Pattern Recognition 161 (2025) 111321

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

RSPCA: Random Sample Partition and Clustering Approximation for ensemble learning of big data

Mohammad Sultan Mahmud [a], Hua Zheng [b], Diego Garcia-Gil [c], Salvador García [d], Joshua Zhexue Huang [a,*]

[a] Big Data Institute, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
[b] College of Software and Artificial Intelligence, Software Engineering Institute of Guangzhou, Guangzhou 510990, China
[c] Department of Software Engineering, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, 18071, Granada, Spain
[d] Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, 18071, Granada, Spain

ARTICLE INFO

Keywords:
Clustering approximation
Ensemble clustering
Incremental clustering
Ensemble learning

ABSTRACT

Large-scale data clustering needs an approximate approach for improving computation efficiency and data scalability. In this paper, we propose a novel method for ensemble clustering of large-scale datasets that uses the Random Sample Partition and Clustering Approximation (RSPCA) to tackle the problems of big data computing in cluster analysis. In the RSPCA computing framework, a big dataset is first partitioned into a set of disjoint random samples, called RSP data blocks that remain distributions consistent with that of the original big dataset. In ensemble clustering, a few RSP data blocks are randomly selected, and a clustering operation is performed independently on each data block to generate the clustering result of the data block. All clustering results of selected data blocks are aggregated to the ensemble result as an approximate result of the entire big dataset. To improve the robustness of the ensemble result, the ensemble clustering process can be conducted incrementally using multiple batches of selected RSP data blocks. To improve computation efficiency, we use the I-niceDP algorithm to automatically find the number of clusters in RSP data blocks and the k-means algorithm to determine more accurate cluster centroids in RSP data blocks as inputs to the ensemble process. Spectral and correlation clustering methods are used as the consensus functions to handle irregular clusters. Comprehensive experiment results on both real and synthetic datasets demonstrate that the ensemble of clustering results on a few RSP data blocks is sufficient for a good global discovery of the entire big dataset, and the new approach is computationally efficient and scalable to big data.

1. Introduction

1.1. Motivation

Ensemble clustering is aimed at an ensemble result that is more robust and better than the single result by a clustering algorithm. However, when a big dataset is encountered, ensemble clustering faces computational challenges because it is impractical, if not impossible, to cluster a big dataset multiple times to generate component results before aggregating them into the ensemble result. Scalability to big data is another computation problem in the current ensemble clustering methods. In this paper, we propose a novel approach to solving the computation problems of ensemble clustering of big datasets.

Clustering seeks to group or cluster data in a certain number of clusters, which is the most important parameter in many clustering algorithms like k-means. In unlabeled data, this parameter is unknown; therefore, users often assume it, but an incorrect guess might lead to inaccurate clustering findings. In practice, many methods, such as elbow, gap statistic, and silhouette, measure the quality of several clustering results with different numbers of clusters to identify the inherent number of clusters in data. They are computationally expensive to use on a big dataset because multiple clustering results must be generated. Therefore, finding the number of clusters in data presents another problem in ensemble clustering of a big dataset using some existing algorithms [1,2].

* Corresponding author.
E-mail addresses: sultan@szu.edu.cn (M.S. Mahmud), zhengh@mail.seig.edu.cn (H. Zheng), djgarcia@ugr.es (D. Garcia-Gil), salvagl@decsai.ugr.es (S. García), zx.huang@szu.edu.cn (J.Z. Huang).

https://doi.org/10.1016/j.patcog.2024.111321
Received 14 June 2024; Received in revised form 11 October 2024; Accepted 24 December 2024
Available online 31 December 2024

MapReduce programming model for distributed computing has attracted attention in the parallel and distributed computing communities. Using MapReduce programming model, clustering algorithms can be rewritten in MapReduce and executed in a parallel and distributed fashion to cluster a big dataset, but repeatedly clustering the same big dataset to produce multiple component clustering results is still computationally expensive. Using random samples of a big dataset can improve the computation efficiency in generating component results, but it requires more complicated aggregation strategies and mechanisms to ensemble the component results. On the other hand, developing distributed clustering algorithms in MapReduce is challenging due to the difficulties of programming and performance optimization [3,4].

To sum up, the challenges of ensemble clustering of big data can be stated as follows:

- Approximate and scalable clustering methods: Traditional clustering ensemble methods integrate the outputs from multiple models or algorithms on the same dataset to obtain a robust result, but they are only appropriate for small or moderate-sized datasets. These methods cannot be used on big datasets because we cannot cluster the same big dataset multiple times in order to get the component results. Since computing on the entire big dataset is impractical, approximate clustering often becomes essential [1, 5]. Therefore, it is necessary to develop approximate ensemble clustering methods that are scalable to big data.
- High-performance sampling: Random sampling is widely used as a cost-effective method in big data analysis. The dilemma is that a small sample is not able to generate an accurate estimation, while a large sample itself results in computing challenges. In this paradoxical case, we can solve this problem by drawing multiple random samples from the big dataset and aggregating the outcomes of those samples into the final ensemble result successfully and rapidly [6].
- Consensus function: Combining the results of different random samples is essential for ensemble clustering of big datasets. The traditional consensus functions are no longer applicable to ensemble the component results from disjoint samples because there is no common object identification reference. Combining different clusterings of disjoint data subsets is a new problem of consensus functions. Indeed, different clusterings of data subsets may produce incompatible data labelings, resulting in intractable correspondence issues, particularly if the numbers of clusters are different. Then, how can the label correspondence issue be resolved?

Therefore, data partitioning, sampling, and approximate computing are all important issues in big data clustering, although these can be independent of each other [5]. We contend that these fundamental issues are more crucial than ever in the era of big data and sophisticated clustering. As a result, big data clustering becomes a challenging task that calls for research on new approaches and solutions, including ensemble clustering, scalable and approximate computing, and distributed and parallel processing.

In response to this call, we propose a new approach of ensemble clustering for big data in this paper with a goal that can be stated as follows: *By representing a big dataset as a set of ready-to-use random samples, sequential clustering algorithms can be used to cluster multiple samples independently (or in parallel) to generate component results, and reliable approximate and accurate clustering results can be obtained from an ensemble of multiple sample outcomes without computing the entire dataset or developing complex parallel clustering algorithms. This clustering approximation can be incrementally improved by appending new outputs from newly analyzed random samples.*

1.2. Contributions

The computational bottleneck of MapReduce distributed model is data communication among nodes that slows down the execution of iterative algorithms. In this paper, we adopt a different computing framework to execute serial clustering algorithms independently on disjoint random data subsets. This is *a non-MapReduce paradigm that benefits big data clustering with the ability to run serial clustering algorithms directly and independently on multiple random samples in distributed computing and ensemble the outcomes of random samples efficiently, so as to extend data scalability*. Our main contributions of this work are as follows:

- A novel approximate clustering ensemble method, called RSPCA standing for Random Sample Partition and Clustering Approximation, is proposed for ensemble clustering of big datasets. The proposed method does not assume the number of clusters in advance; instead, it is discovered automatically from the underlying multiple random samples, i.e., RSP data blocks.
- In RSPCA, to deal with irregular clusters, we adopt two schemes, spectral clustering and correlation clustering, to ensemble the centroids of clusters in the RSP data blocks. Because different numbers of clusters may be identified from different RSP data blocks, RSPCA resolves the incompatible numbers of clusters in the clustering ensemble.
- Since RSPCA allows clustering RSP data blocks independently using a serial clustering algorithm, we propose to use batch component results to generate the ensemble result incrementally so as to improve the robustness of the ensemble result and computation efficiency as well as data scalability of the RSPCA method.
- Extensive experiments were conducted on both synthetic and real-world datasets to demonstrate the effectiveness and stability of the RSPCA approach and compare them with several state-of-the-art methods. These experiment results show that the RSPCA can approximate the true number of clusters and is computationally efficient for large-scale datasets.

The remaining paper is organized as follows: Section 2 briefly reviews the distributed and large-scale data clustering. Section 3 first introduces the major notations and then presents some preliminaries used in this research. Section 4 describes the proposed RSPCA framework. The experimental results are discussed in Section 5. Finally, Section 6 concludes this work and states some future perspectives.

2. Related work

The fundamental task of ensemble clustering, which has been studied extensively in the past decades [7,8], is to achieve enhanced and robust clustering performance. Typically, clustering ensembles are produced from component clustering results of the same dataset, using either a single clustering algorithm with varied input parameters or different algorithms. They are only appropriate for small or moderate-sized datasets. Due to the computational overhead of producing multiple component clustering results from a big dataset, traditional clustering ensemble algorithms are not scalable to big data [2,9]. In fact, a key operation to big data clustering ensembles is the aggregation of different clusterings of component datasets. The component clusterings from disjoint random samples may result in incompatible data labels, leading to complex correspondence problems, especially when the numbers of clusters in different samples are not same. Therefore, more intricate ensemble strategies and mechanisms are required [10].

As the size of datasets increases, traditional centralized clustering algorithms struggle to process them efficiently. To tackle this issue, distributed versions of traditional clustering algorithms have been introduced to support big data solutions. For example, the distributed hierarchical clustering algorithm, PACk, creates the local cluster models

and then combines them to form the global cluster model. The well-known centroid-based clustering method, k-means, is recognized for its high efficiency and has been adopted in various MapReduce-based distributed versions to facilitate large-scale clustering tasks [11,12]. Additionally, MLib of Spark offers a distributed k-means++ [13] implementation. However, k-means only need to input the number of clusters, and the algorithm is highly interpretable. Furthermore, a distributed dual averaging-based data clustering [14] approach has been proposed to handle large-scale clustering tasks. However, it is worth noting that MapReduce computing framework is not efficient in executing iterative algorithms so it is not suitable for big data as it requires generating multiple clusterings to assess the quality of the results with different numbers of clusters.

To analyze big data using density-peak clustering (DPC), significant research has focused on developing distributed DPC algorithms. Various distributed DPC implementations have been developed, including FDDP [15] and Fast LDP-MST [16]. However, current distributed DPC algorithms still have the problem of $\mathcal{O}(N^2)$ computational complexity, as they naively compute pair-wise distances to determine the distance for each point and local density. It is important to note that the distance measure is more costly than other tasks, such as data partitioning. Several variant DPC algorithms, including DCDP-ASC [17] and ADPC-KNN [18], have been proposed due to the effectiveness of the DPC algorithm. The high computational complexity limits their use in large-scale scenarios.

Automatic clustering algorithms are attracting more attention from the academic community, e.g., density-based and data-depth clustering. Density-based algorithms, such as MR-DBSCAN [19], can cluster datasets with convex shapes and skewed data, but it is difficult to determine the density threshold. The depth difference [20] method estimates the depth within and between clusters and uses the depth difference to finalize the optimal value of K. However, it is difficult to identify clustering centroids correctly in datasets with complex decision graphs. A bootstrap method [21] was proposed to estimate the number of clusters in big datasets, which minimizes the corresponding estimated clustering instability. On the other hand, the kluster [22] procedure takes randomly selected clusters as the initial seeds to determine the final number of clusters in a dataset in an iterative manner that confirms the most frequent mean of the resulting clusters from the iterations to be the optimal number of clusters.

Spectral clustering [23] is a flexible clustering procedure that can produce high-quality clusterings on small datasets but has limited applicability to large-scale problems because of its high computational complexity of $\mathcal{O}(N^3)$ with N being the number of data points. Further evidence for spectral clustering comes from substantial theoretical literature [24]. Correlation clustering [25] is able to identify the number of clusters automatically. Furthermore, its "model selection" property can be theoretically justified with a probabilistic interpretation, and theoretical analysis has been conducted for correlation clustering with derived error bounds. Since it was found that the correlation clustering problem is NP-complete [26], most of the work has been focused on finding an approximation solution [27].

3. Preliminaries

This section first presents the major notations and then paves the foundation of the proposed RSPCA method by introducing clustering approximations, random sample partitioning, and the density-peak-based I-niceDP algorithm.

3.1. Notations

Table 1 summarizes the main notations used in this paper.

The aim of RSPCA is to generate clustering ensembles from large-scale datasets. In this process, a critical step is to determine the optimal number of clusters for big data clustering with a particular emphasis on computational feasibility and efficiency, which will be the primary focus of the forthcoming sections.

Table 1
Frequently used notations in this paper.

