Mitigating the Compiler Optimization Phase-Ordering Problem using Machine Learning

Sameer Kulkarni & John Cavazos

University of Minnesota
Driven to Discover℠

Intro

- Why we need code optimization?
  - Prog. Language design flaw
    - e.g. goto statement in C/Cpp
  - People are evil and chaotic
    - e.g. define unused variables inside a loop

Intro

- What do we have so far?
    - Optimization option with fixed order

Intro

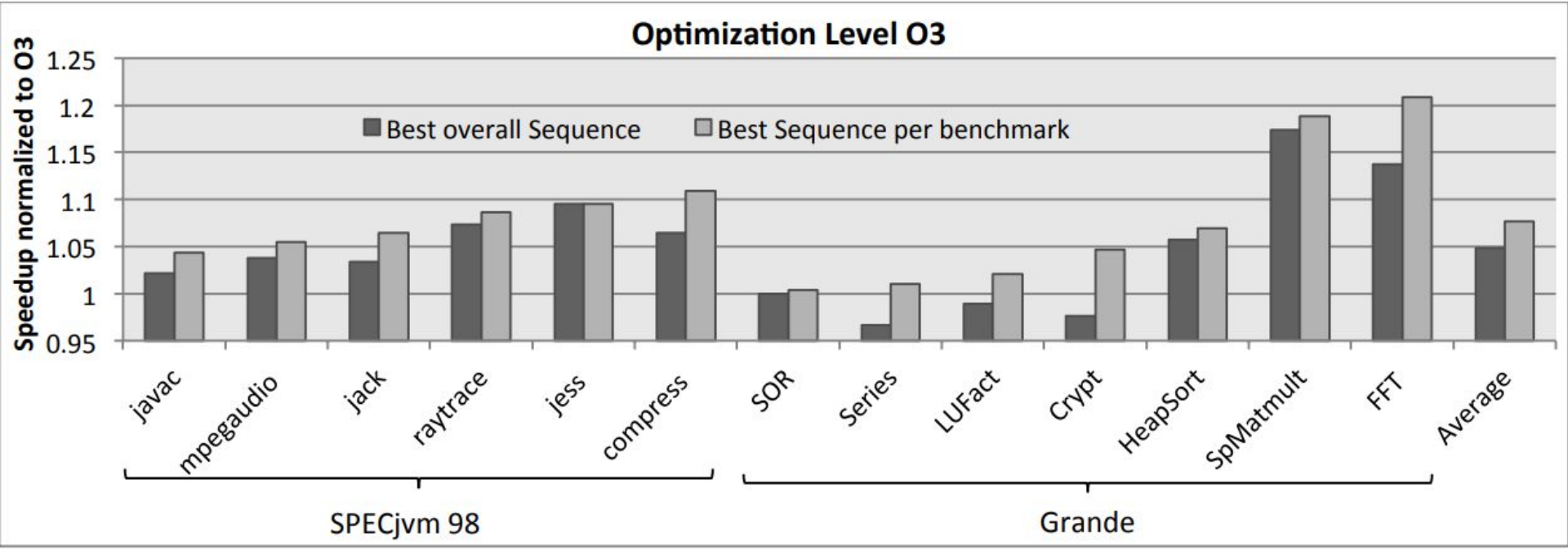
- What do we have so far?

OptKey	*Meaning*
Optimization Level O0	
CSE	Local common sub expression elimination
CNST	Local constant propagation
CPY	Local copy propagation
SA	CFG Structural Analysis
ET	Escape Transformations
FA	Field Analysis
BB	Basic block frequency estimation
Optimization Level O1	
BRO	Branch optimizations
TRE	Tail recursion elimination
SS	Basic block static splitting
SO	Simple optimizations like Type prop, Bounds check elim, dead-code elim, etc.
Optimization Level O2	
LN	Loop normalization
LU	Loop unrolling
CM	Coalesce Moves

Intro

- What do we have so far?
    - Optimization option with fixed order
    - Genetic algorithm that takes care of order

Results from GA



Problem with GA

- Expensive searching time for best result
- No fine-grain method level optimization
    - Unless you have method level timer

