Analysis of Redundancy and Application Balance in the SPEC CPU 2006 Benchmark Suite

ISCA 2007

Phansalkar, Joshi and John

Fast Subsetting to form CPU2006 suite

Computer Architecture News
A Publication of the Association for Computing Machinery Special Interest Group on Computer Architecture
Vol. 35, No. 1 - March 2007

(2) After several development versions of the new suite were built, various voting members of the SPEC CPU subcommittee released data to a trusted third party: non-voting participants from the Laboratory for Computer Architecture at the University of Texas. The University researchers prepared normalized summaries of the data, performed clustering analysis, and presented benchmark similarity dendograms such as the ones shown at [3].

If normalized data from a member showed that a benchmark used few resources, or if analysis from the university researchers showed that two benchmark candidates behaved similarly, this alone was not sufficient to exclude a candidate. But it was a factor that was considered, along with other factors such as application area, coding style, and size of user base.

Performance Counters and Development of SPEC CPU2006

John L. Henning
Sun Microsystems
Contact: john dot henning (at) acm dot org

Motivation

Many benchmarks are similar

Running more benchmarks that are similar will not provide more information but necessitates more effort

One could construct a good benchmark suite by choosing representative programs from similar clusters

Advantages:

- Reduces experimentation effort

Benchmark Reduction

Measure properties of programs (say K properties)

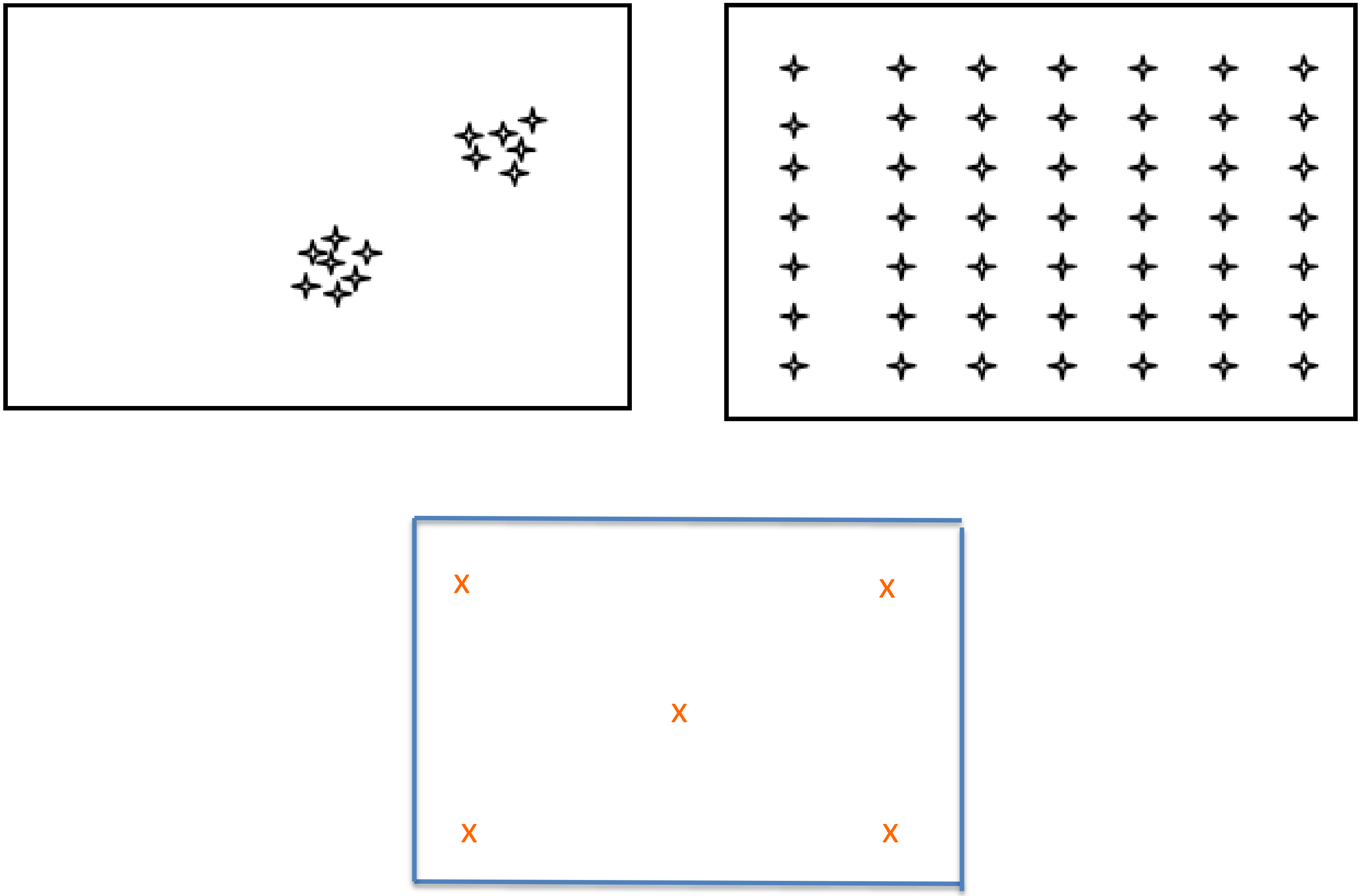
- Microarchitecture independent properties
- Microarchitecture dependent properties

Display benchmarks in a K-dimensional space

Workload space consists of clusters of benchmarks

Choose one benchmark per cluster

Example Workload/Benchmark space Distributions



Benchmark Reduction

Measure properties of programs (say K properties)
- Microarchitecture independent properties
- Microarchitecture dependent properties

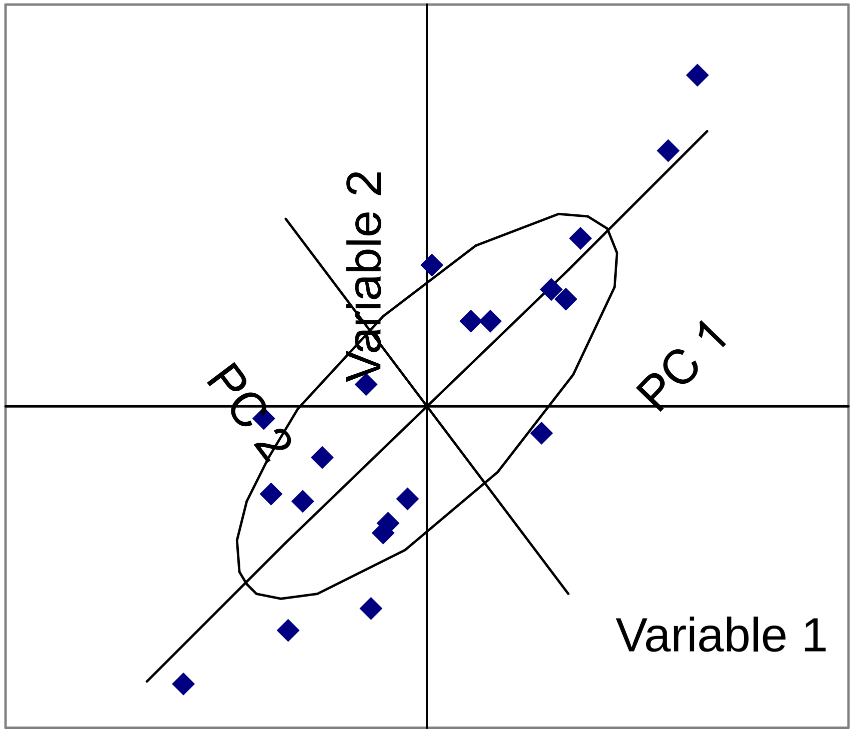
Derive principal components that capture most of the variability between the programs

Workload space consists of clusters of benchmarks in the principal component space

Choose one benchmark per cluster

Principal Components Analysis

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



$$PC1 = a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ...$$
$$PC2 = a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ...$$
$$PC3 = a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + ...$$

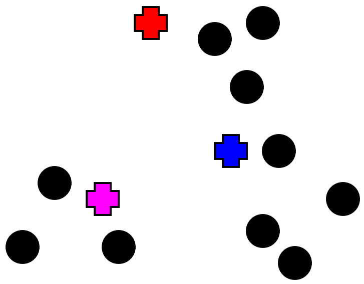
Source:moss.csc.ncsu.edu/pact02/slides/eeckhout_135.ppt

Clustering

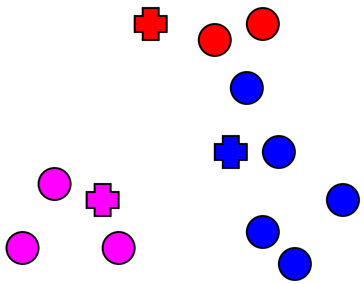
- Clustering algorithms
  - K-means clustering
  - Hierarchical clustering

K-means Clustering

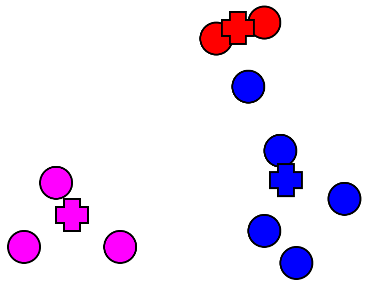
1. Select K, e.g.: K=3
2. Randomly select K cluster centers



