Received: 6 October 2021 | Accepted: 10 December 2021

DOI: 10.1002/int.22791

RESEARCH ARTICLE

WILEY

BenchSubset: A framework for selecting benchmark subsets based on consensus clustering

Hongping Zhan[1] | **Weiwei Lin**[1] | **Feiqiao Mao**[2] | **Minxian Xu**[3] | **Guangxin Wu**[1] | **Guokai Wu**[1] | **Jianzhuo Li**[1]

[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou, China

[2]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China

[3]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

Correspondence

Weiwei Lin, School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China.
Email: linww@scut.edu.cn

Feiqiao Mao, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China.
Email: feiqiao@szu.edu.cn

Funding information

Key-Area Research and Development Program of Guangdong Province, Grant/Award Number: 2021B0101420002; National Natural Science Foundation of China, Grant/Award Numbers: 62072187, 62002078, 61872084; Guangzhou Science and Technology Program key projects, Grant/Award Number: 202007040002; Guangdong Major Project of Basic and Applied Basic Research, Grant/Award Number: 2019B030302002; Guangzhou Development Zone Science and Technology, Grant/Award Number: 2020GH10

Abstract

The redundancy in the benchmark suite will increase the time for computer system performance evaluation and simulation. The most typical method to solve this problem is to select subsets based on clustering. However, it is a challenge to validate benchmark subsetting results for unlabeled benchmark suites when using the clustering method, and existing research has not considered this problem. Also, there is no quantitative evaluation method for subsetting which can reflect the universal and the diversity characteristics of the benchmark suite at the same time. To solve the above problems, we propose BenchSubset, a framework for selecting benchmark subsets based on consensus clustering, which includes Group Principal Components Analysis, consensus clustering, and a new evaluation method considering the universal and the diversity characteristics of the benchmark suite. We conducted SPEC CPU2017 subsetting experiments on Huawei's Taishan 200, then verified the effectiveness of BenchSubset in selecting a benchmark subset. Compared with the mainstream principal components analysis with hierarchical clustering (PCA-H) method, the

benchmark subset selected by BenchSubset performs better in representing the universal and the diversity characteristics of SPEC CPU2017.

KEYWORDS
benchmark subsets, consensus clustering, SPEC CPU2017

1 | INTRODUCTION

Benchmarking has traditionally been focused on competitive system evaluation and comparison, which are mainly reflected in vendors that sell System under Test (SUT) use benchmarks to demonstrate the advantages of their products for marketing. However, in the past couple of decades, the application scenarios of benchmarking have gradually increased. For example, customers can use benchmarks to compare competing products from different suppliers, and researchers use benchmarks to evaluate novel system architectures.[1] However, the redundancy in the benchmark suites will increase the evaluation costs of computer system performance evaluation.[2,3] In addition, when developing, deploying, and operating the SUT, benchmarks can also be used to verify a given hardware and software configuration through simulation, but the simulation time is can be much slower than the actual execution time,[4] for instance, the microservices can be executed within milliseconds in realistic testbed, while the simulation toolkit can consume seconds or minutes for execution.

To solve the above problems, the existing research usually selects a benchmark subset from the benchmark suite to save time, but the randomly selected subset can lead to misleading conclusions. Therefore, to evaluate the performance of different computer systems more scientifically and efficiently and reduce the time of simulation, it is valuable to investigate how to select a benchmark subset that can reflect the performance characteristics and potential of the system from the original benchmark suite. In addition, the investigated approach can not only select typical subsets for existing benchmark suites but also guide the selection process of benchmark for constructing a new benchmark suite.

Although a lot of research has been done on the selection of benchmark subsets in recent years, most of the methods still have certain shortcomings. The benchmark subset selection method based on P&B design[5] costs a large simulation time overhead; the benchmark subset selection method based on genetic algorithm[2,6] has the advantage that the selected subset is relatively objective, but the initial individual selection of the algorithm has a relatively large impact on the final result. The most typical method is clustering.[7–19] When using the clustering method to extract subsets, the collected performance data is preprocessed through Principal Component Analysis (PCA) first, then the optimal size of benchmark subset and the optimal benchmark in each cluster is determined based on principal components (PCs) using cluster algorithm. However, unlike the Server Efficiency Rating Tool (SERT),[20] most of the benchmark suites have no labels. Therefore, it is a challenge to validate benchmark clustering results and determine the optimal size of the benchmark subset. In addition, from the perspective of evaluation methods, existing research mainly includes machine indicators, SPEC scores, and distance quantification,[2,13,15] but the benchmark subset should reflect the universal and the diversity characteristics of the benchmark suite at the same time as much as possible, and there is no comprehensive indicator now.

On the basis of the above research background, we propose the *BenchSubset* framework, which mainly addresses benchmark suite subsetting of the unlabeled benchmark suite. To verify the results of benchmark clustering, we use consensus clustering based on resampling to process the feature data representing the input benchmark suite, and through the consensus matrix generated by consensus clustering, we can determine the optimal size of the benchmark subset and optimal benchmark subset. To make up for the deficiencies of the existing evaluation method, we also proposed a new benchmark subset evaluation method. As a case study, BenchSubset selected the benchmark subsets for the four subsuites of unlabeled SPEC CPU2017. In general, our work has three main *contributions*:

1. We propose BenchSubset, a framework for selecting benchmark subsets based on consensus clustering, which encompasses the entire process of benchmark subset selection and can be extended to servers of any architecture. To the best of our knowledge, BenchSubset is the first framework that considers the validation of the benchmark subset selection results of the unlabeled benchmark suite, then realizes the selection of the benchmark subset based on consensus clustering, including the optimal size of the benchmark subset and the optimal benchmark in each cluster.
2. We present a new benchmark subset evaluation method, which can evaluate if the benchmark subset can reflect the universal and the diversity characteristics of the benchmark suite at the same time as much as possible.
3. When preprocessing the feature data, this paper replaces PCA with Group Principal Components Analysis (GPCA), which individually performs PCA operations on each group of feature data. Compared with PCA, it reduces the data dimension of the feature and eliminates correlations while retaining the various types of features as completely as possible.

The rest of this paper is organized as follows. Section 2 includes related research and SPEC CPU2017 introduction; Section 3 introduces our proposed BenchSubset framework workflow and its detailed design; Section 4 demonstrates the experimental analysis of SPEC CPU2017 subsetting; finally, the summary of the work is concluded in Section 5.

2 | BACKGROUND

In this section, we first analyzed and discovered the problems of the existing benchmark subset selection methods, then introduced SPEC CPU2017's design.

2.1 | Related work

Existing research mainly reduces benchmark execution and simulation time from three aspects, including reducing the input set,[4,21] benchmark subset selection, and statistics sampling.[22] Among them, the research on benchmark subset selection is relatively extensive. The current benchmark subsetting methods are mainly based on the hardware performance counter data of the benchmark at runtime. However, the more precisely we characterize a benchmark's performance on a given system, the less usable it is across different microarchitectures. Yi et al.[5] used the Plackett–Burman design matrix to visually show whether the parameters are in the normal range under different processor configurations, and calculate each parameter's impact on the processor performance.

The benchmark vector is composed of the influence ranking of all parameters. By calculating the Euclidean distance of the two benchmark vectors, and comparing it with the user-defined similarity threshold, if it is lower than the threshold, then the two benchmarks are similar. The smaller the distance, the more similar the benchmarks. The advantage of the Plackett–Burman design is that for a specific component, such as a processor, the computational complexity is linear, but for multiple components, multiple simulations are required. To save simulation time, the benchmark subset selection method combining dimensionality reduction and clustering[7–19] has gradually been applied, among which the commonly used cluster algorithm include hierarchical clustering and *K*-Means. When selecting the benchmark subset of SPEC CPU2017 through hierarchical clustering, Panda et al.[15] manually specified the optimal *K* value, and Limaye and Adegbija[16] used the sum of squares of errors (SSE) to determine the optimally selected benchmark, which is the sum of the Euclidean distances between the class centroids and the data points in the cluster. As the clusters are merged, the SSE value will increase. Finally, the number of clusters is selected according to the Pareto optimal solution of SSE and execution time. However, there is no strategy for selecting the optimal benchmark in each cluster. When applying *K*-Means for subset selection, Ajay Joshi et al.[17] used Bayesian Information Criterion (BIC) to select the optimal *K* value and selected benchmark close to the cluster center as a representative benchmark. Jia et al.[18] verified that the benchmark far from the cluster center is better for BigDataBench. On the basis of the fact that the benchmark has no labels and that it is impossible to judge which clustering method is more effective, Liu et al.[13] proposed a clustering algorithm selector for embedded benchmarks. All the feature data are first processed by PCA, then Linear Discriminant Analysis (LDA) is used to process the feature data labeled by the clustering algorithm, and the generated results are fed back to the clustering algorithm set. Finally, the optimal clustering algorithm and the optimal size of clusters are selected from 10 clustering algorithms through BIC. The optimal benchmark in each cluster is closest to the cluster center. In addition to cluster algorithms, genetic algorithms are also used in the selection of benchmark subsets. The selection method of benchmark subset based on the genetic algorithm needs to define benchmark characterization representation, benchmark coding method, and fitness function of evaluating the effect of benchmark subset. Jin et al.[2,19] used a vector composed of 29 features proposed by Phansalkar to represent the benchmark. The fitness function is determined by the workload space and the execution time. The advantage of this method is that the selected subset is relatively objective, but the design of the fitness function is not particularly reasonable. It does not take the score of the benchmark into account, such as the SPEC CPU2017 score, and the selection of the initial individual of the algorithm has a great impact on the final effect.