Possible solutions

- Predict the complete sequence

- Predict the current best optimization

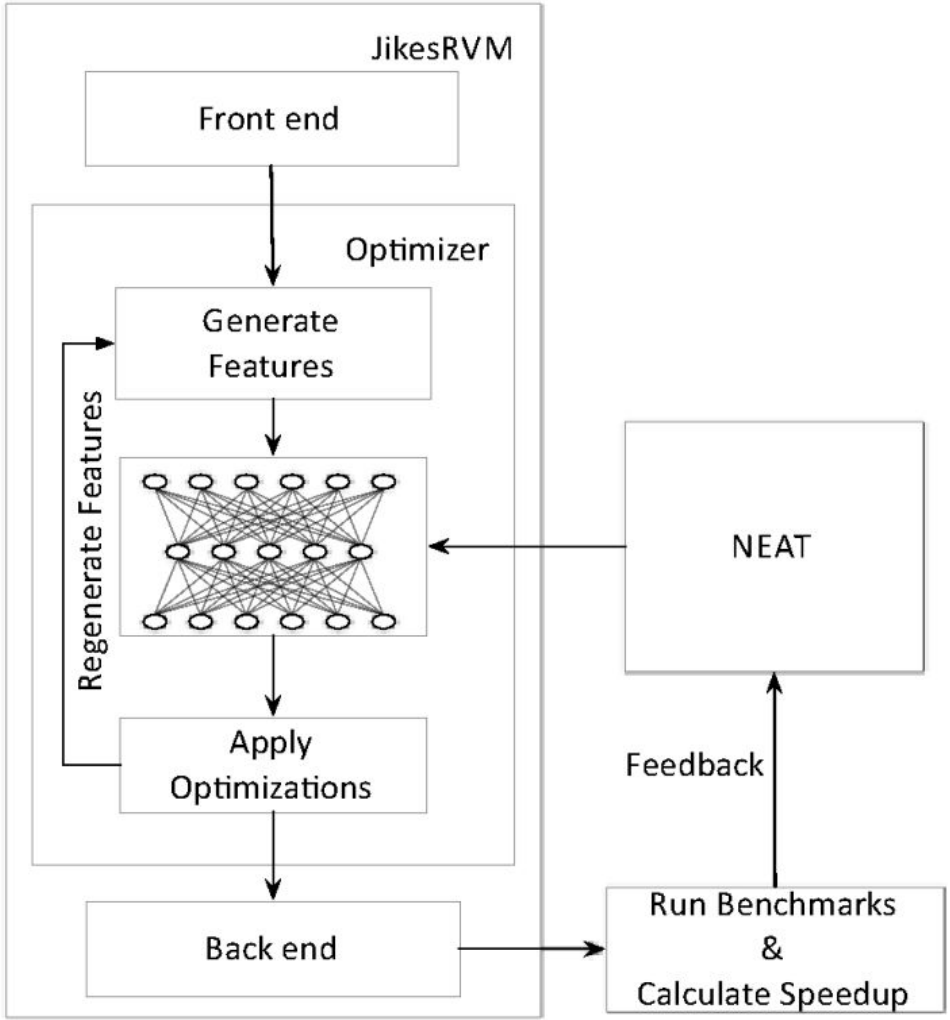
Possible solutions

- Predict the complete sequence
- Predict the current best optimization
  - Markov Property