3. Assign benchmarks to cluster centers



4. Move cluster centers

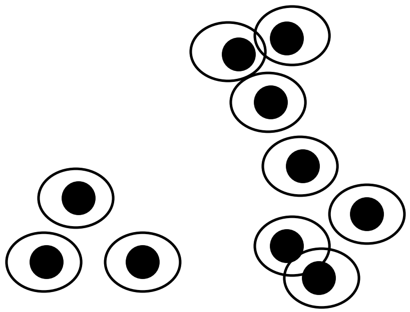


5. Repeat steps 3 and 4 until convergence

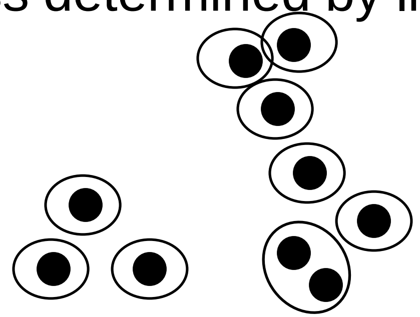
Hierarchical Clustering

Iteratively join clusters

1. Initialize with 1 benchmark/cluster

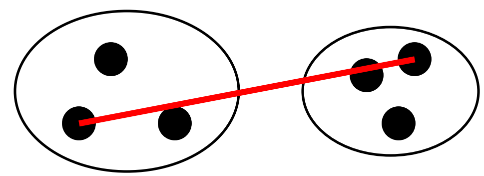


2. Join two “closest” clusters
   Closeness determined by linkage strategy



3. Repeat step 2 until one cluster remains

- Joining clusters
  - Complete linkage

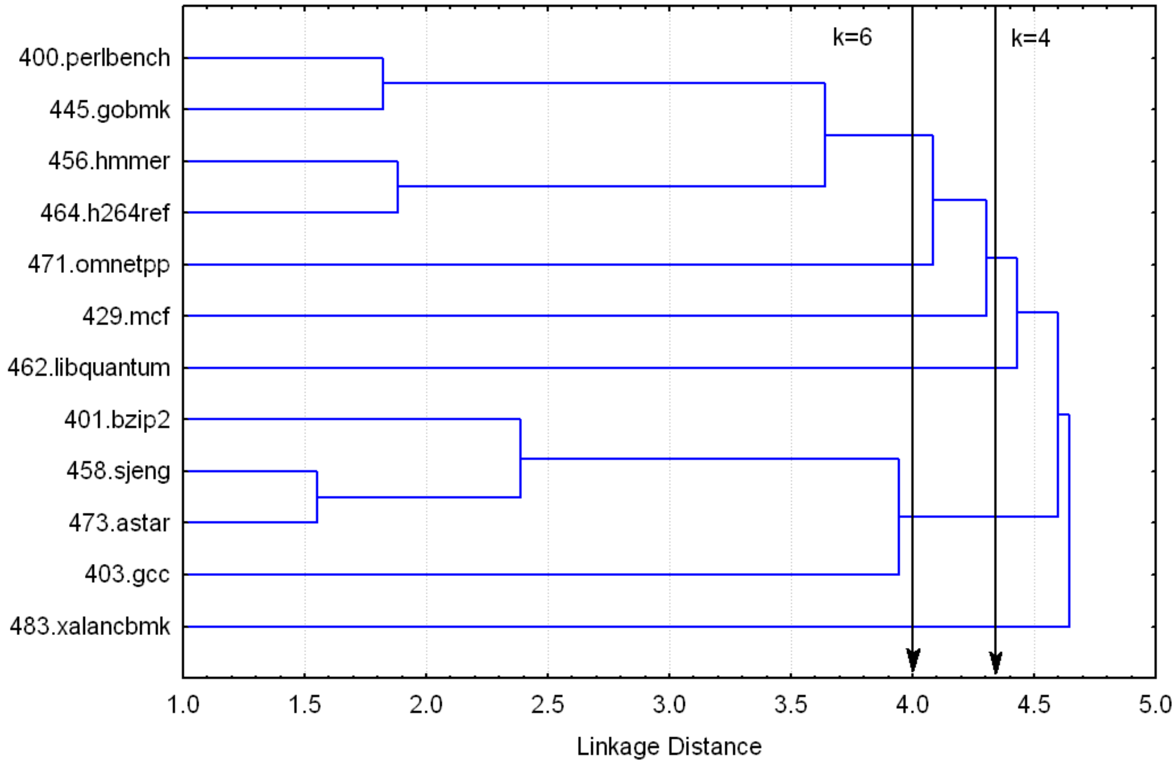


  - Other linkage strategies exist with qualitatively the same results

Distance between clusters

- Euclidian Distance
  - the way the crow flies; sq root of (a^2 +b^2);
- Manhattan Distance
  - The way cars go in manhattan; a+b
- Centroid of clusters
- Distance from centroid of one cluster to another centroid
- Longest distance from any element of one cluster to another

Dendrogram for illustrating Similarity



Single Linkage distance

k=4	**400.perlbench, 462.libquantum,473.astar,483.xalancbmk**
k=6	**400.perlbench, 471.omnetpp, 429.mcf, 462.libquantum, 473.astar, 483.xalancbmk**

Software Packages to do Similarity Analysis

- STATISTICA
- R
- MATLAB

- PCA
- K-means clustering
- Dendrogram generation

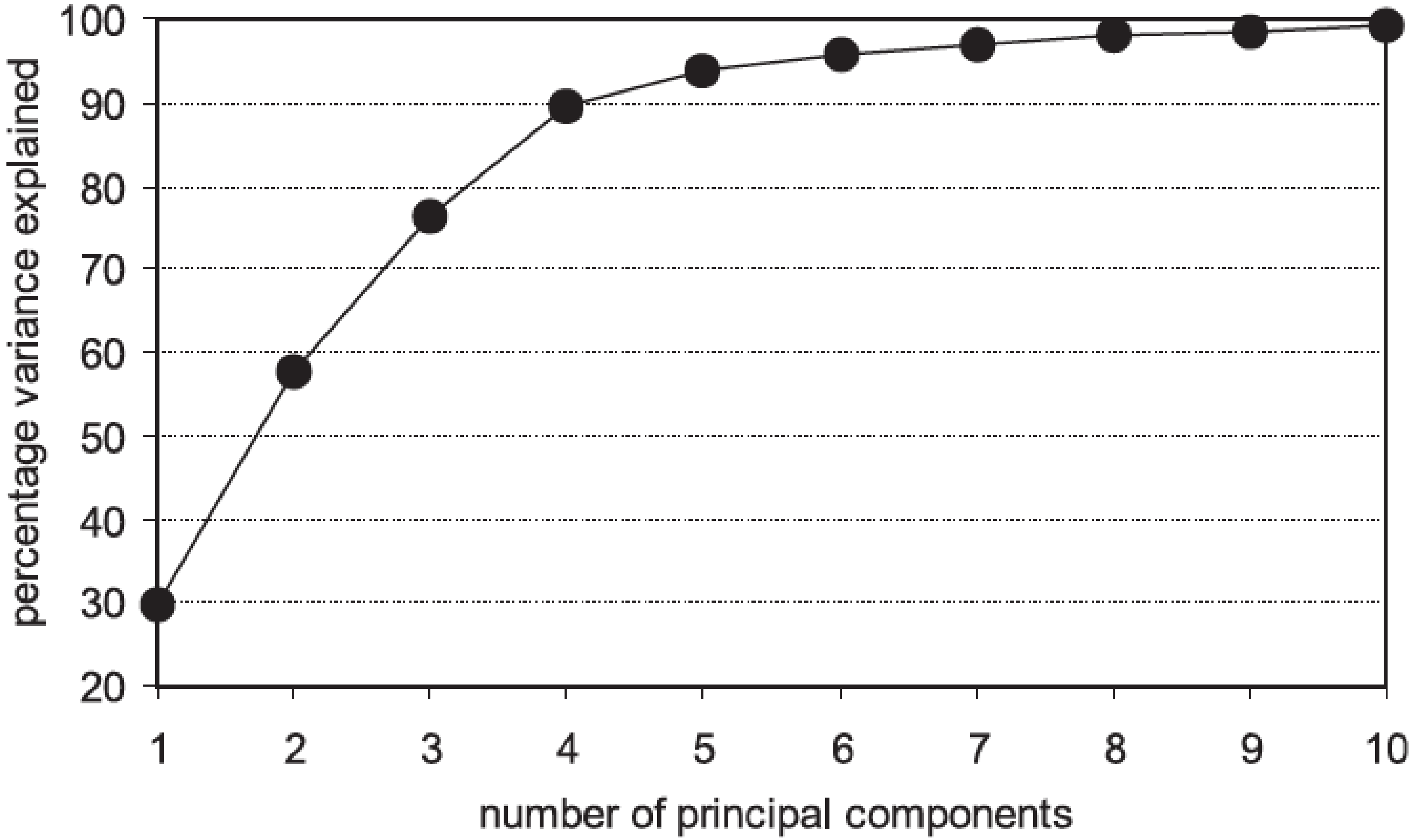


Figure 9.1 Amount of variance explained as a function of the number of principal components.