The BenchSubset framework proposed in this paper is an improvement on the common benchmark subset selection method combining dimensionality reduction and clustering. BenchSubset not only considers the problem of not having a one-size-fit-all clustering algorithm but also determines the optimal size of the benchmark subset and the optimal benchmark in each cluster; compared with BenchPrime,[13] BenchSubset takes the validation of the benchmark subset selection results of the unlabeled benchmark suite into account.

2.2 | SPEC CPU2017

As a motivational example, we selected the benchmark subset for the unlabeled SPEC CPU2017 suite. SPEC CPU2017 is the latest version of SPEC CPU, which mainly stresses the performance of the processor, memory subsystem, and compiler.[23] Compared with SPEC CPU2006, SPEC CPU2017 is larger and more complex. First, CPU2017 updates the compiler, adds more functions, and improves

performance.[24] Second, from the perspective of workload, SPEC CPU2017 replaces some benchmarks in CPU2006 with larger and more complex workloads. It also updates the application domain, such as adding three-dimensional (3D) rendering, biomedical imaging, image processing, and other new areas to the floating point (FP) category, and removing benchmarks in the fields of speech recognition and electronic automation design. In general, SPEC CPU2017's source code has grown rapidly,[23] so there is greater redundancy compared to SPEC CPU2006. Therefore, to improve the efficiency of its benchmarking, it is very significant to select a benchmark subset of SPEC CPU2017. SPEC CPU2017 contains 43 benchmarks, including four subsuites, SPECrate 2017 Integer, SPECrate 2017 Floating Point, SPECspeed 2017 Integer, and SPECspeed 2017 Floating Point (hereinafter referred to as SPECrate INT, SPECrate FP, SPECspeed INT, and SPECspeed FP), the specific workloads are shown in Table 1. The test types include integer arithmetic and floating-point arithmetic, as well as two modes of rate and speed, respectively. Speed is the calculation speed test for a single workload, and the rate is the throughput testing for the system to run multiple of the same workload, for these two modes, threads and copies must be given before running, respectively.

To ensure the fairness of the benchmark, the SPEC CPU benchmark suite uses speed and throughput indicators across different architectures to measure and compare performance. The performance ratio of a benchmark i in speed mode and rate mode is calculated as

$$Ratio_{speed}(i, threads) = \frac{time(ref, 1)}{time(SUT, threads)}, \tag{1}$$

$$Ratio_{rate}(i, copies) = copies * \left(\frac{time(ref, 1)}{time(SUT, copies)} \right), \tag{2}$$

where $time(ref, 1)$ is the running time on the reference machine. The reference machine selected in this article is Sun Fire V490, and the number of concurrent threads and copies in

TABLE 1 SPEC CPU2017

SPECrate 2017 Integer	SPECrate 2017 Floating Point	SPECspeed 2017 Integer	SPECspeed 2017 Floating Point
500.perlbench_r	503.bwaves_r	600.perlbench_s	603.bwaves_s
502.gcc_r	507.cactuBSSN_r	602.gcc_s	607.cactuBSSN_s
505.mcf_r	508.namd_r	605.mcf_s	619.lbm_s
520.omnetpp_r	510.parest_r	620.omnetpp_s	621.wrf_s
523.xalancbmk_r	511.povray_r	623.xalancbmk_s	627.cam4_s
525.x264_r	519.lbm_r	625.x264_s	628.pop2_s
531.deepsjeng_r	521.wrf_r	631.deepsjing_s	638.imagick_s
541.leela_r	526.blender_r	641.leela_s	644.nab_s
548.exchange2_s	527.cam4_r	648.exchange2_s	649.fotonik3d_s
557.xz_r	538.imagick_r	657.xz_s	654.roms_s
	544.nab_r		
	549.fotonik3d_r		
	554.roms_r		

speed and rate modes are both 1.[25] For the benchmark i, $time(SUT, threads)$ is the running time of the SUT when the number of concurrent threads is the threads in speed mode, and $time(SUT, copies)$ is the running time of the SUT when the number of copies is the copies in the rate mode. Finally, the performance score calculation method of CPU2017 is shown in formula (3), where N is the number of benchmarks of CPU2017:

$$SCORE = \left\{ \left(\prod_{i=1}^{N} Ratio(i) \right)^{\frac{1}{N}} \middle| \; i \in SPEC \right\}. \tag{3}$$

3 | BENCHSUBSET

In this section, we first introduced the workflow of the proposed BenchSubset framework, and then introduced the major methods in BenchSubset in detail, including group PCA, benchmark subset selection based on consensus clustering, and new evaluation method.

3.1 | BenchSubset overview

The goal of BenchSubset is to provide its users with a framework for selecting a benchmark subset. Figure 1 shows the workflow of the proposed BenchSubset framework. For a given benchmark suite $BenchSet_{full} = \{bech_1, bech_2, ..., bech_n\}$, the purpose of this framework is to select the most representative benchmark subset $BenchSet_{sub} = \{bech_1, ..., bech_{k_{best}}\}$, where n is the number of benchmarks in the benchmark suite, k_{best} is the number of benchmarks in the benchmark subset.

BenchSubset takes a set of input performance counter data that summarize the characteristics of a given benchmark suite. For example, SPEC CPU2017 emphasizes the performance of processor, memory, and compilers, so we mainly consider the feature related to the processor and memory, including CPI, cache miss rate, memory behavior, and so forth. All the feature data are first processed by GPCA to lower their dimensions and eliminate their correlation with the principal components while keeping each type of feature as complete as possible. In GPA, the feature data are first divided into groups based on computer components, including CPU group, Cache group, and so forth, then each group of data is processed by PCA separately. Considering that it is impossible to directly select the

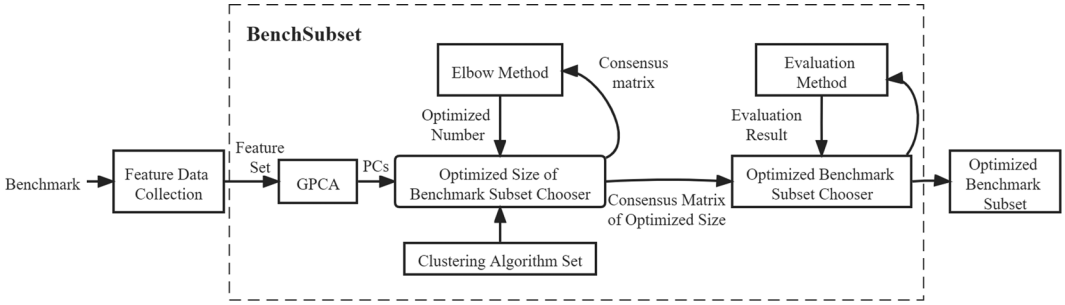

FIGURE 1 BenchSubset framework. GPCA, Group Principal Components Analysis; PCs, principal components

best built-in clustering algorithm in consensus clustering, we provide a clustering algorithm set and select a benchmark subset based on the consensus clustering with multiple clustering algorithms. For a specific consensus clustering and a possible number of the benchmark subset k, the optimized size of the benchmark subset chooser first generates the corresponding consensus matrix which takes the principal components (PCs) as inputs based on consensus clustering. Then, BenchSubset uses the consensus distribution of the consensus matrix and elbow method to determine the optimized number of benchmark subsets. Once the optimal size of the benchmark subset is chosen, the optimized benchmark subset chooser establishes the clusters' boundaries by using the corresponding consensus matrix as a similarity measure to feed to hierarchical clustering with average linkage. In each cluster, BenchSubset chooses the best benchmark based on stability. Finally, the optimized benchmark subset is selected based on our proposed evaluation method by comparing the subsetting result of the cluster in the clustering algorithm set.

3.2 | Design of BenchSubset

3.2.1 | Group PCA

We collected feature data from the system's hardware performance counters by running SPEC CPU2017 on a real system. However, there are some problems to feed the raw measurement data into BenchSubset directly. On the one hand, the number of dimensions in collected feature data is too high. On the other hand, there are correlations between feature data. First, there are many-to-many correlations between functional unit and hardware performance counters, so it is difficult to completely reflect their work through a single performance counter. For example, read access to the memory means that the cache has been accessed. Second, there are complex correlations between the events represented by the performance counters. when a performance counter changes, the performance counters related to it also change accordingly. For example, a hit in the L3 cache means that the performance counter value of the L1/2 cache miss increases. Existing research generally uses PCA to solve these problems, which aims at reducing the dimensionality of features and eliminating the correlation between the features. However, it is too rough to perform PCA processing on all types of feature data directly while each benchmark performs differently in different types of feature data.

