Evaluating Benchmark Subsetting Approaches

Joshua J. Yi[1], Resit Sendag[2], Lieven Eeckhout[3], Ajay Joshi[4], David J. Lilja[5], Lizy K. John[4]

[1]Freescale Semiconductor
[2]University of Rhode Island
[3]Ghent University, Belgium
[4]University of Texas at Austin
[5]University of Minnesota

IISWC — Oct 26, 2006

Introduction

- Architects often select specific benchmarks to:
  - Reduce simulation time
  - Focus on specific characteristics (*e.g.*, memory behavior)
  - Build a benchmark suite

- Key challenge for selecting or subsetting benchmarks is:
  - To select a *representative* subset

Benchmark Subsetting Approaches

- Popular/emerging subsetting approaches include:
  - By principal component analysis (PCA)
  - By performance bottlenecks (Plackett and Burman)
  - By percentage of floating-point instructions (integer vs. floating-point)
  - Compute-bound or memory-bound
  - By programming language
  - Randomly

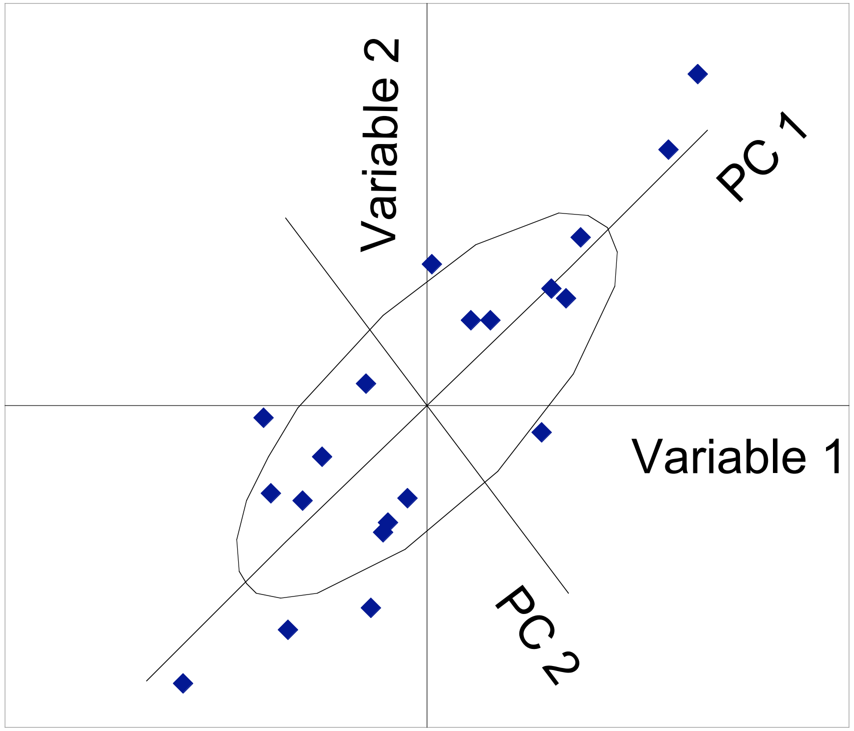
- But, which approach:
  - Produces the most accurate subset for a given subset size?
    - Absolute accuracy vs. relative accuracy
  - Produces the most accurate subset with the least profiling cost?
  - Most efficiently covers the space of benchmark characteristics?

Benchmark Subsetting Approach #1

- By principal component analysis (PCA):
  - Profile benchmarks to collect program characteristics
    - Instruction mix, amount of ILP, I/D footprints, data stream strides, *etc.*
  - Remove correlation between characteristics using Principal Component Analysis
    - Principal components are linear combinations of original characteristics
    - For more information on PCA, see [Eeckhout *et al.*, PACT 2002]
  - Cluster the benchmarks based on their principal components into N clusters
  - Select one representative benchmark from each cluster to form the subset

Removing Correlation using PCA

- Remove correlation between program characteristics
- Principal Components (PCs) are linear combination of original characteristics
- Var(PC1) ≥ Var(PC2) ≥ ...
- PC2 is less important to explain variation
- Reduce No. of variables
- Throw away PCs with negligible variance



$$PC1 = a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ...$$

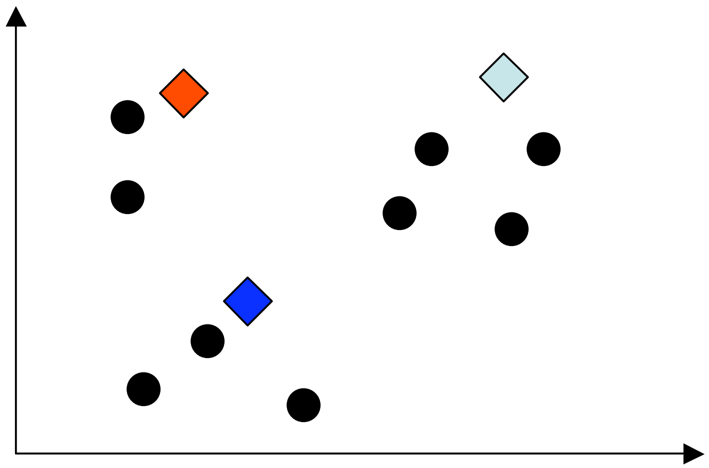
$$PC2 = a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ...$$

