Razavi *et al. J Big Data* (2021) 8:64
https://doi.org/10.1186/s40537-021-00450-w

Journal of Big Data

RESEARCH Open Access

Big data fuzzy C-means algorithm based on bee colony optimization using an Apache Hbase

Seyyed Mohammad Razavi[†], Mohsen Kahani[*†] and Samad Paydar[†]

*Correspondence:
kahani@um.ac.ir
[†]Seyyed Mohammad Razavi and Mohsen Kahani contributed equally to this work
[†]Samad Paydar Advisor contributor
Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

Abstract

Clustering algorithm analysis, including time and space complexity analysis, has always been discussed in the literature. The emergence of big data has also created a lot of challenges for this issue. Because of high complexity and execution time, traditional clustering techniques cannot be used for such an amount of data. This problem has been addressed in this research. To present the clustering algorithm using a bee colony algorithm and high-speed read/write performance, Map-Reduce architecture is used. Using this architecture allows the proposed method to cluster any volume of data, and there is no limit to the amount of data. The presented algorithm has good performance and high precision. The simulation results on 3 datasets show that the presented algorithm is more efficient than other big data clustering methods. Also, the results of our algorithm execution time on huge datasets are much better than other big data clustering approaches.

Keywords: Clustering, Big data, Bee colony, MapReduce

Introduction

Nowadays, scientists believe that the issue of big data is the biggest challenge in computer science. Social websites like Facebook and Twitter have billions of users that produce hundreds of gigabytes of information per minute. There are over one billion users on YouTube that produce and upload hundreds of hours of movie per minute [1, 2].

In order to use and discover knowledge out of this massive amount of data, the preparation of proper tools and infrastructures is essential. Data mining techniques are among the most famous and authentic methods of knowledge discovery [3–5]. Clustering, as one of the main data mining methods, is aimed at partitioning a given dataset into a finite number of groups, called clusters, so that each member of each group is more similar to any other member in the same group than any member in the other groups [1].

Although originally rooted in the domain of data mining, clustering is extensively used to address various problems in different domains, such as bioinformatics, machine learning, networking, and pattern recognition [6–8].

Due to its algorithmic nature, the analysis of time and space complexity of a clustering algorithm is great importance in the evaluation of its performance, and hence its widespread acceptance and use among researchers and practitioners. This becomes more crucial when clustering big datasets, as proposing effective and efficient clustering algorithms for this scale of data is very challenging.

There are different challenges for clustering big data, many of them are rooted in the inherent features of big data. The more important elements of which are [9]:

1. Volume: big data deals with huge amounts of data. While there is not a widely known agreement on the exact size of data that can be considered to be big, but the authors in [10] have provided a simple but reasonable characterization of the volume of big data (1). Since our focus in this paper is on clustering of big data, it is worth noting that determining and analyzing the relations between data elements, which is a primitive operation in clustering methods, is quite challenging in the presence of a massive amount of data. This emphasizes the need for efficient methods for big data clustering.
2. Velocity: the speed with which the data is generated in big data systems, is so high that the system needs to be able to respond to huge amounts of incoming data within a short period of time. This means that a clustering algorithm needs to analyze the relations between data elements so quickly that the fast incoming data does not invalidate the results of the analysis before they are used.
3. Variety: usually, big data systems have to handle a wide range of different types of incoming data, including structured (e.g. CSV data), semi-structured (e.g. HTML content), and unstructured data (e.g. video and image). From the point of view of clustering, it means that the algorithm that is responsible for analyzing and comparing the data elements must be able to handle the different dimensions of heterogeneity of those data elements, which is quite challenging.
4. Variability: the rate with which different types of data, or even different instances of the same type of data, are produced in a big data system has a wide range of variation, as some data instances might be generated per microsecond, while others in seconds or even days. This makes processing and in particular clustering of, big data more difficult as it is hard to make an assumption about the distribution of the presence of different data among the incoming.
5. Complexity: big data systems need to handle data that comes from different, usually heterogeneous, data sources with different data governance or data generation characteristics, making control of the streams of incoming data more challenging. This again affects the performance of clustering methods as they have more problems analyzing the relations among different data elements.