Notation	Description
$\mathbb{D}$	A big dataset;
$N, \mathcal{F}$	Number of objects and dimensions in $\mathbb{D}$, respectively;
RSP	Random sample partitioning;
n	RSP data block size;
m	Total number of RSP data blocks;
es	Ensemble data size;
$\mathscr{P}$	RSP operation on $\mathbb{D}$;
$\mathscr{S}$	Sampling operation;
$\mathscr{L}$	Local clustering model;
$\mathscr{E}$	Ensemble learning;
$\mathscr{A}$	An approximate clustering result;
b	Randomly selected a subset of data blocks, ($b < m$);
D_i	ith RSP data block;
C_i, c_i	ith cluster and centroid of C_i, respectively;
K	Number of clusters;
O_i	An observation point;
GMM	Gamma mixture model;
$\Pi^\star$	Discovered set of cluster centroids of b samples;
$\hat{n}$	Number of objects in $\Pi^\star$;
$\mathbf{S}$	A similarity matrix;
$\mathbf{A}$	An affinity matrix;
$\mathbf{D}$	A degree matrix;
$\mathbf{L}$	A Laplacian matrix;
$\mathbf{P}$	A probability;
SC	Spectral clustering;
CC	Correlation clustering;

3.2. Approximate clustering of large-scale data

Assume a dataset $\mathbb{D}$ is too large and cannot be held in memory. We cannot run a clustering function on $\mathbb{D}$ to estimate the number of clusters. If we could obtain the number of clusters of the entire dataset $\mathbb{D}$ by analyzing a subset to satisfy the limited available computing resources, we would be able to reduce computational costs and accomplish the task of big data clustering.

We know that the precondition for approximate analysis of big data with a subset is that the distribution of data in the subset is required to be similar to the distribution of the original full dataset. However, the data subsets generated by the common data partitioning methods, i.e., range and hash partitioning, do not necessarily satisfy the precondition since they do not consider the properties of the statistical distribution in the data. In fact, the records of a big dataset are rarely arranged randomly [5]. However, statistical and inferential thinking requires random samples for approximate clustering of large-scale datasets [6].

For approximately clustering of a big dataset, random sampling is a popular and cost-effective method. However, sampling techniques for big data clustering are not always feasible because of the sampling efficiency issue and the poor match between the sampling design. A promising solution is block-level sampling, which is considerably more efficient than record-level sampling. Another issue is that we can avoid sampling bias by using multiple random samples of a large dataset in a clustering ensemble. The RSPCA adapts an incremental approximate computing approach to estimate the number of clusters in a large dataset.

3.3. RSP data model

Suppose given a big dataset $\mathbb{D}$, we want to analyze $\mathbb{D}$, but it is too big to be analyzed due to the limitations of the computing resources. A random sample partitioning (RSP) operation on $\mathbb{D}$ is applied to produce random samples $\{D_1, D_2, \ldots, D_m\}$ that hold

$$\begin{cases} D_i \neq \emptyset, \\ D_i \cap D_j = \emptyset, \\ \bigcup_{i=1}^{m} D_i = \mathbb{D}, \\ F(D_i) \approx F(\mathbb{D}), \qquad 1 \leq i \leq m, \end{cases} \tag{1}$$

Algorithm 1: Random Sample Partitioning (RSP).

```
Input  : 𝔻: A big dataset.
         n: RSP sample size.
1 begin
2   N = getNumberOfObjects(𝔻);  /* estimate the number
      of objects in the dataset */
3   m = getNumberOfPartitions(N, n);  /* estimate the
      number of total RSP data blocks */
4   rand = generateRandomNumbers(N);
5   𝔻 ← rand.append(𝔻);
6   J = getOrder(𝔻);  /* ascending or descending order,
      sorted by rand */
7   for i = 1 to m do
8     D_i = getRS(J, m); /* sequentially cut from sorted
        data */
Output : {D_1, D_2, ..., D_m}, a set of random sample of 𝔻.
```

where $F(.)$ is the cumulative distribution function. The first three relations define a data partition of $\mathbb{D}$, whereas the last relation categorizes a random sample partition.

In the RSP data model, all RSP data blocks $\{D_1, D_2, \ldots, D_m\}$ satisfy the definition of a random sample[1] of $\mathbb{D}$ and are ready-to-use. To create multiple random samples, we can simply randomly select a few RSP data blocks from the RSP data model without repeatedly going through all the records in $\mathbb{D}$. Therefore, the sampling process for multiple random samples has been significantly improved. Since D_i preserves statistical properties as $\mathbb{D}$ does and is decidedly smaller than $\mathbb{D}$ in size, it can be processed and analyzed efficiently using existing sequential as well as parallel algorithms.

In this work, we adopt the RSP data model [28] to represent a big dataset as a set of ready-to-use random sample data blocks, so the block-level sampling method is used to select multiple random samples efficiently. The RSP data block generation is explained in Algorithm 1. The inputs of the algorithm are a big dataset $\mathbb{D}$ and the size of each RSP data block n. The output is a set of m RSP data blocks, where $m = N/n$. This process is carried out as follows:

Lines 2–3 compute the total number of objects N and the number of RSP blocks m, respectively. Line 4 generates a series of N unique random numbers with a uniform distribution. Line 5 appends the random numbers as one additional ID in $\mathbb{D}$. Line 6 sorts the random number ID in order to randomize the $\mathbb{D}$ records. Lines 7–8 cuts the sequence of randomized records of $\mathbb{D}$ into m sub-sequences, consecutively, each subsequence being an RSP data block. *We state that the sequence represents the sequence of the sorted random numbers, the records in each subsequence are unrelated and the order of data records in all subsequences are completely random.*

3.4. Identifying the number of clusters and initial centroids in RSP data blocks

We consider that natural clusters in data have a single modal distribution. A dataset with multiple clusters can be modeled with Gaussian or Gamma distributions. In these distributions, the significant feature is density peaks, each representing the modal of a cluster. We consider the number of density peaks to be the number of clusters, and peaks as cluster locations. Since the number of clusters in data is an input parameter, we need to know or guess in order to run the clustering algorithm. The density-peak-based I-niceDP algorithm [29] is suitable for identifying the number of clusters modeled as Gamma distribution. In RSPCA, we used I-niceDP as an operator to find the number of clusters and initial cluster centroids from RSP data blocks. The procedure of I-niceDP is outlined in Algorithm 2. The input to the algorithm is RSP data blocks, and the output is the number of clusters and centroids discovered in the RSP data blocks. Each RSP data block is calculated as follows:

Line 3 sets an observation point as a reference to calculate the distance distribution of objects. In line 4, the distance vector originates by calculating the distances between the observation point and the data points of an RSP sample. Line 5 employs the kernel density estimation (KDE) method to determine the maximum number of GMM components $\mathcal{M}_{\max}$ (i.e., the number of clusters), where the probable number of components is regulated by two thresholds, Δ_1 and Δ_2. In lines 6–9, a set of GMMs with a number of components less than or equal to the maximal number $\mathcal{M}_{\max}$ are computed from the distance vector, and each GMM model is constructed using the EM algorithm. Lines 10–12 determine which model is the best fit using the AICc criterion. Lines 13–15 use the density peaks (DP) mechanism to identify the high-density data points for each GMM component, and these high-density data points are used as the initial cluster centroids. Finally, lines 16–17 allocate the initial cluster centroids to the k-means scheme to cluster the input data and optimize the discovered cluster centroids as the outcome of the random sample.

Generally, I-niceDP combines the findings of multiple observation points to obtain a correct estimation of the distance distributions of objects, allowing more accurate detection of a large number of clusters. Like other density-peak-based clustering algorithms, I-niceDP considers the number of peaks to be the number of clusters, and peaks as cluster locations in this way. The advantage of multiple observation points is that if a cluster is missed by one observation point, it can be found by other observation points. However, calculating the Euclidean distances between the multiple observation points and data points, albeit at an increased computing cost. Ideally, in RSPCA, we deal with a subset of a big dataset, and each RSP data block produces a local clustering outcome using the I-niceDP scheme with a single observation point. Afterwards, a collaborative result is dynamically formed from multiple RSP data block outcomes. Collectively, multiple RSP data blocks can portray multiple views (a set of RSP data blocks can be considered as multi-observations) of a dataset simultaneously, and thus it can capture the latent knowledge within the data in a more comprehensive manner. Our experiments have demonstrated that single observation point-based I-niceDP is capable of determining the high-density areas in the original data space. We consider that each RSP data block is a single-observation process to produce local clustering insights, and the collaborative step is to share information about their memberships among different observation views of RSP data blocks.

4. Clustering approximation with ensemble of multiple samples

In this section, we first formulate the clustering problem for very large-scale datasets by making the memory and time constraints explicit. Then, we present our proposed clustering ensemble solution, RSPCA, to approximate big data clustering in detail.

4.1. Problem definition

Assume that $\mathbb{D}$ is a big dataset of N records and cannot be analyzed efficiently on a single machine directly. To enable clustering analysis, $\mathbb{D}$ is divided into m non-overlapping RSP data blocks of equal size (each with $n \ll N$ records), where the following relation holds:

$$\bigcup_{i=1}^{m} D_i = \mathbb{D} \quad \text{and} \quad D_i \cap D_j = \emptyset, \quad \text{for } \forall i \neq j \text{ and } i, j \in \{1, 2, \ldots, m\}. \tag{2}$$

Now, we assume that a randomly selected set of RSP data blocks fit into memory and can be analyzed independently using a sequential clustering algorithm (the computation can be conducted in a parallel

[1] Let D is a subset of big dataset $\mathbb{D}$, i.e., $D \subset \mathbb{D}$. D is a random sample of $\mathbb{D}$ if $F(D) \approx F(\mathbb{D})$, where $F(.)$ is the cumulative distribution function.

Algorithm 2: I-niceDP.

Input : A set of RSP data blocks, $\{D_1, D_2, ..., D_b\}$.

```
begin
    forall D^b_{i=1} do
        Generate a random observation point, O;
        Calculate the Euclidean distance vector X_O between the data points of D_i and O;
        Estimate the number of GMM components M_max using the KDE;
        for M = M_max − Δ1; M ≤ M_max + Δ2; M++ do
            Model X_O to GMM(O, M);
            Apply the EM algorithm to solve GMM(O, M);
            Compute AICc(M) of GMM(O, M);
        Choose the best-fitted model GMM(O) with the minimum AICc;
        Determine the GMM(O) as the final model GMM_f and K as the number of clusters;
        Obtain the IDs of data points for each GMM component of the GMM_f;
        Compute the local density of each data point in the k-th GMM component;
        Select the data points with the top 50% local density scores;
        Find the high-density points in the k-th GMM component via the DP mechanism;
        Apply the k-means with assigning K and the initial cluster centroids;
        Determine the final clusters number K and cluster centroids;
```

Output : b sets of cluster centroids $\{c^i_1, c^i_2, ..., c^i_{k_i}\}$, $i = 1, 2, ..., b$.

and distributed manner). In practice, we can randomly select a few RSP data blocks $\{D_1, D_2, \dots, D_b\}$, where $b < m$, as random samples and use them for approximate ensemble clustering as the estimated result of $\mathbb{D}$. In general, a big dataset is unknown and the knowledge of the number of clusters in the dataset is not available. However, we can estimate the number of clusters and centroids of a dataset. Assume that b data blocks provide b sets of cluster centroids, formulated as follows:

$$\pi(D_i) \leftarrow \Phi(D_i), \quad \pi(D_i) = (k_i, C_i), \tag{3}$$

where $\Phi(.)$ is a clustering function on D_i and returns $\pi(D_i)$ with two values. The first term k_i is the number of clusters in D_i, and the second term, $C_i = \{c_{i_1}, c_{i_2}, \dots, c_{i_k}\}$, is the set of centroids of the k clusters. The b sets of results are represented as

$$\Pi^\star = \{\pi(D_1), \pi(D_2), \dots, \pi(D_b)\}. \tag{4}$$

The set $\Pi^\star$ is the input to a consensus function to compute the ensemble result by aggregating the b sets of results. A key challenge is how to combine the b sets of results that are generated from disjoint data blocks into an ensemble. There is no common object IDs for reference, as used in the classical ensemble clustering process. To solve this problem, we take two scenarios into consideration, discussed below.

4.2. The proposed RSPCA approach

In this section, we present RSPCA, a novel approximate clustering ensemble method that makes use of the recent techniques of random sample partition and clustering approximation for big data. The computation framework of RSPCA is depicted in Fig. 1. We summarize the basic steps in the following:

1. First, the random sample partitioning (RSP) operation $\mathscr{P}$ is performed to generate a set of random samples (i.e., RSP data blocks) for a given big dataset $\mathbb{D}$.
2. Then, using block-level sampling operation $\mathscr{S}$, we randomly select a subset of RSP data blocks to find the number of clusters and centroids. Each RSP data block provides a local model $\mathscr{L}$ of clustering.
3. Use the outcomes of selected random samples and ensemble operation $\mathscr{E}$ to get a global approximation $\mathscr{A}$.
4. Finally, the k-means algorithm is used to cluster all RSP data blocks and save the output with cluster IDs.