$$PC3 = a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + ...$$

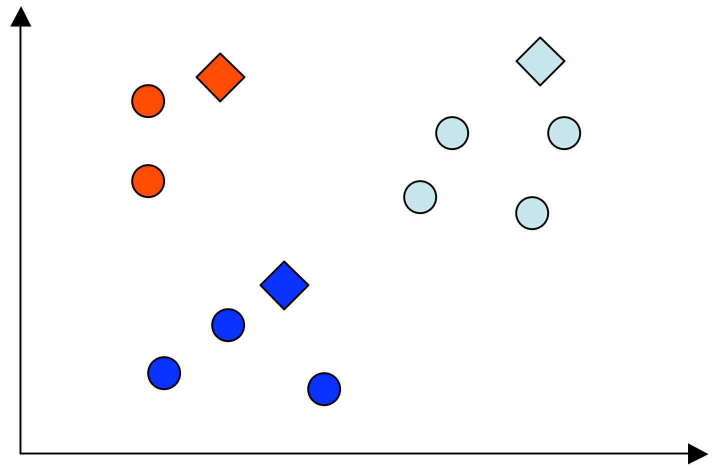
Clustering using k-means, Part 1

Cluster Analysis

K-Means Clustering algorithm



Step 1: Randomly select K cluster centroids

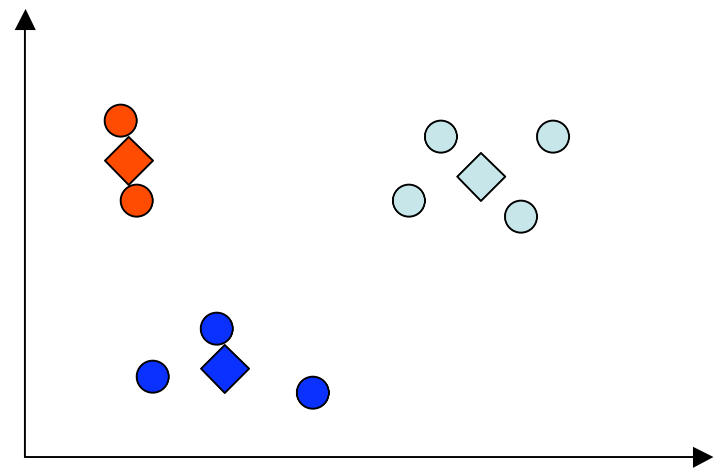


Step 2: Assign benchmarks to nearest cluster centroids

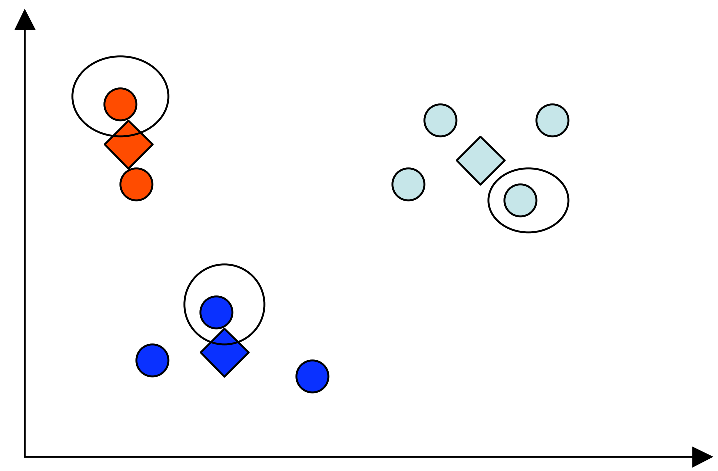
Clustering using k-means, Part 2

Cluster Analysis

K-Means Clustering algorithm



Step 3: Recalculate centroids and repeat Step 2 and 3 until algorithm converges



Step 4: Choose representative programs that are closest to the centroid of the clusters

Benchmark Subsetting Approach #2

- By performance bottlenecks (Plackett and Burman – P&B)
  - Use P&B design to quantify the magnitudes of all performance bottlenecks (CPI) in the processor and memory subsystem
    - Rank microarchitecture parameters based on their impact on overall performance
    - For more information on the P&B design, see [Yi *et al.*, HPCA 2003]
  - Cluster the benchmarks into N clusters based on:
    - Rank of magnitudes
    - Magnitudes
    - Percentage of CPI variation due to single bottlenecks
    - Percentage of CPI variation due to single bottlenecks and all interactions
  - Bottlenecks can be determined
    - Per benchmark
    - Across all benchmarks
  - Select one benchmark from each cluster to form the subset

Benchmark Subsetting Approaches #3 – #6

- By percentage of floating-point instructions (integer vs. floating-point)
  - SPECint vs. SPECfp
- Compute-bound vs. memory-bound
  - Compute-bound vs. memory-bound
    - Compute-bound: less than 6% L1 D$ miss rate for a 32KB cache
- By programming language
  - C vs. FORTRAN
- Randomly

Randomly choose benchmarks from each group
Form 30 different subsets for each group and report average results

Benchmark Subsetting Approach #7

- High-frequency
  - The de facto approach by computer architects
  - Form subsets based on descending order of frequency-of-use [Citron 2003, ISCA 2003 panel]
    - Choose most frequently used benchmark when subset size is 1
    - Choose two most frequently used benchmarks when subset size is 2
    - *etc*.

Methodology and Experimental Setup

- PCA profiling: ATOM
- Simulator:
  - SMARTS simulation framework (based on SimpleScalar)
    - U=1000 instructions, W=2000 instructions
    - 99.7% confidence interval, ±3% confidence level
  - P&B profiling: Added user-configurable latencies and throughputs