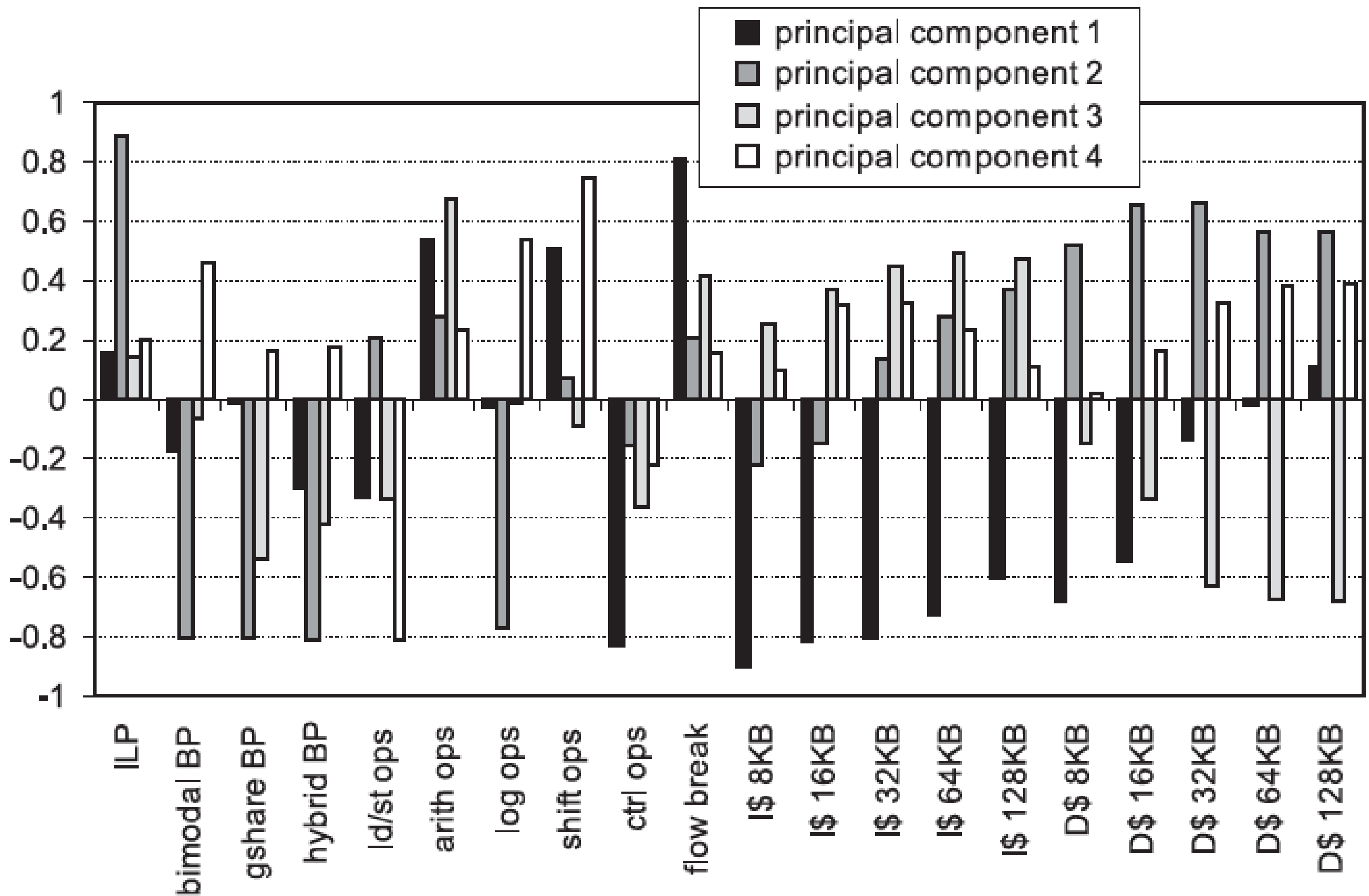


Figure 9.2 Factor loadings.

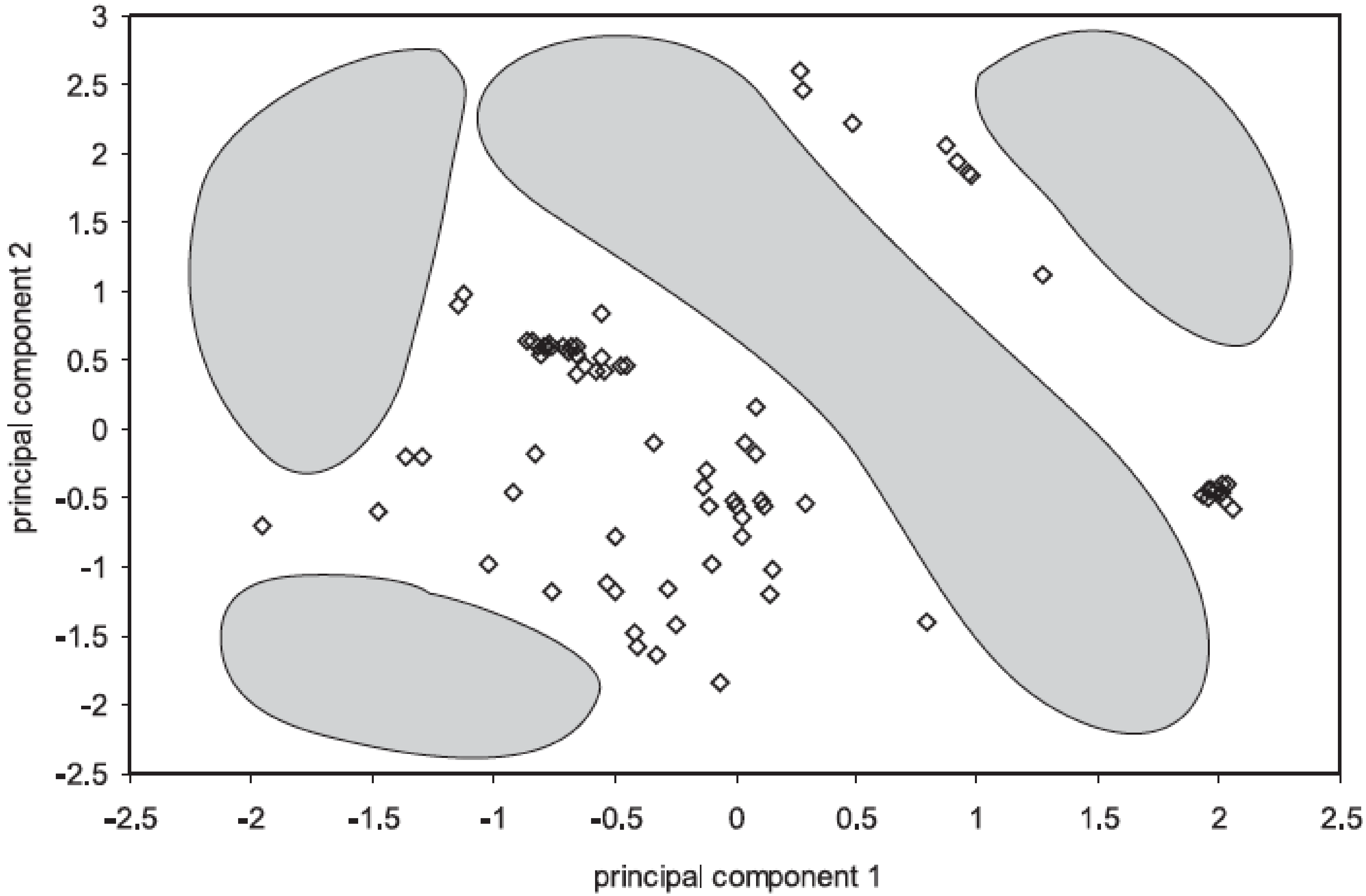


Figure 9.5 Weak spot detection.

Are features of equal weight? Need for Normalizing Data

	feature 1	feature 2
bench1	0.01	20
bench2	0.1	40
bench3	0.05	50
bench4	0.001	60
bench5	0.03	25
bench6	0.002	30
bench7	0.015	70
bench8	0.5	60
	0.0885	44.375
	0.169483	18.40759

