Hybrid Optimizations: Which Optimization to Use?

John Cavazos

Institute for Computing Systems Architecture

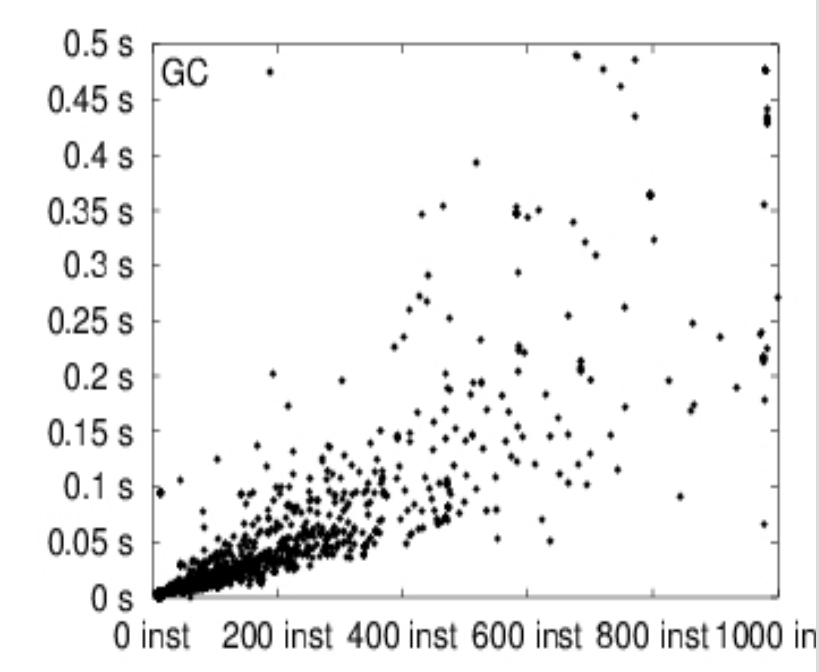
School of Informatics

University of Edinburgh, UK

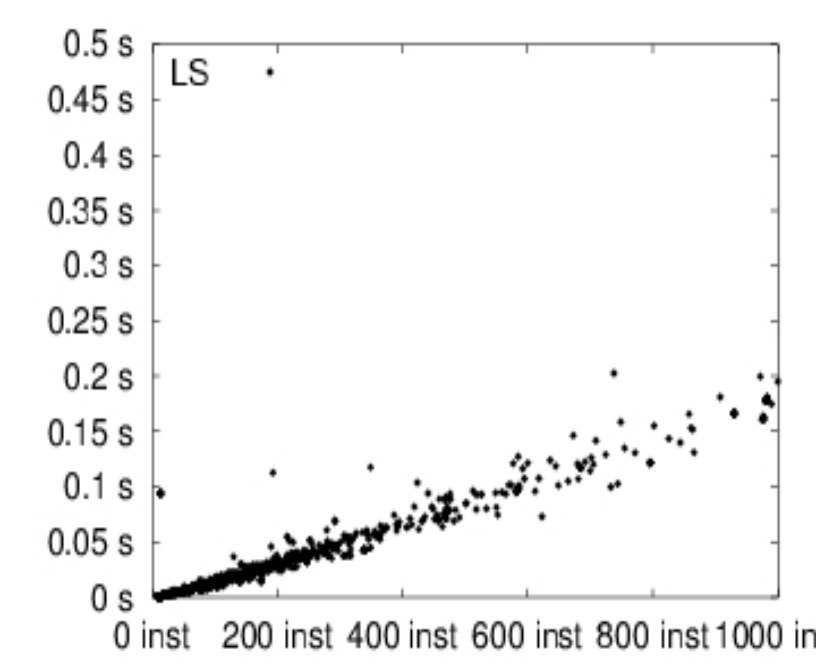
Motivation

- Register Allocation: important
  - Effective use of registers
- Different Algorithms to choose from
  - Graph coloring: possibly expensive
  - Linear scan: not always effective
- Important for Dynamic Compilation
- Which algorithm to apply?