NEAT

- Neuro-Evolution of Augmenting Topologies
  - GA with ANN flavor

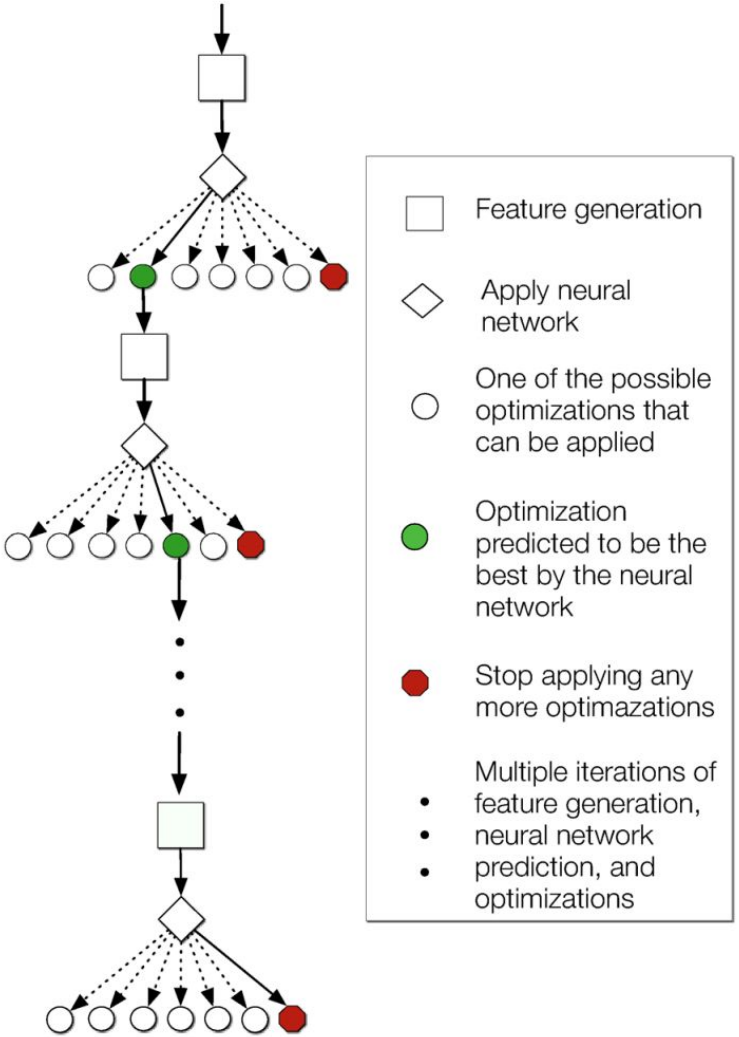
NEAT


JikesRVM
Front end
Optimizer
Generate Features
Regenerate Features
Apply Optimizations
Back end
NEAT
Feedback
Run Benchmarks & Calculate Speedup

The ANN of NEAT

- Takes feature as input, predict best optimization to apply
- Sometimes the second best option to avoid dead loop