Variance 1 > Mean 1

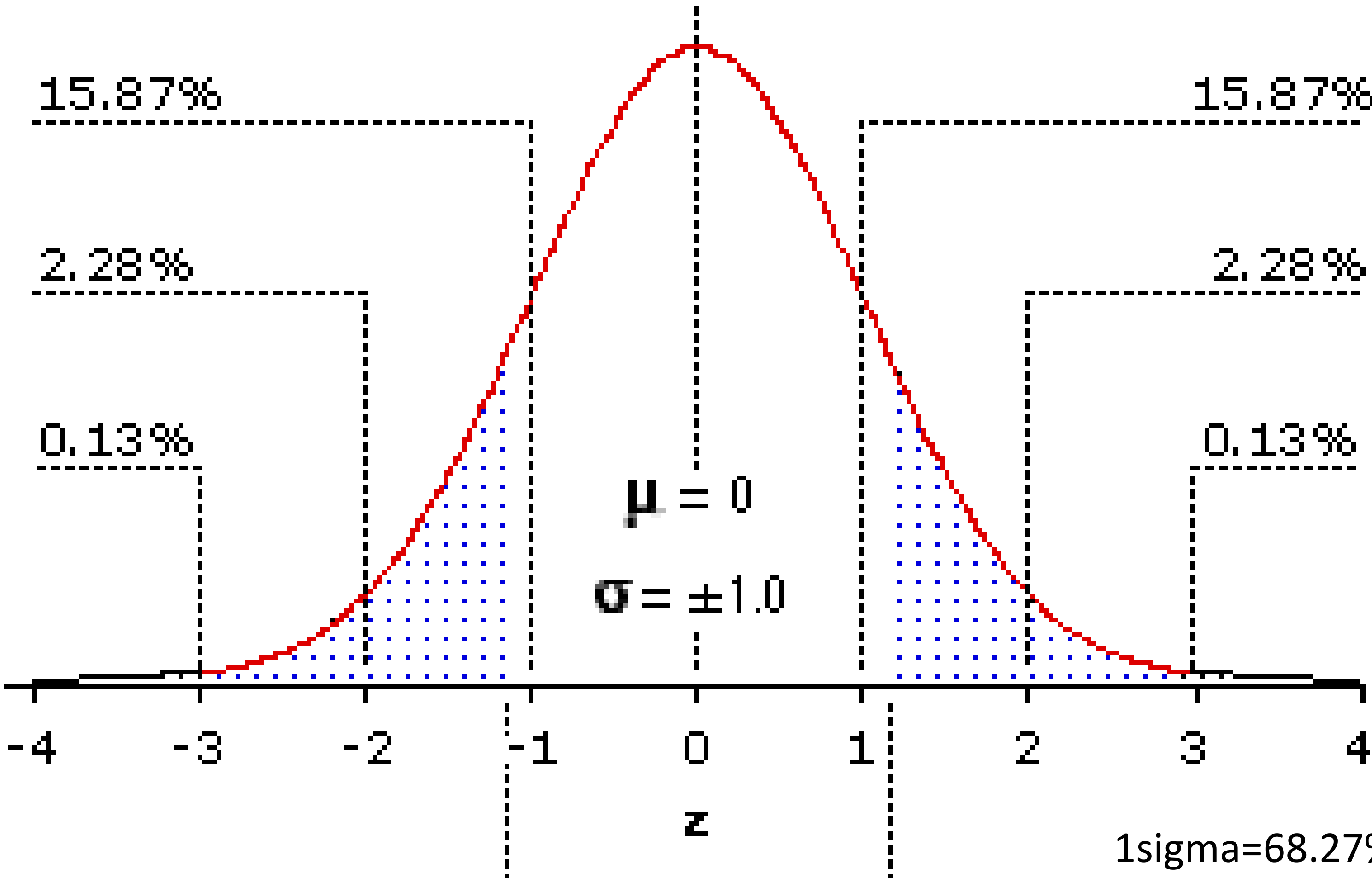
Variance 2 << Mean 2

Feature 1 numeric values << Feature 2 numeric val

Compute distance from 0 to bench 4, and 0 to bench 8

Feature 1 has low effect on distance

Unit normal distribution



1sigma=68.27%
2 sigma=95.45%
3 sigma=99.73%

Normalizing Data (Transforming to Unit-Normal)

The converted data is also called standard score.

How do you convert to a distribution with mean = 0 and std dev = 1?

The standard score of a raw score *x* [1] is

$$z = \frac{x - \mu}{\sigma}$$

where:

μ is the mean of the population;

σ is the standard deviation of the population.

Normalizing Data

	feature 1	feature 2	norm feat 1	norm feat 2
bench1	0.01	20	-0.46317	-1.32418
bench2	0.1	40	0.067853	-0.23767
bench3	0.05	50	-0.22716	0.305581
bench4	0.001	60	-0.51628	0.848835
bench5	0.03	25	-0.34517	-1.05256
bench6	0.002	30	-0.51037	-0.78093
bench7	0.015	70	-0.43367	1.392089
bench8	0.5	60	2.427969	0.848835
	0.0885	44.375	0	0
	0.169483	18.40759	1	1

Convert to a distribution with mean = 0 and std dev = 1

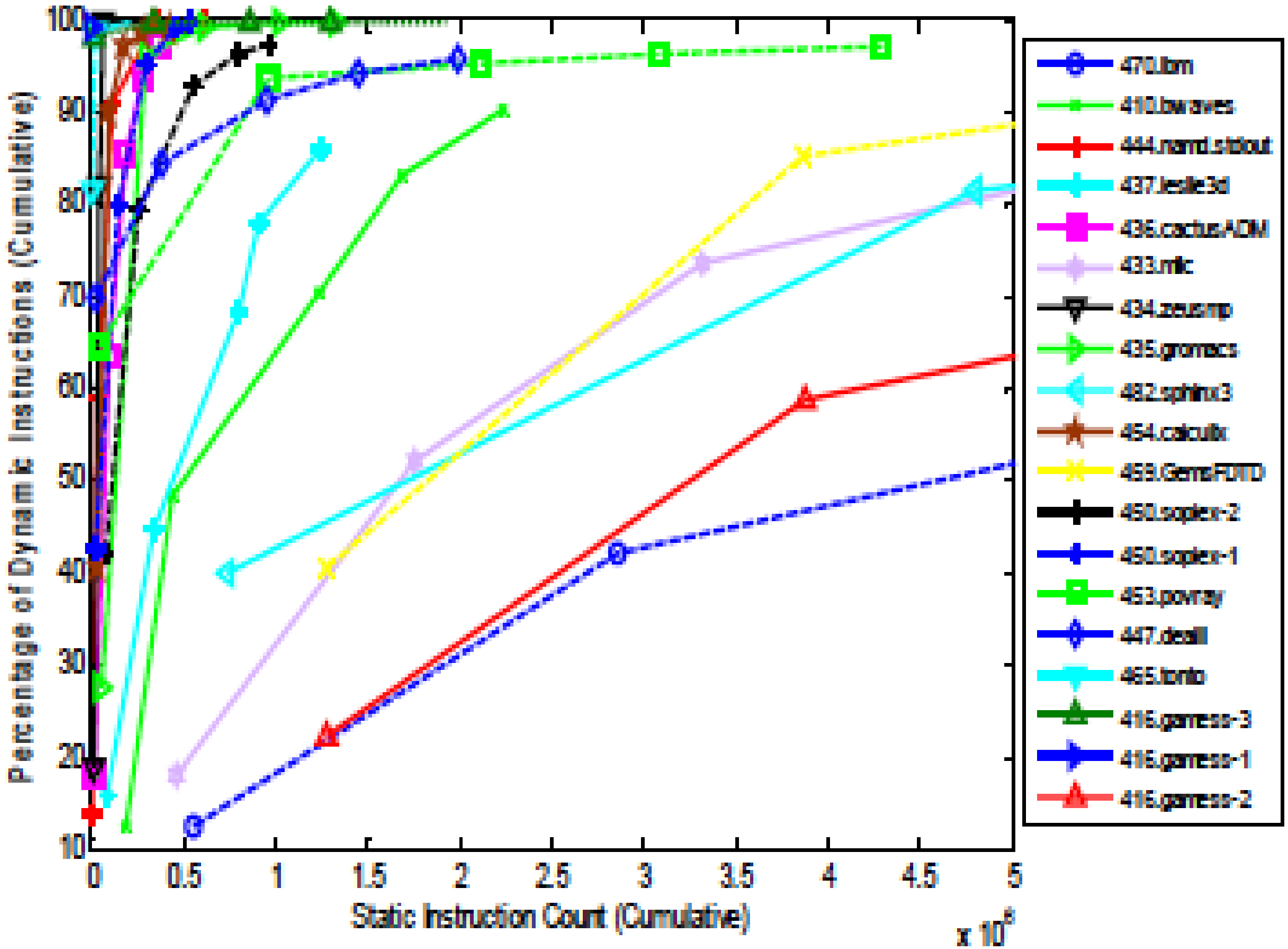
With normalized data, bench8 is far from bench 4

Mahalanobis distance

- Mahalanobis distance
  - How many standard deviations away a point P is from the mean of a distribution
  - If all axes are scaled to have unit variance, Mahalanobis distance = Euclidian distance

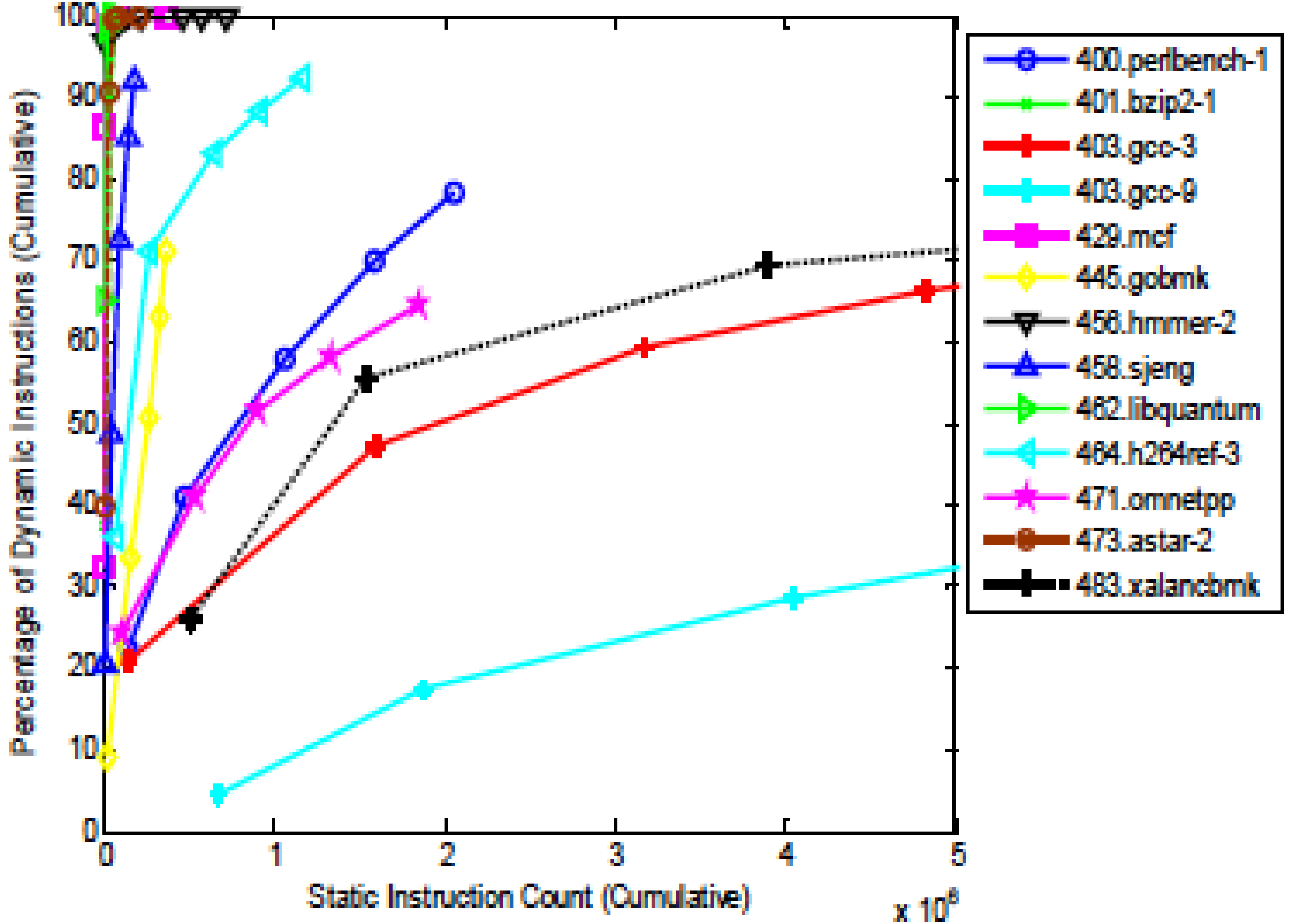
Name – Language	Inst. Count (Billion)	Branches	Loads	Stores
CINT 2006				
400.perlbench –C	2,378	20.96%	27.99%	16.45%
401.bzip2 – C	2,472	15.97%	36.93%	12.98%
403.gcc – C	1,064	21.96%	26.52%	16.01%
429.mcf –C	327	21.17%	37.99%	10.55%
445.gobmk –C	1,603	19.51%	29.72%	15.25%
456.hmmer –C	3,363	7.08%	47.36%	17.68%
458.sjeng –C	2,383	21.38%	27.60%	14.61%
462.libquantum-C	3,555	14.80%	33.57%	10.72%
464.h264ref- C	3,731	7.24%	41.76%	13.14%
471.omnetpp- C++	687	20.33%	34.71%	20.18%
473.astar- C++	1,200	15.57%	40.34%	13.75%
483.xalancbmk- C++	1,184	25.84%	33.96%	10.31%