While being directly associated with the nature of big data, the challenges described above have different levels of severity and likelihood for different application domains. For instance, the diagram shown in Fig. 1 demonstrates the severity of each challenge in three data analytics tasks, namely clustering, stream classification, and noise detection.

Traditional clustering techniques cannot be used for such a large volume of data because of their high time complexity and execution time. For instance, the k-means

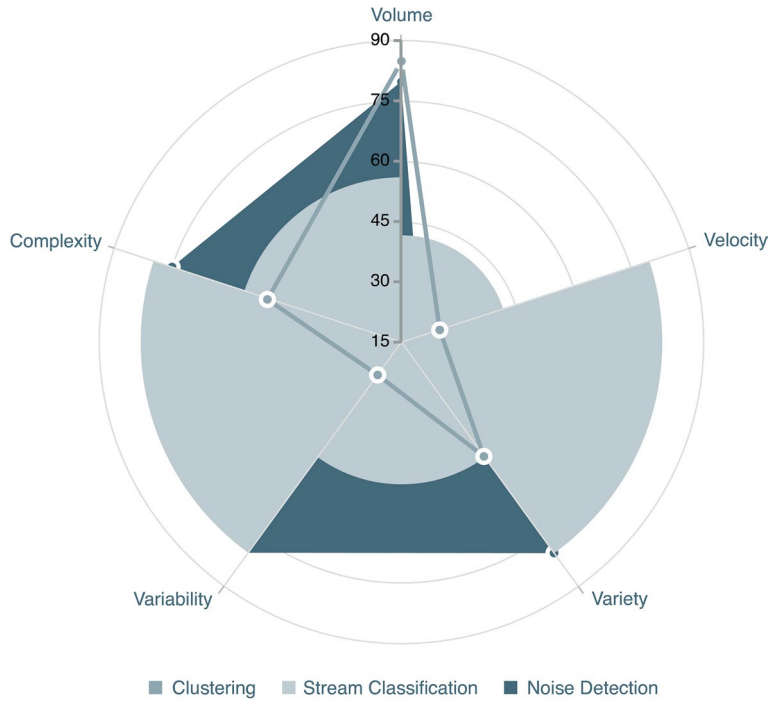

Fig. 1 The difference in the challenges mentioned in different applications in big data

clustering problem is NP-hard, even if K equals 2. The main goal is to accelerate the clustering algorithm with the least effect on their clustering quality.

Therefore, the main problem presented for this solution in this paper is Clustering large volumes of data (in terabytes and petabytes) using Map-Reduce architecture so that the quality of clustering is not compromised.

The paper is organized as follows. "Related work" section describes the related work on clustering techniques for big data, including single-machine clustering techniques and multi-machine clustering techniques. "Preliminary" section introduces the proposed big data clustering algorithm and "Proposed method" section discusses the experimental evaluation of this algorithm. Finally, "Results" section concludes the paper.

Related work

As mentioned before, services and sources like sensor networks, cloud storage, social networks and etc. produce a massive amount of data that its management, reuse, and analysis are an indispensable need of today's world. The main goal of clustering techniques on massive data is to improve the partition and segregation of this data. The resulting data, then, can be used with less complexity and at a higher pace.

In the following, different types of big data clustering algorithms are discussed. Generally big data clustering algorithms are divided into two categories: (1) single machine clustering algorithms and (2) Multiple machine clustering algorithm. In recent years, multiple machine clustering algorithms have been paid more attention than single machine clustering algorithms, because of their less execution time and capability to handle bigger size of data. As shown in Fig. 2, single and multiple machine algorithms