Algorithm 3 is an illustration of the RSPCA's computation process and is executed as follows: The inputs are a big dataset ($\mathbb{D}$) and the RSP sample size (n). First, in line 2, the RSP(.) operator converts $\mathbb{D}$ into a set of RSP data blocks. Line 3 randomly selects a subset of RSP blocks. In lines 4–6, I-niceDP(.) and k-means(.) operators compute each selected data block independently and estimate the local number of clusters and centroids. Line 7 obtains the meta-data of the selected RSP data blocks. Line 8 uses ensemble operators, spectral clustering SC or correlation clustering CC, to integrate the centroids into the final set of the approximate number of clusters of $\mathbb{D}$. Lines 9–10 use the k-means scheme to cluster all the data blocks. Finally, the algorithm outputs the set of centroids and the clustering results.

We can improve the clustering results gradually by combining new outputs from other random samples until a stable result is obtained or all samples are used up. The size of RSP data blocks is decidedly smaller than the entire big dataset, so it can be efficiently processed on a traditional or virtual machine as well as in a parallel and distributed manner.

The key steps of the proposed RSPCA method are explained below.

4.2.1. Generating RSP data model and block-level sampling

Random sample partitioning (RSP) is the basis of this work. For big data analysis, we represent a big dataset as a set of disjoint random sample data blocks so that each is used as a random sample of the big dataset. Therefore, for the clustering approximation of a big dataset, we convert it to a set of ready-to-use RSP data blocks for efficient random sample selection as

$$\{D_1, D_2, \dots, D_m\} = \mathbf{RSP}(\mathbb{D}), \tag{5}$$

where **RSP**(.) is the conversion function to an RSP data model.

Then, using block-level sampling, a subset of RSP data blocks is randomly selected as follows

$$\{D_1, D_2, \dots, D_b\} = \mathbf{Sampling}(\mathbb{D}, b), \quad b < m, \tag{6}$$

where b is the user-specified number of RSP data blocks.

4.2.2. Finding the number of clusters and centroids in RSP data blocks

In this step, we estimate the number of clusters and centroids of selected b RSP data blocks. For each RSP data block, we determine the number of clusters and initial centroids independently using the density-peak-based I-niceDP algorithm. The high-density peaks for the

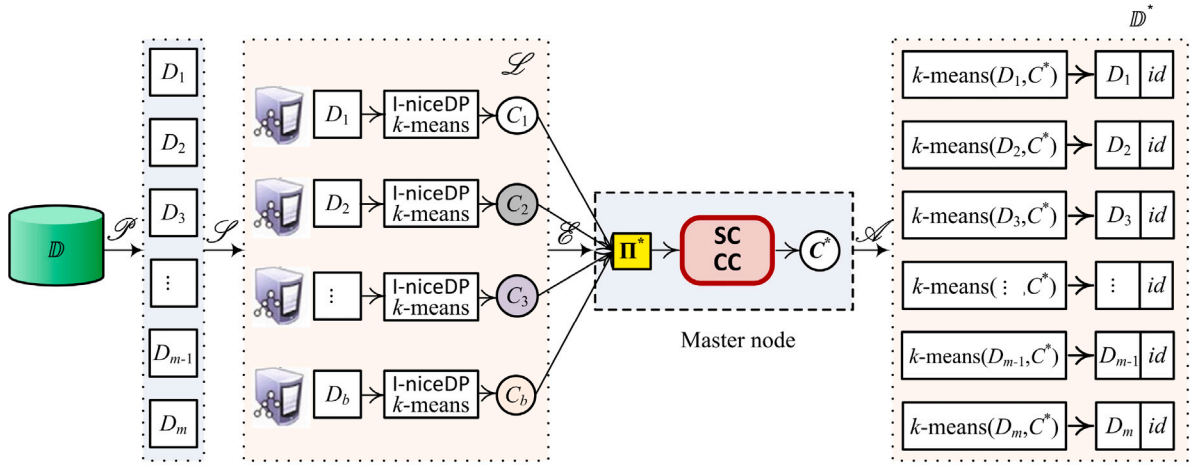


Fig. 1. The computation framework of RSPCA.

Algorithm 3: RSPCA.

Input : $\mathbb{D}$: A big dataset.
n: Sample size.

begin
  $\{D_1, D_2, ..., D_m\} \leftarrow$ **RSP**$(\mathbb{D}, n)$; /* generates the RSP data model of $\mathbb{D}$ */
  $\{D_1, D_2, ..., D_b\} \subset \{D_1, D_2, ..., D_m\}$; /* randomly select a subset of RSP data blocks */
  forall $D_i \in \{D_1, D_2, ..., D_b\}$ **do**
    $(k_i, C_i) =$ **I-niceDP**(D_i); /* finding the initial number of clusters and centroids */
    $C_i^* = k$-**means**(k_i, D_i, C_i); /* refining the initial centroids */
  $\Pi^\star = C_1 \cup C_2 \cup ... \cup C_b$; /* collect all the centroids of selected RSP data blocks */
  $C^\star =$ **SC**$(\Pi^\star)$ or **CC**$(\Pi^\star)$; /* integrate the centroids into the final estimate K */
  forall D_i **do**
    $\mathbb{D}^\star = k$-**means**(K_i, D_i); /* use k-means to cluster all RSP data blocks */

Output : K: The number of clusters and $\mathbb{D}^\star$: Clustering result.

candidate cluster centroids are identified via the I-niceDP operation as follows

$$(k_i, C_i') = \textbf{I-niceDP}(D_i), \tag{7}$$

where **I-niceDP**(.) is an operator on RSP data block D_i, and (k_i, C_i') are the two return values of the function. The first term is the number of clusters in D_i, and the second term, $C_i' = \{c'_{1i}, c'_{2i}, \dots, c'_{ki}\}$, is the set of centroids of the k clusters.

I-niceDP provides preliminary insights into clusters, and the k-means algorithm refines them. The output of I-niceDP is used as initial centroids in the k-means clustering scheme to determine the precise centroids for each RSP data block locally, as follows

$$C_i = k\text{-}\textbf{means}(k_i, C_i', D_i), \qquad 1 \le i \le b, \tag{8}$$

where $C_i = \{c_{1i}, c_{2i}, \dots, c_{ki}\}$ is the set of the refined centroids of k_i clusters in D_i.

Applying the two operators I-niceDP and k-means to selected b random samples $\{D_1, D_2, \dots, D_b\}$, we obtain b sets of local centroids and make a union of these sets to form a new set as

$$\Pi^* = C_1 \cup C_2 \cup \cdots \cup C_b. \tag{9}$$

The set Π^* contains totally $\cup_{i=1}^{b} k_i$ cluster centroids from b RSP data blocks. Since the RSP samples are taken from the big dataset, they should have similar inherent clusters. Therefore, the numbers of clusters in them should be very close to each other, and the centroids of the same clusters in different random samples should also be located closely. Considering these characteristics, in the next section, we suggest using the spectral and correlation clustering techniques to combine the centroids in Π^* into an ensemble set of centroids that will serve as estimated centroids of the big dataset.

4.2.3. Ensemble the clusters of RSP data blocks

A measure is needed to combine multiple clustering outcomes. More specifically, in this case, we aggregate the "centroids" of clusters in the selected data subsets. A cluster ball model was proposed to combine the centroids of data subsets in [30]. A limitation of the cluster ball ensemble is that it does not perform well when applied to a dataset with irregular-shaped clusters, such as moon-shaped and Swiss-roll data. We solve this problem with two different graph-based schemes to merge the centroids of subsets as follows:

Scheme 1: RSP ensemble using spectral clustering (RSPCA-SC) Spectral clustering (SC) is a graph-based method for locating k arbitrarily shaped clusters in data. Based on pairwise closeness or similarity, spectral clustering groups objects in clusters. The spectral clustering function is implemented as follows:

1. Using Euclidean distance to calculate a similarity matrix **S** from Π^*.
2. Transform the similarity matrix **S** to an affinity matrix **A**.
3. Calculate the degree matrix **D** and the Laplacian matrix **L** = **D** − **A**.
4. Calculate the eigenvalues and eigenvectors of the Laplacian matrix **L**. The number of zero eigenvalues infers the number of connected components in a similarity graph, which provides a reliable estimate of the number of clusters in the data.
5. Finally, cluster the graphs with the k-means scheme.

Scheme 2: RSP ensemble using correlation clustering (RSPCA-CC) Correlation clustering (CC) is also graph-based clustering. This clustering method represents a set of points in an arbitrary feature space as a weighted or unweighted complete undirected graph, where the vertices of the graph represent the items and edges are labeled either + or − indicating whether their end vertices are similar or

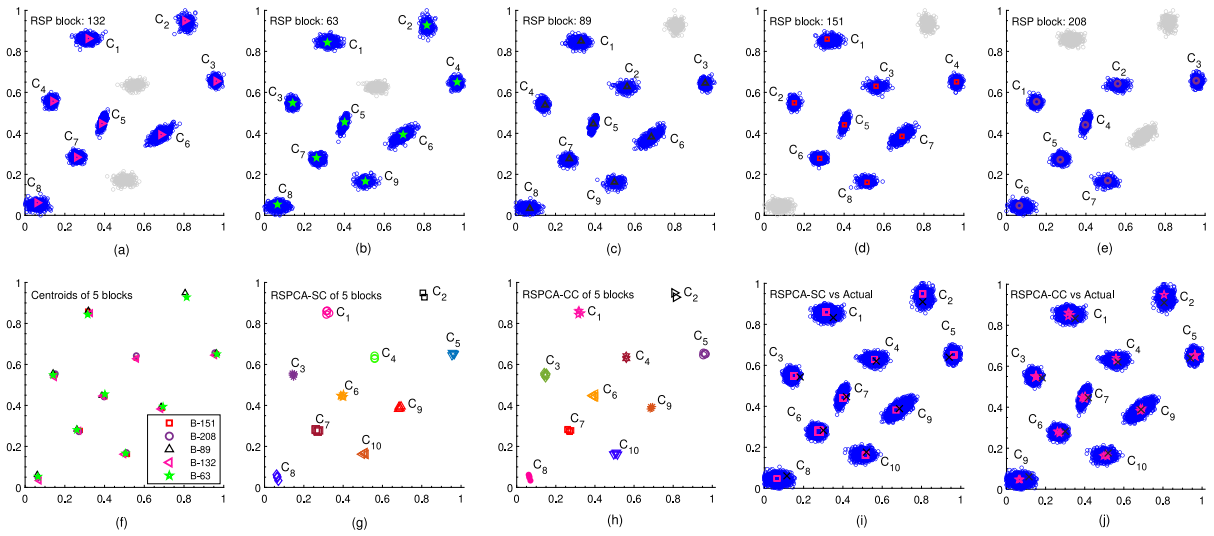


Fig. 2. An illustration of RSPCA approach using 2M2D10C dataset. (a-e) Randomly selected five RSP data blocks with 5000 points each and individually obtained cluster centroids using I-niceDP. (f) The discovered cluster centroids from the five RSP data blocks. (g-h) The final ensemble cluster centroids using SC and CC, respectively. (i-j) For the RSPCA-SC and RSPCA-CC ensembles, the matching clustering results, respectively, with the actual centroids indicated by the "×". Different colors and symbols represent individual cluster assignments of each observation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

different. The aim is to produce a partition of the vertices (clustering) that maximizes agreement with the edge label.

To start, we compute a similarity matrix $\mathbf{S}$ for Π^* using the negative squared Euclidean similarity function, which takes a numerical value s_{ij} for each pair of points i, j. Then, construct a sparse graph, $\mathcal{G} = (v, e, w)$, in which the data objects are represented by the set of vertices v, the edges e, and the weights of the edges e. A weight of an edge, reflecting with a probability $\mathbf{P}_{ij}$ the strength of the relationship between corresponding objects, indicates whether nodes i and j are members of the same cluster. Our goal is to find a clustering that is described as a new graph $\mathcal{G}'$ with edges $x_{ij} \in \{0, 1\}$, where nodes i and j are assigned to the same cluster if $x_{ij} = 1$. To ensure consistency, the edges must specify an equivalence relationship: $x_{ii} = 1$ and $x_{ij} = x_{jk} = 1$ implies $x_{ij} = x_{ik}$.