CFP 2006				
410.bwaves – Fortran	1,178	0.68%	56.14%	8.08%
416.gamess – Fortran	5,189	7.45%	45.87%	12.98%
433.milc – C	937	1.51%	40.15%	11.79%
434.zeusmp–C,Fortran	1,566	4.05%	36.22%	11.98%
435.gromacs-C, Fortran	1,958	3.14%	37.35%	17.31%
436.cactusADM-C, Fortran	1,376	0.22%	52.62%	13.49%
437.leslie3d – Fortran	1,213	3.06%	52.30%	9.83%
444.namd – C++	2,483	4.28%	35.43%	8.83%
447.dealII – C++	2,323	15.99%	42.57%	13.41%
450.soplex – C++	703	16.07%	39.05%	7.74%
453.povray – C++	940	13.23%	35.44%	16.11%
454.calculix –C, Fortran	3,041	4.11%	40.14%	9.95%
459.GemsFDTD – Fortran	1,420	2.40%	54.16%	9.67%
465.tonto – Fortran	2,932	4.79%	44.76%	12.84%
470.lbm – C	1,500	0.79%	38.16%	11.53%
481.wrf - C, Fortran	1,684	5.19%	49.70%	9.42%
482.sphinx3 – C	2,472	9.95%	35.07%	5.58%



(b) CFP2006 Benchmarks

Figure 1. Instruction locality based on code reuse in the top 20 hot subroutines for SPEC CPU2006 benchmarks.



(a) CINT2006 Benchmarks

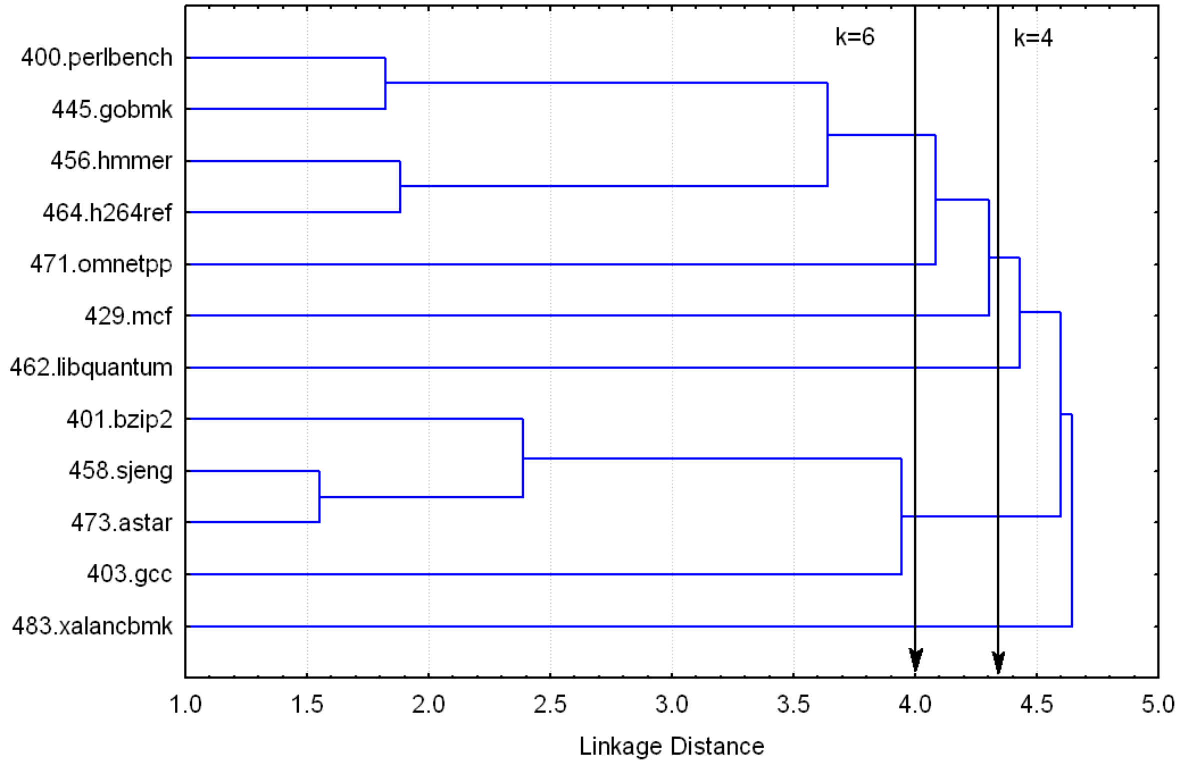
Table 2: Range of important performance characteristics of SPEC CPU2006 benchmarks

Metric	Min	Max
I-cache miss ratio	~ 0	1.7%
L1 D-cache miss ratio	6.3%	33%
L2 cache misses per instruction (per L2 access)	~0 (0.01%)	2.4% (49%)
DTLB miss ratio	0.2%	8.4%

Table 3. Program Characteristics used for measuring similarity between Integer and Floating-Point programs.

Integer benchmarks	**Floating-Point benchmarks**
Integer operations per instruction	Floating point operations per instruction
L1 instruction cache misses per instruction	Memory references per instruction
Number of branches per instruction	L2 data cache misses per instruction
Number of mispredicted branches per instruction	L2 data cache misses per L2 accesses
L2 data cache misses per instruction	Data TLB misses per instruction
Instruction TLB misses per instruction	L1 data cache misses per instruction

Dendrogram for illustrating Similarity



Single Linkage distance

k=4	**400.perlbench, 462.libquantum,473.astar,483.xalancbmk**
k=6	**400.perlbench, 471.omnetpp, 429.mcf, 462.libquantum, 473.astar, 483.xalancbmk**

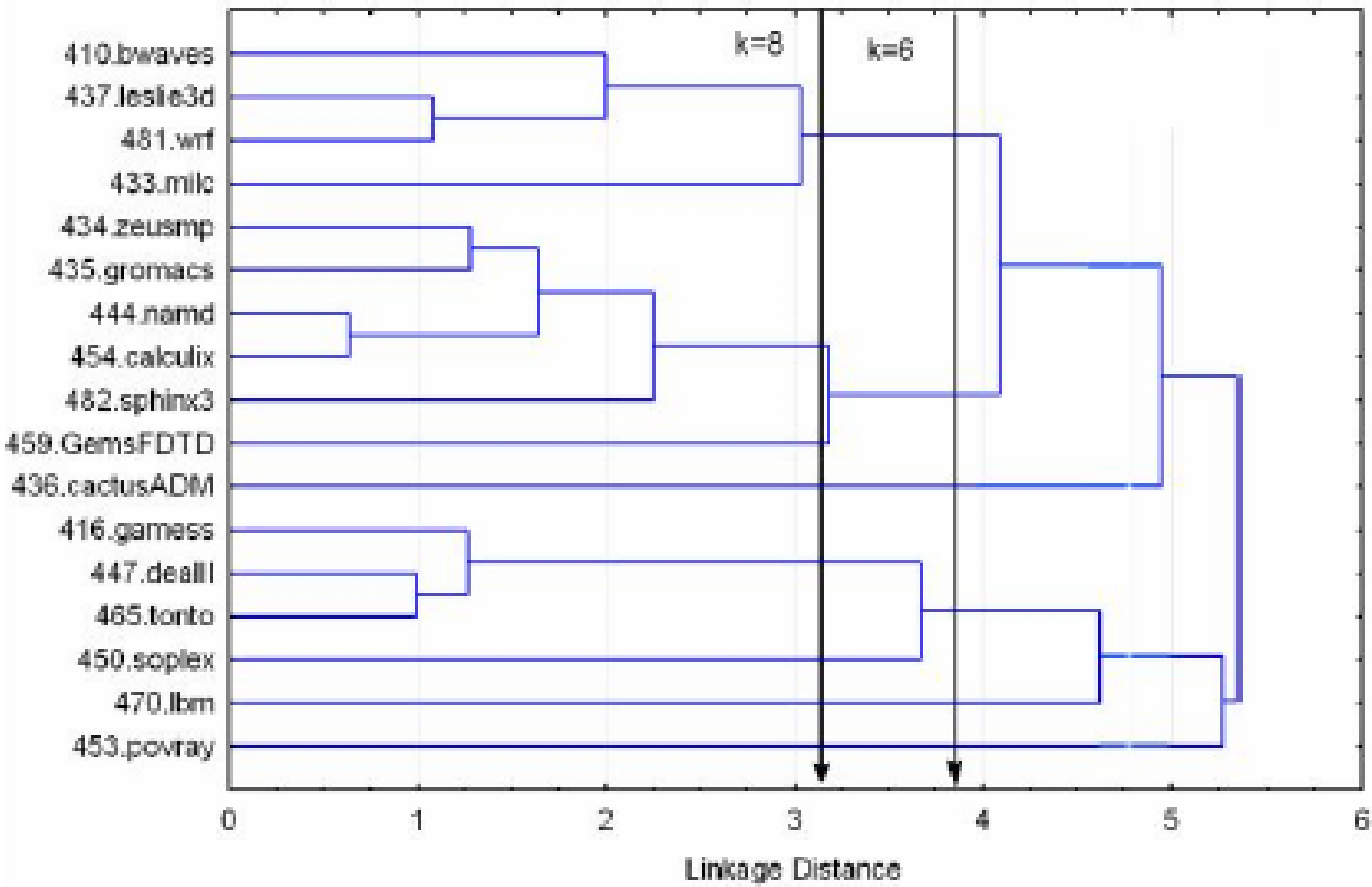


Figure 3. Dendrogram showing similarity between CFP2006 Programs.

