

# Introduction

- Performance evaluation
  - Most popular method: running benchmarks
  - Takes time
    - E.g.: SPEC CPU2000: 44 min. on a AMD Athlon 64\*
    - E.g.: TPC-C: 3-4 hours
    - At least 3 orders of magnitude slower when simulated
- But ... lot's of needless work!
  - Benchmarks have similar characteristics
  - Running still another benchmark doesn't always provide more information

\*FX-55, MSI Neo Platinum motherboard, fastest SPEC INT on Oct. 22 2004

# Problem Statement

- If some benchmarks are similar ...
  - Can we reduce the number of benchmarks?
    - The reduced benchmark suite should be **representative** for the full benchmark suite
    - The reduction should be representative in all circumstances, i.e. it should be **reliable**
  - How should we do this?
    - Detect **benchmark similarity**
    - Reduce the number of benchmarks by **clustering** similar benchmarks
    - [Saavedra TC'96, Lafage WWC'00, Sherwood ASPLOS'02, Eeckhout PACT'02, Vandierendonck CAECW'04, ISPASS'04]

# Benchmark Similarity

- Inherent property of programs
- Characterize similarity in two steps
  1. Measure K selected **workload characteristics**
  2. Display benchmarks in a **K-dimensional space**
    - Coordinates of the benchmarks correspond to the values measured for the workload characteristics
    - **Similarity = distance in the workload space**
- Reduce benchmark suite
  - Workload space contains clusters of benchmarks
  - Basic idea: select 1 benchmark per cluster

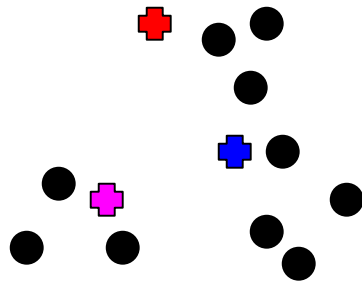
# Two Main Parameters

- Choice of Clustering algorithm
  - K-means clustering
  - Hierarchical clustering
  - Methods based on principal components analysis
- Choice of Workload characteristics
  - Execution time on a range of computers [Vandierendonck CAECW'04]
  - Execution time on a subset of those computers
  - Instruction mix, cache miss rates, ... [Eeckhout PACT'02]
- Goal: Investigate impact of both parameters on representativeness and reliability

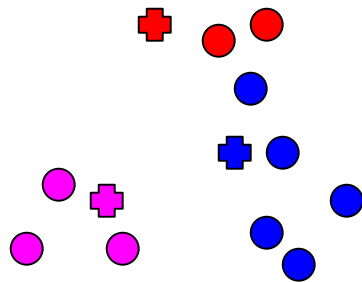
# K-means Clustering

1. Select K, e.g.: K=3

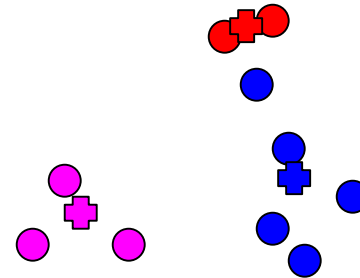
2. Randomly select K cluster centers



3. Assign benchmarks to cluster centers



4. Move cluster centers



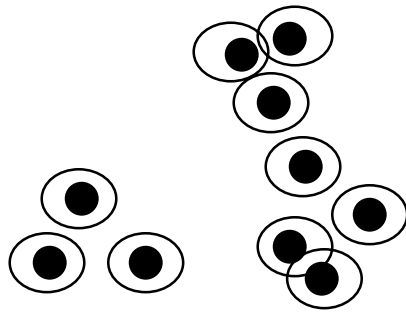
5. Repeat steps 3 and 4 until convergence

