Parallel Tempering for Stochastic Superoptimization

Mark Dubynskyi, Raghu Guggilam, Safiuddeen Salem, Eric Zipor, mentored by Dr. Michael Jarret and Anthony E. Pizzimenti

Introduction

Background: Stochastic search methods, such as simulated annealing (SA) and parallel tempering (PT), are powerful tools for minimizing complex functions.

- **SA:** Gradually reduces exploration ("riskiness") to converge, but may accept suboptimal solutions to escape local minima.
- **PT:** Runs parallel searches at different "temperatures", swapping states to improve global exploration.

Adaptation to compiler optimization: We apply SA and PT to compiler-based logic synthesis, comparing their performance in large search spaces. Target metric includes: Instruction count, Execution time, Gate complexity.

STOKE: MCMC-Based Superoptimization

Key Ideas: ***STOKE*** is a stochastic superoptimizer that uses **Markov Chain Monte Carlo (MCMC)** to discover highly efficient x86 assembly code. Unlike traditional compilers:

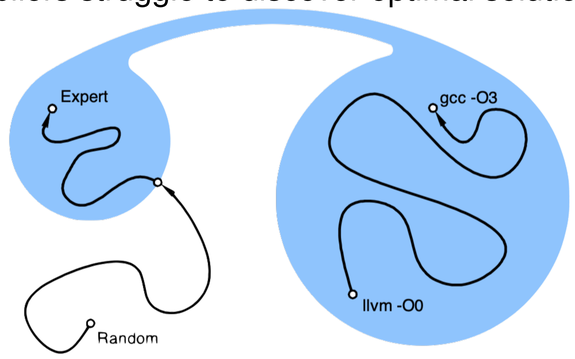
- Treats optimization as a ***stochastic search*** over loop-free code sequences.
- Explores non-obvious transformations missed by deterministic heuristics.

How It Works: STOKE iteratively:

- Proposes small code changes via MCMC sampling.
- Accepts/rejects changes based on correctness and performance.
- Escapes local minima, enabling global optimization.

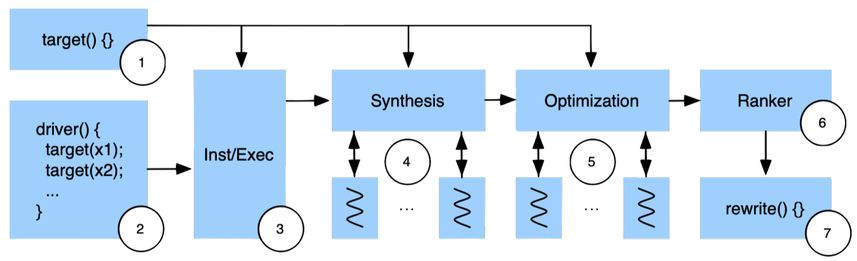
Implementation

Search Space Analysis: Figure 1 visually demonstrates how compiler-generated code (O0/O3) clusters in dense regions, while expert optimizations occupy isolated areas. This explains why traditional compilers struggle to discover optimal solutions.



System Architecture:
STOKE's pipeline (Fig. 2) operates in four phases:

1. Compile reference implementation
2. Generate test cases
3. Propose and refine rewrites using parallel MCMC
4. Validate and rank solutions



Optimization Method's Core: STOKE's search is guided by a dual-objective cost function

$$c(\mathcal{R}, \mathcal{T}) = \text{eq}(\mathcal{R}, \mathcal{T}) + \lambda \cdot \text{perf}(\mathcal{R})$$

where:

- eq verifies correctness (test cases $\rightarrow$ SMT)
- perf $= \sum \text{latency}(i)$ estimates performance
- MCMC accepts moves with $P = \min(1, e^{-\beta \Delta c})$

Parallel Tempering Enhancement

To improve STOKE's exploration capability, we replace simulated annealing with parallel tempering in the MCMC engine. This addresses SA's tendency to become trapped in local minima when optimizing complex assembly code spaces.

Key advantages of parallel tempering:

- Maintains multiple parallel searches at fixed temperature levels
- Enables periodic state swaps between temperature regimes
- Allows thorough high-temperature exploration before lower-temperature refinement
- Particularly effective for x86 assembly's high-dimensional search space

Challenges

- Requires fine tuned selection of temperatures and schedules
- Determining appropriate search length for an amount of replicas is problem-dependent

Acknowledgements

We want to thank Anton Lukyanenko, Swan Klein, Kelsi Listman, Tim Banks, and all those who run and support MEGL.

References

[1] Eric Schkufza, Rahul Sharma, and Alex Aiken. "Stochastic superoptimization". In: ***SIGPLAN*** (2013), pp. 305–316. DOI: 10.1145/2499368.2451150.