Table 5. Representative subset of SPEC CFP2006 programs.

Subset of Six Programs	437.leslie3d, 454.calculix, 436.cactusADM, 447.dealII, 470.lbm, 453.povray
Subset of Eight Programs	437.leslie3d, 454.calculix, 459.GemsFDTD,436.cactusADM, 447.dealII, 450.soplex, 470.lbm, 453.povray

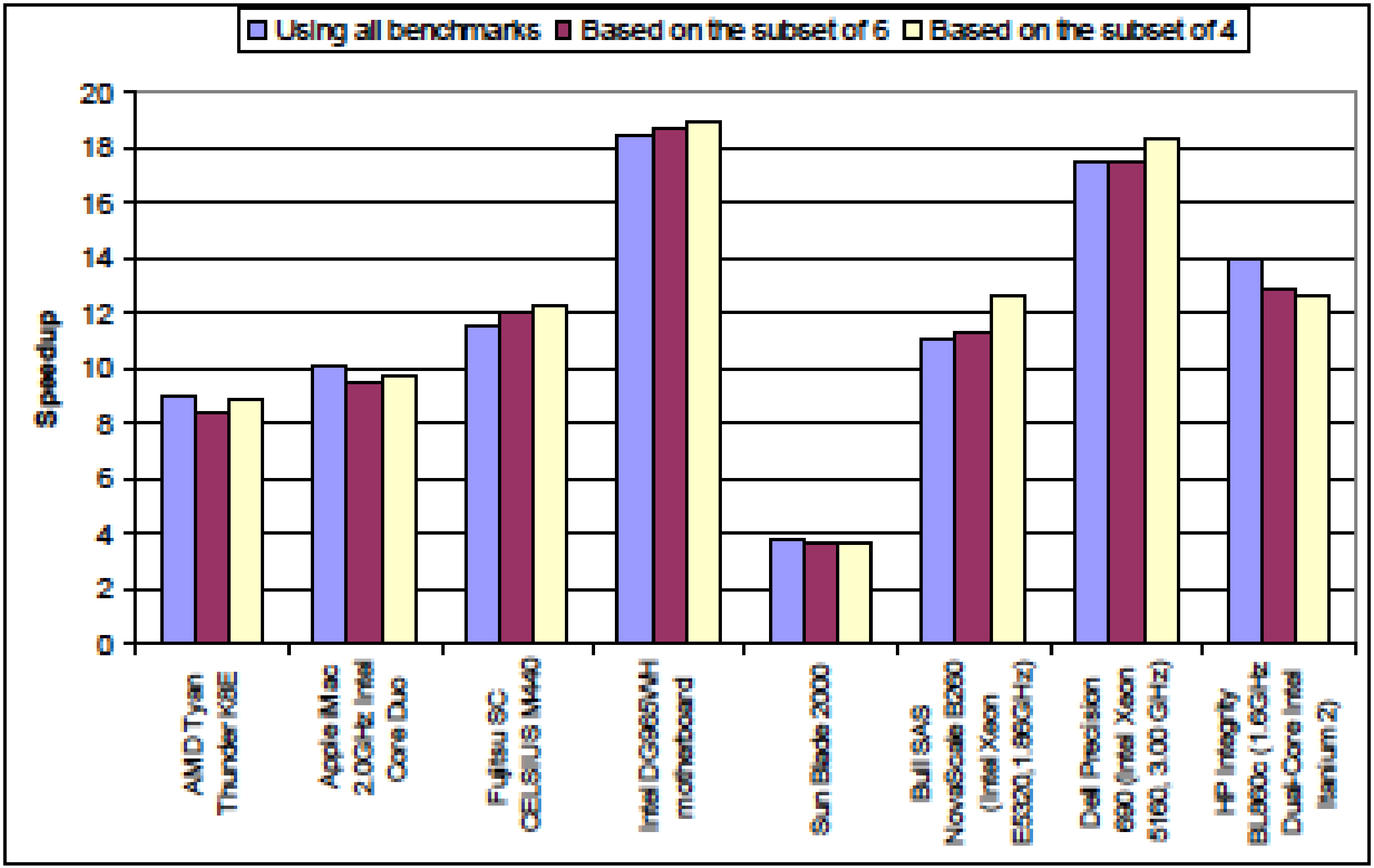


Figure 4. Validation of CINT2006 subset using performance scores of eight systems from the SPEC CPU website.

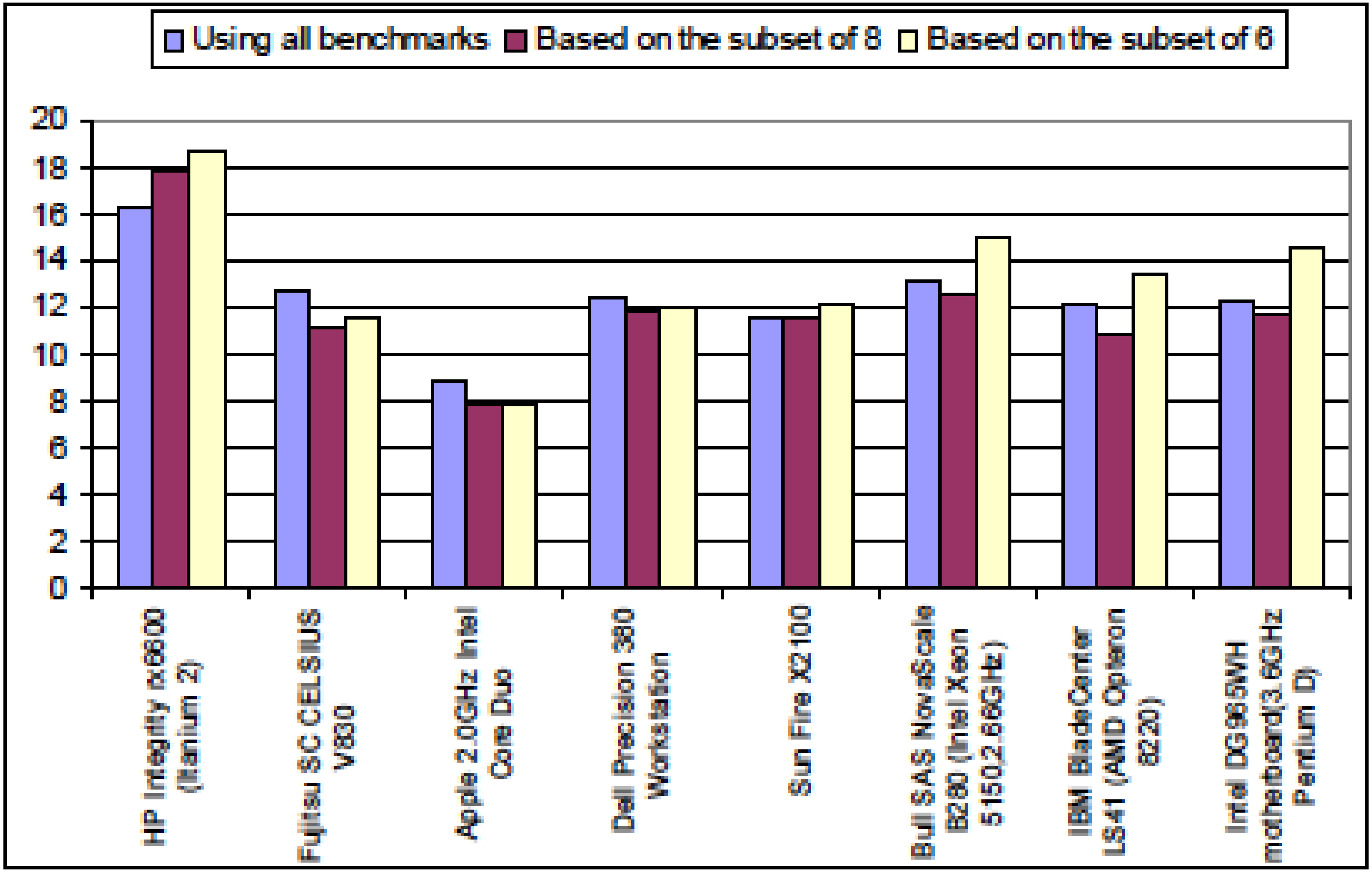


Figure 5. Validation of CFP2006 subset using performance scores of eight systems from the SPEC CPU website.