To overcome the above difficulties, we propose GPCA based on PCA, which individually performs PCA operations on each group of feature data. In this way, GPCA can reduce the data dimension and eliminate correlations while retaining the various types of feature data as completely as possible. we divide the overall collected feature data into five types as follows, then perform PCA processing on these five groups of data, respectively:

- *CPU group*: Processor-related performance events, including CPU utilization, IPC, and so forth.
- *Cache group*: Performance events related to the cache, can be subdivided into four groups: L1 cache, L2 cache, L3 cache, and TLB.
- *Branch group*: Branch prediction error rate.
- *Memory group*: Performance events related to memory read and write.
- *Power group*: The power data of the server, including the average power, standard deviation, first quartile, median, third quartile, and maximum and minimum values.

3.2.2 | Benchmark subset selection based on consensus clustering

1. *Consensus clustering*

While many criteria are used to determine the optimal size of benchmark subset based on the cluster algorithm, that is, the K value, the general clustering algorithm does not consider the validation of the clustering results. On the basis of the assumption that the more the attained clusters are robust to sampling variability, the more we can confident that these clusters represent the real structure, consensus clustering[26] simulates the perturbations of the original data by resampling techniques to verify the clustering results. Consensus clustering represents and quantifies the agreement among the clustering runs over the perturbed data sets using consensus matrix, which stores, for each pair of benchmarks, the proportion of clustering runs in which two benchmarks are clustered together.

BenchSubset uses subsampling techniques, whereby a subset of benchmarks is sampled without replacement from the benchmark suite. Let $BenchSet^{(1)}$, $BenchSet^{(2)}$, ..., $BenchSet^{(H)}$ be the list of H perturbed data sets obtained by subsampling the original benchmark data sets $BenchSet_{full} = \{bech_1, bech_2, ..., bech_n\}$, $D^{(1)}$, $D^{(2)}$, ..., $D^{(H)}$ is the corresponding feature data after the GPCA processing. $M^{(h)}$ represents the consensus matrix of $BenchSet^{(h)}$, that is, the result of processing $D^{(h)}$ through consensus clustering. The entries of $M^{(h)}$ are defined as the formula (4). Due to the use of subsampling, the benchmarks in the benchmark suite are not always included in the subsampled data set. Therefore, it is necessary to record the benchmarks when subsampling. Consensus clustering uses indicator matrix $I^{(h)}$ to record it, the entries of $I^{(h)}$ are equal to 1 if both $bech_i$ and $bech_j$ are present in the data set $D^{(h)}$, and 0 otherwise. Finally, the consensus matrix can be defined as a properly normalized sum of the connection matrices of all the perturbed data sets as shown in formula (5):

$$M^{(h)}(i,j) = \begin{cases} 1 & bench_i \ \ and \ \ bench_j \ \ \text{cluster together,} \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

$$C(i,j)^{(k)} = \frac{\sum_h M^{(h)}(i,j)}{\sum_h \mathrm{I}^{(h)}(i,j)}. \tag{5}$$

(2) *Clustering algorithm candidate*

Consensus clustering requires a built-in clustering algorithm (the Cluster in Algorithm 1), in addition, there is no one-size-fit-all clustering algorithm, the optimal clustering algorithm varies depending on different benchmark suites. For this purpose, BenchSubset considers three clustering algorithms in sklearn.cluster[27] (the Clusters in Algorithm 1): (i) Hierarchical clustering, and (ii) K-means, and (iii) Spectral, then determines three algorithms, including consensus clustering with hierarchical clustering (CC_{HC}), consensus clustering with K-means (CC_{Kmeans}), consensus clustering with Spectral ($CC_{spectral}$).

(3) *Determining the optimal size of the benchmark subset*

On the basis of the consensus matrix generated by the consensus clustering, BenchSubset can determine the optimal size of the benchmark subset. Algorithm 1 details how to perform this process taking a GPCA generated data set D. BenchSubset first loops all the initial built-in clustering algorithm of consensus clustering, then loops each possible number of benchmark subsets k to find the optimal size of benchmark subset for each built-in consensus clustering. The range of k $[2, k_{max}]$ is affected by the subsampling rate and the

size of the input benchmark suite, for example, for the case study in Section 4, SPECrate INT contains 10. When the subsampling rate is 80%, then the value of k_{max} cannot exceed 8. For a given k, BenchSubset first loops the current iteration h and resamples the benchmark suite data D until the number of iterations reaches the given H. On the basis of the resampled data $D^{(h)}$, the connection matrix $M^{(h)}$ and the indicator data $I^{(h)}$ are obtained. Finally, we can obtain the consensus matrix $C^{(k)}$.

Algorithm 1. Determining the optimal number of benchmark subset

Input: GPCA generated data set D; clustering algorithms set *Clusters*; max number of benchmark subset k_{max}; number of iterations H.
Ouput: k_{best}, best number of benchmark subset.

```
function FINDBESTK
  for Cluster in Clusters do
    for k ∈ [2, k_max] do
      M ← ∅
      I ← ∅
      for h = 1, 2, ..., H do
        D^(h) ← Resample(D)
        I^(h) ← D^(h)
        M^(h) ← Cluster(D^(h), k)
        M ← M ∪ M^(h)
        I ← I∪I^(h)
      end for
      C^(k) ← compute from M^(h) and I^(h)
    end for
    k_best ← compute based on consensus distribution
  end for
end function
```

Once the consensus matrix for each possible number of benchmark subset k is obtained, BenchSubset uses the consensus distribution, and the proportion change in the area under the consensus Cumulative Distribution Function (CDF) to determine the optimal size of benchmark subset. If the distribution of the consensus matrix is skewed towards 0 and 1, then it can be decided that the k value is the optimal size of the benchmark subset k_{best}. For the case of no obvious skew, it is necessary to calculate the CDF and its area, then evaluate the optimal size of the benchmark subset by comparing the change of the CDF area with the increase of k. The calculation of the CDF and its area change are shown in the formulas (6) and (7):

$$CDF(c) = \frac{\sum_{i<j} 1\{C(i,j) \le c\}}{\frac{n(n-1)}{2}}, \tag{6}$$

$$A(k) = \sum_{i=2}^{M} [x_i - x_{i-1}] CDF(x_i), \tag{7}$$

where $1\{C(i,j) \le c\}$ is the indicator function that is equal to 1 when $C(i,j) \le c$ is true, and 0 otherwise. The set $\{x_1, x_2, ..., x_M\}$ is the sorted set entries of the consensus matrix $C^{(k)}$ (with

$M = n(n-1)/2$. When the built-in clustering algorithm is hierarchical clustering, the CDF area change of the consensus matrix is shown in the formula (8):

$$\triangle(k) = \begin{cases} A(k) & \text{if} \quad k = 2, \\ \dfrac{A(k+1) - A(k)}{A(k)} & \text{if} \quad k > 2, \end{cases} \tag{8}$$

where $A(k)$ is the CDF area corresponding to the value of k. When the built-in algorithm is not hierarchical clustering, replace $A(k)$ with $\hat{A}(k) = \max_{k' \epsilon \{2, \ldots, k\}} A(k')$ to solve. When the CDF area change tends to be stable, the point is deemed to be k_{best} of the input benchmark suite, which can be judged by the elbow method.

(4) *Determining optimal benchmark in each cluster*

For the selected consensus clustering algorithm (such as CC_{HC}), once the optimal size of the benchmark subset k_{best} is determined, BenchSubset takes the consensus matrix $C^{(k_{best})}$ as a similarity measure to feed to a hierarchical clustering with average linkage, then the benchmark suite is divided into k_{best} clusters. For the benchmark cluster $P = \{P_1, \ldots, P_{k_{best}}\}$, the optimal benchmark selection strategy in the cluster P_k is: when there is only one benchmark, it is obvious that the benchmark is the optimal benchmark; when there are two benchmarks, the benchmark with a shorter running time is selected; when the number of benchmarks in the cluster is greater than 2, we can select the optimal benchmark by comparing the consensus index. The calculation of the consensus index of the benchmark i in the cluster P_k is shown in formula (9):

$$m_i(P_k) = \frac{1}{N_k - 1\{i \in I_k\}} \sum_{\substack{i \in I_k \\ j \neq i}} C(i, j), \tag{9}$$

where I_k is the benchmark set belonging to P_k, and N_k is the number of benchmark in P_k. When benchmark $i \in I_k$, the value of $1\{i \in I_k\}$ is 1, and 0 otherwise. The benchmark with the largest consensus index in the cluster P_k is the optimal benchmark in the cluster. The set of the optimal benchmark in all clusters $BenchSet_{sub}^{(cluster_i)} = \{bech_1, \ldots, bech_{k_{best}}\}$ is the benchmark subset of selected consensus clustering.

3.2.3 | Evaluation method

A representative benchmark subset should reflect the universal feature of the benchmark suite first, but the redundant benchmarks in the benchmark suite have similar performance, which may skew the general characteristics towards the redundant benchmark, so the benchmark program in a representative benchmark subset also should reflect the diverse characteristics. Overall, a representative benchmark subset should be able to reflect not only the universal but also the diversity characteristics of the benchmark suite as much as possible. On the basis of the analysis of References [2,13,15], we proposed a new benchmark subset evaluation method-representativeness score (RS), which combines the universality score (US) similarity and diversity score (DS) similarity.