6. Subsetting step: select K benchmarks closest to the cluster centers

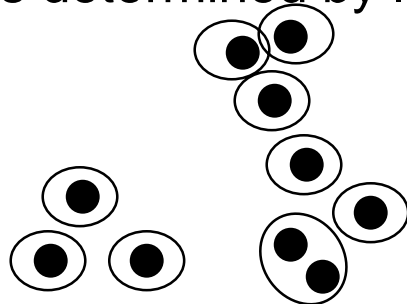
# Hierarchical Clustering

Iteratively join clusters

1. Initialize with 1 benchmark/cluster

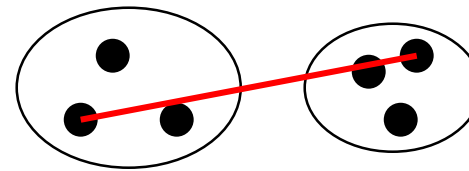


2. Join two “closest” clusters  
Closeness determined by linkage strategy



3. Repeat step 2 until one cluster remains

- **Joining clusters**
  - Complete linkage



- Other linkage strategies exist with qualitatively the same results

# Principal Components Analysis

- Linear correlation between characteristics
- Principal components
  - Linear combination of original characteristics  $X_j$
  - $X_j$  is the execution time of the  $j$ -th benchmark
  - $Z_i, i=1..N$       $Z_i = \sum_j a_{ij} X_j$
- Properties
  - Total variance (information content) is unchanged
  - $Z_i$  and  $Z_j$  are uncorrelated,  $i \neq j$
  - $\text{Var}(Z_1) \geq \text{Var}(Z_2) \geq \text{Var}(Z_3) \geq \dots \geq \text{Var}(Z_N)$

# PCA: Forward Method

- Rank benchmarks
  - $Z_1$  contains most information, so the benchmark most similar to  $Z_1$  is the most interesting benchmark
- For  $I=1,\dots,N$  ( $N$  = number of benchmarks)
  - Inspect  $I$ -th principal component
  - Find  $j$  for which  $|a_{ij}|$  is maximal
  - Benchmark  $j$  is the  $I$ -th most interesting
  - Unless it has already been selected, in which case we look for the second largest  $|a_{ij}|$
  - [Dunteman '89, Vandierendonck CAECW'04]

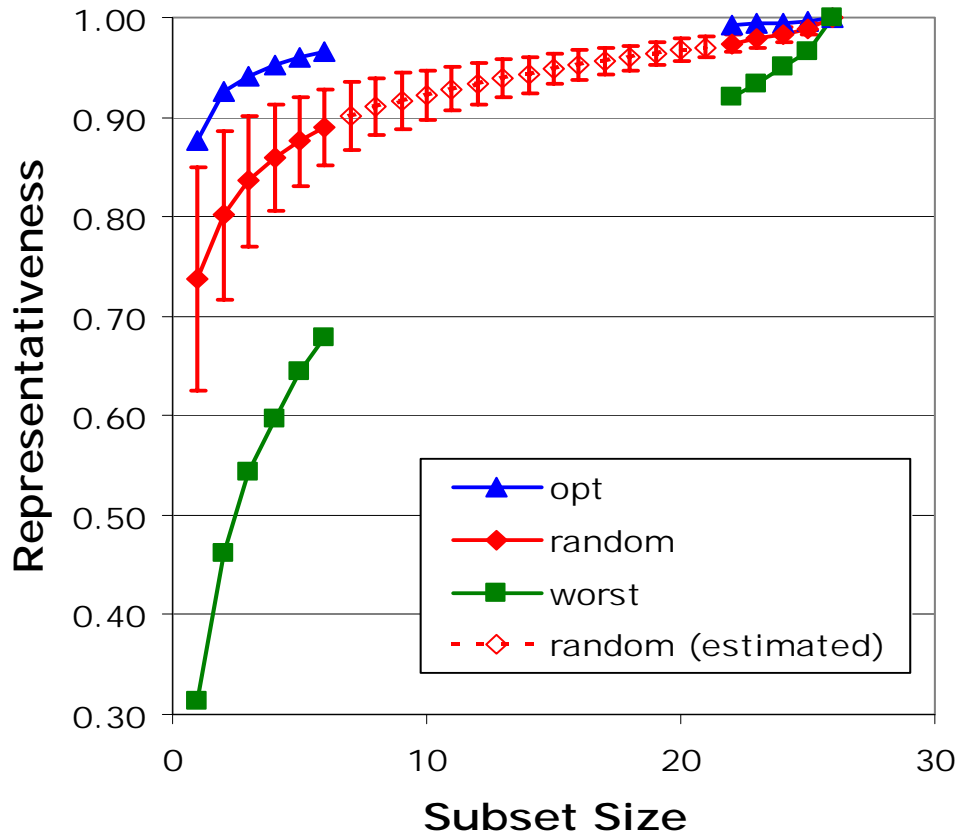
# PCA: Backward Method

- Rank benchmarks
  - $Z_N$  contains the least information, so the benchmark most similar to  $Z_N$  is the least interesting benchmark
- For  $I=N, \dots, 1$  ( $N = \text{number of benchmarks}$ )
  - Inspect  $I$ -th principal component
  - Find  $j$  for which  $|a_{ij}|$  is maximal
  - Benchmark  $j$  is the  $I$ -th most interesting ( $N-I$ -th least interesting)
  - Unless it has already been selected, in which case we look for the second largest  $|a_{ij}|$