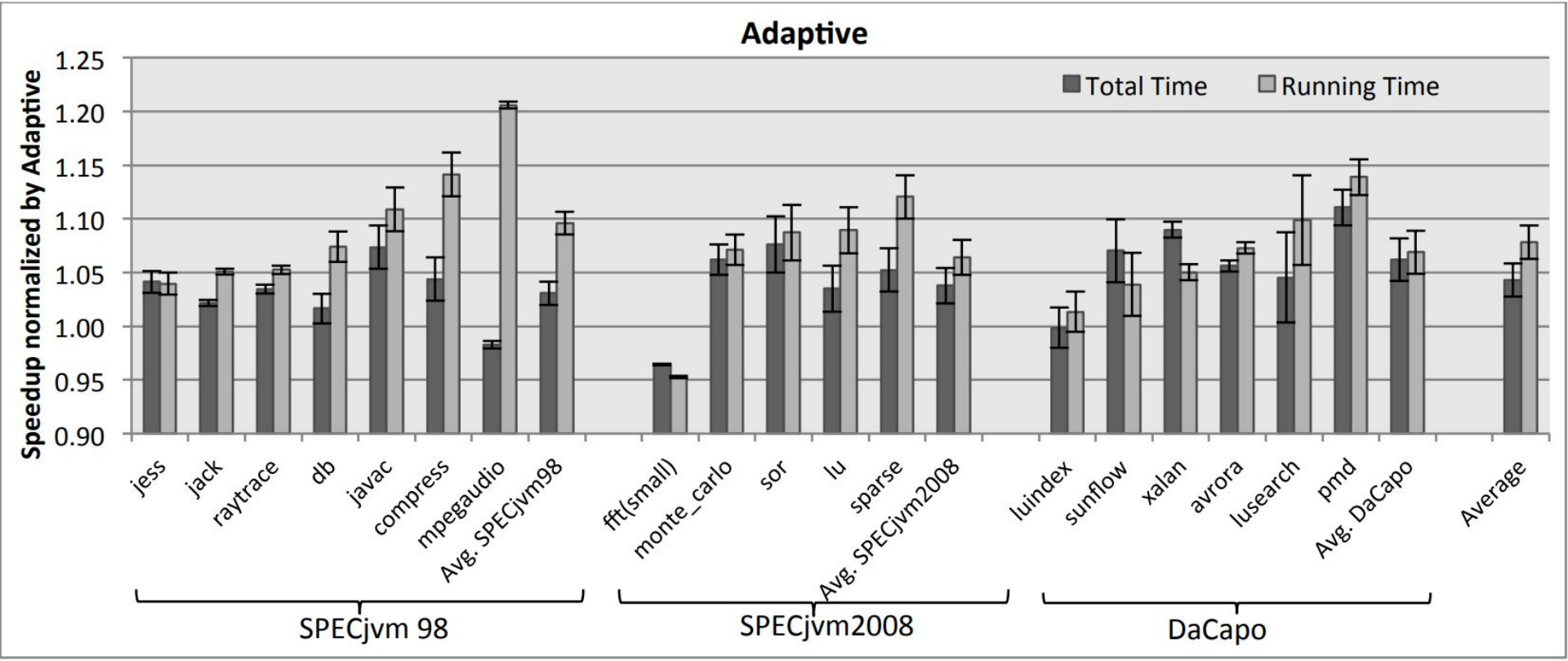
The ANN of NEAT



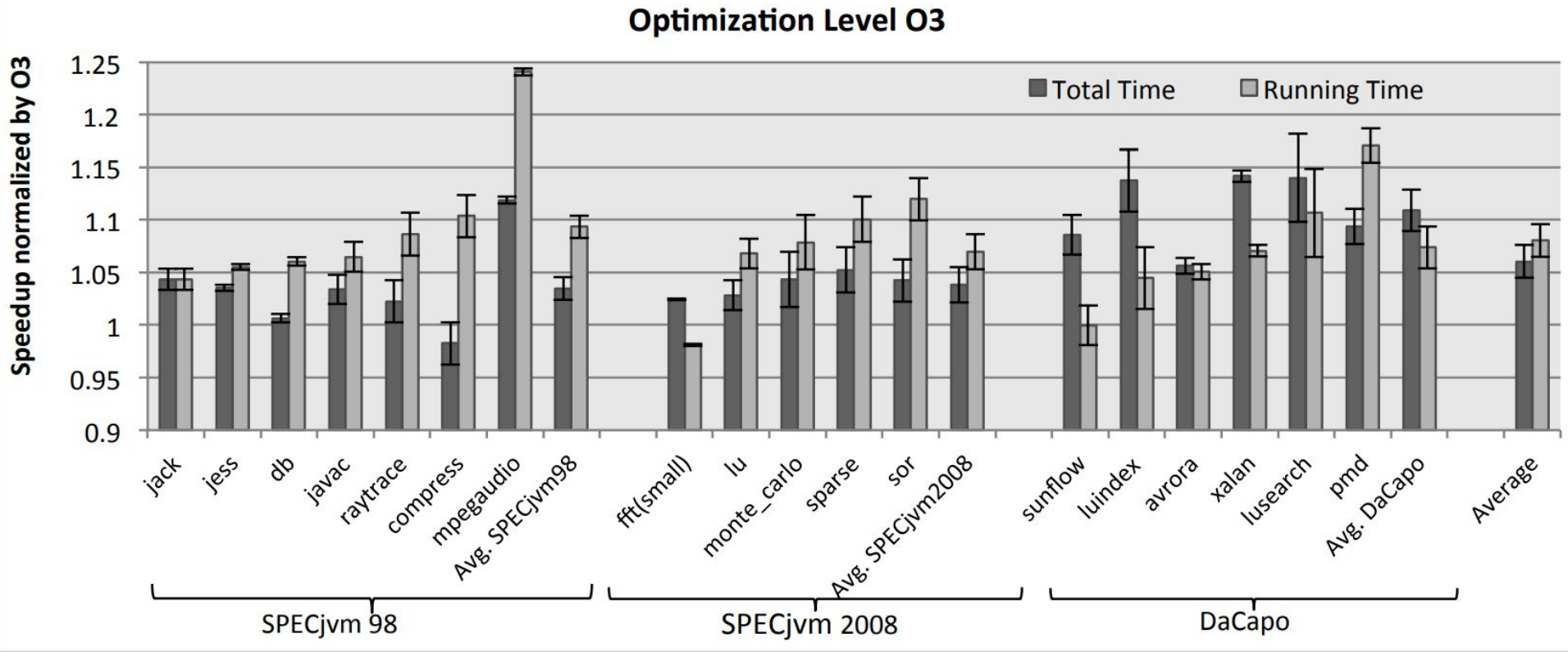
Implementation

- 60 ANNs each generation
- 300 generations
- Trained on seven benchmarks from the Java Grande benchmark suite
- Tested on SPECjvm98, SPECjvm2008, and DaCapo