We regard each benchmark in the benchmark suite (denoted as BenchSet) as a point in a multidimensional space, which can be represented by the vector $bench^T = (e_1, e_2, ..., e_m)^T$, where e_i represents a specific basic feature, and m is the number of basic feature. This paper selects the basic features belonging to six categories, including cycle instructions, cache, branch prediction, and memory read and write which reflect the processor architecture, and average power, SPEC CPU score which with regard to performance and energy consumption. For a BenchSet, we use the mean vector of all benchmarks in BenchSet to express the universality, and the standard deviation vector expresses the diversity, as shown in the following formulas (10) and (11):

$$\overline{BenchSet}^T = (\overline{e_1}, \overline{e_2}, ..., \overline{e_m})^T, \tag{10}$$

$$BenchSet^T_{std_{dev}} = (\sigma_1, \sigma_2, ..., \sigma_m)^T, \tag{11}$$

where $\overline{e_i}$ is the mean value of the basic feature e_i of the benchmarks in BenchSet, σ_i corresponds to the standard deviation, and the calculation of $\overline{e_i}$ and σ_i are as follows (N represents the number of benchmark in the BenchSet, e_{ij} is the $\overline{e_i}$ value of the benchmark j in the BenchSet):

$$\overline{e_i} = \frac{\sum_{j=1}^{N} e_{ij}}{N}, \tag{12}$$

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{N} (e_{ij} - \overline{e_i})^2}{N}}. \tag{13}$$

To measure the similarity of universality and diversity between BenchSet and its subset, we use the mean vector's cosine similarity of them to measure the similarity of the universality, and standard deviation's cosine similarity of them to measure the diversity, the calculations are shown in the following formulas (14) and (15). The closer the cosine value is to 1, the more similar the two vectors are, and the greater the difference otherwise. To evaluate the representativeness of the benchmark subset, this paper uses the geometric mean of the similarity of universality and diversity as the final evaluation index, as shown in formula (16). The closer the value is to 1, the better the representativeness of the subset, the worse otherwise.

$$US = \frac{\overline{BenchSet_{full}} \cdot \overline{BenchSet_{sub}}^T}{\|\overline{BenchSet_{full}}\| \times \|\overline{BenchSet_{sub}}\|}, \tag{14}$$

$$DS = \frac{BenchSet_{stddev_{full}} \cdot BenchSet_{stddev_{sub}}{}^T}{\|BenchSet_{stddev_{full}}\| \times \|BenchSet_{stddev_{sub}}\|}, \tag{15}$$

$$RS = \sqrt{US * DS}, \tag{16}$$

where $\overline{BenchSet_{full}}$ is the universality of the benchmark suite, while $\overline{BenchSet_{sub}}^T$ is the universality of its subset, $BenchSet_{stddev_{full}}$ is the diversity of the benchmark suite, while

$BenchSet_{stddev_{sub}}$ is the diversity of its subset. By comparing the RS values of the benchmark subsets selected by three specific consensus clustering algorithms, that is, CC_{HC}, CC_{kmeans}, $CC_{spectral}$, we can choose the benchmark subset closest to 1 as the optimal benchmark subset $BenchSet_{sub} = \{bech_1, ..., bech_{k_{best}}\}$.

4 | SUBSETTING SPEC CPU2017

As a case study, we use BenchSubset to select subsets for SPEC CPU2017. We first determined the experimental environment and the variable settings in the consensus clustering. Second, we verified the effectiveness of the GPCA and evaluation method. Finally, based on the evaluation method in Section 3.2.3, we compared the subsetting results of BenchSubset and principal component analysis with hierarchical clustering (PCA-H) method.

4.1 | Experimental setup

4.1.1 | Hardware and software setup

To verify the effectiveness of the benchmark subset selection and evaluation method we proposed, we conducted SPEC CPU2017 subsetting experiments on Huawei's Taishan 200. As ARM servers are gradually emerging in high-performance computing (HPC),[28,29] the application of SPEC CPU2017 on the arm server will be more and more extensive. The hardware and software configurations of Taishan 200 are shown in Table 2. The number of copies of the Rate suite and the number of threads of the Speed suite in SPEC CPU2017 are uniformly set to 96.

To reduce the overhead of data collection,[30] we choose Linux kernel tool perf to collect performance data, and Taishan 200's iBMC to collect power data. We started from the two aspects of the processor and the memory subsystem, combined with the armv8-A white paper[30] and the kernel code of Taishan 200[32,33] to select the performance events. The selection results are shown in Table 3, where $k = 1, 2$, for example, l<k>d_cache_rd includes l1d_cache_rd and

TABLE 2 Taishan 200's Configuration

Processors	Taishan 200 (Model 2280)—Kunpeng 920 processor, Dual socket armv8 architecture 48 cores-can execute 48 threads (per processor)
L1d cache	64K (per core)
L1i cache	64K (per core)
L2 cache	512K (per core)
L3 cache	32768K (four cores)
OS	CentOS Linux release 7.9.2009
	Linux kernel: 4.18.0-193.1.2.el7.aarch64
	perf version: 4.18.0-193.1.2.el7.aarch64
	GCC: 8.2.0

l2d_cache_rd, o represents instruction and data, for example, <o>TLB-loads includes iTLB-loads and dTLB-loads. For ×86 servers, we only need to select performance events of the corresponding category according to the performance counter parameters provided by perf. The performance events used in the evaluation method are shown in Table 4.

4.1.2 | Consensus clustering setup

Consensus clustering requires iterations H and subsampling rate R, but the existing research has not analyzed the determination of these two parameters. Therefore, we take CC_{HC} as an example, for each k, we run H from 1000 to 7000, at each iteration, the perturbed benchmarks are obtained by subsampling R from 60% to 90%, and we compared the result using the evaluation method proposed in Section 3.2.3. Figure 2A–D shows the comparison results of SPECrate INT, SPECrate FP, SPECspeed INT, and SPECspeed FP, respectively. For SPECrate INT, when $H = 4000$, the overall performance of the subset selected by SPECrate INT is the best; when $R = 60\%$ or 70%, regardless of the number of iterations, the subset selection results are better. For SPECrate FP, when $H = 7000$ and $R = 60\%$, the selected subset can best represent the benchmark suite; when $R = 60\%$, it performs best under most iterations, except for the case of $H = 2000$. When $H = 7000$, the subset selected of SPECspeed INT performs best overall, and the best performance is when $R = 90\%$. When $H = 3000$, the overall performance of the subset selected by SPECspeed FP is the best; among them, when $R = 60\%$ or 70% is relatively better while $R = 60\%$ performs better only when $H = 3000$; for other subsampling rates, performance stability is better in any number of iterations. On the whole, for different CPU2017 subsuite,

TABLE 3 Performance events

Component	Type	Performance
Processor	CPU	instructions, cycles, task-clock, context-switches, CPU-migrations, stalled-cycles-backend, stalled-cycles-frontend
	Cache	L1-dcache-loads, L1-dcache-load-misses, L1-icache-loads, L1-icahce-load-misses, l<k>d_cache_rd, l<k>d_cache_wr, l<k>d_cache_refill_rd, l<k>d_cache_refill_wr, l<k>d_cache_wb_victim, l<k>d_cache_wb_clean, l<k>d_cache_inval armv8_pmuv3_0/l<k><o>_cache_refill/, armv8_pmuv3_0/l<k><o>_cache/, l1i_cache_prf, l1i_cache_prf_refill
	TLB	<o>TLB-loads, <o>TLB-load-misses, l1d_tlb_rd, l1d_tlb_wr, l1d_tlb_refill_rd, l1d_tlb_refill_wr, armv8_pmuv3_0/l<k><o>_tlb_refill/, armv8_pmuv3_0/l<k><o>_tlb/, armv8_pmuv3_0/<o>tlb_walk/
	Branch	branch-misses, armv8_pmuv3_0/br_mis_pred/, armv8_pmuv3_0/br_pred/
Memory	DDRC/HHA	uncore_hisi_ddrc.act_cmd, uncore_hisi_hha.rd_ddr_128b, uncore_hisi_hha.rd_ddr_64b, uncore_hisi_hha.rx_ops_num, uncore_hisi_hha.rx_outer, uncore_hisi_hha.rx_sccl, uncore_hisi_hha.wr_ddr_128b, uncore_hisi_hha.wr_dr_64b
	NUMA	armv8_pmuv3_0/remote_access_rd/, armv8_pmuv3_0/remote_access/, armv8_pmuv3_0/bus_access/, armv8_pmuv3_0/bus_cycles/, armv8_pmuv3_0/mem_access/

TABLE 4 Performance events in evaluation method

Performance events	Type
SPEC CPU Score	SPEC CPU score
IPC	IPC
L1 cache miss	Cache
L2 cache miss	
L3 cache miss	
TLB miss	
Branch prediction miss	Branch
hha_rd_ddr	Memory read and write
hha_wr_ddr	
scoket_rd_hits	
Bus access per cycle	
Mean power	Power