Our goal is to discover a clustering that is as consistent as possible with the idea that edges with high probability should not cross cluster boundaries, while those with low probability should. We define the cost of cutting edges i and j as w_{ij}^{+}, whose probability is $\mathbf{P}_{ij}$, and the cost of keeping them as w_{ij}^{-}, respectively. Mathematically, this objective can be expressed as follows, as described in [27]:

$$\begin{aligned} &\min \sum_{i<j} w_{ij}^{-} x_{ij} + w_{ij}^{+}(1 - x_{ij}), \\ &\text{s.t. } x_{ij} + x_{jk} \geq x_{ik} \ \forall i, j, k, \\ &\qquad x_{ij} \in \{0, 1\} \ \forall i, j, \end{aligned} \tag{10}$$

where $x_{ij} = 0$ indicates that vertices i and j belongs to a common cluster, and $x_{ij} = 1$ indicates they are in separate clusters.

There are two plausible formulations, additive and logarithmic weights, for the costs w^{+} and w^{-}, both of which have gained support in literature. We can use either $w_{ij}^{+} = \mathbf{P}_{ij}$ and $w_{ij}^{-} = 1 - \mathbf{P}_{ij}$ as in [27], or $w_{ij}^{+} = \log(\mathbf{P}_{ij})$ and $w_{ij}^{-} = \log(1 - \mathbf{P}_{ij})$ as in [31]. Since the logarithmic technique chooses maximum-likelihood clustering, based on the assumption that $\mathbf{P}_{ij}$ are independent and identically distributed, given for the edge ij in actual clustering, it has a weak mathematical basis.

4.3. A numerical example

To better illustrate the key steps of the proposed clustering approximation, we provide a small exaggerated numerical example as shown in Fig. 2. In this example, for the 2M2D10C dataset, which has 10 clusters. Five RSP data blocks (i.e., block id $\#\{132, 63, 89, 151, 208\}$) of 5000 objects each were randomly taken. Intuitively, using the I-niceDP algorithm in each selected RSP data block, the clusters correspond to $\{8, 9, 9, 8, 7\}$. In other words, the I-niceDP and k-means combination calculates the centroids as illustrated in Fig. 2(a-e).

We gather the associated set of centroids in Fig. 2(f) that results in a conclusion synthesized by an ensemble function. The collective centroids are compact and coreset representations of subsets. Consequently, to solve this ensemble problem, we employ two schemes, SC and CC. According to both RSPCA-SC and RSPCA-CC aggregation processes, the number of approximations is 10 clusters, as shown in Fig. 2(g-h). Finally, Fig. 2(i-j) demonstrate the impact of ensemble results by comparing them with the actual cluster centroids.

4.4. Computational complexity analysis

We analyze the computational complexity of RSPCA. It has three major steps that need to be conducted: (1) performing an RSP operation to generate RSP data blocks and randomly selecting a subset of data blocks; (2) determining the initial centroids and number of clusters using the I-niceDP scheme and refining explored centroids via the k-means algorithm on each selected RSP data block; and (3) gathering the meta-data of intermediate results in RSP data blocks and performing an ensemble operation to obtain the final clustering results.

Suppose we have a dataset with N objects. We generate m RSP data blocks, each containing n data points, and then randomly select b subsets, where $b < m$. The first step involves applying an RSP operation with $\mathcal{O}(n \log(N/n))$ complexity. When considering the RSP operation as a data preprocessing step, its computational complexity can be ignored. Then, the I-niceDP scheme generates one-dimensional data and density peaks from each RSP data block to define initial cluster centroids with $\mathcal{O}(nf)$ complexity, where f is the dimension of the data. Then, to estimate the number of clusters and the initial centroids, a Gamma mixture model (GMM) is solved using the EM technique. It has a computational cost of $\mathcal{O}(\rho\mu nk)$, where ρ is the number of iterations needed to determine α for the GMM and μ is the number of EM algorithm iterations. Refining centroids using the k-means scheme requires $\mathcal{O}(ntfk)$ operations, where t is the maximum number of iterations and k is the number of clusters. Consequently, the I-niceDP algorithm requires $\mathcal{O}(nf + \rho\mu nk + ntfk)$ operations on a single node. The final step is to merge the individual results of RSP

data blocks, for which we propose two graph-based ensemble schemes: spectral and correlation clustering.

In the RSPCA-SC process, spectral clustering (SC) involves computing affinity and Laplacian matrices, resulting in a computational complexity of $\mathcal{O}(\hat{n}^3)$ for $\hat{n}$ objects. Here, $\hat{n}$ represents a set of centroids, and in practice, $\hat{n} \ll n \ll N$. In the RSPCA-CC, correlation clustering (CC) utilizes a signed graph as input. Similar to SC, CC also calculates a similarity matrix $\hat{n} \times \hat{n}$ among the objects (centroids), incurring a complexity of $\mathcal{O}(\hat{n}^2)$. The optimization problems of CC are known to be NP-complete. The overall computation required by CC is $\mathcal{O}(\hat{n}\log^2 \hat{n})$ with a high probability. In summary, the total computational complexities of the two proposed RSPCA-SC and RSPCA-CC clustering ensemble schemes are approximately $\mathcal{O}(n\log(N/n) + \rho\mu nk + ntfk + \hat{n}^3)$ and $\mathcal{O}(n\log(N/n) + \rho\mu nk + ntfk + \hat{n}\log^2 \hat{n})$, respectively.

We can operate a distributed and parallel computing framework to utilize a computing cluster with Q nodes, such as Apache Spark or Hadoop. The individual result vectors are then sent back to the master node with a communication complexity of $\mathcal{O}(1)$. Additionally, the RSPCA is distributed computing framework, enabling the execution of serial algorithms independently on local nodes without the need for data communication among the nodes.

5. Experiments

In this section, we report on extensive experiments to evaluate the performance of the proposed RSPCA. First, we introduce the datasets used, including the experimental setting. Then, we illustrate the parameter sensitivity and convergence of the RSPCA. Finally, we present a performance comparison with several state-of-the-art algorithms.

5.1. Datasets

We evaluated the proposed clustering methods using both synthetic and real-world datasets. The characteristics of the seven datasets are detailed in Table 2. To illustrate the advantages of RSPCA, we commenced the evaluation with two-dimensional (2D) synthetic datasets. The datasets used for this purpose are as follows:

- *3M2D5C* is designed for clustering tasks and is used in [32]. This synthetic dataset comprises 3 million objects drawn from a mixture of five bivariate normal distributions. Note that there are overlapping objects from components in the mixture.
- *2M2D10C* and *2M2D20C* are two synthetic datasets from the multivariate Gaussian distribution models $\mathcal{N}(0,1)$. The observations have independent and identically distributed dimensions with different parameters. Our focus is on varying sizes and the number of clusters.
- *Covertype (CT)*[2] contains 581,012 objects and describes seven forest cover types using 54 different geographic measurements. It is important to note that 84% of the items belong to 2 classes (36.5% type-1 and 48.7% type-2).
- *KDDcup'99 (KDD)*[3] is from the KDD Cup 1999 competition and contains network intrusion detection data. There are about 4.9 million objects with 42 features. There are 23 classes in this dataset, and 98.3% of the objects belong to 3 classes (normal 19.6%, neptune 21.6%, and smurf 56.8%).
- *PokerHand (PH)*[4] has about 1.2 million data points with ten predictive features. In this dataset, each record is an example of a hand consisting of five playing cards drawn from a standard deck of 52. There are ten classes, and two dominant classes account for over 90% of the samples (nothing in hand 49.9% and one pair 42.4%).
- *SUSY*[5] has been produced using Monte Carlo simulations. There are 5 million objects, and each object has 18 numerical features to help discriminate between two classes.

We retain all the characteristics of real-world datasets and convert the categorical features into numerical ones. Then, we create two-dimensional data using the truncated SVD (also known as LSA) tool. Additionally, we normalize features for all datasets using the max–min normalization technique.

5.2. Computation environment

All experiments are conducted on an Apache Spark cluster of 30 nodes equipped with Intel Xeon E5-2650 v2 2.60 GHz, 128 GB of memory, and 1.6 TB of SSD. All algorithms are implemented on the Python Anaconda platform.

To improve computational efficiency and data scalability, multiple samples are analyzed independently and in parallel, with no communication overhead between nodes during the computing. The clustering components are very small in size, so this process needs short time and almost negligible. This means that the overhead of generating clustering components of RSP data blocks constitutes the main portion of the CPU runtime in this framework, and it is linearly proportional to the data scale. It is worth noting that the Spark uses executors as the actual computing units. In this experiment, each executor is allocated 4 cores and 16 GB of memory, which means that the maximum parallelism supported by the experimental cluster is 240.

5.3. Experiment setup and parameter analysis

Table 3 summarizes the experimental parameters of the RSPCA in order to fairly test the influence of RSP data block size (n) and ensemble size (es). In the experiment, we investigate the performance of RSPCA-SC and RSPCA-CC schemes under various parameter settings. In order to explore the effect of the parameters, the value of each of them is trialed 20 times independently.

5.4. Competitors

We compare the RSPCA-SC and RSPCA-CC schemes with several state-of-the-art clustering algorithms that can process big datasets and estimate the number of clusters, including CAMUSA [33], kluster [22], nbootstrap [21], SNN-DPC [34], and U-SENC [2]. We conducted our experiments using the following settings:

COMUSA [33] is a similarity graph-based algorithm. It constructs a similarity graph by calculating the co-associations of objects in the input data. To start a new cluster, COMUSA selects a pivot vertex, and the cluster expands by adding its neighbors if they are the most similar to the pivot.

kluster [22] is a scalable method for estimating the number of clusters using the BIC (Bayesian Information Criterion), PAM (Partitioning Around Medoids), AP (Affinity Propagation), and CAL (Calinski and Harabasz Index) approaches. It applies these statistical methods iteratively to small data subsets and gives the most frequent and average approximate number of clusters.

nbootstrap [21] involves selecting the number of clusters through resampling. Two bootstraps are drawn from the dataset, and the number of clusters is determined by optimizing an instability estimation from these pairs. In this experiment, 20 resamples are utilized.

SNN-DPC [34] is a clustering approach based on the fast search and density peaks mechanism. The parameter k is an important factor that determines the granularity of clusters in SNN-DPC. When k is small, the algorithm identifies small and tightly packed clusters, whereas a large

[2] https://archive.ics.uci.edu/ml/datasets/covertype.
[3] https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data.
[4] https://archive.ics.uci.edu/ml/datasets/Poker+Hand.
[5] https://archive.ics.uci.edu/ml/datasets/SUSY.

Table 2
Characteristics of the datasets (N: number of observations, $\mathcal{F}$: number features, K: number of clusters or classes).

Datasets	N	$\mathcal{F}$	K	Cluster distributions
3M2D5C	3,000,000	2	5	300,000 × (2 : 2 : 3 : 1 : 2)
2M2D10C	2,000,000	2	10	100,000 × (2 : 3 : 3 : 1 : 1 : 2 : 2 : 3 : 1 : 2)
2M2D20C	2,000,000	2	20	10,000 × (15 : 5 : 13 : 7 : 10 : 12 : 8 : 10 : 15 : 5 : 8 : 12 : 13 : 7 : 10 : 13 : 7 : 12 : 8 : 10)
CT	581,012	2	7	211,840 : 283,301 : 35,754 : 2747 : 9493 : 17,367 : 20,510
KDD	4,898,431	42	23	972,781 : 2,807,886 : 1,072,017 : 15,892 : 12,481 : 10,413 : 2316 : 2203 : 1020 : 979 : 264 : 53 : 30 : 21 : 20 : 12 : 10 : 9 : 8 : 7 : 4 : 3 : 2
PH	1,025,010	10	10	513,702 : 433,097 : 48,828 : 21,634 : 3978 : 2050 : 1460 : 236 : 17 : 8
SUSY	5,000,000	18	2	2,712,173 : 2,287,827

Minor classes are indicated as noise data, which is underlined.

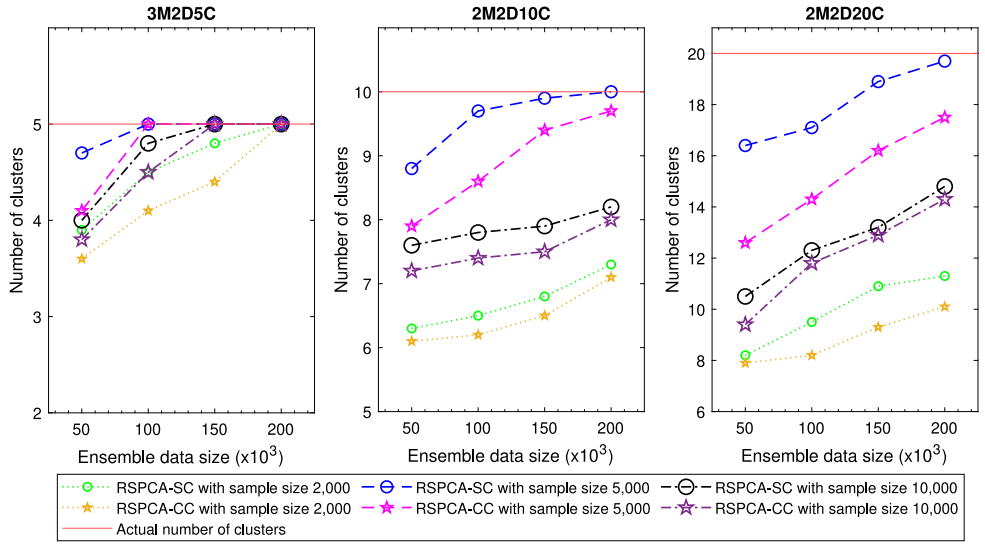


Fig. 3. Influence of RSP block size and ensemble size on the estimated number of clusters using RSPCA (average of 20 trails).