Allocation Cost
Graph Coloring vs Linear Scan



Graph Coloring

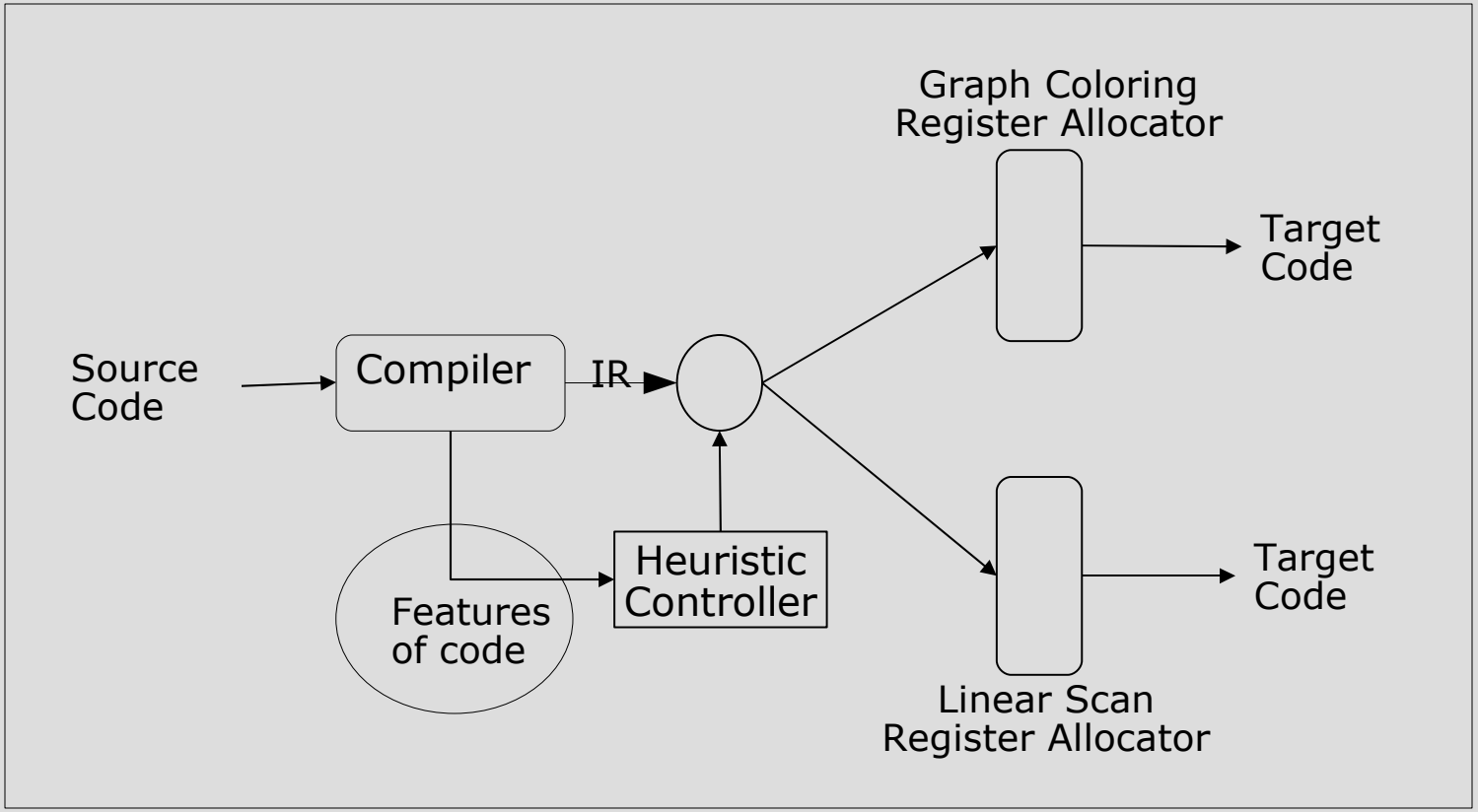


Linear Scan

Solution

- **Features** predict which algorithm to use
- Heuristic function controls allocator
  - Reduces **cost** significantly
  - Retains **benefit**
- Successful with simple features
- Applicable to other optimizations