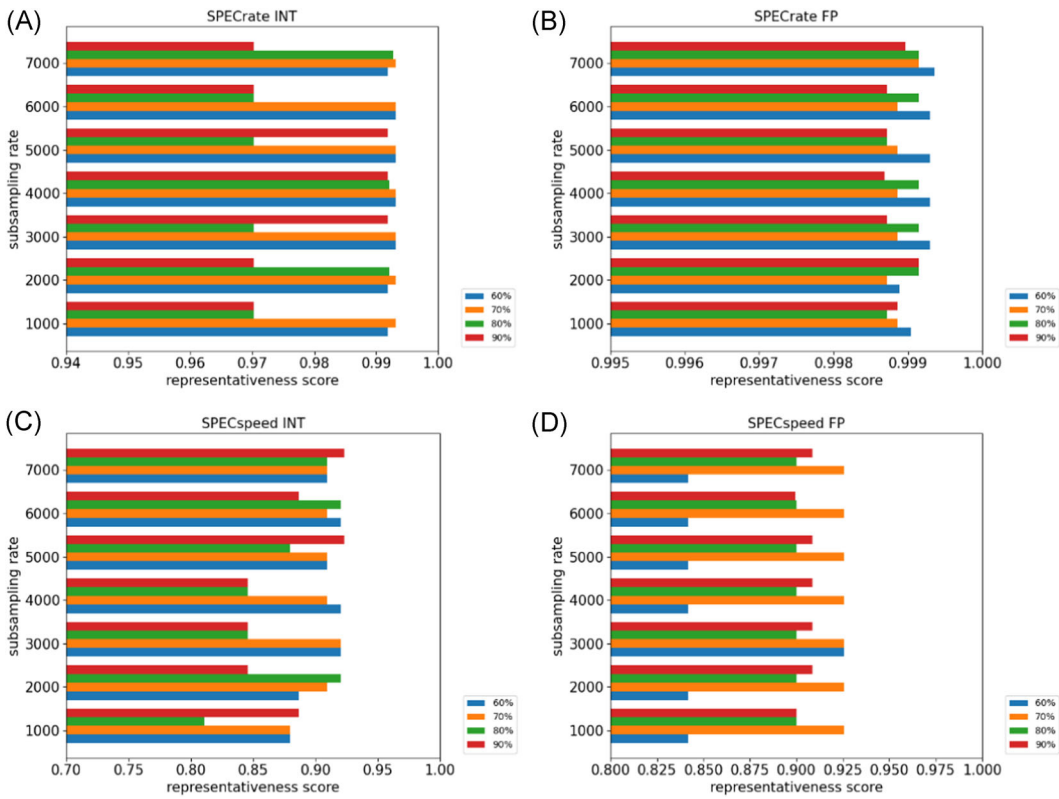

FIGURE 2 Consensus clustering setup's comparison results [Color figure can be viewed at wileyonlinelibrary.com]

neither the number of iterations nor the subsampling rate has an absolute relationship with the quality of the benchmark subset selection result. To ensure the consistency of the experimental environment, we unified the consensus clustering parameters to the number of iterations $H = 6000$ and the $R = 80\%$.

4.2 | Effectiveness analysis of GPCA

To verify the effectiveness of the GPCA in Section 3.2.1, We used Pycharm to perform the GPCA on the 43 benchmarks of CPU2017, and reduced our data dimensionality from $[43 \times 64]$ to $[43 \times 18]$, covering over 95% of the variance, where the CPU group dimensionality is reduced from $[43 \times 7]$ to $[43 \times 2]$, the L1 cache group dimensionality is reduced from $[43 \times 8]$ to $[48 \times 3]$, the TLB group dimensionality is reduced from $[43 \times 9]$ to $[43 \times 2]$, and the memory group dimensionality is reduced from $[43 \times 5]$ to $[43 \times 2]$. Figure 3A–D shows the scatter plots of the first two PC values of CPU group, L1 cache group, TLB group, and Memory group, respectively. We use square boxes to highlight benchmarks which PC values close together in Figure 3. The benchmarks with similar PC values will show similar performance characteristics, but it is obvious that the performance characteristics of benchmarks are not consistent in different groups. As seen in Figure 3, 628.pop2_s is isolated in the CPU group and L1 cache group, but it is close to other benchmarks in the other two groups. 620.omnetpp_s and 623.xalancbmk_s are close together in the CPU group but are isolated in the L1 cache group. Therefore, we can conclude that the benchmarks have different performance characteristics in different categories, and the GPCA can reduce the feature dimension and eliminate correlations while retaining the characteristics of each category.

4.3 | Result of SPEC CPU2017 subsetting

Considering that there might be deviations in data collection using perf, we use the arithmetic average of the collected data as the formal experimental data. We conduct benchmark subset selection experiments for SPECrate INT, SPECrate FP, SPECspeed INT, and SPECspeed FP of SPEC CPU2017 on the Taishan 200 server. For SPECrate FP, when consensus clustering is CC_{HC} with $H = 6000$, and the $R = 80\%$, consensus matrix's distributions of $k = 2$ and $k = 3$ are as shown in Figure 4A,B: It can be seen that compared to $k = 3$, the consensus distribution of $k = 2$ is skewed towards 0 and 1, so the optimal size of SPECrate FP's subset is 2. But for the case where there is no skew to the consensus distribution, as shown in Figure 4C,D is the consensus distribution of SPECrate INT (the conditions are the same as SPECrate FP), it needs to be judged according to the CDF area change of the consensus matrix distribution, the optimal benchmark subset size determined by the elbow rule is 3 as shown in Figure 5.

The analysis of other subsuites of SPEC CPU2017 is the same and the final result is shown in Table 5 (omitting the suffix of the benchmark). From the perspective of the optimal size of the benchmark subset, the decision results of CC_{HC} and CC_{kmeans} are the same first while $CC_{spectral}$ is 3 for all the subsuites. Judging from the decision results of subsetting, there are differences between the three algorithms, but there are also overlapping parts. For example, 602.gcc_s and 519.lbm_r are selected in all three algorithms.

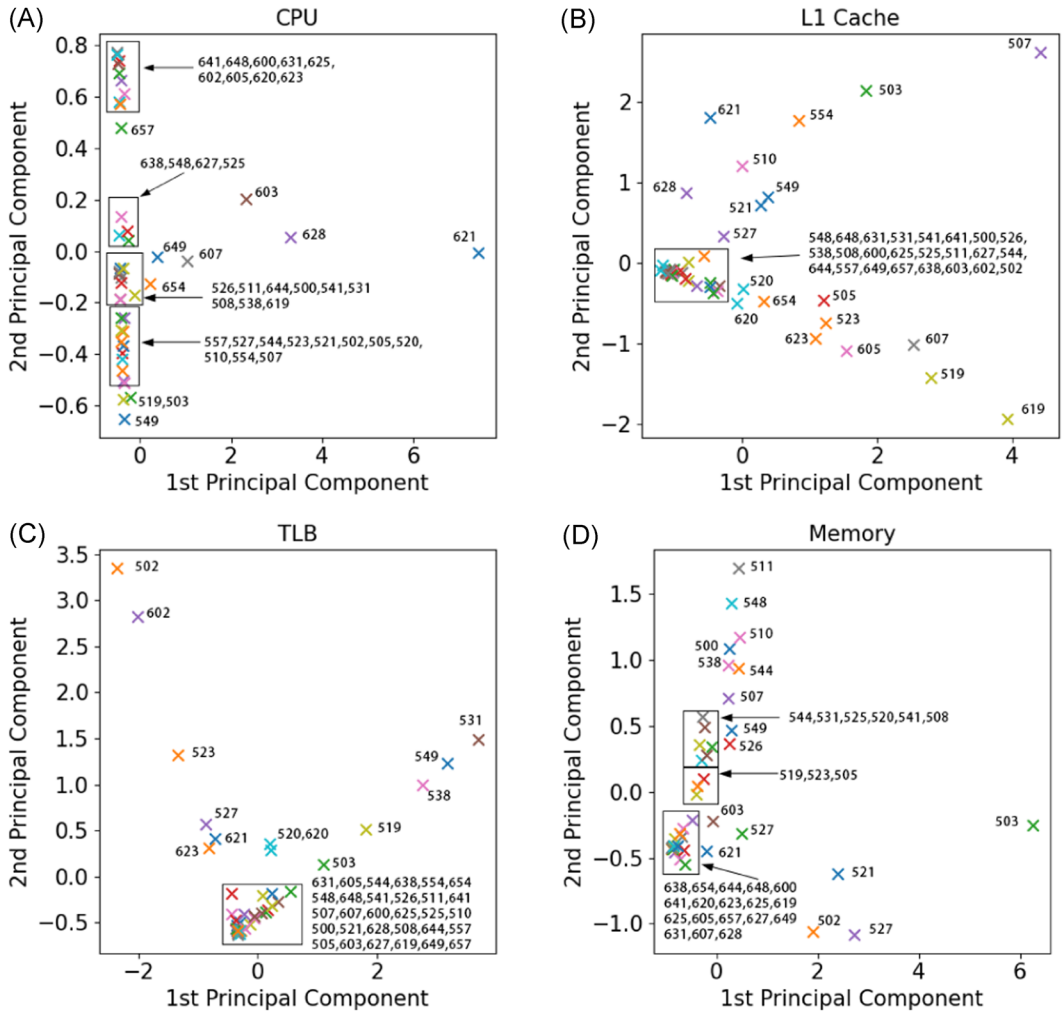

FIGURE 3 Scatter plot of CPU2017's PC value [Color figure can be viewed at wileyonlinelibrary.com]

4.4 | Comparison with ratio-based evaluation

To verify the effectiveness of the evaluation method proposed in Section 3.2.3, we compare it with the ratio-based method used in paper 15 which is determined by the usefulness of a subset by comparing the SPEC scores of the benchmark suite and subset. The results are shown in Table 6, where $SCORE_{full}$ is the SPEC score of the input benchmark suite, $SCORE_{sub}$ is the SPEC score of the benchmark subset.