Table 3
Summary of some key parameters.

Parameters	Values
RSP data block size (n)	2000; 5000; 10 000
Ensemble data size (es)	5%; 10%; 15%; 20%
Number of observation points in I-niceDP	1

k detects larger and well-separated clusters. In our experiment, we set k to 20.

U-SENC [2] depends on the efficient construction of an affinity submatrix and a bipartite graph formulation. The U-SENC method involves a common parameter, p. In the experiments, we use $p = 1000$, and the number of clusters is determined by the k smallest eigenvectors close to zero that are obtained.

5.5. Evaluation metrics

It is challenging to effectively compare various clustering algorithms, especially when they yield different results. We use two widely used internal evaluation measures, the Davies–Bouldin index (DBI) [35] and Calinski–Harabasz index (CHI) [36], and three external evaluation measures, the adjusted mutual information (AMI) [37], adjusted rand index (ARI) [37], and Fowlkes–Mallows index (FMI) [38], to assess their effectiveness.

5.6. Experimental result analysis

5.6.1. Influence of RSP data block size and ensemble size

In order to assess the influence of the RSP data block size (n) and ensemble size (es) in cluster approximations, we tested three different sizes of RSP data blocks: 2000, 5000, and 10 000. The performances of RSPCA-SC and RSPCA-CC approaches on three synthesis datasets are illustrated in Fig. 3. From Fig. 3, we can see that small samples (n = 2000) are unable to identify the clusters, as are large samples (n = 10 000). On the other hand, moderate-sized samples (n = 5000) can more efficiently capture the clusters with increasing ensemble data size. Empirically, a relatively larger ensemble data size is beneficial.

5.6.2. Comparison with state-of-the-art algorithms

Table 4 displays the number of clusters identified using various algorithms across a range of ensemble sizes for different datasets. The values in parentheses for the three synthetic datasets represent the percentage of relative error in the estimated number of clusters determined by the applied algorithms. The results indicate that the RSPCA-SC, RSPCA-CC, kluster, COMUSA, and U-SENC algorithms can correctly identify clusters, while the nselecboot and SNN-DPC algorithms fail in doing so. For three synthetic datasets, the number of clusters identified by U-SENC and SNN-DPC is much lower than the actual number. Neither kluster and COMUSA nor U-SENC can accurately recognize the cluster centroids. The nselecboot algorithm splits the cluster into multiple clusters, rendering it unsuitable for centrocentric selection, leading to unsatisfactory outcomes. Visual judgment and results reveal that the SNN-DPC and U-SENC algorithms struggle to identify small clusters.

In our experiment, we considered four different ensemble sizes, including 5%, 10%, 15%, and 20%, that are much bigger than the classical statistical method. The fact that it is derived from several random samples, i.e., RSP data blocks, and is much bigger than the classical statistical method. It has broader coverage to identify all clusters. The RSPCA-SC and RSPCA-CC schemes accurately identify both the clusters and their corresponding centroids. We observe that as

Table 4
Comparison results of the cluster number approximation (mean ± std over 20 runs). Values in parenthesis () show relative errors.

Datasets	*es*	COMUSA	kluster	nselectboot	SNN-DPC	U-SENC	RSPCA-SC	RSPCA-CC
3M2D5C	5%	3.2±0.8 (40.0)	4.0±0.8 (20.0)	2.6±0.5 (48.0)	2.5±0.5 (50.0)	3.2±0.6 (35.0)	**4.7±0.5 (8.0)**	4.4±0.8 (12.0)
	10%	3.4±0.5 (30.0)	4.5±0.6 (10.0)	2.8±0.5 (44.0)	2.8±0.6 (45.0)	3.7±0.8 (25.0)	**5.0±0.0 (0.0)**	**5.0±0.5 (0.0)**
	15%	4.0±0.6 (20.0)	4.6±0.5 (6.7)	3.0±0.0 (40.0)	2.8±0.9 (35.0)	4.1±0.8 (20.0)	**5.0±0.0 (0.0)**	**5.0±0.0 (0.0)**
	20%	4.2±0.5 (20.0)	4.8±0.6 (5.0)	3.0±0.0 (40.0)	3.3±0.8 (35.0)	4.2±0.5 (16.0)	**5.0±0.0 (0.0)**	**5.0±0.0 (0.0)**
2M2D10C	5%	7.1±0.7 (30.0)	8.6±0.6 (15.0)	18.5±1.2 (85.0)	4.2±1.1 (57.5)	5.7±1.1 (42.5)	**8.8±0.9 (14.0)**	7.9±0.8 (24.0)
	10%	7.8±0.6 (22.5)	8.7±0.5 (12.5)	18.1±2.0 (80.0)	4.6±1.7 (55.0)	7.3±0.9 (27.5)	**9.7±0.6 (4.0)**	8.6±0.6 (16.0)
	15%	8.0±0.7 (20.0)	8.6±0.5 (13.3)	18.2±1.6 (83.0)	5.4±1.6 (45.0)	7.5±0.7 (25.0)	**9.9±0.3 (1.6)**	9.4±0.5 (14.0)
	20%	8.3±0.5 (17.5)	9.1±0.7 (7.5)	17.4±1.9 (78.0)	6.2±1.5 (37.5)	8.2±0.7 (20.0)	**10.0±0.0 (0.0)**	9.7±0.5 (6.0)
2M2D20C	5%	12.0±1.5 (40.0)	12.5±1.1 (36.7)	33.1±2.4 (65.0)	7.7±1.6 (61.3)	10.5±1.4 (47.5)	**16.4±1.4 (29.0)**	12.6±1.4 (48.0)
	10%	12.1±2.0 (38.8)	13.6±1.4 (31.8)	32.5±2.1 (63.0)	8.2±1.1 (58.8)	12.3±2.1 (38.7)	**17.1±1.2 (13.0)**	14.3±1.1 (40.0)
	15%	13.7±1.1 (31.3)	14.4±0.6(28.3)	32.6±2.5 (63.3)	9.2±1.9 (55.0)	13.1±1.7 (33.8)	**18.9±0.8 (8.0)**	16.2±1.3 (21.0)
	20%	14.0±1.3 (30.0)	14.7±0.5 (26.7)	33.3±1.4 (67.0)	9.7±1.5 (51.3)	14.0±0.8 (30.0)	**19.7±0.6 (3.0)**	17.5±0.9 (15.0)
CT	5%	8.3±1.1	10.3±0.6	2.2±0.6	3.4±0.8	5.4±1.4	4.5±0.6	5.1±0.8
	10%	10.2±1.5	11.0±0.8	2.3±0.6	3.5±1.1	7.1±0.9	4.9±0.6	5.7±0.6
	15%	10.1±1.5	10.8±0.6	2.2±0.5	4.4±0.9	7.7±0.9	5.1±0.5	5.9±0.6
	20%	10.3±1.6	10.6±0.5	2.1±0.5	4.7±1.2	9.2±0.8	5.2±0.5	6.1±0.4
KDD	5%	6.7±0.8	9.7±1.1	2.7±0.7	4.2±1.1	8.7±1.6	3.8±1.1	4.1±0.6
	10%	8.3±1.1	10.6±0.8	2.6±0.7	6.8±0.7	9.6±1.1	4.1±0.9	4.9±0.5
	15%	9.9±1.8	10.5±0.6	2.5±0.6	8.2±0.8	10.3±1.5	4.4±0.8	5.1±0.5
	20%	11.3±2.1	9.9±1.7	2.5±0.5	8.5±1.1	10.6±1.1	4.5±0.6	5.2±0.4
PH	5%	3.5±1.1	3.5±0.6	2.7±0.6	3.5±0.7	3.6±0.8	3.1±0.8	3.4±0.9
	10%	4.2±0.9	3.8±0.6	2.4±0.8	4.8±1.5	4.2±0.5	3.6±0.5	3.7±0.7
	15%	4.6±1.3	4.2±0.5	2.4±0.7	5.3±1.2	4.4±0.6	3.8±0.6	4.1±0.5
	20%	5.5±1.1	4.1±0.5	2.3±0.5	6.2±1.5	4.6±0.6	3.9±0.5	4.2±0.6
SUSY	5%	5.3±0.9	8.2±0.7	2.5±0.8	4.5±1.5	8.2±0.9	2.5±0.5	3.1±0.5
	10%	6.8±1.3	9.1±0.8	2.6±0.5	4.9±1.2	9.1±1.4	2.8±0.4	3.3±0.6
	15%	6.7±1.1	8.7±0.5	2.3±0.7	5.5±1.4	8.8±1.5	3.1±0.4	3.4±0.5
	20%	7.5±1.4	9.2±0.6	2.4±0.5	6.0±0.9	9.1±1.2	3.1±0.5	3.4±0.6

RSP block size $n = 5000$. The best and second best-results are highlighted in **bold** and underlined, respectively.

the ensemble sizes increase, the clustering approximation of the RSPCA-SC and RSPCA-CC schemes gradually improves. We were interested in determining the amount of data needed for the RSPCA-SC and RSPCA-CC schemes to achieve the most stable and optimal solution. The results show that the RSPCA-SC and RSPCA-CC schemes exhibit stability when they utilize $\approx 15-20\%$ of the entire dataset. Tables 5 and 6 presented clustering quality using internal and external validation measures. Both RSPCA-SC and RSPCA-CC consistently outperform other clustering methods in terms of DBI, CHI, AMI, ARI, and FMI for both synthetic and real-world datasets across different ensemble sizes. Additionally, it is worth noting that the clustering performance of all these methods improves as the percentage of data sizes increases.

Our primary goal is to develop effective and scalable methods for clustering big data. These methods should address issues such as high computational complexity and the challenge of determining the number of clusters when dealing with large datasets. We have introduced RSPCE [30], ACEM [39], and MSFC [40] solutions, which involve using multiple disjoint samples of a large dataset and combining their results to approximate the characteristics of the entire dataset. To ensure a fair comparison, we have conducted a comparative analysis of six datasets using multiple sample ensemble clustering techniques for cluster estimation, as shown in Fig. 4. The corresponding averages of 20 trials are included in the analysis.

Note that the RSPCE method yields higher accuracy in estimating the number of clusters for three synthetic datasets. However, it does not perform well when dealing with high-dimensional and complex real-world datasets. One limitation of RSPCE's cluster ball ensemble algorithm is its poor performance when applied to datasets with irregular-shaped clusters, such as moon-shaped and Swiss-roll data. On the other hand, we found that graph-based clustering ensemble algorithms demonstrate outstanding performance in estimating the number of irregularly shaped clusters in complex real-world datasets.

5.6.3. Scalability with ensemble data size

We study the scalability of RSPCA with varying ensemble data sizes. In the experiment, we randomly select four ensemble sizes, including 5%, 10%, 15%, and 20% of the entire dataset, to evaluate the efficiency of the methods. Fig. 5 depicts the running time of our RSPCA methods and other state-of-the-art clustering algorithms with varying ensemble data sizes on the tested datasets. We can see that RSPCA-SC, RSPCA-CC, and U-SPEC generally require less computational time than the other methods across most datasets. The CPU time required by COMUSA, SNN-DPC, and nselectboot algorithms increases linearly with the ensemble data size and is mainly independent of K. We were interested in knowing the computation duration of RSPCA-SC and RSPCA-CC schemes scales with ensemble cardinality to achieve the optimal solution. Remarkably, both the RSPCA-SC and RSPCA-CC schemes perform substantially better and exhibit advantageous scalability within nearly the same amount of time.

It is worth noting that in this experiment, the RSPCA uses at most 20% ensemble data size (e.g., 80 RSP data blocks for 2M2D10C and 2M2D20C datasets), which is significantly less than the maximum parallelism of 240 supported by the experimental environment. This indicates that the experiment can complete all RSP data block clustering in parallel with one batch. Thus, the CPU time in the experimental results can be regarded as the sum of the CPU time of a single machine and the CPU time for ensemble the clustering components, with the latter being very short, typically requiring only about 5–10 s.