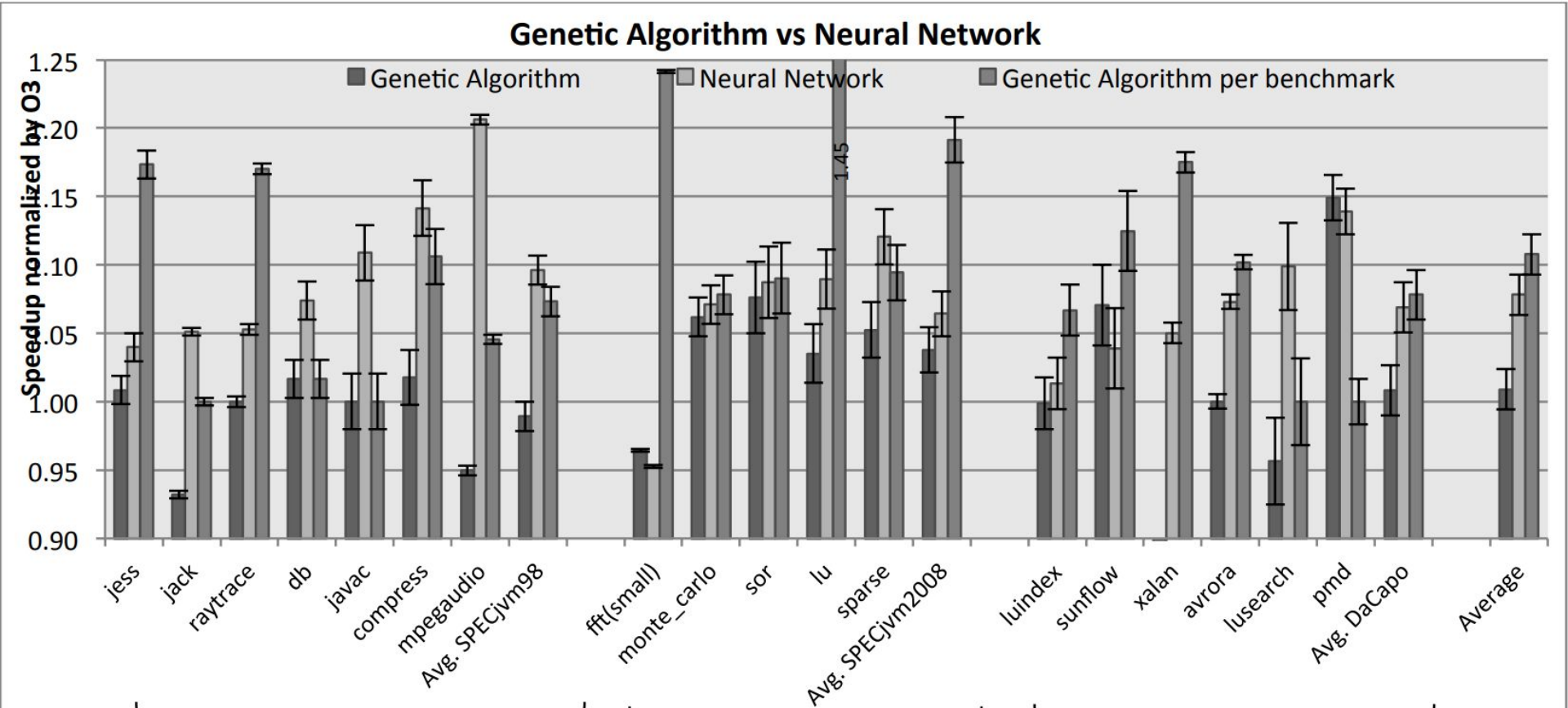
Result from NEAT



Result from NEAT



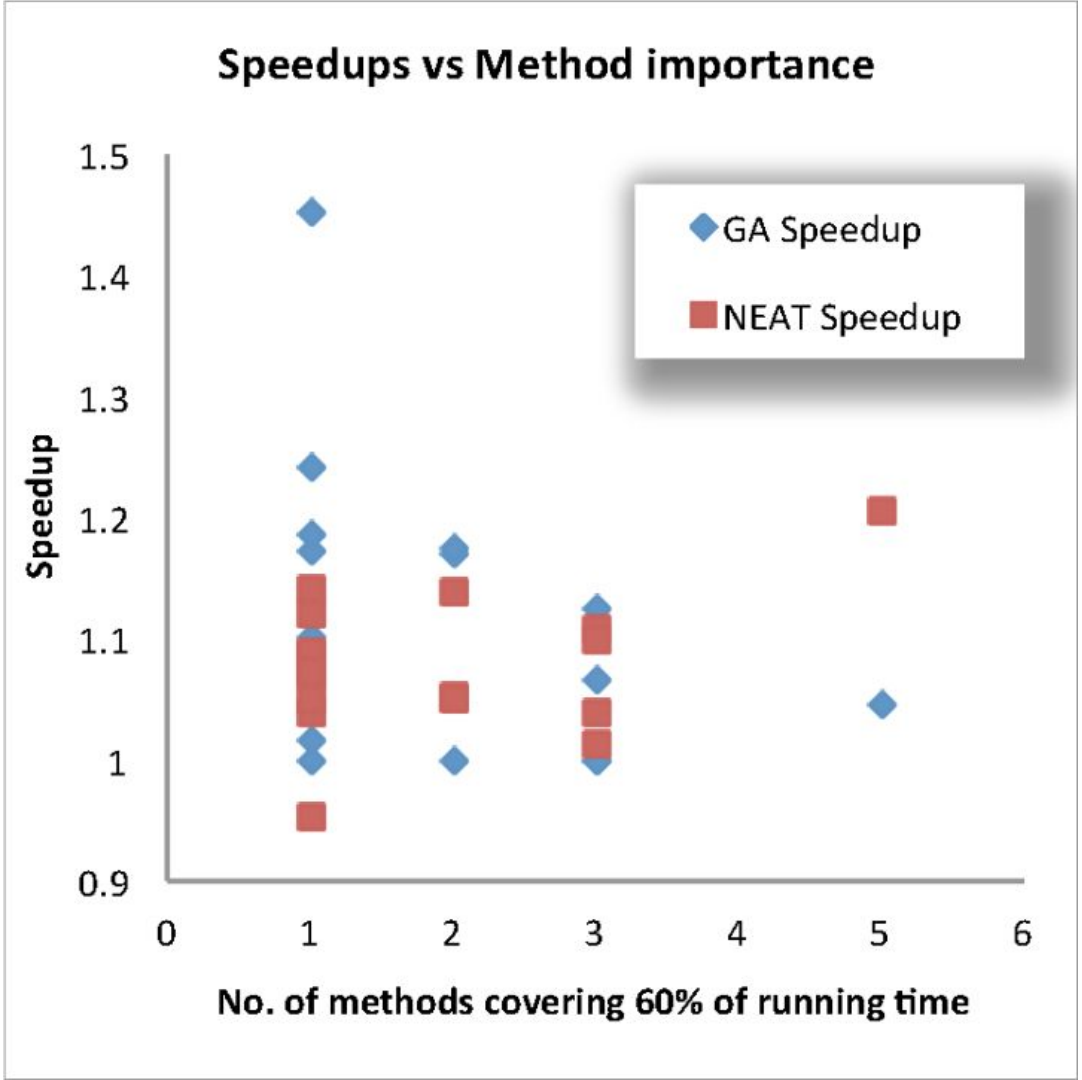
GA vs NEAT


Genetic Algorithm vs Neural Network
Genetic Algorithm
Neural Network
Genetic Algorithm per benchmark
Speedup normalized by O3
1.25
1.20
1.15
1.10
1.05
1.00
0.95
0.90
1.45
jess
jack
raytrace
db
javac
compress
mpegaudio
Avg. SPECjvm98
fft(small)
monte_carlo
sor
lu
sparse
Avg. SPECjvm2008
luindex
sunflow
xalan
avrora
lusearch
pmd
Avg. DaCapo
Average

GA vs NEAT

- Costs for training GAs and NEAT
  - GA : 11.4 days
  - NEAT : 13.2 days
  - GA per benchmark : 70 days

GA vs NEAT


Speedups vs Method importance
1.5
1.4
1.3
1.2
1.1
1
0.9
Speedup
GA Speedup
NEAT Speedup
0
1
2
3
4
5
6
No. of methods covering 60% of running time