Taking the subset selection of SPECrate INT under the CC_{HC} as an example, from the evaluation method proposed in this paper, the selection results of CC_{kmeans} have the best overall performance while the selected subset of CC_{HC} performs best according to the ratio-based method among the three consensus algorithms. To verify the correctness of the two evaluation methods, we observe the performance of the benchmark subset relative to the benchmark suite from specific performance events, such as IPC, L1 cache miss rate, and so forth, as shown in Figure 6. The subset selection result of CC_{kmeans} is 520.omnetpp_r, 523.xalancbmk_r, 531.deepsjeng_r while the subset selection result of CC_{HC} is 505.mcf_r, 500.perlbench_r,

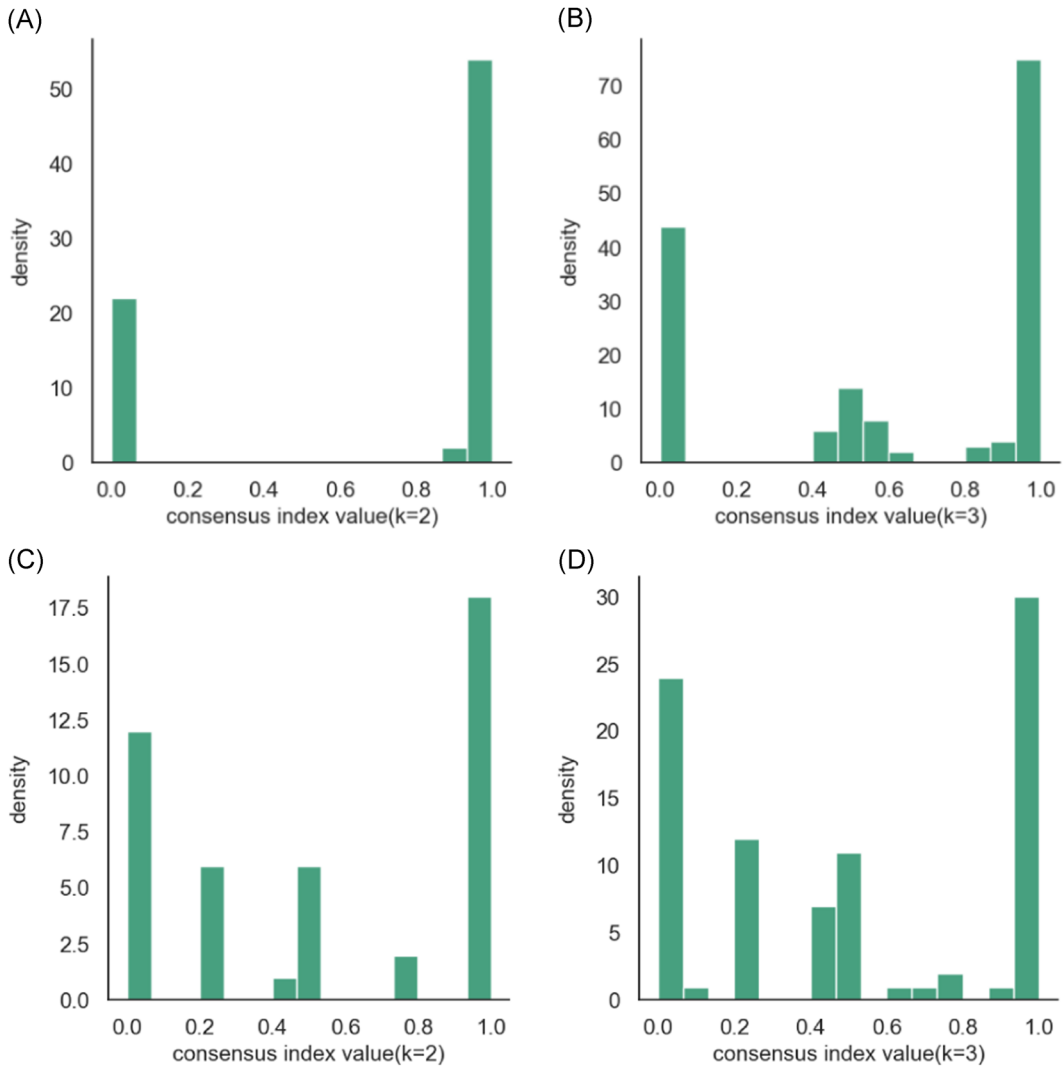

FIGURE 4 Consensus distribution of subsuites: SPECrate FP, (A) $k = 2$ and (B) $k = 3$; SPECrate INT, (C) $k = 2$ and (D) $k = 3$ [Color figure can be viewed at wileyonlinelibrary.com]

531.deepsjeng_r. Figure 6A is the IPC distribution of the benchmark in SPECrate INT. It can be observed that the selection results of CC_{kmeans} are relatively evenly distributed in the whole, but the values of 500.perlbench_r and 531.deepsjeng_r in the selection results of CC_{HC} are similar, so its selection diversity on IPC is not good. As shown in Figure 6B, THE results of the two algorithms have diversity in the branch prediction error rate. In terms of L1 cache miss rate, the selection diversity of CC_{HC} is not enough, and the values of 500.perlbench_r and 531.deepsjeng_r are similar (as shown in Figure 6C). As shown in Figure 6D, 520.omnetpp_r is very different from other benchmarks in the iTLB miss rate, but the benchmark subset selected by the CC_{HC} algorithm does not include it. Therefore, starting from the specific distribution of performance events, compared with the ratio-based evaluation method, the evaluation method proposed in this paper obviously takes the diversity into account. The analysis of the other three subsuites is similar.

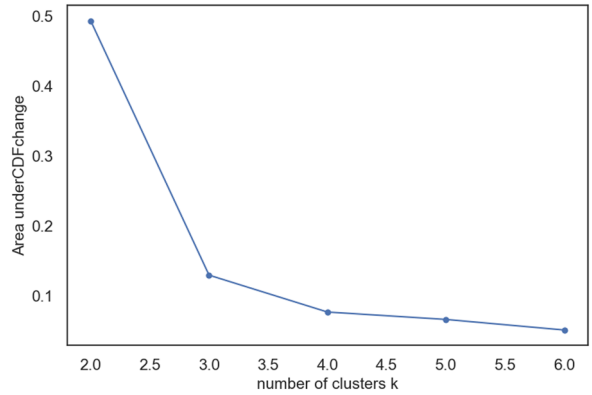

FIGURE 5 the proportion change in the area under Cumulative Distribution Function (CDF) [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 5 subsetting results of consensus clusterings

	CC_{HC}		CC_{Kmeans}		$CC_{Spectral}$	
	k_{best}	**Subset**	k_{best}	**Subset**	k_{best}	**Subset**
SPECrate INT	3	505, 500, 531	3	520, 523, 531	3	505, 541, 502
SPECrate FP	2	510, 519	2	527, 519	3	521, 519, 511
SPECspeed INT	2	602, 641	2	623, 602	3	605, 631, 602
SPECspeed FP	3	621, 619, 644	3	649, 638, 621	3	654, 621, 638

TABLE 6 Comparison with ratio-based evaluation

		CC_{HC}		CC_{Kmeans}		$CC_{Spectral}$	
	$SCORE_{full}$	***RS***	$SCORE_{sub}$	***RS***	$SCORE_{sub}$	***RS***	$SCORE_{sub}$
SPECrate INT	81.33373	0.97028	82.09279	0.99118	66.01479	0.99002	69.53008
SPECrate FP	62.78908	0.99915	28.69329	0.99897	36.32031	0.9989	51.26534
SPECspeed INT	4.34710	0.92038	3.94950	0.81076	4.46222	0.94519	4.16834
SPECspeed FP	35.30240	0.89994	28.56895	0.84808	28.83916	0.84193	38.06775

4.5 | Comparison with PCA-H subsetting

To prove the effectiveness of BenchSubset, we conducted a comparative experiment with the PCA-H method based on the evaluation method proposed in Section 3.2.3. Panda et al.[15] used the PCA-H method to select the benchmark subsets of the four subsuites of SPEC CPU2017, based on the benchmark subset size of 3, for a cluster with more than two benchmarks, Panda et al. selected the benchmark with the smallest link distance, but they ignored the selection of benchmark with the same link distance. Figure 7A–D is the dendrograms of SPECrate INT, SPECrate FP, SPECspeed INT, and SPECspeed FP by PCA-H method, respectively, which can represent the similarity between benchmarks. According to the selection strategy, taking