5.6.4. Statistical analysis

To evaluate the estimated centroids of different algorithms, we have also used the two-sample Kolmogorov–Smirnov (KS) test to compare the results in Table 7. The KS test helps us determine whether two distributions (in this case, the centroids distance distributions of estimated and actual centroids) are different or the same. We have examined the cumulative distribution functions (CDF) of the centroids distance distributions, which are compared in Fig. 6 across three different synthesis datasets with two varying ensemble sizes, i.e., $es = \{10\%, 20\%\}$.

Table 5
Clustering performance comparison in terms of internal validation measures (average scores over 20 runs).

Datasets	Methods	DBI (lower values better)				CHI (higher values better)			
		es = 5%	10%	15%	20%	5%	10%	15%	20%
3M2D5C	COMUSA	0.635	0.550	0.512	0.512	2.5e+5	5.7e+5	1.1e+5	1.4e+6
	kluster	0.512	0.475	0.466	0.460	3.9e+5	8.8e+5	1.4e+6	2.0e+6
	nselectboot	0.723	0.687	0.596	0.596	2.1e+5	4.4e+5	7.5e+5	1.0e+6
	SNN-DPC	0.728	0.665	0.715	0.613	2.1e+5	4.5e+5	6.6e+5	1.0e+6
	U-SENC	0.571	0.536	0.512	0.497	2.7e+5	6.9e+5	1.1e+6	1.7e+6
	RSPCA-SC	**0.504**	0.447	**0.445**	**0.445**	**4.0e+5**	**1.1e+6**	**1.7e+6**	**2.3e+6**
	RSPCA-CC	0.594	**0.446**	0.446	0.446	3.9e+5	**1.1e+6**	**1.7e+6**	**2.3e+6**
2M2D10C	COMUSA	0.479	0.437	0.410	0.393	2.4e+5	6.0e+5	1.1e+6	1.5e+6
	kluster	0.371	0.349	0.356	0.285	4.4e+5	1.0e+6	1.4e+6	3.9e+6
	nselectboot	0.789	0.758	0.817	0.773	**2.0e+6**	**4.2e+6**	**6.4e+6**	8.4e+6
	SNN-DPC	0.748	0.731	0.574	0.567	1.3e+5	2.9e+5	6.2e+5	8.0e+5
	U-SENC	0.583	0.465	0.448	0.409	1.7e+5	5.4e+5	8.5e+5	1.4e+6
	RSPCA-SC	**0.350**	**0.226**	**0.216**	**0.162**	5.4e+5	1.1e+6	4.1e+6	**8.9e+6**
	RSPCA-CC	0.440	0.375	0.369	0.243	2.9e+5	8.9e+5	1.3e+6	5.5e+6
2M2D20C	COMUSA	0.517	0.521	0.443	0.521	2.3e+5	5.1e+5	8.8e+5	1.2e+6
	kluster	0.489	0.434	0.393	0.392	2.5e+5	6.1e+5	9.5e+5	1.3e+6
	nselectboot	0.712	0.687	0.701	0.702	**8.2e+6**	**1.6e+7**	**2.5e+7**	**3.3e+7**
	SNN-DPC	0.647	0.651	0.636	0.716	1.5e+5	3.3e+5	5.3e+5	7.5e+5
	U-SENC	0.601	0.521	0.490	0.422	1.9e+5	5.1e+5	8.2e+5	1.2e+6
	RSPCA-SC	**0.419**	**0.285**	**0.212**	**0.109**	2.9e+5	1.0e+6	3.8e+6	2.1e+7
	RSPCA-CC	0.552	0.489	0.343	0.314	1.9e+5	4.8e+5	1.2e+6	2.0e+6
CT	COMUSA	0.828	0.828	0.828	0.827	2.7e+4	5.4e+4	8.0e+4	1.0e+5
	kluster	**0.806**	**0.810**	**0.811**	**0.804**	2.7e+4	5.4e+4	8.1e+4	1.0e+5
	nselectboot	0.879	0.837	0.837	0.837	**3.1e+4**	**6.5e+4**	**9.8e+4**	**1.3e+5**
	SNN-DPC	0.936	0.895	0.912	0.905	2.8e+4	5.7e+4	8.2e+4	1.0e+5
	U-SENC	0.862	0.849	0.848	0.826	2.8e+4	5.4e+4	8.0e+4	1.0e+5
	RSPCA-SC	0.869	0.854	0.852	0.830	2.8e+4	5.4e+4	8.1e+4	1.1e+5
	RSPCA-CC	0.856	0.870	0.828	0.825	2.9e+4	5.4e+4	8.0e+4	1.1e+5
KDD	COMUSA	0.482	0.496	0.562	0.600	1.3e+7	**2.7e+7**	**4.2e+7**	5.3e+7
	kluster	0.578	0.602	0.627	0.566	1.3e+7	2.6e+7	4.1e+7	**5.4e+7**
	nselectboot	**0.283**	**0.318**	0.425	0.403	3.2e+6	5.9e+6	7.1e+6	1.1e+7
	SNN-DPC	0.890	0.843	0.853	0.848	3.4e+4	6.9e+4	1.3e+5	2.3e+5
	U-SENC	0.524	0.554	0.591	0.619	**1.4e+7**	2.6e+7	4.1e+7	5.3e+7
	RSPCA-SC	0.373	0.360	0.401	0.396	5.4e+6	1.8e+7	2.7e+7	4.1e+7
	RSPCA-CC	0.310	0.367	**0.384**	**0.368**	6.8e+6	2.0e+7	3.4e+7	4.6e+7
PH	COMUSA	0.993	0.917	0.901	0.875	3.2e+4	6.6e+4	1.3e+5	2.3e+5
	kluster	0.921	0.917	0.904	0.907	3.3e+4	6.8e+4	2.4e+5	1.3e+5
	nselectboot	1.123	1.108	1.103	1.192	2.9e+4	5.9e+4	1.1e+5	2.1e+5
	SNN-DPC	**0.348**	**0.353**	**0.403**	**0.561**	**6.1e+6**	**1.8e+7**	**3.2e+7**	**4.9e+7**
	U-SENC	0.921	0.917	0.903	0.901	3.3e+4	6.7e+4	2.4e+5	1.3e+5
	RSPCA-SC	0.998	0.984	0.987	0.903	3.0e+4	6.7e+4	1.4e+5	2.4e+5
	RSPCA-CC	0.915	0.905	0.837	0.845	3.4e+4	6.6e+4	1.3e+5	2.4e+5
SUSY	COMUSA	**0.827**	0.834	0.841	0.845	1.6e+5	3.7e+5	5.9e+5	7.7e+5
	kluster	0.838	0.831	0.837	0.844	1.6e+5	3.7e+5	5.8e+5	7.6e+5
	nselectboot	0.943	0.952	0.924	0.937	1.7e+5	**3.9e+5**	**6.1e+5**	**8.0e+5**
	SNN-DPC	0.880	0.835	0.839	0.836	**1.8e+5**	3.8e+5	5.8e+5	5.9e+5
	U-SENC	0.835	0.839	**0.832**	0.841	1.7e+5	3.7e+5	5.8e+5	7.7e+5
	RSPCA-SC	0.914	0.922	0.901	0.824	**1.8e+5**	3.6e+5	5.8e+5	7.5e+5
	RSPCA-CC	0.852	**0.820**	0.835	**0.809**	**1.8e+5**	3.8e+5	5.6e+5	7.5e+5

RSP block size $n = 5000$. The best and second best-scores are highlighted in **bold** and underlined, respectively.

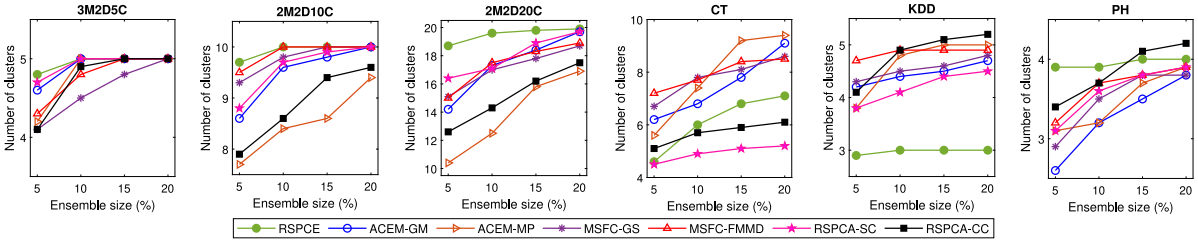


Fig. 4. A comparison of multiple sample ensemble clustering methods for cluster estimation.

To determine whether the algorithm is significantly different from others, we compute the z and p-values of the KS test. The z-value measures the distance between two distributions. The p-value represents the probability of observing a test statistic as extreme as the observed value under the null hypothesis. The null hypothesis assumes that the distance distribution between the actual and estimated centroids of the algorithm is the same, while the alternative hypothesis assumes that the estimated centroid's distribution is larger or different compared to the actual one. Our results show that the proposed RSPCA-SC and RSPCA-CC are statistically better than the five competitors at the 95%

Table 6
Clustering performance comparison in terms of external validation measures (average scores over 20 runs).