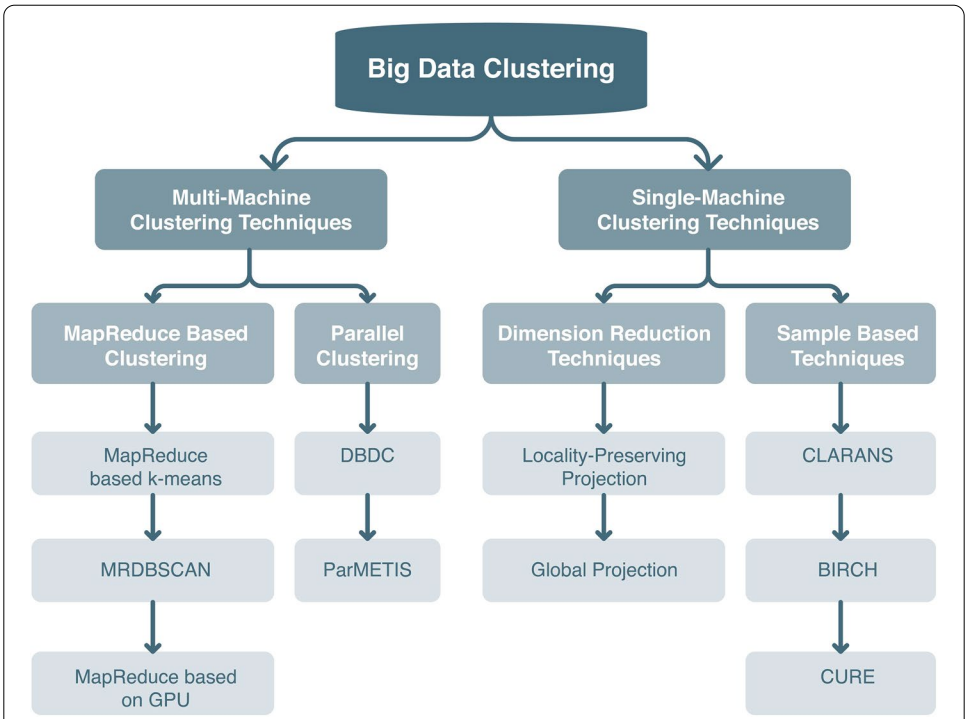

Fig. 2 Classification of big data clustering algorithms [55]

are composed of different techniques: (1) Single machine clustering that contains sampling techniques or dimension reduction techniques, and (2) Multiple machine clustering algorithms that contains parallel clustering or MapReduce [11] based clustering. Based on the above categorization.

Single-machine clustering techniques

Single machine techniques consist of sampling and dimensionality reduction techniques, which are elaborated in the following. These algorithms are the first group of clustering algorithms that were used for performance promotion and scale development and their goal is to contrast with the exponential space state in the clustering.

These algorithms are considered as sampling because they perform the clustering process on a sample of dataset and generalize the results to the entire dataset rather than just performing the clustering on the entire dataset. This makes the algorithm faster because the computations are done on a smaller volume of data and as a result the time and space complexity of these algorithms are lower.

CLARANS

Before addressing CLARANS algorithm [12], its primary version, CLARA [13], is discussed. This method is more powerful and capable of performing on big datasets compared with PAM [14] that partition around the center. CLARA reduces the quadratic and time complexity needed for algorithm execution to the linear function of the number of data. PAM computes all the non-similarity matrices among data for the entire

data set and saves them in RAM. So it has $O(n^2)$ space complexity that cannot be used for large amounts of n.

To overcome this problem, CLARA doesn't compute the non-similarity matrix for the entire dataset. PAM and CLARA can be considered equal to graph search problems that each vertex is a possible solution for clustering and two vertexes are connected when they have differences in one center. PAM starts with a randomly chosen vertex and greedily moves to the neighbor vertexes unless it finds no better neighbor. CLARA does this search in a sub graph, instead of the whole graph, which has a number of $O(k)$. In order to improve the quality and scalability of CLARA, another algorithm CLARANS was suggested. That was the combination of sampling technique and PAM algorithm. Unlike CLARA, this algorithm does not restrict itself to constant samples resulted from sampling. CLARANS is presented for the improvement of CLARA. Similar to PAM, CLARANS algorithm searches all the graphs to find the optimal solution. However, here, in each step it investigates some instances of the current neighbor vertexes. Both CLARA and CLARANS use the sampling technique, but their difference is in how they execute this technique. Sampling in CLARA happens at the beginning and the search space limits to a sub-graph while in CLARANS, the sampling applies dynamically in each step of algorithm execution. Results show that CLARANS use of dynamic sampling made it more efficient compared to CLARA [12].