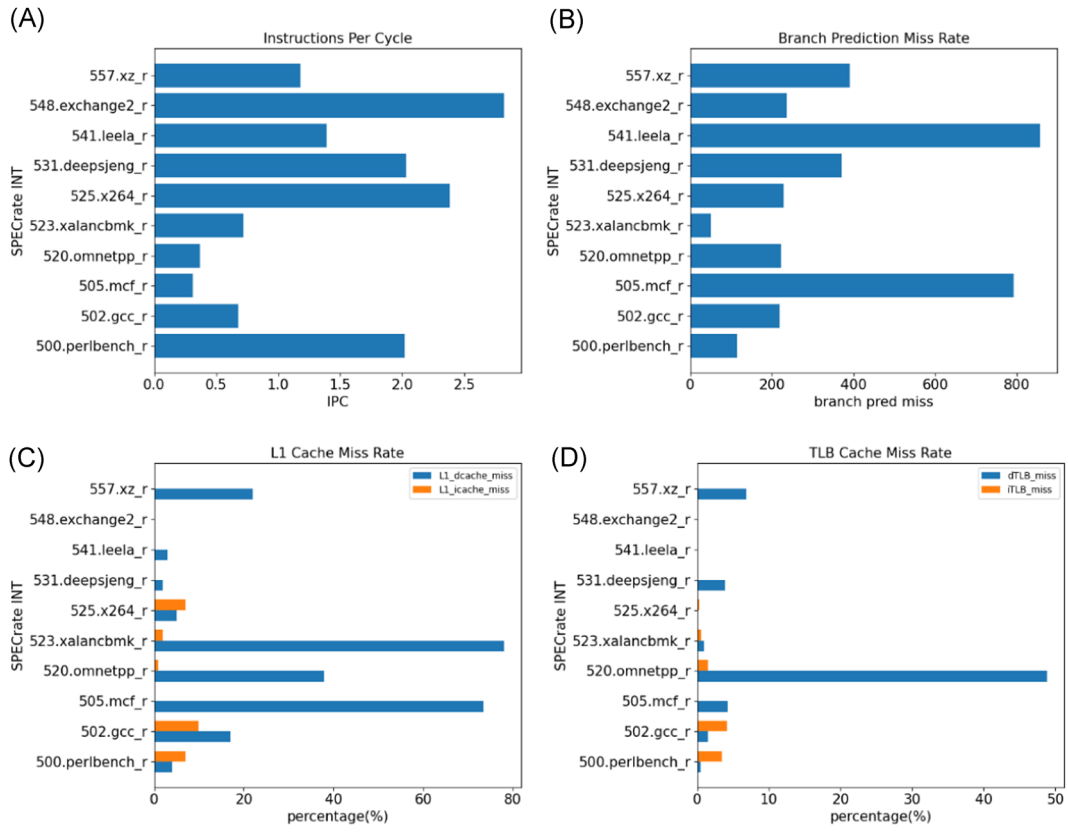

FIGURE 6 performance events distribution of SPECrate [Color figure can be viewed at wileyonlinelibrary.com]

SPECrate INT as an example, the selected subset is 531.deepsjeng_r, 505.mcf_r/557.xz_r, 500.perlben-ch_r/525.x264_s, the subset selection results of the four subsuites are shown in Table 7 (the suffixes of the benchmarks are omitted below).

Because the PCA-H method does not have the strategy of selecting the optimal benchmark from the two benchmarks with the same link distance, to prove the effectiveness of the optimal benchmark selection method in this paper, for the selection of the benchmark of PCA-H in a cluster should be different from the subset selected by BenchSubset as much as possible. The comparison results of CC_{HC}, CC_{Kmeans}, $CC_{Spectral}$ with PCA-H method are shown in Tables 8–10. From the perspective of the benchmark subsuite, all the consensus clustering methods have a poorer subset selection effect on the SPECspeed FP comparing to other subsuites. The subsetting result of CC_{HC} is always better than the PCA-H method, the selection effect of CC_{Kmeans} in speed suite is poor, and the selection effect of $CC_{Spectral}$ in a floating-point suite is relatively poor.

Combining the selection results of CC_{HC}, CC_{Kmeans}, $CC_{Spectral}$, the benchmark subset selection results of BenchSubset are shown in Table 11. The optimal benchmark subset of SPECrate INT is 520.omnetpp_r, 523.xalancbmk_r, 531.deepsjeng_r, the optimal benchmark subset of SPECrate FP is 510.parest_r, 519.lbm_r, the optimal benchmark subset of SPECspeed INT is 605.mcf_s, 631.deepsjing_s, 602.gcc_s, the optimal benchmark subset of SPECspeed FP is 621.wrf_s, 619.lbm_s, 644.nab_s. Judging from the evaluation results, the selection result of

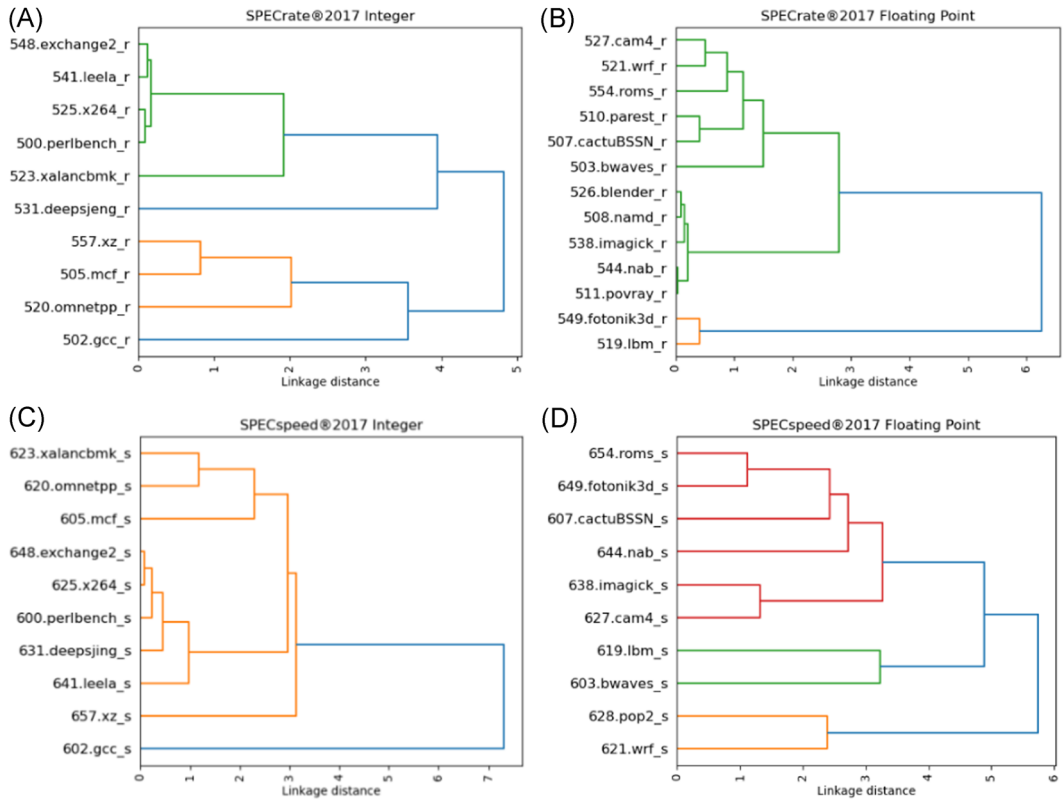

FIGURE 7 Dendrogram showing similarity [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 7 Subsetting result of PCA-H method

SPECrate INT	531, 505/557, 500/525
SPECrate FP	519/549, 511/544, 507/510
SPECspeed INT	602, 657, 648/625
SPECspeed FP	621/628, 603/619, 649/654

TABLE 8 Subsetting result comparison of PCA-H method and CC_{HC}

	PCA-H		CC_{HC}	
SPECrate INT	525, 557, 531	0.96881	505, 500, 531	0.97028
SPECrate FP	549, 511, 507	0.99895	510, 519	0.99915
SPECspeed INT	602, 657, 648	0.90993	602, 641	092038
SPECspeed FP	628, 603, 649	0.89512	621, 619, 644	0.89994

TABLE 9 Subsetting result comparison of PCA-H method and CC_{Kmeans}

	PCA-H		CC_{Kmeans}	
SPECrate INT	531, 505, 500	0.9702	520, 523, 531	0.99118
SPECrate FP	549, 511, 507	0.99895	527, 519	0.99897
SPECspeed INT	602, 657, 648	0.90993	623, 602	0.81076
SPECspeed FP	628, 603, 654	0.89517	649, 638, 621	0.84808

TABLE 10 Subsetting result comparison of PCA-H method and $CC_{Spectral}$

	PCA-H		$CC_{Spectral}$	
SPECrate INT	531, 557, 500	0.97405	505, 541, 502	0.99002
SPECrate FP	549, 544, 507	0.99913	521, 519, 511	0.9989
SPECspeed INT	602, 657, 648	0.90993	605, 631, 602	0.94519
SPECspeed FP	628, 603, 649	0.89512	654, 621, 638	0.84193

TABLE 11 Subsetting result comparison of PCA-H method and BenchSubset

	PCA-H		BenchSubset	
SPECrate INT	531, 505, 500	0.9702	520, 523, 531	0.99118
SPECrate FP	549, 511, 507	0.99895	510, 519	0.99915
SPECspeed INT	602, 657, 648	0.90993	605, 631, 602	0.94519
SPECspeed FP	628, 603, 649	0.89512	621, 619, 644	0.89994

BenchSubset is always better than the PCA-H method, the specific comparison is shown in Figure 8. In summary, the benchmark subset selected by BenchSubset proposed in this paper is more representative, which reflects the universal and the diversity characteristics of the benchmark suite at the same time as much as possible.