Datasets	Methods	AMI (higher values better)				ARI (higher values better)				FMI (higher values better)			
		es = 5%	10%	15%	20%	5%	10%	15%	20%	5%	10%	15%	20%
3M2D5C	COMUSA	0.699	0.777	0.838	0.839	0.585	0.679	0.773	0.773	0.728	0.776	0.842	0.842
	kluster	0.844	0.900	0.912	0.935	0.782	0.867	0.904	0.924	0.850	0.904	0.930	0.945
	nselectboot	0.618	0.655	0.725	0.726	0.448	0.528	0.605	0.605	0.674	0.695	0.735	0.735
	SNN-DPC	0.619	0.670	0.645	0.723	0.489	0.545	0.526	0.620	0.675	0.702	0.697	0.749
	U-SENC	0.754	0.813	0.838	0.872	0.644	0.737	0.773	0.829	0.759	0.820	0.842	0.881
	RSPCA-SC	**0.920**	**0.968**	**0.968**	**0.968**	**0.910**	**0.978**	**0.980**	**0.980**	**0.930**	**0.982**	**0.980**	**0.980**
	RSPCA-CC	0.878	**0.968**	**0.968**	0.967	0.856	0.976	0.979	0.978	0.892	0.973	0.974	0.978
2M2D10C	COMUSA	0.903	0.940	0.948	0.955	0.753	0.857	0.875	0.885	0.808	0.880	0.895	0.903
	kluster	0.960	0.965	0.963	0.975	0.900	0.912	0.910	0.948	0.902	0.915	0.923	0.955
	nselectboot	0.878	0.886	0.883	0.884	0.673	0.703	0.695	0.691	0.734	0.778	0.754	0.748
	SNN-DPC	0.745	0.760	0.852	0.854	0.508	0.555	0.685	0.664	0.646	0.678	0.765	0.748
	U-SENC	0.848	0.919	0.933	0.947	0.655	0.804	0.849	0.874	0.741	0.843	0.876	0.896
	RSPCA-SC	**0.974**	**0.990**	**0.995**	**0.998**	**0.939**	**0.972**	**0.972**	**1.000**	**0.939**	**0.976**	**0.977**	**1.000**
	RSPCA-CC	0.946	0.957	0.966	0.985	0.852	0.894	0.915	0.958	0.889	0.922	0.930	0.964
2M2D20C	COMUSA	0.903	0.908	0.930	0.933	0.725	0.733	0.790	0.795	0.768	0.775	0.818	0.760
	kluster	0.913	0.927	0.940	0.940	0.753	0.778	0.805	0.813	0.790	0.817	0.830	0.837
	nselectboot	0.916	0.926	0.926	0.923	0.762	0.770	0.770	0.767	0.796	0.797	0.800	0.797
	SNN-DPC	0.804	0.824	0.833	0.860	0.521	0.558	0.579	0.633	0.623	0.649	0.665	0.639
	U-SENC	0.876	0.906	0.923	0.932	0.665	0.731	0.773	0.794	0.726	0.775	0.806	0.823
	RSPCA-SC	**0.933**	**0.973**	**0.984**	**0.994**	**0.807**	**0.917**	**0.952**	**0.980**	**0.832**	**0.925**	**0.958**	**0.982**
	RSPCA-CC	0.882	0.935	0.952	0.968	0.692	0.754	0.868	0.895	0.724	0.798	0.870	0.905
CT	COMUSA	0.086	**0.091**	**0.093**	0.093	**0.012**	**0.012**	**0.014**	0.013	0.244	0.228	0.229	0.230
	kluster	**0.089**	**0.091**	0.091	**0.094**	**0.012**	**0.012**	**0.014**	0.014	0.225	0.220	0.220	0.223
	nselectboot	0.057	0.052	0.052	0.052	−0.009	−0.015	−0.016	0.001	**0.410**	**0.429**	**0.429**	**0.428**
	SNN-DPC	0.065	0.067	0.070	0.075	−0.001	−0.002	0.001	0.003	0.362	0.355	0.327	0.314
	U-SENC	0.076	0.085	0.089	0.093	0.000	0.007	0.010	0.013	0.298	0.258	0.251	0.237
	RSPCA-SC	0.081	0.084	0.088	**0.094**	0.006	0.007	0.012	**0.015**	0.354	0.322	0.325	0.297
	RSPCA-CC	0.082	0.088	0.091	0.093	0.007	0.009	**0.014**	**0.015**	0.322	0.287	0.287	0.258
KDD	COMUSA	0.805	0.817	0.794	0.780	0.902	0.908	0.891	0.881	0.937	0.946	0.935	0.929
	kluster	0.794	0.784	0.769	0.795	0.891	0.887	0.872	0.891	0.935	0.933	0.924	0.935
	nselectboot	0.804	0.791	0.756	0.791	0.831	0.800	0.799	0.800	0.917	0.903	0.877	0.904
	SNN-DPC	0.801	0.843	0.812	0.839	0.838	0.906	0.903	0.920	0.920	0.944	0.939	0.953
	U-SENC	0.800	0.802	0.782	0.779	0.900	0.898	0.882	0.881	0.931	0.940	0.930	0.928
	RSPCA-SC	0.806	**0.857**	0.815	0.848	0.848	**0.929**	0.860	0.923	0.925	**0.958**	0.928	0.955
	RSPCA-CC	**0.877**	0.856	**0.850**	**0.854**	**0.943**	0.928	**0.923**	**0.925**	**0.966**	0.957	**0.954**	**0.959**
PH	COMUSA	0.006	**0.008**	0.007	0.009	**0.004**	0.005	0.005	**0.006**	0.371	0.329	0.323	0.292
	kluster	0.005	0.006	0.007	0.007	0.003	0.003	0.003	0.003	0.357	0.349	0.331	0.332
	nselectboot	0.001	0.001	0.000	0.000	0.001	0.001	0.000	0.000	**0.438**	**0.436**	**0.437**	**0.455**
	SNN-DPC	**0.007**	**0.008**	**0.010**	**0.010**	**0.004**	**0.006**	0.006	**0.006**	0.310	0.264	0.245	0.239
	U-SENC	0.005	0.007	0.007	0.008	0.003	0.004	0.004	**0.006**	0.357	0.337	0.324	0.317
	RSPCA-SC	0.002	0.005	0.005	0.006	0.003	0.004	0.005	0.005	0.414	0.379	0.369	0.347
	RSPCA-CC	0.006	0.006	0.009	0.009	**0.004**	0.005	**0.007**	**0.006**	0.339	0.295	0.264	0.256
SUSY	COMUSA	**0.083**	**0.084**	**0.084**	**0.083**	0.066	0.070	0.070	0.062	0.404	0.384	0.389	0.360
	kluster	0.082	0.083	0.083	0.082	0.058	0.058	0.059	0.057	0.345	0.335	0.340	0.334
	nselectboot	0.065	0.077	0.081	0.081	**0.078**	**0.091**	**0.100**	**0.103**	**0.535**	**0.527**	**0.544**	**0.549**
	SNN-DPC	0.079	0.082	0.087	0.086	0.067	0.076	0.077	0.071	0.453	0.442	0.429	0.399
	U-SENC	0.081	0.080	0.082	0.080	0.062	0.059	0.059	0.058	0.354	0.338	0.340	0.338
	RSPCA-SC	0.058	0.066	0.067	0.072	0.079	0.070	0.078	0.084	0.534	0.518	0.524	0.497
	RSPCA-CC	0.065	0.071	0.072	0.078	0.072	0.069	0.069	0.069	0.505	0.487	0.468	0.440

RSP block size $n = 5000$. The best and second best-scores are highlighted in **bold** and underlined, respectively.

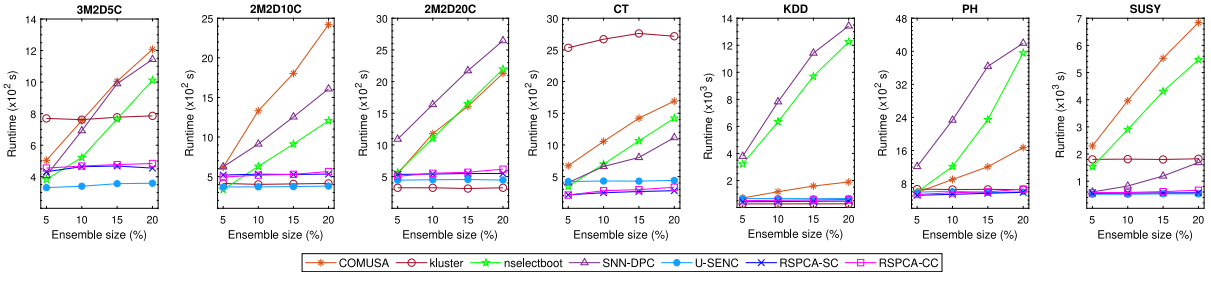
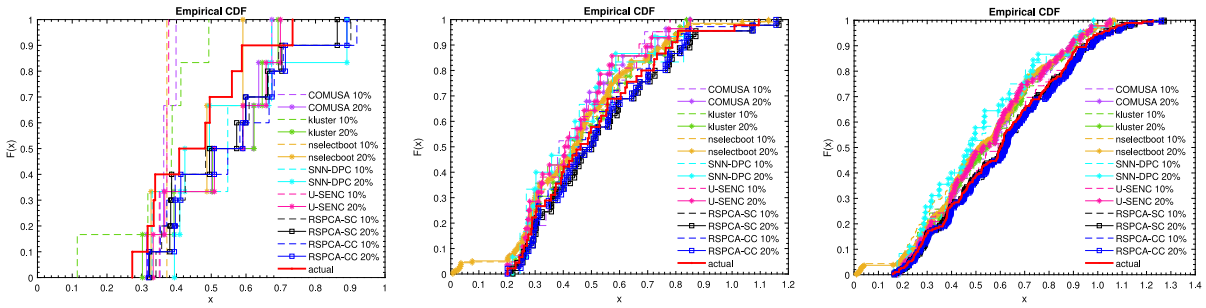


Fig. 5. Execution time (average of 20 runs) of our methods and the compared clustering methods.



(a) 3M2D5C (b) 2M2D10C (c) 2M2D20C

Fig. 6. The cumulative distribution functions show the centroid's distance distributions of three synthesis datasets. Two distributions are shown for 10% and 20% of the ensemble sizes.

Table 7
Results of the Kolmogorov–Smirnov (KS) test. z and p represent the test statistics and the probability of observing a test statistic, respectively. A higher p value indicates better results.

Datasets	Methods	$es = 10\%$		$es = 20\%$	
		z	p	z	p
3M2D5C	COMUSA	0.600	0.242	0.400	0.473
	kluster	0.333	0.603	0.400	0.472
	nselectboot	0.600	0.589	0.267	0.785
	SNN-DPC	0.366	0.725	0.400	0.473
	U-SENC	0.600	0.242	0.400	0.472
	RSPCA-SC	0.400	0.313	0.300	0.675
	RSPCA-CC	0.400	0.312	0.300	0.671
2M2D10C	COMUSA	0.168	0.773	0.167	0.677
	kluster	0.168	0.671	0.163	0.705
	nselectboot	0.102	0.884	0.114	0.747
	SNN-DPC	0.244	0.859	0.333	0.263
	U-SENC	0.152	0.865	0.186	0.546
	RSPCA-SC	0.111	0.930	0.111	0.929
	RSPCA-CC	0.117	0.961	0.083	0.998
2M2D20C	COMUSA	0.094	0.754	0.110	0.426
	kluster	0.111	0.553	0.117	0.347
	nselectboot	0.118	0.086	0.122	0.043
	SNN-DPC	0.176	0.275	0.213	0.063
	U-SENC	0.088	0.927	0.093	0.709
	RSPCA-SC	0.066	0.894	0.053	0.952
	RSPCA-CC	0.061	0.900	0.068	0.751

confidence level. In other words, the proposed RSPCA-based schemes are the "winners" in this scenario.

5.6.5. Discussions

The current mainstream big data paradigms use the distributed and parallel data clustering model, but the high computational cost makes computing an entire dataset inefficient. In the RSPCA computing framework, we eliminate inter-node data communication by not iterating on the entire dataset. We have demonstrated that by representing a large dataset as an RSP data model, we can replace the costly record-level sample with an efficient block-level sample clustering process. Additionally, the big dataset clustering analysis transforms into the analysis of multiple random RSP data blocks. Once we convert a large dataset into an RSP data model, we no longer need to analyze it all at once to estimate clustering properties. Consequently, we compute ensemble estimations from multiple RSP samples via a serial clustering algorithm in a parallel manner. As we see in the experiment results, a few RSP sample ensembles can obtain the global clustering approximation. Also, we can enrich results incrementally by appending new clustering outcomes from newly analyzed RSP data blocks.

Merits and Limitation. The experimental results reveal some attractive merits of RSPCA. Specifically, RSPCA can efficiently identify the number of clusters in large-scale data, is scalable, is not sensitive to noise, and is parameter-free. Nevertheless, we also found the limitation of RSPCA is ineffective on high-dimensional datasets.

6. Conclusions

In this paper, we propose a novel approximate clustering ensemble method, RSPCA, for big data clustering to improve the robustness of the result, the computation efficiency, and data scalability. Specifically, to make the large-scale data clustering problem tractable, in RSPCA, we introduced a non-MapReduce computing framework, that allows execution of serial algorithms independently to generate component results so as to improve the computation efficiency and data scalability. Subsequently, in RSPCA, two clustering ensemble schemes, spectral and correlation, were designed to efficiently integrate the clustering outcomes of multiple samples for handling irregular clusters. Extensive experiments have been conducted, and the results demonstrate the scalability and approximation robustness of the new approach. Besides, RSPCA yields a more accurate and interpretable solution compared to the existing clustering methods. Therefore, computational efficiency, data scalability and easy-to-use make RSPCA a desirable tool for exploration of big data.

In the future, we will conduct a stringent theoretical investigation on ensemble clustering with multiple random samples and investigate the problems of RSPCA's poor performance on high-dimensional datasets. A possible solution would be to explore a subspace cluster identification method for multiple random samples of a big dataset. Clearly, as a general clustering ensemble framework for large-scale datasets, our method has a potential to be applied to new data domains, such as stream data. But special operations must be implemented to compute the component clustering results and the clustering ensemble. Moreover, the general idea of RSPCA can be extended to evolutionary learning algorithms, e.g., genetic algorithms (GA) and particle swarm optimization (PSO), to optimize the number of clusters in multiple random samples in large-scale data clustering ensembles.

CRediT authorship contribution statement

Mohammad Sultan Mahmud: Writing – original draft, Software, Methodology, Conceptualization. **Hua Zheng:** Writing – review & editing, Investigation, Formal analysis. **Diego Garcia-Gil:** Writing – review & editing, Validation, Formal analysis. **Salvador García:** Writing – review & editing, Supervision, Conceptualization. **Joshua Zhexue Huang:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has been supported by the Key Basic Research Foundation of Shenzhen under Grant No. JCYJ20220818100205012 and partially supported by Project PID2023-150070NB-I00 by MICINN/AEI and is part of the I+D+i project granted by C-ING-250-UGR23 co-funded by Consejería de Universidad, Investigación e Innovación and for the European Union related to the FEDER Andalucía Program 2021-2027.

Data availability

Data will be made available on request.

References

[1] S. Ma, J. Huai, Approximate computation for big data analytics, ACM SIGWEB Newsl. (2021) 1–8, http://dx.doi.org/10.1145/3447879.3447883.

[2] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, C.-K. Kwoh, Ultra-scalable spectral clustering and ensemble clustering, IEEE Trans. Knowl. Data Eng. 32 (6) (2020) 1212–1226, http://dx.doi.org/10.1109/TKDE.2019.2903410.

[3] F. Li, Y. Qian, J. Wang, C. Dang, L. Jing, Clustering ensemble based on sample's stability, Artificial Intelligence 273 (2019) 37–55, http://dx.doi.org/10.1016/J.ARTINT.2018.12.007.

[4] J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García, F. Herrera, Big Data Preprocessing: Enabling Smart Data, Springer Nature, 2020, http://dx.doi.org/10.1007/978-3-030-39105-8.