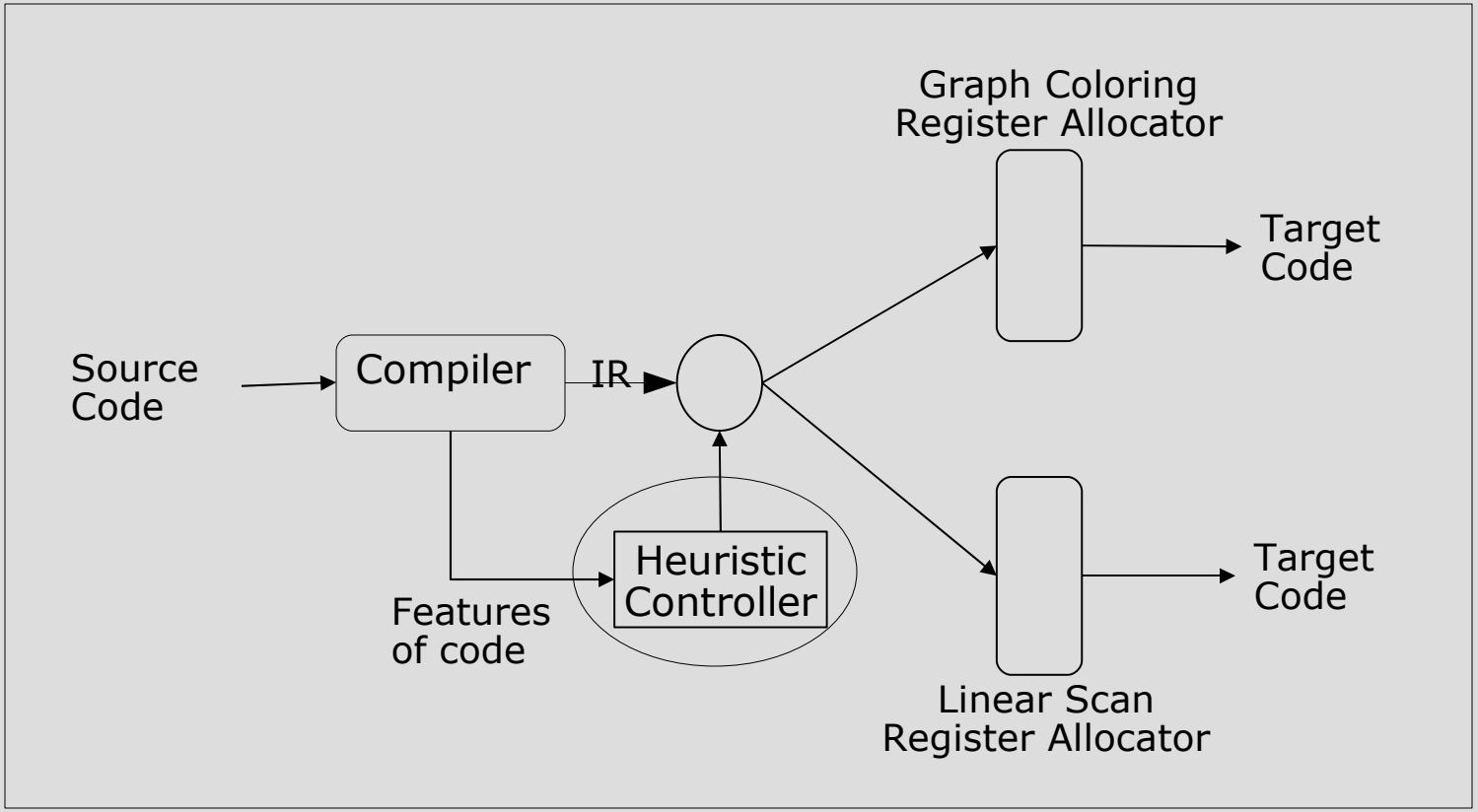
Hybrid Register Allocation



Features of Methods

Features	Meaning
Out, In, and Exception Out Edges	Out, in, and exception out edges in CFG (total, avg)
Live on Entry Live on Exit	Number of edges live on entry and exit (total, min, max)
Insts and Blocks	Number of instructions and blocks in method (total)
Block size	Size of blocks (max, min, avg)
Intervals	Number of live intervals (max, total, avg)
Symbolics	Number of symbolics (total, avg)

