parameters.
- Benchmarking with BenchExec - Provides consistent runtime analysis and performance normalisation for reliable benchmarking.
- Parallelization - Utilizes multi-threading to speed up the algorithm processing significantly.
- Coverage Metrics - Integration with TestCov enables detailed branch coverage analysis, while Testcomp facilitates the benchmarking of testing results.

References:

Sikraken Development and User Guide (Dr Chris Meudec - 2024). Research strategy [online]. Available from: https://docs.google.com/document/d/1uDLnlrFGWUNYyzsotZAZ_jFVrRSPEoixVMgw1UZZ0ug/edit?tab=t.0[accessed 25 October 2024].

uk.mathworks.com (2024). Research strategy [online]. Available from: https://uk.mathworks.com/videos/what-is-a-genetic-algorithm-100904.html[accessed 25 October 2024].

wikipedia.org/Hessian_matrix (2024). Research strategy [online]. Available from: https://en.wikipedia.org/wiki/Hessian_matrix[accessed 25 October 2024].

wikipedia.org/Simulated_annealing (2024). Research strategy [online]. Available from: https://en.wikipedia.org/wiki/Simulated_annealing[accessed 25 October 2024].

geeksforgeeks.org(2024). Research strategy [online]. Available from: https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/[accessed 25 October 2024].

test-comp(2025). Research strategy [online]. Available from: https://test-comp.sosy-lab.org/2025/[accessed 25 October 2024].

github.com/sosy-labs/benchexec(2025). Research strategy [online]. Available from: https://github.com/sosy-lab/benchexec/tree/main[accessed 25 October 2024].