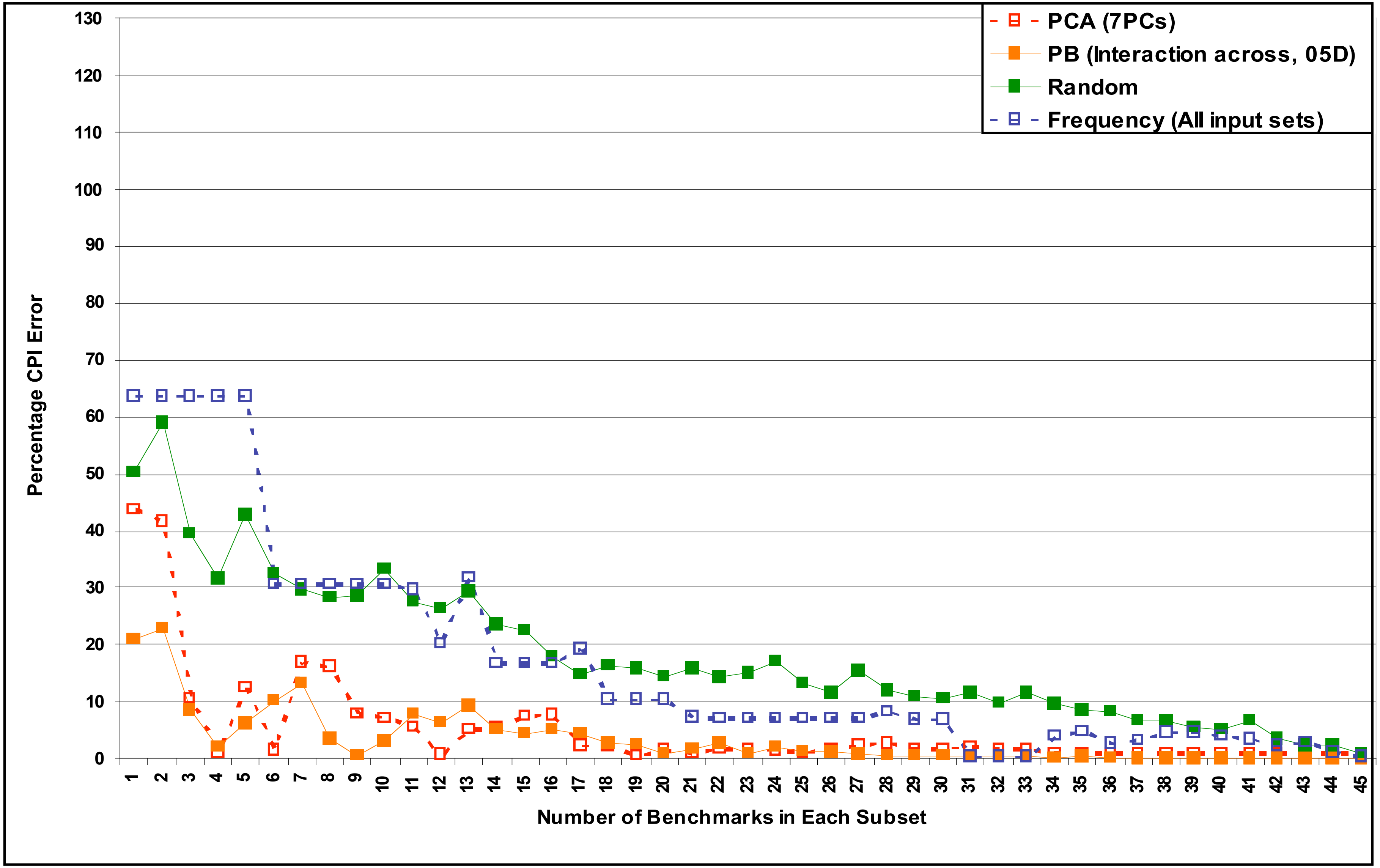
- Benchmark information
  - All SPEC CPU 2000 benchmarks and input sets
    - Except *vpr-place* and *perlbmk-perfect* crash SMARTS
  - Benchmark-input pair used synonymously with benchmark

- Processor configurations:
  - 4 4-way issue configurations, 4 8-way configurations
  - For each issue width, configurations represent range of configurations