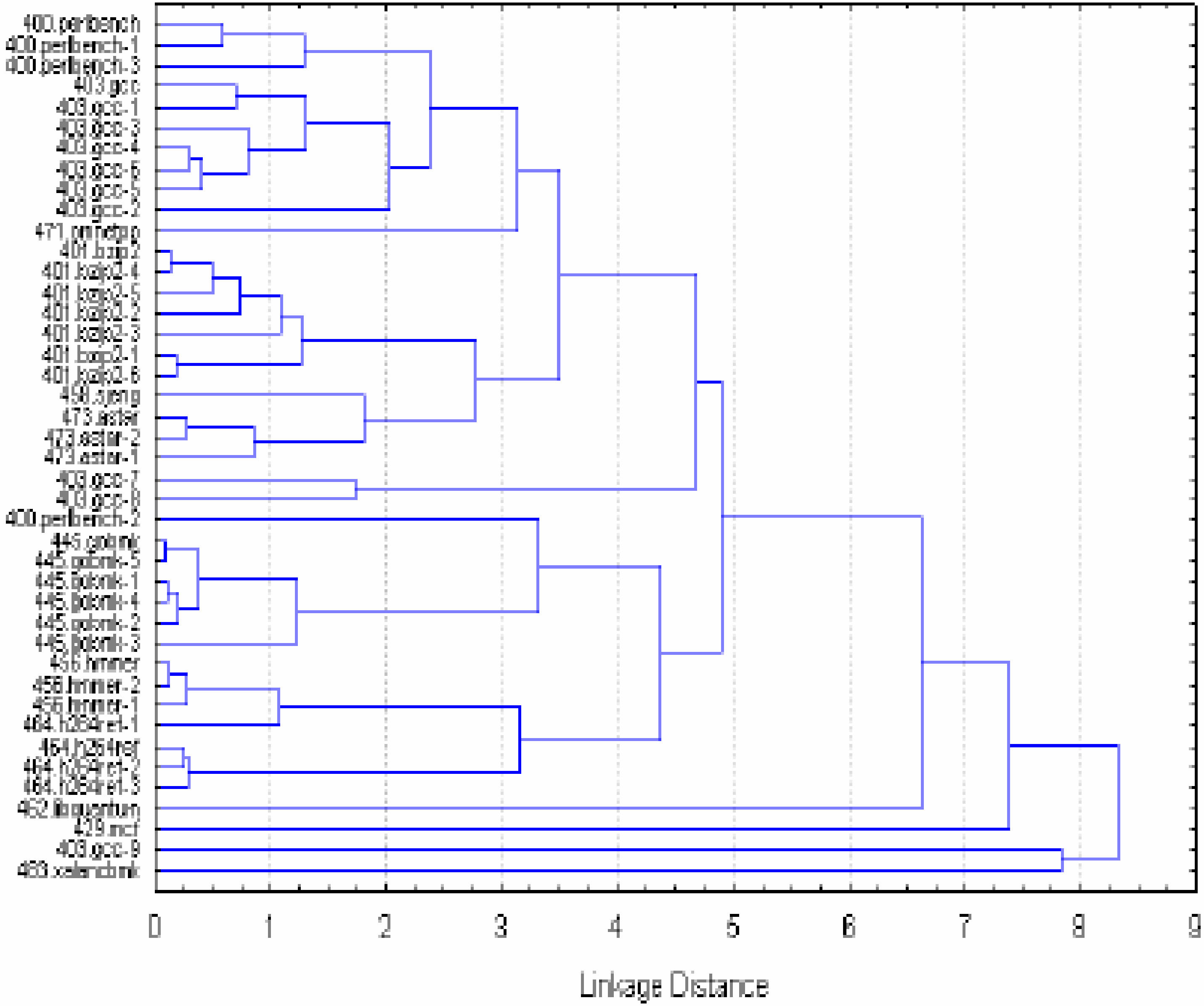


Figure 6. Dendrogram showing similarity between program-input set for each benchmark in the SPEC CPU2006 suite.

Table 6. List of representative input sets for SPEC CPU2006 programs.

CINT2006 benchmarks	464.h264avc - input set 2
400.perlbench - input set 1	473.astar - input set 2
401.bzip2 - input set 4	
403.gcc - input set 1	**CFP2006 benchmarks**
445.gobmk - input set 5	416.gamess - input set 3
456.hmmer - input set 2	450.soplex - input set 1