# Evaluation

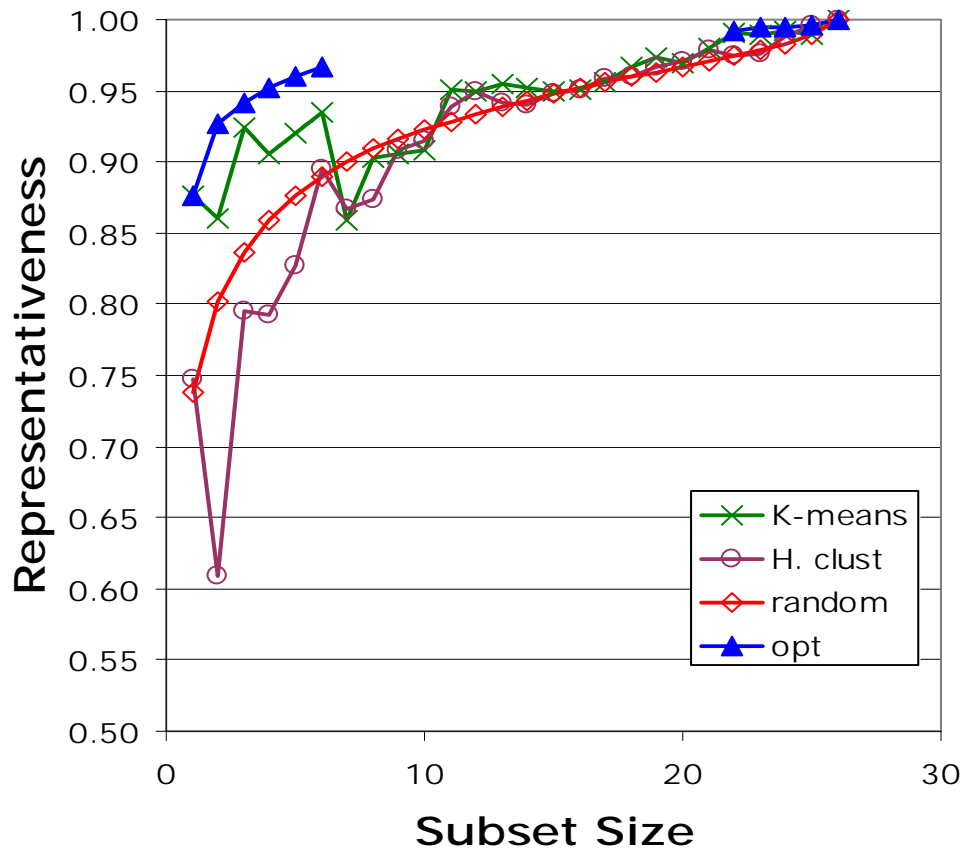
- Reduce SPEC CPU 2000
  - **Goal**: reduced suite should rank computers in same way as full suite
  - Tested on performance results published by SPEC for 340 computers (see paper for details)
  - **Metric**: rank correlation coefficient between rankings measured using **reduced** and **full** suites
  - **Good rankings**: coefficient  $\geq 0.90$  or  $0.95$  correspond to 5% and 2.5% of incorrectly ranking two machines