BIRCH

If data exceed RAM capacity the I/O operations amplify the computation cost as well as execution time. BIRCH algorithm [15] is a solution to this problem. BIRCH uses its exclusive data structure and called clustering feature(CF) and its tree which is CF-tree (Clustering Feature tree). In fact, CF consists of a summary of each cluster. In BIRCH, CF metadata is saved in the main RAM. Each CF is a triple containing $< N, LS, SS >$ which respectively shows the number of data in the cluster, the aggregation of cluster data, and the sum of data cluster squares. In order to merge two clusters, their CF triples are mutually added. The advantage of this method is that for merging two clusters, just CF triples are added without any demand for the data into clusters and no computation complexity is imposed on the algorithm. There are two main phases in BIRCH: (1) Dataset is checked out and the CF-tree is created in main memory, and (2) Clustering is done on the CF-tree. Experiments show that CLARANS had a better performance both in time and space complexity in comparison with BIRCH and also has a more efficient execution when facing noise data.

CURE

In the aforementioned algorithms, each cluster represents by a point and the clustering algorithm started data clustering based on this approach. Until data are spherical, this process is suitable but in the real world, data might be much more complicated. To address this issue, CURE [16] algorithm uses more than one points to show the concept of clustering. In fact, this algorithm assumes each data as a cluster and step by step merges different clusters to reach the number of predefined clusters. In the process used for merging two clusters at each stage, two data with the least corresponding distance merge with each other. Two data structures, heap, and K-d tree [17] are used in CURE

to enhance the clustering quality. Heap is used to preserving the distance among clusters and K-d is used to save the points showing the clusters. CURE also uses sampling techniques to accelerate the computations. First, it uses a sample of the dataset and then performs the above-mentioned process on it. Chernoff bound is used to specify the sample volume. The volume sample is high if the data is massive and the repetition of sampling consumes much time. The CURE uses partitioning to accelerate the algorithm. If we assume n as the main dataset and m as the sample dataset, CURE partitions m into p parts and executes clustering in a hierarchical manner on p to reach the termination conditions of the algorithm. Then another clustering algorithm is done on the entire p. Finally, all samples excluded in m are assigned to the closest cluster. The results show that the algorithm execution of CURE is slow but this algorithm is more resistant to noise data.

BKSK

[18] proposes an algorithm based on batches that are clustered in parallel on single machine. The proposed algorithm with a split dataset consists of several stages. The input dataset is divided into batches. Clustering is applied to each batch as a separate dataset. The initial centroids are selected randomly. Each batch is processed in parallel until the convergence condition is met. The algorithm minimizes the sum of squared errors for all clusters. As a result, Data partitioning into the batches before clustering and their parallel processing reduce the computation time.

Although the time and place complexity of clustering algorithms is dependent on the number of data in the dataset, on the other hand, the many dimensions of data is another crucial challenge. In fact, more data dimensions are equal to more features which results in longer algorithm execution time. Sampling techniques decrease the volume of the dataset but have no effect on dimension decrease.

Locality-preserving projection

In locality-preserving projection method, it's essential that after projecting data with many dimensions into fewer dimensions, the distance among points still remains and these dimension reductions have no effect on points distance and do not harm data generality. So data distance in reduced dimension space should be an optimal approximation of points distance in the primary dimension space. Random projection is done by data is a linear transformation matrix that contains the primary data of the dataset. If matrix R considered as a rotation matrix and $d \times t (t << d)$ (which d is the number of R dimensions and t is the number of projected matrix dimensions), and each cell of R as $R(i, j)$ is a random variable then $A^{'} = A.R$ is the matrix projection of A with t dimensions. Creating a rotation matrix is different from other projection algorithms. The rotation matrix is created by random values that have a normal distribution with mean 0 and variance 1. Of course, this is just one of the available methods to create the aforementioned matrix. Clustering applies after transformation and projection of matrix A to $A^{'}$. [19] and [20] are some examples of implementing this algorithm and recently are presented in method [21] to reduce the execution time and improve its function.