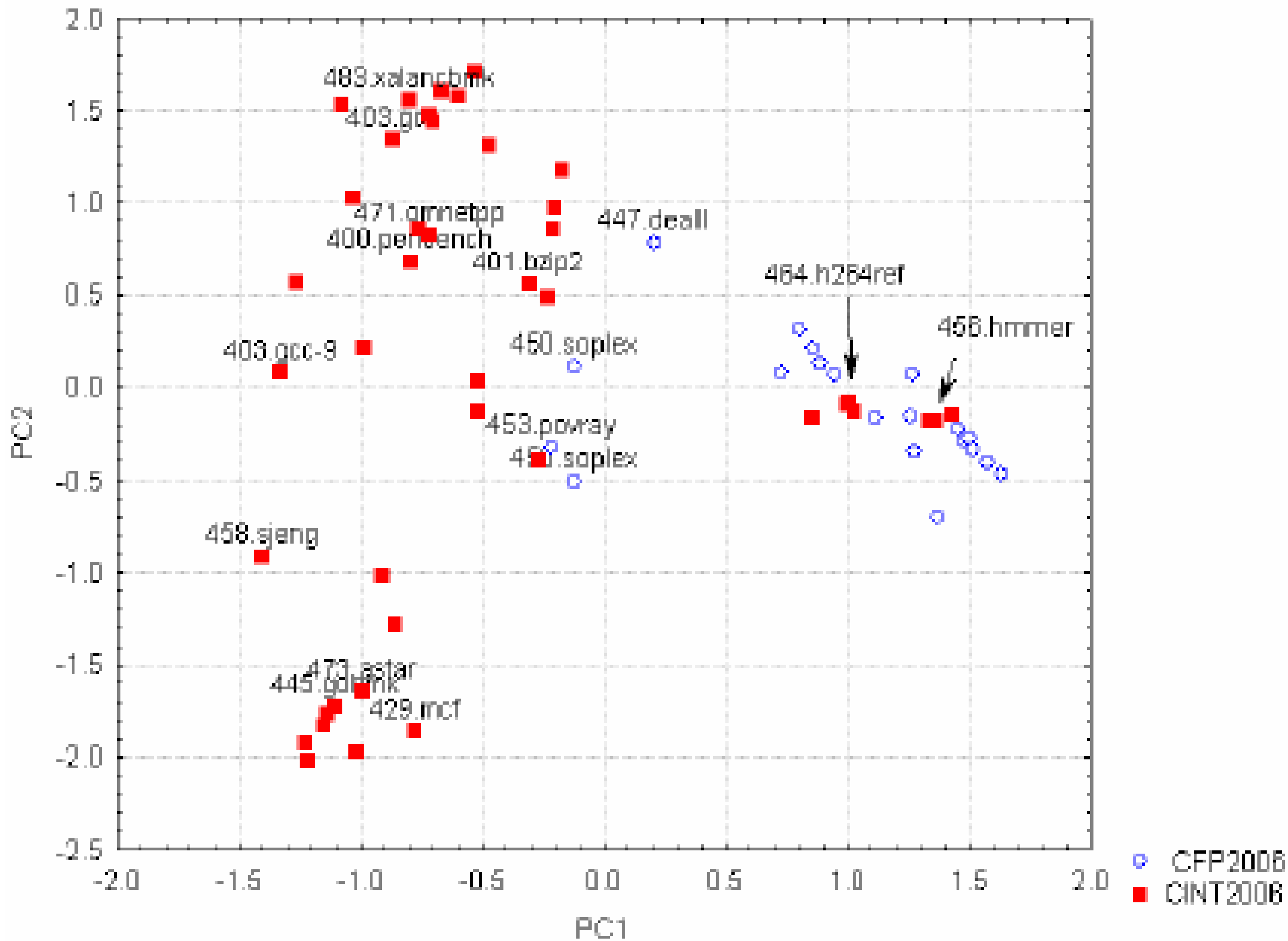
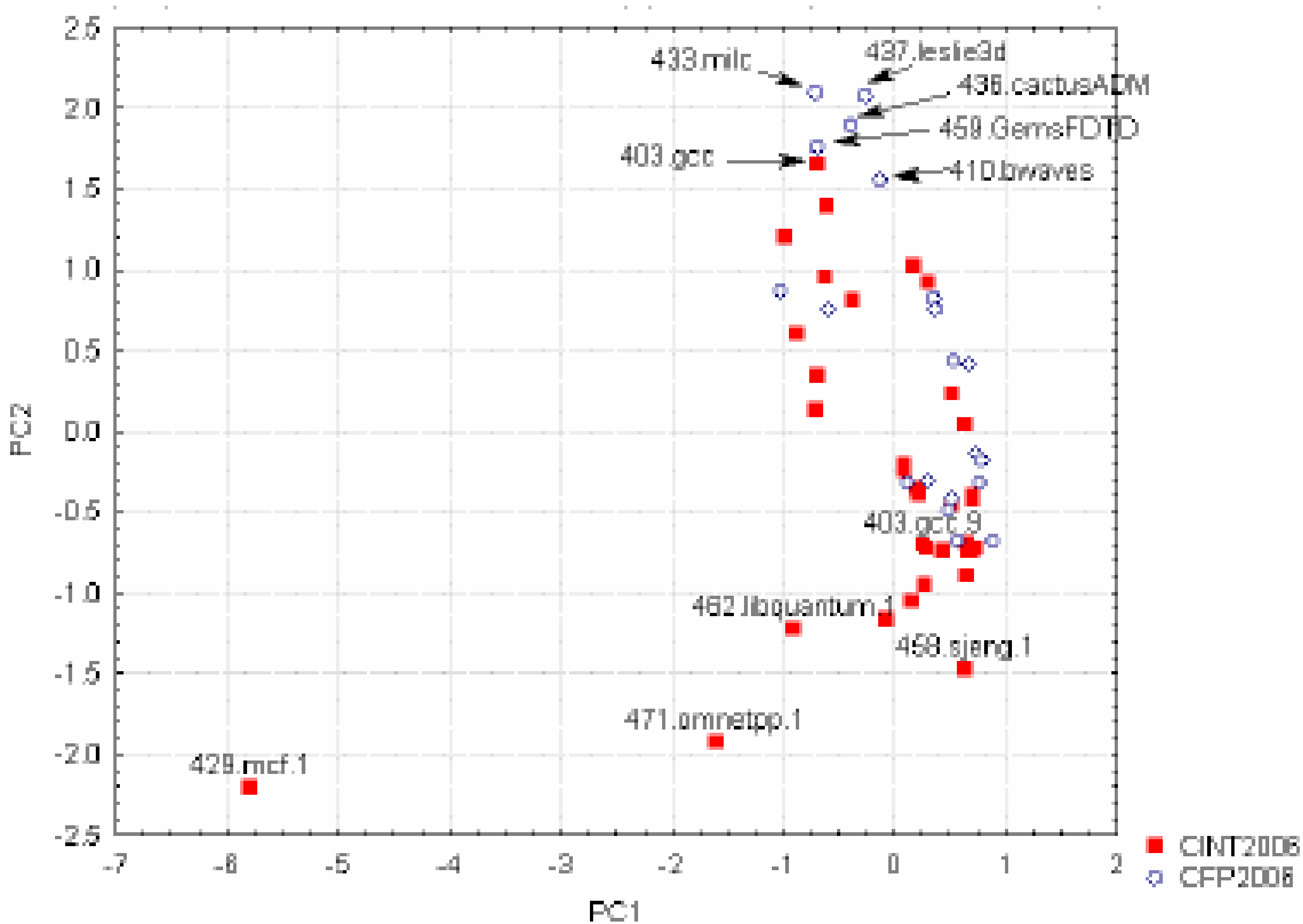
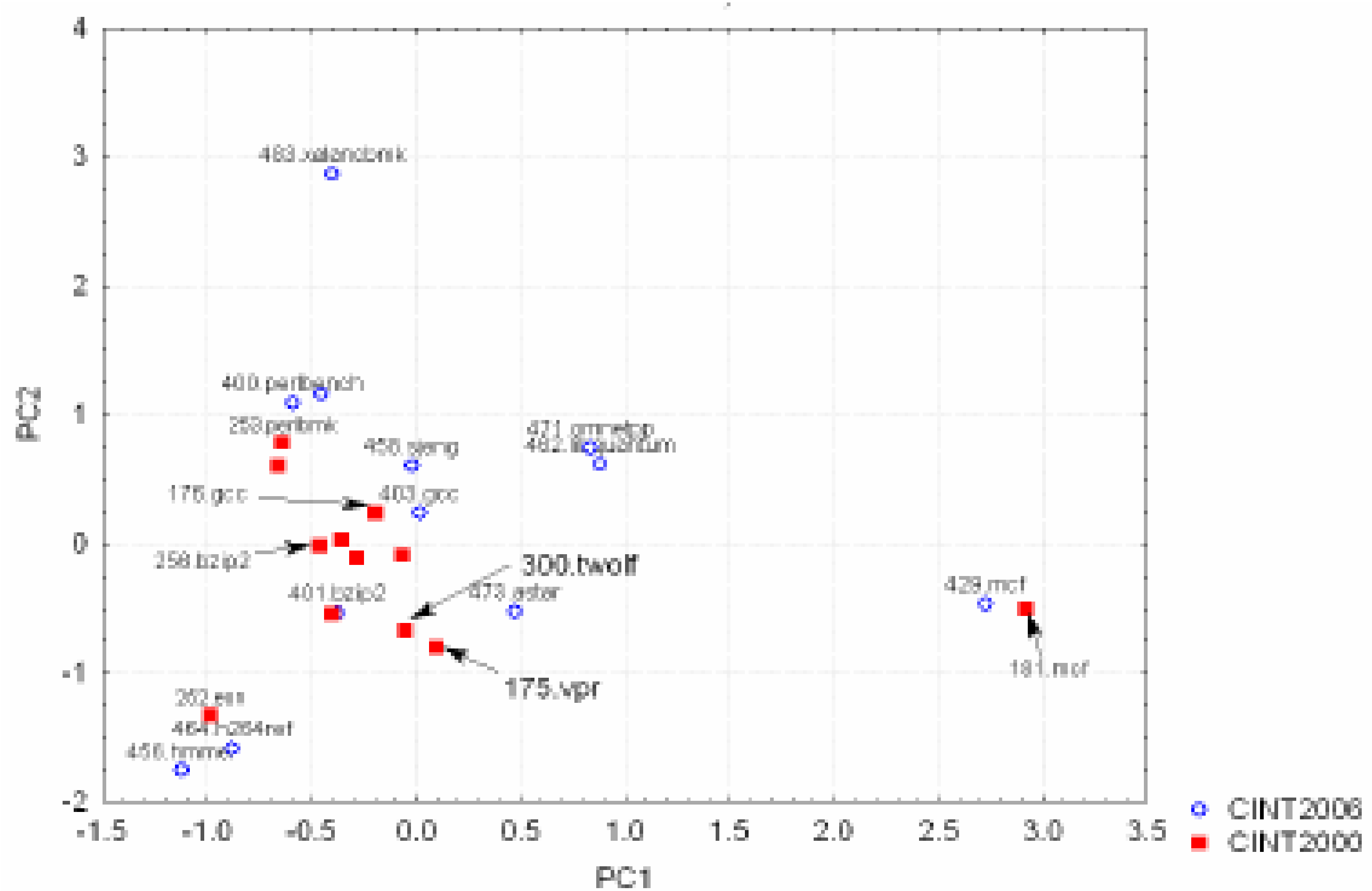


Figure 8. CINT and CFP programs in the PC workload space using branch predictor characteristics.



(a) PC1 Vs. PC2

Memory Characteristic space



(a) PC1 Vs. PC2

Figure 10. Scatterplot showing position of EDA applications in the workload space.

Table 7. Classification of programs based on application areas.

Application area	Benchmarks
Artificial Intelligence	**458.sjeng**, 445.gobmk, **473.astar**
Equation solver	436.cactusADM, 459.GemsFDTD
Fluid Dynamics	**410.bwaves**, 434.zeusmp, **437.leslie3D**, 470.lbm
Molecular Dynamics	**435.gromacs**, **444.namd**
Quantum Chemistry	**465.tonto**, **416.gamess**
Engineering and Operational Research	454.calculix, 447.dealII, 450.soplex, 453.povray

Table 8. Sensitivity of Programs to Branch Misprediction Rate and L1 D-cache Miss-rate across five different platforms.

Branch Prediction	
High	456.hmmer-1, 456.hmmer, 456.hmmer-2
Medium	471.omnetpp, 429.mcf, 473.astar-1, 473.astar, 464.h264ref-1, 473.astar-2, 400.perlbench-1, 401.bzip2-4, 462.libquantum,, 401.bzip2-3, 401.bzip2-2, 400.perlbench, 401.bzip2, 445.gobmk-3, 401.bzip2-1, 464.h264ref, 401.bzip2-5,, 403.gcc-8, 458.sjeng,401.bzip2-6, 403.gcc-4
Low	464.h264ref-3, 445.gobmk, 445.gobmk-1, 445.gobmk-4, 445.gobmk-2, 445.gobmk-5, 400.perlbench-2, 464.h264ref-2, 403.gcc-7, 403.gcc-6, 400.perlbench-3, 483.xalancbmk, 403.gcc-2, 403.gcc-5, 403.gcc-1, 403.gcc, 403.gcc-9, 403.gcc-3

Table 8. Sensitivity of Programs to Branch Misprediction Rate and L1 D-cache Miss-rate across five different platforms.

	L1 D-cache
High	462.libquantum, 464.h264ref-2, 464.h264ref-3, 464.h264ref, 456.hmmer-1
Medium	456.hmmer, 456.hmmer-2, 400.perlbench-2, 400.perlbench-3, 445.gobmk-3, 403.gcc-7
Low	400.perlbench, 403.gcc-8, 483.xalancbmk, 473.astar-2, 403.gcc, 400.perlbench-1, 473.astar, 464.h264ref-1, 445.gobmk, 473.astar-1, 445.gobmk-4, 471.omnetpp, 429.mcf, 403.gcc-9, 403.gcc-3, 445.gobmk-2, 401.bzip2-3, 401.bzip2-5, 445.gobmk-1, 403.gcc-6, 403.gcc-5, 401.bzip2-2, 401.bzip2-6, 403.gcc-2, 403.gcc-1, 401.bzip2-1, 401.bzip2, 403.gcc-4, 401.bzip2-4, 445.gobmk-5, 458.sjeng

[23] H. Vandierendonck, K. Bosschere, "Many Benchmarks Stress the Same Bottlenecks", *Proc. of the Workshop on Computer Architecture Evaluation using Commerical Workloads (CAECW-7)*, pp. 57-71, 2004.

3. In order to measure the sensitivity of a program to branch predictor and L1 D-cache configuration, for every machine we ranked programs based on these characteristics. The difference in ranks of a program across all machines is then computed. The resulting number is indicative of sensitivity of that program for a given characteristic.

We will discuss this after Plackett and Burman method – Yi et al – in a few weeks