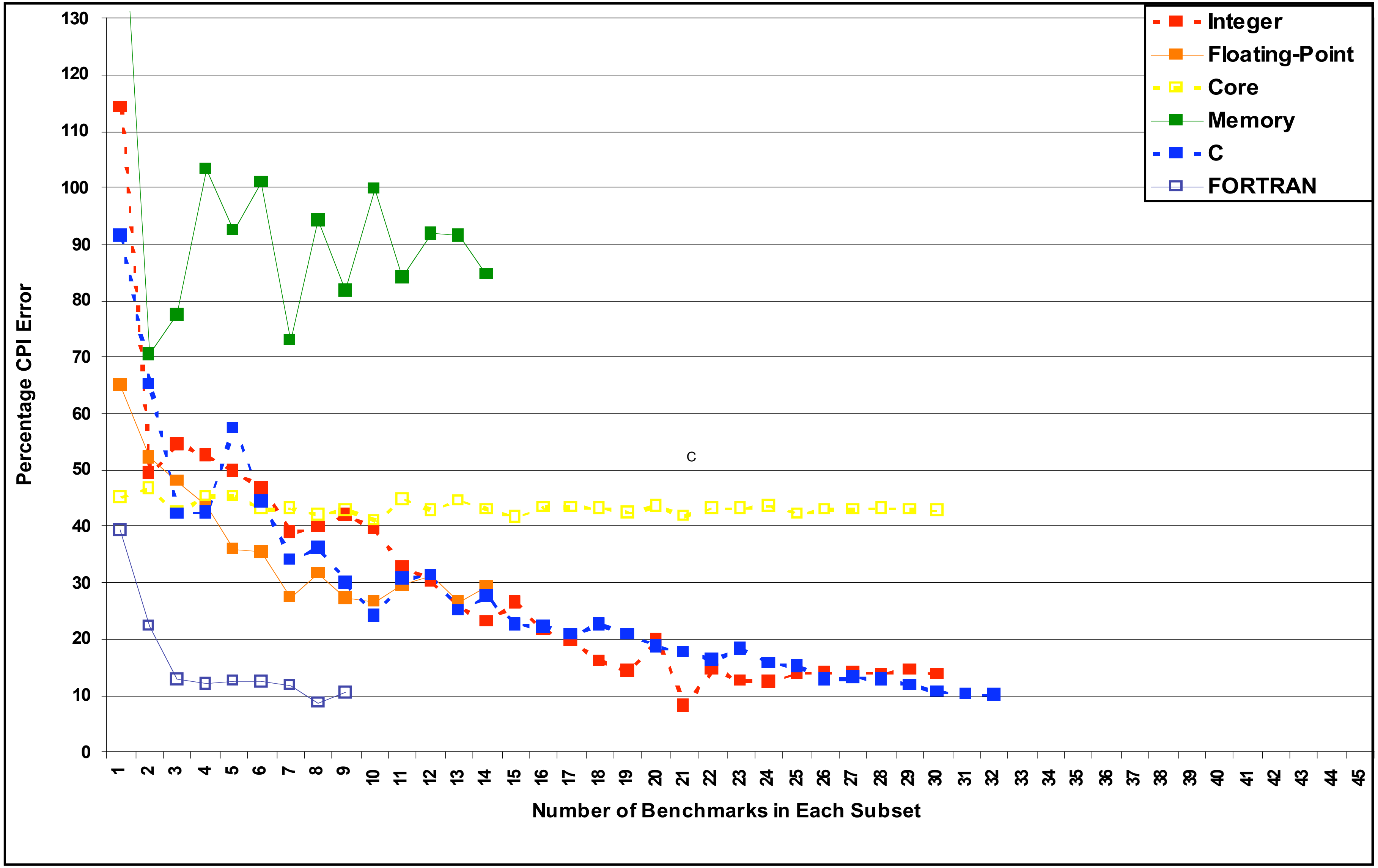
Quantifying Representativeness

- Absolute accuracy
  - Important when extrapolating results of subset for performance prediction of entire suite
  - Error in estimated CPI or EDP when using subset vs. full suite
- Relative accuracy
  - Important when comparing alternative designs during early design space exploration studies
  - Error in estimated speedup when using subset vs. full suite
- Coverage of the workload space
  - Important when selecting a subset of programs when designing a benchmark suite
  - Minimum Euclidean distance of the benchmark's characteristics of each subset away all individual benchmarks