Hybrid Register Allocation



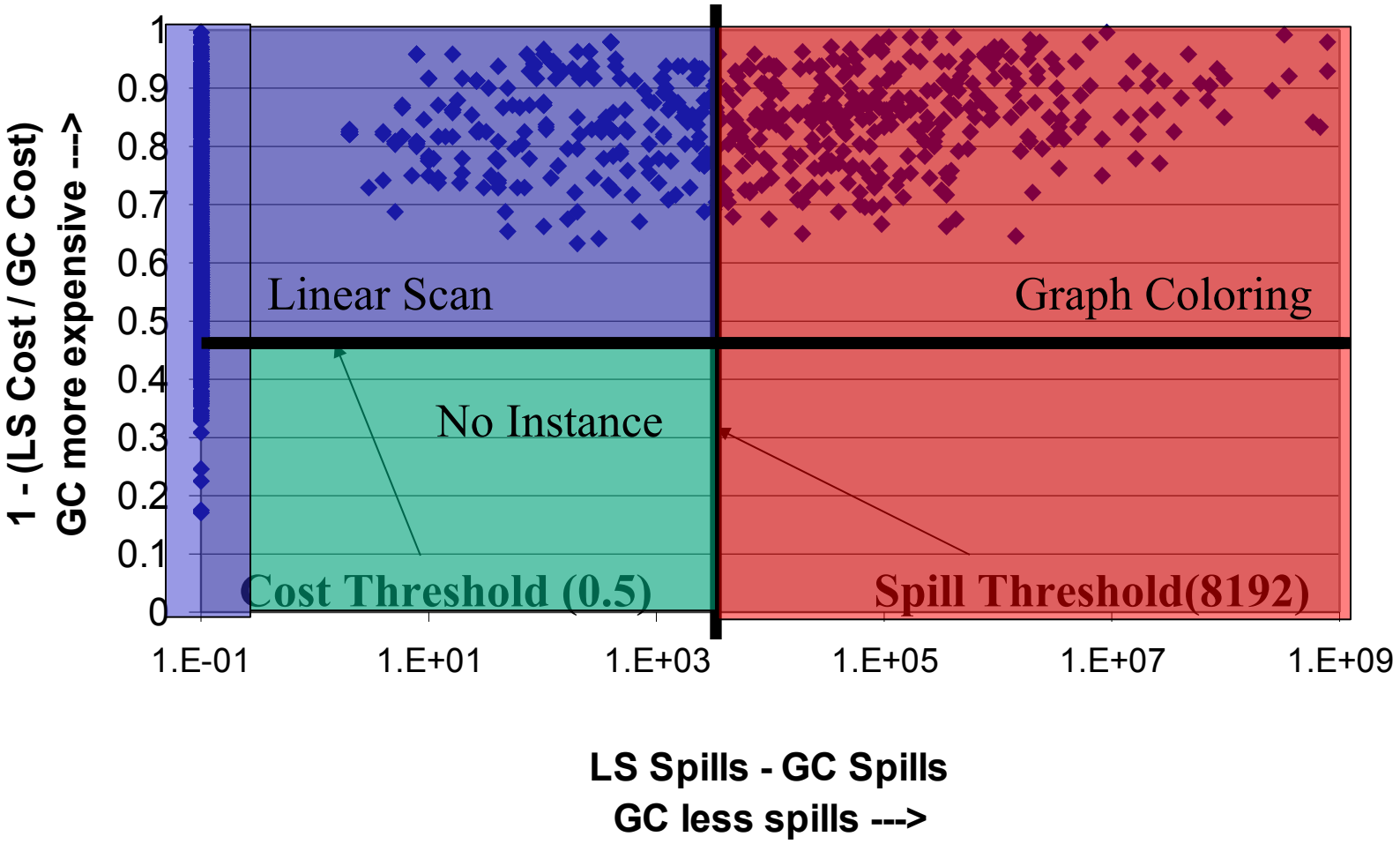
Inducing Heuristic Controller

- For each method generate **raw** training data
  - Features of method
  - Dynamic spills incurred
  - Cost of allocation algorithms
- Process raw data to generate training set
- Leave-one-out cross-validation
- Rule-Induction
- Learning output **heuristic controller**

Labelling Training Instances

- Two factors:
  - **Cost** of register allocation
  - **Spill benefit** of different allocators
- Prefer graph coloring
  - If **spill benefit** above threshold
- Prefer linear scan
  - If graph coloring **cost** above threshold
  - No **spill benefit**