[5] M.S. Mahmud, J.Z. Huang, S. Salloum, T.Z. Emara, K. Sadatdiynov, A survey of data partitioning and sampling methods to support big data analysis, Big Data Min. Anal. 3 (2) (2020) 85–101, http://dx.doi.org/10.26599/BDMA.2019.9020015.

[6] X.-L. Meng, Statistical paradises and paradoxes in big data (I): Law of large populations, big data paradox, and the 2016 US presidential election, Ann. Appl. Stat. 12 (2) (2018) 685–726, http://dx.doi.org/10.1214/18-AOAS1161SF.

[7] N. Iam-On, T. Boongoen, S. Garrett, C. Price, A link-based approach to the cluster ensemble problem, IEEE Trans. Pattern Anal. Mach. Intell. 33 (12) (2011) 2396–2409, http://dx.doi.org/10.1109/TPAMI.2011.84.

[8] D. Huang, J. Lai, C.-D. Wang, Ensemble clustering using factor graph, Pattern Recognit. 50 (2016) 131–142, http://dx.doi.org/10.1016/j.patcog.2015.08.015.

[9] F. Cicalese, E.S. Laber, Information theoretical clustering is hard to approximate, IEEE Trans. Inform. Theory 67 (1) (2021) 586–597, http://dx.doi.org/10.1109/TIT.2020.3031629.

[10] X. Niu, C. Zhang, X. Zhao, L. Hu, J. Zhang, A multi-view ensemble clustering approach using joint affinity matrix, Expert Syst. Appl. 216 (2023) 119484, http://dx.doi.org/10.1016/j.eswa.2022.119484.

[11] R. Mussabayev, N. Mladenovic, B. Jarboui, R. Mussabayev, How to use K-means for big data clustering? Pattern Recognit. 137 (2023) 109269, http://dx.doi.org/10.1016/j.patcog.2022.109269.

[12] A.M. Ikotun, A.E. Ezugwu, L. Abualigah, B. Abuhaija, J. Heming, K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data, Inform. Sci. 622 (2023) 178–210, http://dx.doi.org/10.1016/j.ins.2022.11.139.

[13] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, S. Vassilvitskii, Scalable k-means++, Proc. VLDB Endow. 5 (7) (2012) 622–633, http://dx.doi.org/10.14778/2180912.2180915.

[14] M. Servetnyk, C.C. Fung, Distributed dual averaging based data clustering, IEEE Trans. Big Data 9 (1) (2023) 372–379, http://dx.doi.org/10.1109/TBDATA.2022.3146169.

[15] J. Lu, Y. Zhao, K. Tan, Z. Wang, Distributed density peaks clustering revisited, IEEE Trans. Knowl. Data Eng. 34 (8) (2022) 3714–3726, http://dx.doi.org/10.1109/TKDE.2020.3034611.

[16] T. Qiu, Y.-J. Li, Fast LDP-MST: An efficient density-peak-based clustering method for large-size datasets, IEEE Trans. Knowl. Data Eng. 35 (5) (2023) 4767–4780, http://dx.doi.org/10.1109/TKDE.2022.3150403.

[17] D. Cheng, J. Huang, S. Zhang, X. Zhang, X. Luo, A novel approximate spectral clustering algorithm with dense cores and density peaks, IEEE Trans. Syst. Man Cybern.: Syst. 52 (4) (2022) 2348–2360, http://dx.doi.org/10.1109/TSMC.2021.3049490.

[18] L. Yaohui, M. Zhengming, Y. Fang, Adaptive density peak clustering based on K-nearest neighbors with aggregating strategy, Knowl.-Based Syst. 133 (2017) 208–220, http://dx.doi.org/10.1016/j.knosys.2017.07.010.

[19] Y. He, H. Tan, W. Luo, S. Feng, J. Fan, MR-DBSCAN: a scalable MapReduce-based DBSCAN algorithm for heavily skewed data, Front. Comput. Sci. 8 (1) (2014) 83–99, http://dx.doi.org/10.1007/S11704-013-3158-3.

[20] C. Patil, I. Baidari, Estimating the optimal number of clusters k in a dataset using data depth, Data Sci. Eng. 4 (2019) 132–140, http://dx.doi.org/10.1007/s41019-019-0091-y.

[21] Y. Fang, J. Wang, Selection of the number of clusters via the bootstrap method, Comput. Statist. Data Anal. 56 (3) (2012) 468–477, http://dx.doi.org/10.1016/j.csda.2011.09.003.

[22] H. Estiri, B.A. Omran, S.N. Murphy, Kluster: An efficient scalable procedure for approximating the number of clusters in unsupervised learning, Big Data Res. 13 (2018) 38–51, http://dx.doi.org/10.1016/j.bdr.2018.05.003.

[23] U. von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416, http://dx.doi.org/10.1007/S11222-007-9033-Z.

[24] L. Ding, C. Li, D. Jin, S. Ding, Survey of spectral clustering based on graph theory, Pattern Recognit. 151 (2024) 110366, http://dx.doi.org/10.1016/J.PATCOG.2024.110366.

[25] N. Bansal, A. Blum, S. Chawla, Correlation clustering, Mach. Learn. 56 (2002) 89–113, http://dx.doi.org/10.1023/B:MACH.0000033116.57574.95.

[26] J. Hua, J. Yu, M. Yang, Star-based learning correlation clustering, Pattern Recognit. 116 (2021) 107966, http://dx.doi.org/10.1016/J.PATCOG.2021.107966.

[27] N. Ailon, M. Charikar, A. Newman, Aggregating inconsistent information: Ranking and clustering, J. ACM 55 (5) (2008) 1–27, http://dx.doi.org/10.1145/1411509.1411513.

[28] S. Salloum, J.Z. Huang, Y. He, Random sample partition: A distributed data model for big data analysis, IEEE Trans. Ind. Inform. 15 (11) (2019) 5846–5854, http://dx.doi.org/10.1109/TII.2019.2912723.

[29] Y. He, Y. Wu, H. Qin, J.Z. Huang, Y. Jin, Improved I-nice clustering algorithm based on density peaks mechanism, Inform. Sci. 548 (2021) 177–190, http://dx.doi.org/10.1016/j.ins.2020.09.068.

[30] M.S. Mahmud, J.Z. Huang, R. Ruby, K. Wu, An ensemble method for estimating the number of clusters in a big data set using multiple random samples, J. Big Data 10 (1) (2023) 40, http://dx.doi.org/10.1186/S40537-023-00709-4.

[31] J.R. Finkel, C.D. Manning, Enforcing transitivity in coreference resolution, in: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, in: HLT-Short'08, 2008, pp. 45–48, http://dx.doi.org/10.5555/1557690.1557703.

[32] X. Zhao, J. Liang, C. Dang, A stratified sampling based clustering algorithm for large-scale data, Knowl.-Based Syst. 163 (2019) 416–428, http://dx.doi.org/10.1016/j.knosys.2018.09.007.

[33] S. Mimaroglu, E. Erdil, Combining multiple clusterings using similarity graph, Pattern Recognit. 44 (3) (2011) 694–703, http://dx.doi.org/10.1016/j.patcog.2010.09.008.

[34] R. Liu, H. Wang, X. Yu, Shared-nearest-neighbor-based clustering by fast search and find of density peaks, Inform. Sci. 450 (2018) 200–226, http://dx.doi.org/10.1016/j.ins.2018.03.031.

[35] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 1 (2) (1979) 224–227, http://dx.doi.org/10.1109/TPAMI.1979.4766909.

[36] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Commun. Stat. 3 (1) (1974) 1–27, http://dx.doi.org/10.1080/03610927408827101.

[37] N.X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance, J. Mach. Learn. Res. 11 (2010) 2837–2854, http://dx.doi.org/10.5555/1756006.1953024.

[38] E.B. Fowlkes, C.L. Mallows, A method for comparing two hierarchical clusterings, J. Amer. Statist. Assoc. 78 (383) (1983) 553–569, http://dx.doi.org/10.1080/01621459.1983.10478008.

[39] M.S. Mahmud, J.Z. Huang, R. Ruby, A. Ngueilbaye, K. Wu, Approximate clustering ensemble method for big data, IEEE Trans. Big Data 9 (4) (2023) 1142–1155, http://dx.doi.org/10.1109/TBDATA.2023.3255003.

[40] M.S. Mahmud, J.Z. Huang, S. García, Clustering approximation via a fusion of multiple random samples, Inf. Fusion 101 (2024) 1–13, http://dx.doi.org/10.1016/J.INFFUS.2023.101986.

Mohammad Sultan Mahmud received the Ph.D. degree from Shenzhen University, China, in 2023, and the master's degree from King Mongkut's University of Technology North Bangkok, Thailand, in 2014. Currently, Dr. Mahmud is a Research Fellow with the College of Computer Science and Software Engineering, Shenzhen University, China. He is one of the pioneers in the distributed clustering ensemble of big data research. Dr. Mahmud has published research articles in esteemed journals such as Information Fusion, IEEE Transactions on Big Data, Big Data Mining and Analytics, and the Journal of Big Data. He has organized a special issue in Information Fusion (Elsevier) entitled Mixture of Experts (MoE) and Ensemble Learning for Big Data. He was a recipient of several awards or honours, notable among which are the Guangdong Government Outstanding International Student Scholarship in 2022 and 2023, the

Excellent Paper Award 2021 Big Data Mining and Analytics, the Shenzhen Universiade International Student Scholarship in 2018, and the Outstanding Doctoral Student of Shenzhen University in 2017. His current research focuses on big data mining, distributed and parallel computing, ensemble learning, and mixture of experts.

Hua Zheng received a Ph.D. degree from Bournemouth University, UK, in 2024. He is an Associate Professor at the Software Engineering Institute Guangzhou, China. His research interests include Graph Neural Networks, Graph Transformers, Geometric Deep Learning, Graph Clustering, Computational Biology, and AI Pharmaceuticals.

Diego García-Gil received his M.Sc. degree in computer science in 2015, and Ph.D. degree in 2020, both from the University of Granada, Spain. Currently, Dr. García-Gil is an Assistant Professor in the Department of Software Engineering at the University of Granada and a Research Fellow at the Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI Institute). He has authored several scientific articles in esteemed journals such as Information Fusion, International Journal of Intelligent Systems, and Information Sciences. Additionally, he has published a monograph with the prestigious Springer, titled "Big Data Preprocessing - Enabling Smart Data," released in 2020. He has also delivered numerous courses, both nationally and internationally, related to his research area. His career has been marked by the study and development of methodologies in various fields of data science and computational intelligence, focusing on ensemble methods for classification and data preprocessing in Big Data. Within artificial intelligence, his main research interests focus on big data science, ensemble learning, mixture of experts, and anomaly detection.

Salvador García received the Ph.D. degree in Computer Science from the University of Granada, Granada, Spain, in 2008. Currently, he is a Professor at the Department of Computer Science and Artificial Intelligence, University of Granada, Spain. Prof. García has published more than 80 papers in international journals (more than 60 in Q1), an h-index of 43, and over 60 papers in international conference proceedings (data from Web of Science). He has organized several special sessions and workshops related to data preprocessing and evolutionary learning in conferences such as "Hybrid Intelligent Systems", "Intelligent Systems Design and Applications" and "International Joint-Conference of Neural Networks". He has been associated with the international program committees and organizing committees of several regular international conferences, including IEEE CEC, ICPR, ICDM, IJCAI, etc. As part of his editing activities, he has co-edited two special issues in international journals. He is the Chief Editor of "Information Fusion" (Elsevier) and "Swarm and Evolutionary Computation" (Elsevier), and he is Co-Editor in Chief of the international journal "Progress in Artificial Intelligence" (Springer). He is a co-author of the books entitled "Data Preprocessing in Data Mining" and "Learning from Imbalanced Data Sets," both published by Springer. His research interests include data science, data preprocessing, Big Data, evolutionary learning, Deep learning, metaheuristics and biometrics.

Joshua Zhexue Huang received the Ph.D. degree from the Royal Institute of Technology, Sweden, in 1993. He is a Distinguished Professor at Shenzhen University, China. Also, he is the Director of the Big Data Institute and the Deputy Director of the National Engineering Laboratory for Big Data System Computing Technology. His main research interests include big data technology and applications. Prof. Huang has published over 260 research papers in conferences and journals. In 2006, he received the most influential paper award at the First Pacific-Asia Conference on Knowledge Discovery and Data Mining. Prof. Huang is known for his contributions to the development of a series of k-means type clustering algorithms in data mining, such as k-modes, fuzzy k-modes, k-prototypes, and w-kmeans, that are widely cited and used, and some of which have been included in commercial software. He has extensive industry expertise in business intelligence and data mining, and has been involved in numerous consulting projects in Australia and China.