# Optimal/Random Subsets



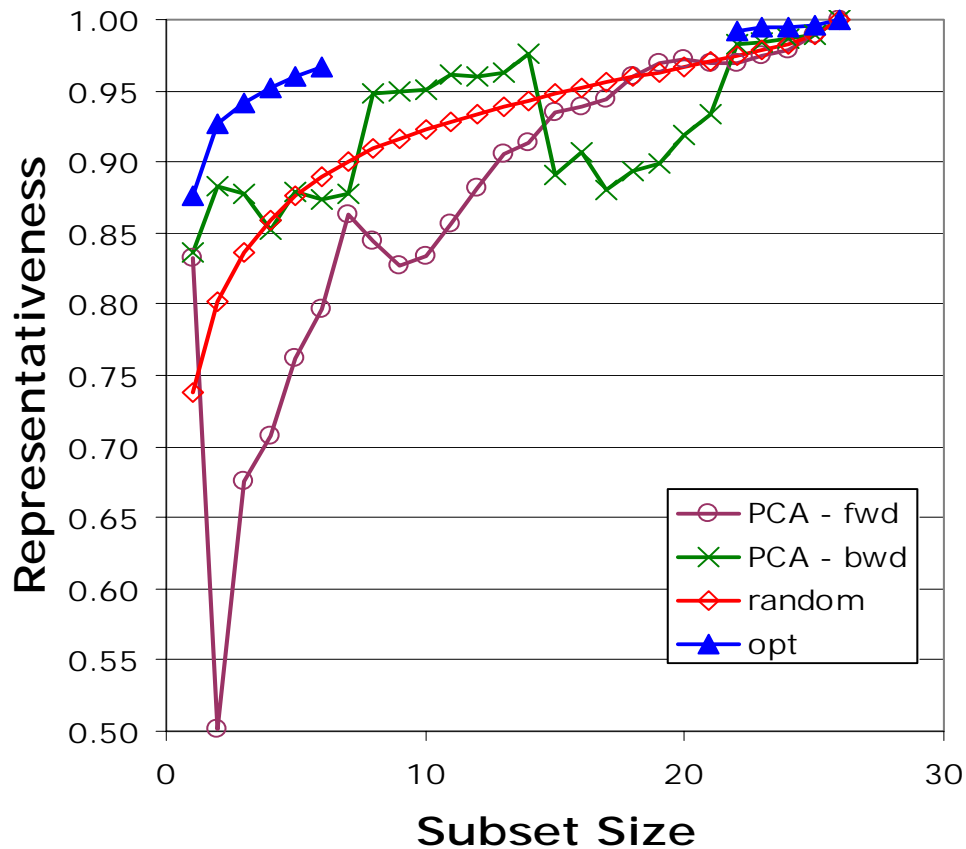
- Randomly selected subsets are relatively bad: 14 needed to obtain 0.9 with 95% probability
- A 22-benchmark subset may be worse than a properly chosen 2-benchmark subset
- Confidence intervals contain 68% of all subsets

# K-means and Hierarch. Clust.



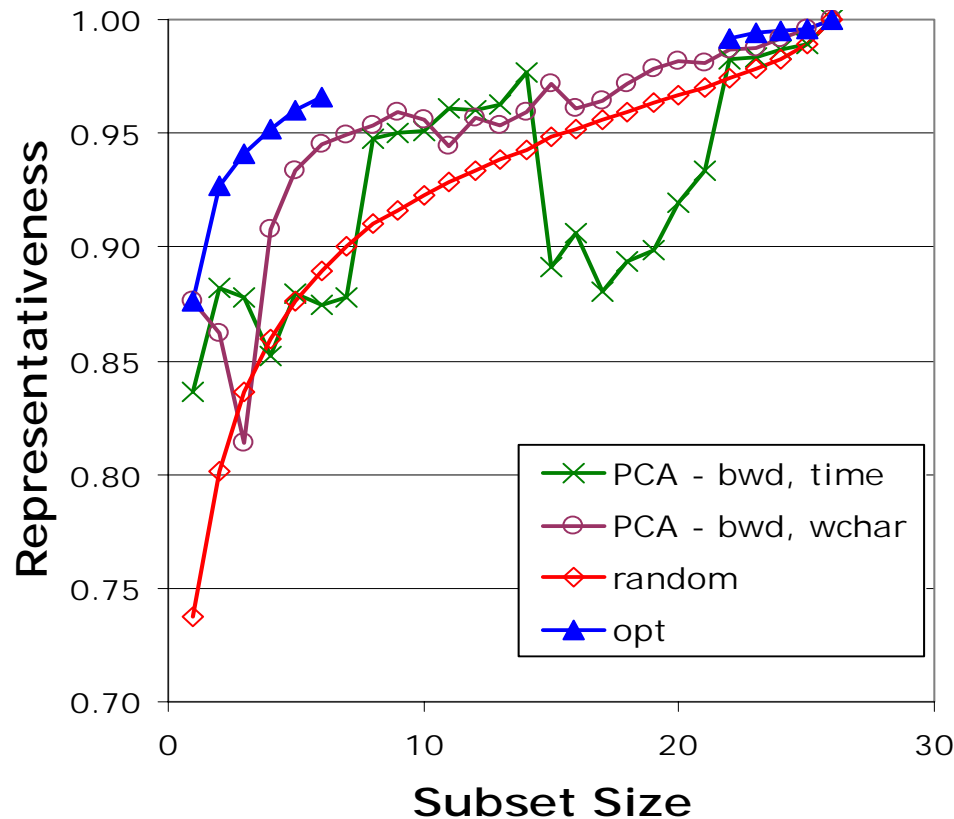
- Representativeness is not a monotonic function of subset size!
- Most of the time: only as good as a random subset