5 | CONCLUSION AND FUTURE WORK

Modern benchmarks have been evolved to cover many application scenarios. The redundancy in the benchmark suite leads to time-consuming when evaluating system architectures or simulating. In recent years, researchers usually select subsets based on clustering to solve this problem. However, it is a challenge to validate benchmark clustering results for unlabeled benchmark suites. And existing evaluation methods for subsetting results have not considered both the universal and the diversity characteristics of the benchmark suite. In this paper, we proposed a framework based on consensus clustering called BenchSubset which takes the above problems into account. In BenchSubset, all the feature data representing the characteristics of the given benchmark suite is processed based on the GPCA first, then consensus clustering and a new evaluation method are used to achieve the optimal size of the benchmark

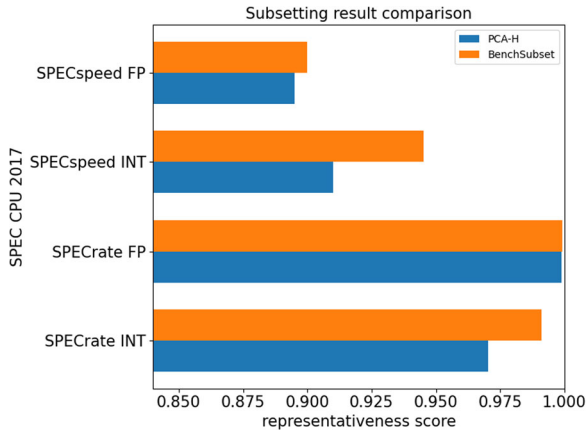

FIGURE 8 Subsetting result comparison of PCA-H method and BenchSubset [Color figure can be viewed at wileyonlinelibrary.com]

subset and the optimal subset decision. In the experimental part, we first verified the effectiveness of the GPCA and evaluation method. Second, as a case study, we selected benchmark subsets based on Huawei's Taishan 200 for SPEC CPU2017. Compared with the mainstream PCA-H method, BenchSubset's selected benchmark subset is more representative.

Although BenchSubset can effectively make decisions about the optimal size of the benchmark subset and the optimal benchmark in each cluster, there is still room for improvement in the framework. In future work, we will comprehensively consider accuracy and reusability[34] when collecting feature data, and we will consider extending Chauffeur provided by the SPEC organization to add the function of benchmark subset selection.

ACKNOWLEDGMENTS

This study is supported by Key-Area Research and Development Program of Guangdong Province (2021B0101420002), National Natural Science Foundation of China (62072187, 62002078, and 61872084), Guangzhou Science and Technology Program key projects (202007040002), Guangdong Major Project of Basic and Applied Basic Research (2019B030302002), and Guangzhou Development Zone Science and Technology (2020GH10).

CONFLICT OF INTERESTS

The authors declare that there are no conflict of interests.

ORCID

Weiwei Lin http://orcid.org/0000-0001-6876-1795

REFERENCES

1. Kounev S, Lange KD, von Kistowski J. *Systems Benchmarking*. Springer International Publishing; 2020.
2. Zhanpeng J, Cheng AC. SubsetTrio: an evolutionary, geometric, and statistical benchmark subsetting framework. *ACM Trans Model Comput Simul*. 2011;21(3):1-23. Article 21.
3. Christopoulos VN, Lilja DJ, Schrater PR, et al. Independent component analysis and evolutionary algorithms for building representative benchmark subsets. In: *ISPASS 2008—IEEE International Symposium on Performance Analysis of Systems and software*. IEEE; 2008:169-178.

4. Alameldeen AR, Martin MMK, Mauer CJ, et al. Simulating a $2 M commercial server on a $2 K PC. *Computer*. 2003;36(2):50-57.
5. Yi JJ, Lilja DJ, Hawkins DM. A statistically rigorous approach for improving simulation methodology. In: *Proceedings of the Ninth International Symposium on High-Performance Computer Architecture, 2003 (HPCA-9 2003)*. IEEE; 2003:281-291.
6. Taylor G, Davies P, Farmwald M. The TLB slice—a low-cost high-speed address translation mechanism. In: *Proceedings of the 17th Annual International Symposium on Computer Architecture*. ACM; 1990:355-363.
7. Eeckhout L, Vandierendonck H, De Bosschere K. Workload design: selecting representative program-input pairs. In: *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*. IEEE; 2002:83-94.
8. Phansalkar A, Joshi A, John Lk. Analysis of redundancy and application balance in the SPEC CPU2006 Benchmark Suite. In: *Proceedings Of The 34th Annual International Symposium on Computer Architecture*. Computer Architecture News; 2007:412-423.
9. Panda R, John LK. Data analytics workloads: characterization and similarity analysis. In: *IEEE 33rd International Performance Computing and Communications Conference* (*IPCCC*). IEEE; 2014:1-9.
10. Trancoso P, Adamou C, Vandierendonck H. Reducing TPC-H benchmarking time. In: *Panhellenic Conference on Informatics*. Springer; 2005:641-650.
11. Vandierendonck H, Trancoso P. Building and validating a reduced TPC-H benchmark. In: *14th IEEE International Symposium on Modeling, Analysis, and Simulation*. IEEE; 2006:383-392.
12. Adhinarayanan V, Feng W. An automated framework for characterizing and subsetting GPGPU workloads. In: *IEEE International Symposium on Performance Analysis of Systems and Software* (*ISPASS*). IEEE; 2016: 307-317.
13. Liu Q, Wu X, Kittinger L, et al. *BenchPrime: accurate Benchmark Subsetting with Optimized Clustering Algorithm Selection*. Department of Computer Science, Virginia Polytechnic Institute & State University; 2018.
14. George VM. 3D workload subsetting for GPU architecture pathfinding. In: *IEEE International Symposium on Workload Characterization*. IEEE; 2015:130-139.
15. Panda R, Song S, Dean J, et al. Wait of a decade: did SPEC CPU 2017 broaden the performance horizon? In: *IEEE International Symposium on High Performance Computer Architecture* (*HPCA*). IEEE; 2018: 271-282.
16. Limaye A, Adegbija T. A workload characterization of the SPEC CPU2017 benchmark suite. In: *IEEE International Symposium on Performance Analysis of Systems and Software* (*ISPASS*). IEEE; 2018:149-158.
17. Ajay Joshi J, Aashish Phansalkar P, Eeckhout L, John LK. Measuring benchmark similarity using inherent program characteristics. *IEEE Trans Comput*. 2006;55(6):769-782.
18. Jia Z, Zhan J, Wang L, et al. Characterizing and subsetting big data workloads. In: *IEEE International Symposium on Workload Characterization* (*IISWC*). IEEE; 2014:191-201.
19. Jin Z, Cheng AC. Evolutionary benchmark subsetting. In: *IEEE Micro*. Vol 28, No. 6, pp. 20-36.
20. Lange KD, Tricker MG. The design and development of the server efficiency rating tool (SERT). In: *Proceedings of the 2nd ACM/SPEC International Conference on Performance engineering*. DBLP; 2011:145-150.
21. KleinOsowski AJ, Lilja DJ. MinneSPEC: a new SPEC benchmark workload for simulation-based computer architecture research. *IEEE Comput Archit Lett*. 2002;1(1):7.
22. Singh S, Awasthi M. Efficacy of statistical sampling on contemporary workloads: the case of SPEC CPU2017. In: *IEEE International Symposium on Workload Characterization* (*IISWC*). IEEE; 2019:70-80.
23. https://www.spec.org/cpu2017/Docs/overview.html. Accessed October 6, 2021.
24. Bucek J, Lange KD, Kistowski Jv. SPEC CPU2017: next-generation compute benchmark. In: *Companion of the ACM/SPEC International Conference on Performance Engineering*. ACM; 2018:41-42.
25. https://www.spec.org/cpu2017/results/cpu2017.html. Accessed October 6, 2021.
26. Monti S. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Mach Learn*. 2003;52(1):91-118.
27. Kramer O. Scikit-learn. In: *Machine Learning for Evolution Strategies*. Springer; 2016:45-53.
28. Wang YC, Chen JK, Li BR, et al. An empirical study of HPC workloads on Huawei Kunpeng 916 processor. In: *IEEE 25th International Conference on Parallel and Distributed Systems* (*ICPADS*). IEEE; 2019:360-367.

29. Saastad OW, Kapanova K, Markov S, et al. PRACE Best Practice Guide 2020: Modern Processors. PRACE aisbl; 2020.
30. Vitillo R. Performance tools developments. *Future Comput Part Phys*. 2011.
31. https://developer.arm.com/documentation/#q=armv8-a&sort=relevancy. Accessed October 6, 2021.
32. https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/tools/perf/pmu-events/arch/arm64/hisilicon/hip08. Accessed October 6, 2021.
33. https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/perf/hisi-pmu.rst. Accessed October 6, 2021.
34. Hassan M, Park CH, Black-Schaffer D. A reusable characterization of the memory system behavior of SPEC2017 and SPEC2006. *ACM Trans Archit Code Optim (TACO)*. 2021;18(2):1-20.

How to cite this article: Zhan H, Lin W, Mao F, et al. BenchSubset: A framework for selecting benchmark subsets based on consensus clustering. *Int J Intell Syst*. 2021;1-24. doi:10.1002/int.22791