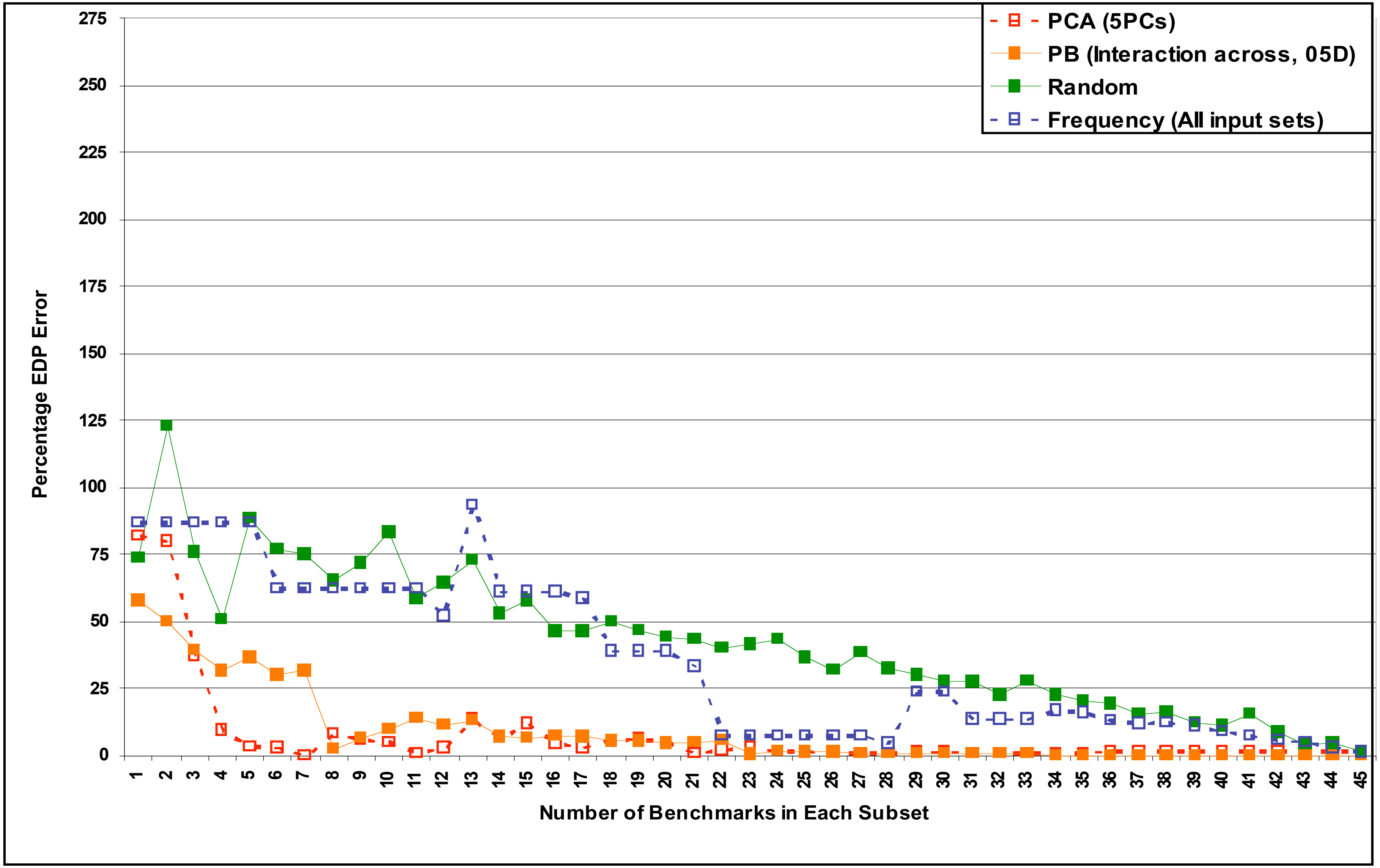
Absolute CPI Accuracy, Part 1



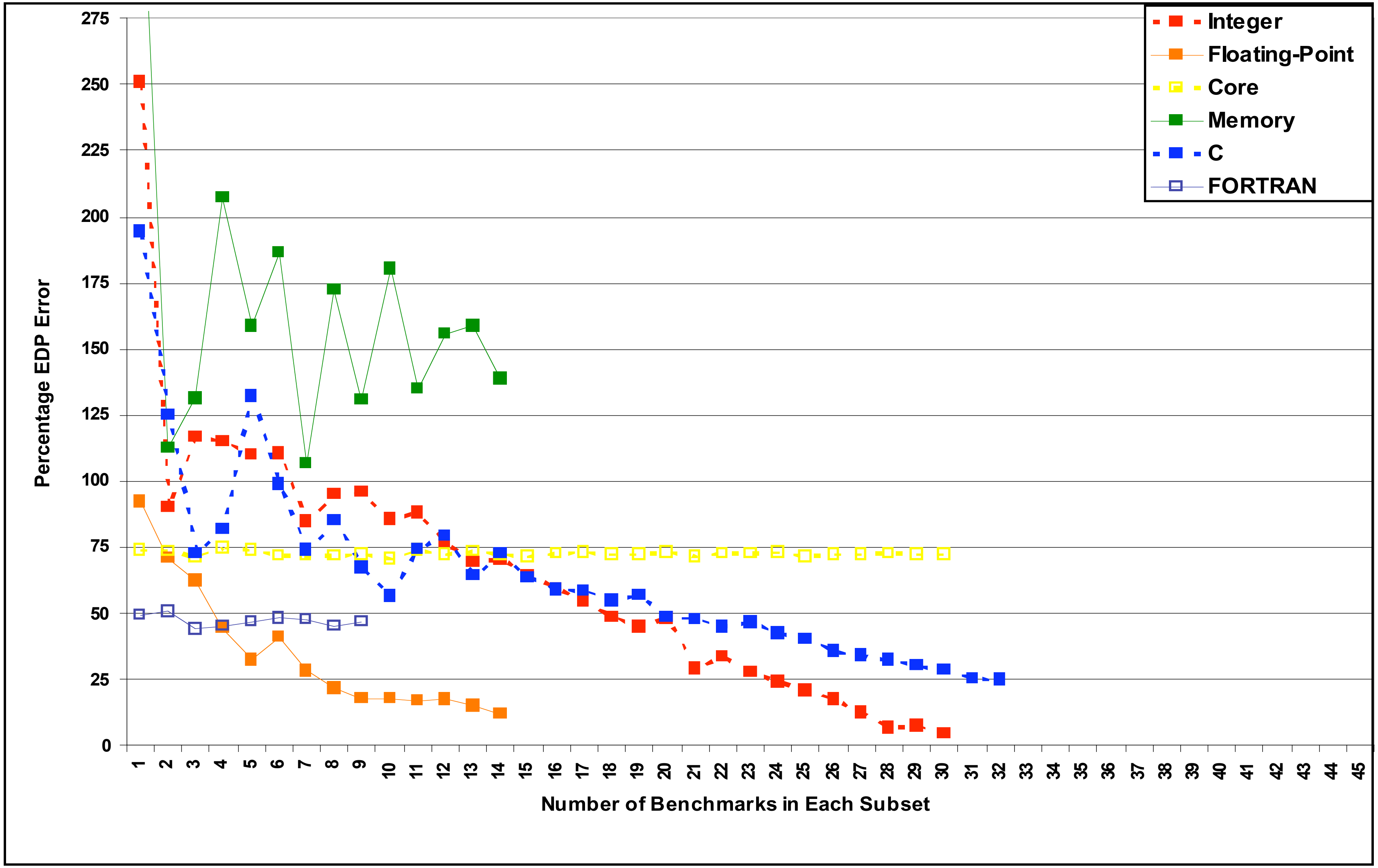
Absolute CPI Accuracy, Part 2



Absolute EDP Accuracy, Part 1



Absolute EDP Accuracy, Part 2



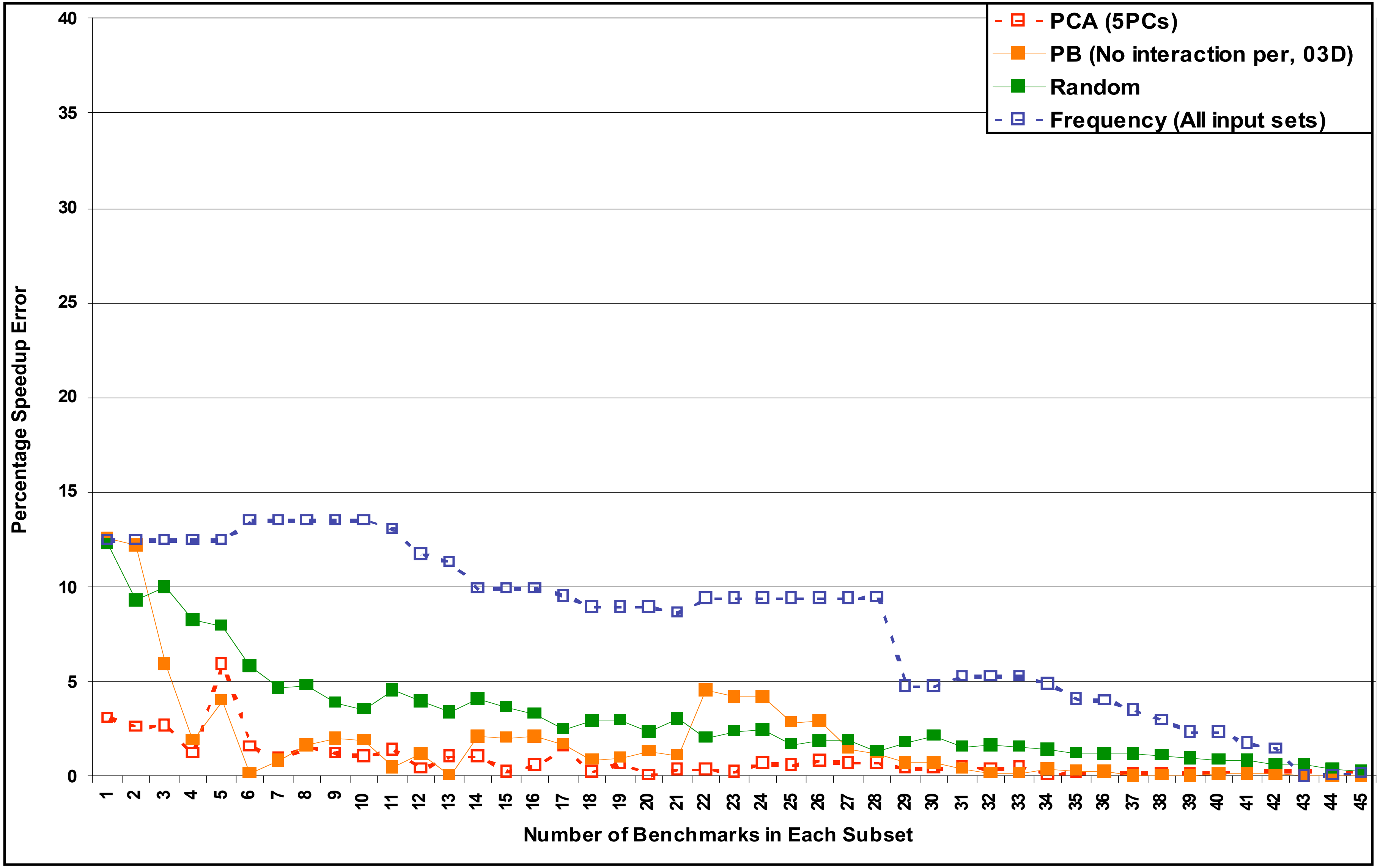
Key Conclusions for Absolute Accuracy

- Most accurate approaches:
  - PCA with 7 principal components
  - P&B using Top 5 bottlenecks
  - If want < 5% CPI error, need at least 17 benchmark-input pairs (1/3 of the entire suite)
- Int vs. float, compute vs. memory, language, and random approaches have poor and inconsistent CPI/EDP
  - Results based on these approaches may be misleading
- High-frequency approach
  - Overly optimistic DL1 and L2 cache hit rates
  - Some subsets may be pessimistic about branch prediction accuracy
- Statistical approaches are the most reliable way to subset benchmarks

Computing Relative Accuracy

- Compute the average speedup across entire benchmark suite for the following enhancements:
  - 4X larger ROB and LSQ
  - Next-line prefetching with prefetch buffers
  - 4X larger DL1 and L2 caches, 8-way associativity, same hit latency