# Algorithms Based on PCA



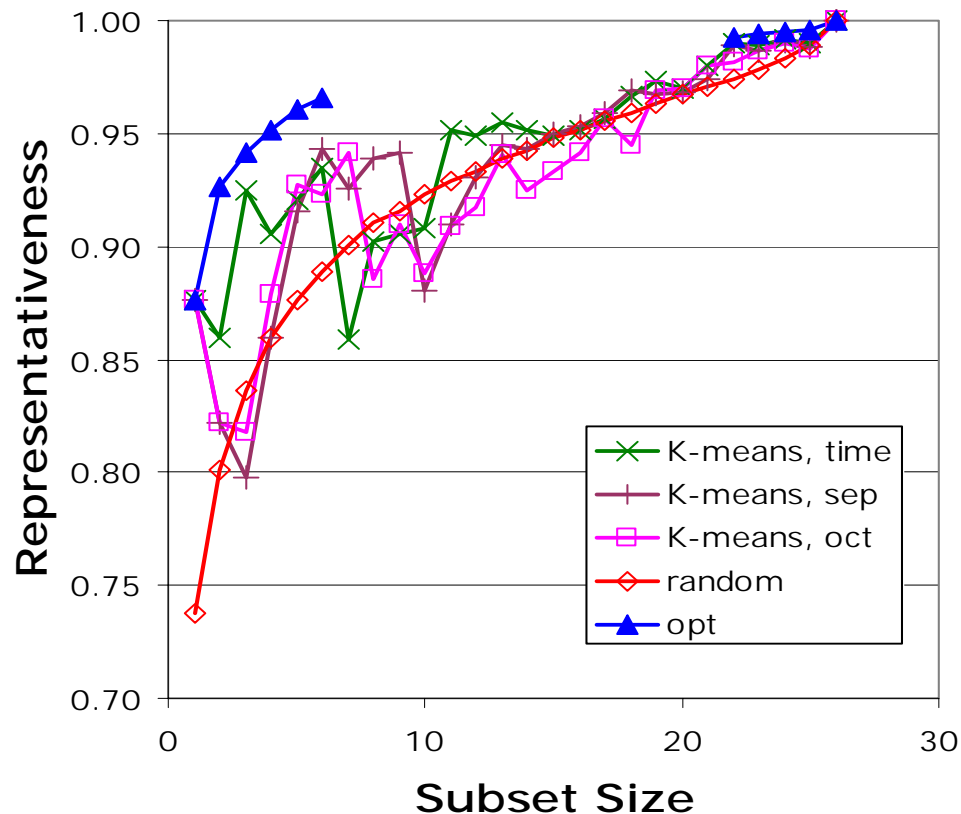
- Perform worse than K-means and hierarchical clustering

# Workload Characteristics



- Use workload characteristics to place benchmarks in workload space:
  - Instruction mix
  - I/d-cache miss rates
  - Branch pred. accuracy (see paper for full list)
- Backward PCA algorithm is suddenly much more reliable
- K-means and hierarchical are less reliable

# Using Early Computers



- Use performance measurements for a limited number of computers to reduce a benchmark suite
  - Sep 2000: 31 computers
  - Oct 2000: 53 computers
- Less representative than using all computers

# Conclusion

- Subset a benchmark suite
  - Representative and reliable
- Tested several algorithms
  - Sometimes representative, but all unreliable
  - Hence, always need verification!
  - A randomly selected subset may not be significantly worse than an algorithmically chosen subset
  - Found backward PCA + workload characteristics to be most reliable...
- Workload characteristics
  - Have less influence on representativeness than the clustering algorithms