Thresholding



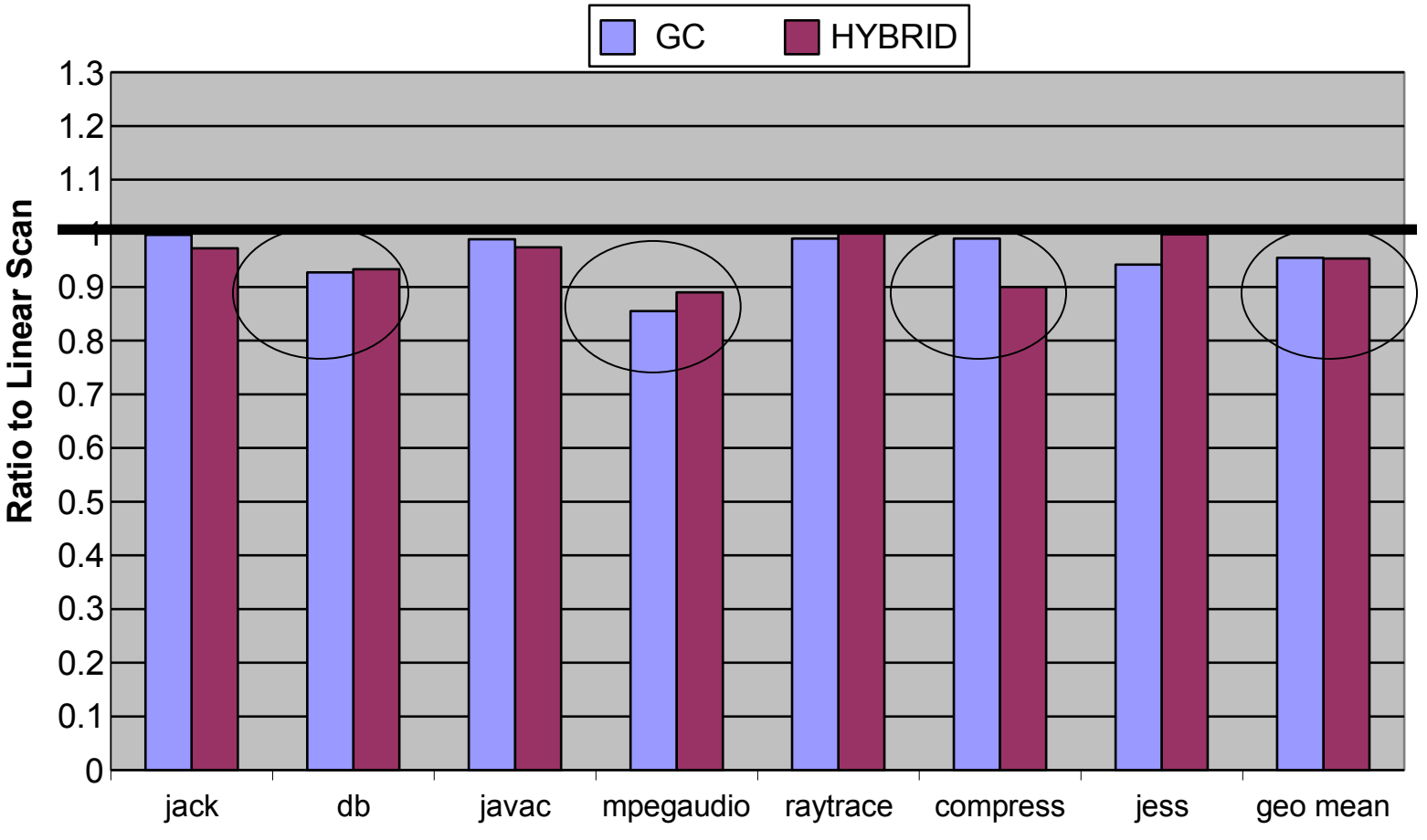
Motivation for Threshold Technique

- Noise reduction technique
- Simplifies learning
- Removes cases of fine distinction
- Separation by a threshold gap