- Compute the average speedup across benchmarks in each subset
- Compute speedup error when using a subset and when using the entire suite
  - Relative error = $(\text{Speedup}_{w/SS} - \text{Speedup}_{wo/SS}) / \text{Speedup}_{wo/SS} * 100$

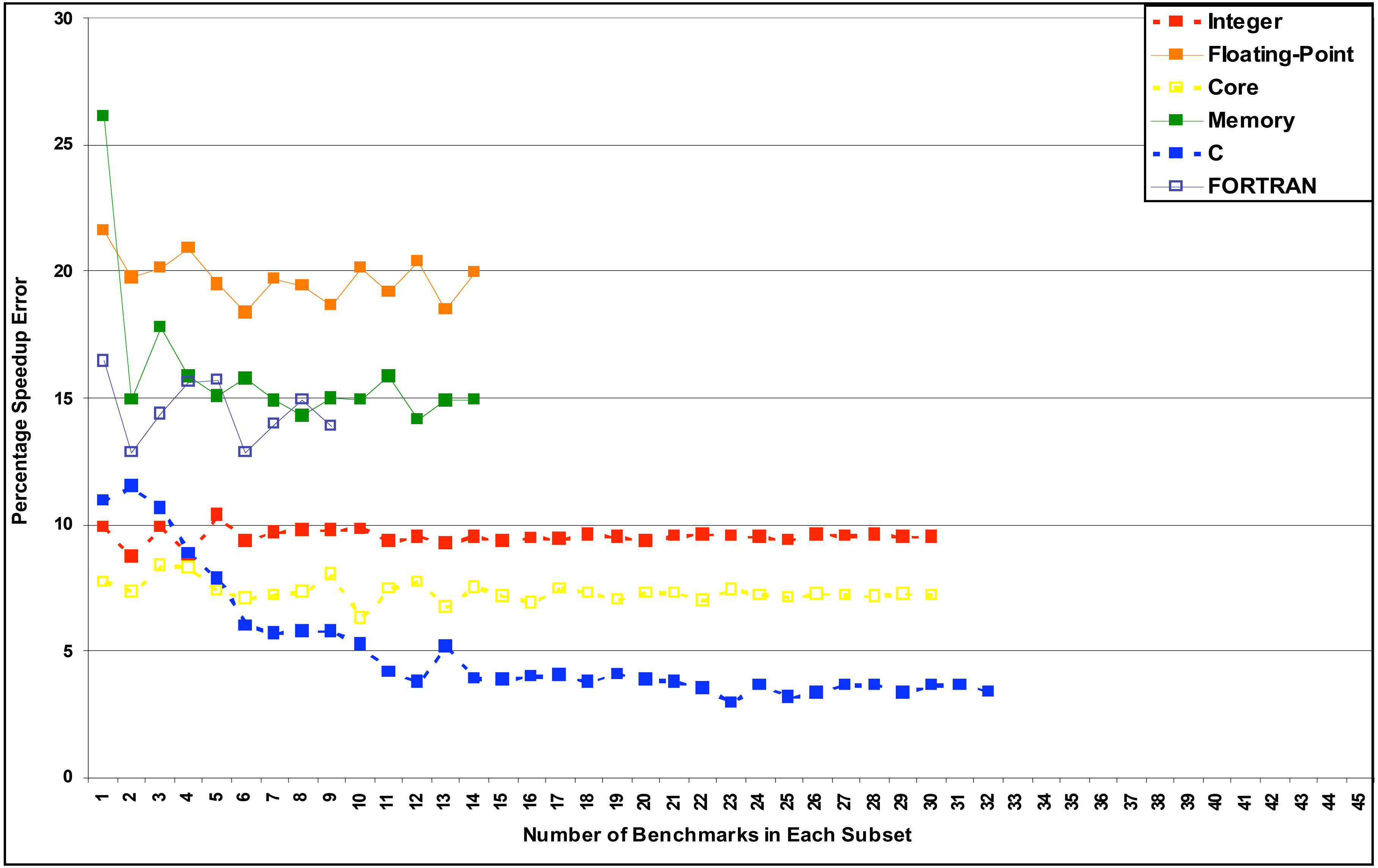
Relative CPI Accuracy (ROB), Part 1



J. Yi et al.

Freescale, Rhode Island, Ghent, Texas, Minnesota

Relative CPI Accuracy (ROB), Part 2



Key Conclusions for Relative Accuracy

- Conclusions are similar to those for absolute accuracy
  - PCA and P&B are more accurate
  - Other 5 approaches are not accurate
- Accuracy generally improves with larger subset sizes
- Similar results across all processor configurations
- Key difference: Relative error is lower than absolute error
  - Relative error is typically < 20% for most approaches/subset sizes
  - Absolute error is typically > 20% for most approaches/subset sizes
  - Less variation in CPI across configurations (*i.e.*, for relative accuracy) than across benchmarks (*i.e.*, for absolute accuracy)
    - Matched-pairs comparison

