THE GENERAL IDEA

Genetic Improvement

Source Code

Computer System

Modifies

Modifies

Runs on

Measurements guide
The General Idea

Typically, in GI we optimize software with respect to a static target computer system.
The General Idea

We could also use GI to optimize a computer system with respect to a static software target.

Isn’t source-code the definition of computer system behavior?
Plenty of applications exist that run intensely and regularly on a computer system.

These computer systems could be optimized to these workloads. No need for general purpose systems. They could be specialized.
OTHER MOTIVATIONS

This is the SiFive Core Design Tool

Open Architecture projects such as RISC-V are making this easier.

Costs of silicon customization, “design to tape-out” are dropping dramatically.
AT WHAT LEVEL SHOULD WE WORK?

Logic gates, transistors, circuit-level?

A few cons:

- It’s been done!
- Run into scalability problems
- Customizing at this level is very expensive
AT WHAT LEVEL SHOULD WE WORK?

Clusters, computer networks, etc.?

A few cons:

- Difficult to simulate workloads sufficiently for optimization
- (I’m not really all that interested!)
AT WHAT LEVEL SHOULD WE WORK?

Computer architecture?

Some Pros:

- Common APIs and standards allow for interchanging components (think genes and alleles)
- We can utilize off the shelf components and designs.
- Plenty of research opportunity --- computer architecture designing and optimization is very manual.
HOW DO WE EVALUATE ARCHITECTURE DESIGNS?

SIMULATION
MODERN SIMULATORS

They provide a language for G1 to modify

Computer System Specification

Workload Specification

Statistics
MODERN SIMULATORS

That language follows a grammar like any other

```python
from gem5.components.boards.simple_board import SimpleBoard
from gem5.components.cachehierarchies.classic.no_cache import NoCache
from gem5.components.memory.single_channel import SingleChannelDDR3_1600
from gem5.components.processors.simple_processor import SimpleProcessor
from gem5.components.processors.cpu_types import CPUCores
from gem5.resources.resource import Resource
from gem5.simulate.simulator import Simulator

# Obtain the components.
cache_hierarchy = NoCache()
memory = SingleChannelDDR3_1600("16GiB")
processor = SimpleProcessor(cpu_type=CPUCores.ATOMIC, num_cores=1)

# Add them to the board.
board = SimpleBoard(
    clk_freq="300MHz",
    processor=processor,
    memory=memory,
    cache_hierarchy=cache_hierarchy,
)

# Setup the workload.
binary = Resource("s56-hello64-static")
board.set_se_binary_workload(binary)

# Setup the Simulator and run the simulation.
simulator = Simulator(board=board)
simulator.run()
```

The “variables” in a design are the components and their properties

For example, we can swap out the memory system with a different type (single channel or dual) or configure its size.
MODERN SIMULATORS

A good target for Grammar-based GP

Diagram:
- Design
  - CPU
    - ISA
    - Extens.
  - Memory
    - No. Cores
    - Size
  - Cache
    - L1 Size
  - OS
    - L2 Size
A FRAMEWORK

Computer System Specification

Workload Specification

Grammar-based GP

Statistics
WHAT'S NEEDING DONE?

STATS
Can we create simulations with good enough fidelity?

ACCURACY
What stats do we need from our simulator, what are we optimizing for??

COST
How do we estimate the cost of a design so we can determine the trade off?

MODEL
How do we do 1000s of evaluations when 1 can take hours?

SPEED

BENCHMARKS
What workloads should we optimize and are they meaningful?
ANY (NICE) QUESTIONS?

Bobby R. Bruce
bbruce@ucdavis.edu
https://www.bobbybruce.net