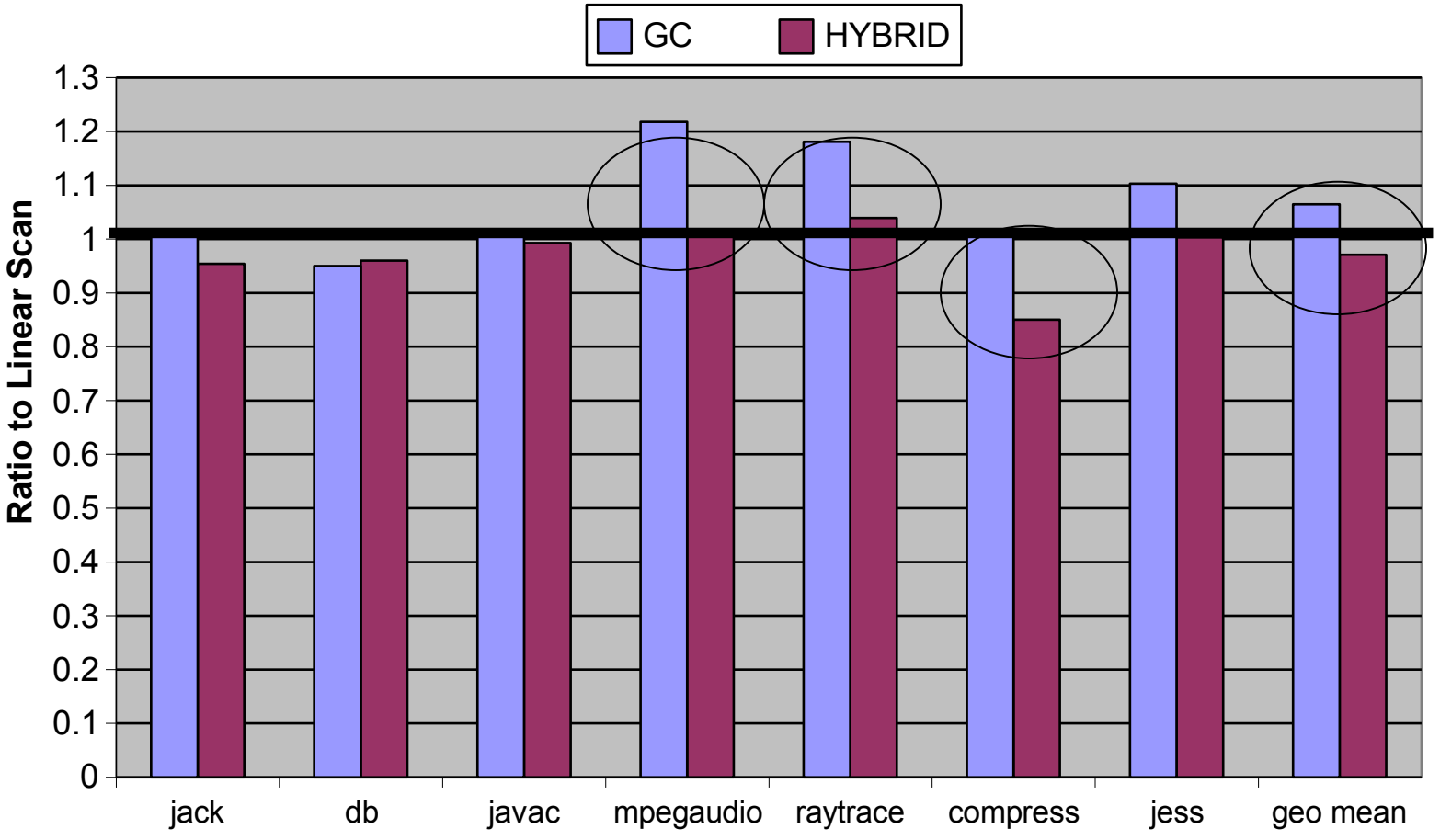
Experimental Setup

- Jikes RVM
  - JIT / Adaptive compilation
- PowerPC 533 MHz G4, model 7410
- 10 Registers (5 volatiles/5 non-volatiles)
- SPEC JVM benchmarks
- Running Time (NO compile time)
  - Min of 25 runs
- Total Time (WITH compile time)

Benchmark Running Times



Benchmark Total Times



Sample Induced Heuristic

GC ← avgLiveExitBB ≥ 3.8 ^ avgVirtRegBB ≥ 13

GC ← avgLiveEntryBB ≥ 4 ^ avgCFGInEdgesBB ≥ 1.4 ^
avgLiveExitBB ≥ 5.5 ^ numberInsts < 294

GC ← avgLiveExitBB ≥ 4.3 ^ maxLiveEntry ≤ 13

GC ← avgLiveExitBB ≥ 3.7 ^ maxLiveEntry ≥ 9 ^
numVirtReg ≥ 895 ^ maxLiveIntervals ≥ 38 ^
maxLiveIntervals ≥ 69

LS ←

Conclusion

- Hybrid Register Allocation successful
- Preserves Benefit Graph Colloring
- Reduces total (allocation) time

More Information

- COLO (**CO**mpilersThat **L**earn to **O**ptimise)
  http://www.anc.ed.ac.uk/machine-learning/colo/
- My Website:
  http://homepages.inf.ed.ac.uk/jcavazos/