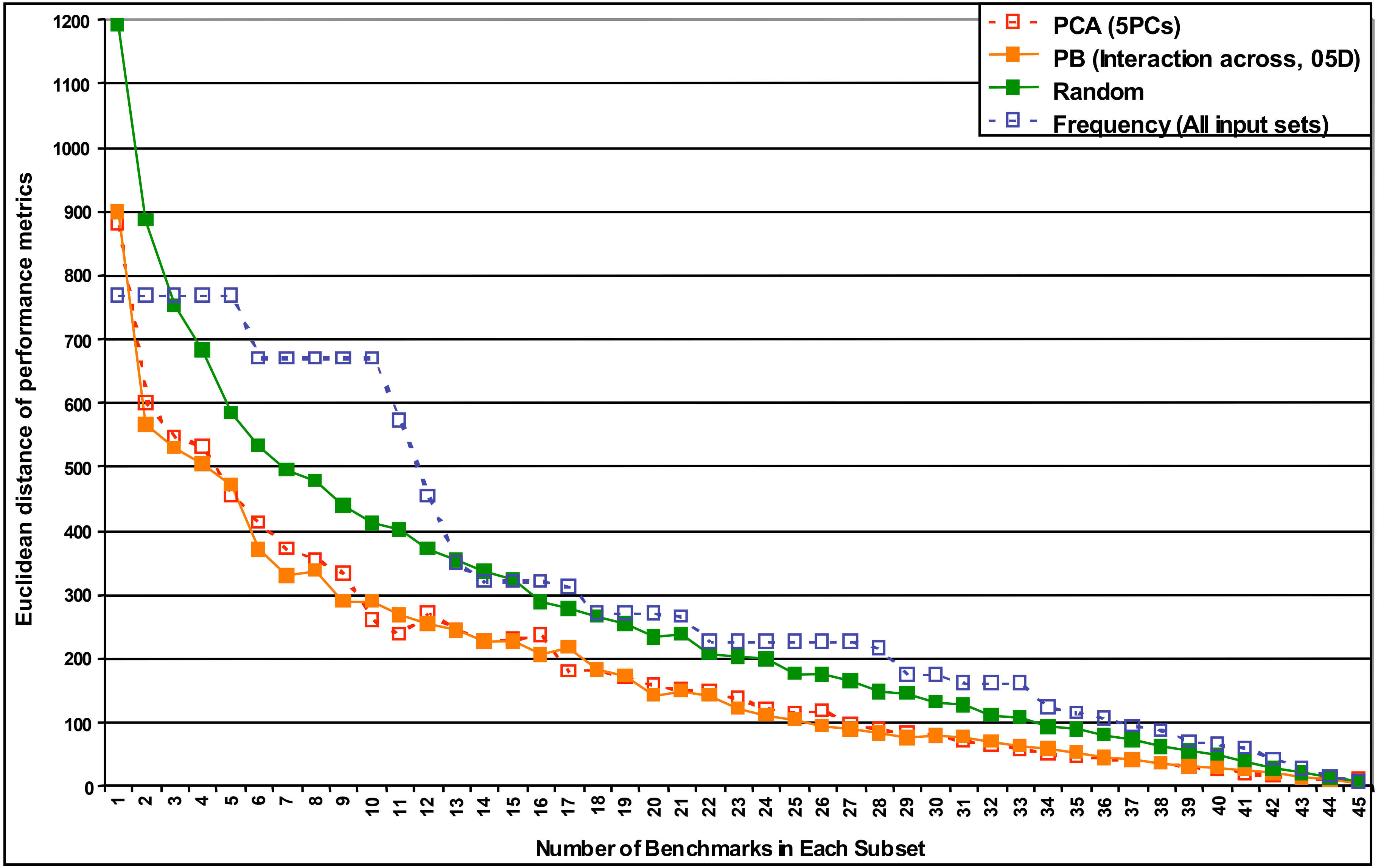
Computing Coverage

- Vectorize the performance and power metrics for each benchmark
  - Performance metrics: IPC; branch prediction accuracy; L1 D-cache, L1 I-cache, L2 cache hit rates, D-TLB, and I-TLB hit rates
  - Power metrics: Power for rename logic, branch predictor, reorder buffer, load-store queue, register file, L1 D-cache, L1 I-cache, L2 cache, functional units, result bus, and clock network
- Normalize each metric and scale to 100
  - Normalize performance metrics to maximum possible value
    - Maximum IPC = Issue width
  - Normalize power metrics to their percentage of the total power consumption
- Compute the Euclidean distance between each benchmark NOT in the subset to each benchmark IN the subset
- For each benchmark NOT in the subset, assign the minimum Euclidean distance as its distance
- Sum the Euclidean distances for all benchmarks NOT in the subset and assign that number as the total minimum Euclidean distance for that subset size

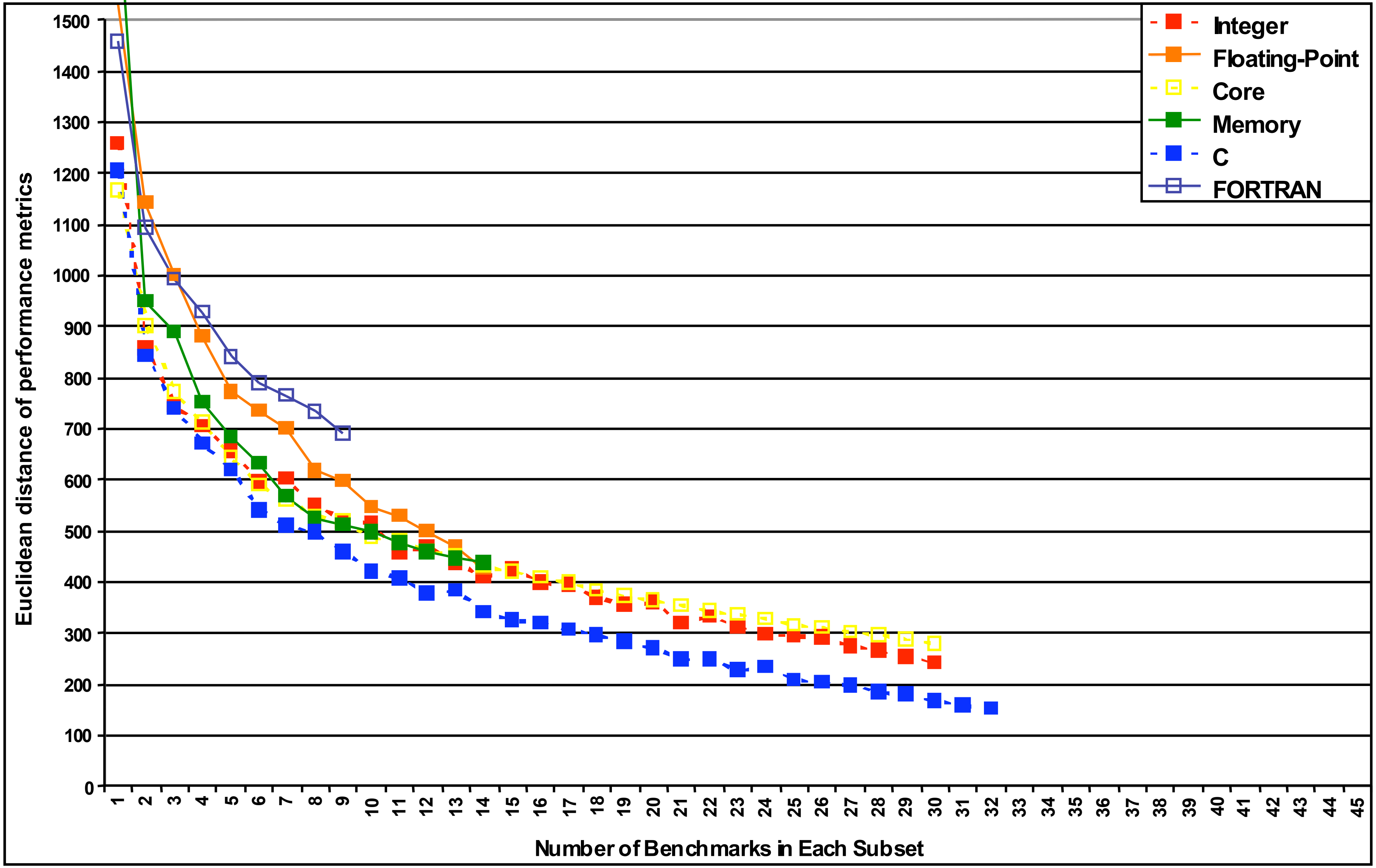
Coverage Intuition

- Total Euclidean distance represents how well the benchmarks in the subset are spread throughout the entire space of benchmarks
- A smaller total Euclidean distance means that benchmarks that are not in the subset are very close to a benchmark in the subset
  - Benchmarks not in the subset are accurately represented by a benchmark in the subset
  - Or, from the viewpoint of coverage, the benchmarks in the subset effectively cover the benchmark suite

Performance Coverage, Part 1



Performance Coverage, Part 2



J. Yi et al.

Freescale, Rhode Island, Ghent, Texas, Minnesota

Key Conclusions for Coverage

- Conclusions are similar to those for absolute accuracy
  - PCA and P&B have good coverage
  - Other 5 approaches do not have good coverage
  - Same conclusions for absolute/relative accuracy and coverage
- Coverage generally improves with larger subset sizes
- Similar results across all processor configurations
- Smaller Euclidean distances for power metrics:
  - Smaller maximum values for power metrics
  - Less variability in power results

Accuracy vs. Profiling Cost

- Subsetting approaches have different accuracies, but what is their profiling cost?
  - PCA
    - Specialized functional simulator or instrumentation
    - Single run or many runs
    - Requires a couple of months to gather profiling data
  - P&B
    - Requires performance simulator
    - 88 very different processor configurations (*i.e.*, some very slow)
    - Requires several months to gather profiling data
  - No profiling cost for the other 5 approaches

- Based on accuracy and profiling cost, we recommend using PCA to subset benchmark suites

Conclusions

- Computer architects frequently use subsetting…
  … but accuracy of subsetting approaches is unknown
  - Absolute accuracy
  - Relative accuracy
  - Coverage

- PCA and P&B design
  - Have the best absolute and relative CPI/EDP accuracy
  - Error less than 5% for 20+ benchmark-input pairs
  - Most efficiently cover the space of performance and power characteristics
- Other 5 approaches have poor accuracy and coverage

- PCA has the highest accuracy at the lowest profiling cost

Thank you

Evaluating Benchmark Subsetting Approaches

Joshua J. Yi, Resit Sendag, Lieven Eeckhout, Ajay Joshi, David J. Lilja, Lizy K. John

Acknowledgements

- This research has been supported by the following:
  - National Science Foundation (CCF-0541162, 0429806)
  - University of Minnesota Digital Technology Design Center
  - University of Minnesota Supercomputing Institute
  - European HiPEAC Network of Excellence
  - Fund for Scientific Research – Flanders (Belgium) (F.W.O Vlaanderen)
  - European SCALA project No. 27648.
  - IBM Centers for Advanced Studies
  - Intel