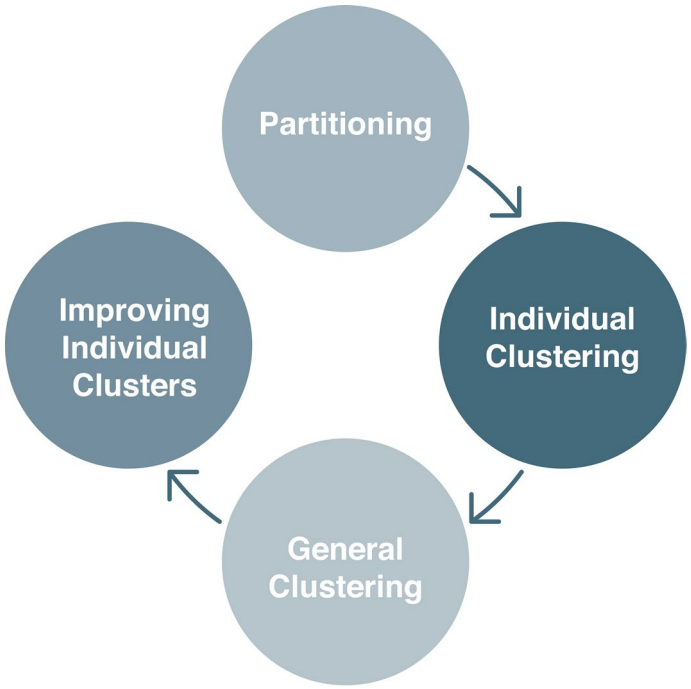

Fig. 3 General procedure of multi-machine clustering algorithms [55]

Global projection

In global projection, the main goal for the projected data is to be very close to the main data, but in local projection, the goal is that points in initial space and the projection space have a good approximation from each other. In other words, if the data primary matrix is A and the approximation matrix is $A^{'}$, in global projection the goal is that the amount of $\left\|A^{'}-A\right\|$ is minimized. Singular value decomposition (SVD) [22], CX/CUR [23], CMD [23] and Colibri [24] are among global projection algorithms.

Multi-machine clustering techniques

Sampling and reduction methods improve the performance of clustering algorithms on big datasets, But due to the growing data, these methods are no longer effective. That's why we need to look for algorithms that can be executed non-centrally like parallel and distributed paradigms.

Generally, non-centralized clustering algorithms can be categorized in to parallel paradigm-based and MapReduce-based algorithms.

In parallel clustering [25], developer are concerned with not only parallelization challenges but also with data partition challenges. The difference between parallel paradigm and MapReduce paradigm is the comfort for the developers. That is, in MapReduce paradigm all data partitioning and data transitions among machine are done by the system. This feature improves parallelization and reliability.

The overall procedure of non-centralized clustering algorithms is shown in Fig. 3, In the first step the data is segmented among different machines. Then, each machine starts clustering its dataset. The main challenges here are minimizing the data transfer traffic

among machines, and lower accuracy compared to sequential model. The lower accuracy happens because of the following reasons: (1) different machines can execute different clustering algorithms, and (2) the manner of data segmentation may changes the final algorithm results.

In parallel algorithms, the data is in a shared storage. By accessing this shared storage, different processors pick up data pieces and process them. On the other hand, in distributed algorithms, data is distributed across multiple physical machines, and each physical machine performs the necessary processing on its own data. In other words, in parallel algorithms, processing is parallelized, but in distributed algorithms, both data and processing are parallelized.

Parallel clustering methods

parallel clustering algorithm are complex for the developers, they are still available solution when volume data increases. In this subsection some of these algorithms are discussed in the following:

DBDC [26, 27] is a parallel density based [28] algorithm. In density based algorithms, the main goal is identifying clusters from the data shape. The density of points inside the cluster are much more than outside of the cluster. In additions, the density of noise points is much less than the density of clusters. here clustering performs locally and globally. For local clustering a default algorithms is used. For global clustering a density based algorithm named DBSCAN is used to finalize the results [29]. The results show that DBSCAN preserves the quality of clustering. Its execution time is 30 times faster than the sequential version of it.

parMETIS [30] is the parallelized version of the METIS [31]. parMETIS is the graph clustering and partitioning algorithms. The parMETIS algorithm is implemented in 3 general steps. In the first step, a general subgraph of the data is extracted. This subgraph is selected according to the degree of vertices. In the next step, the initial division is performed and the initial subgraph is clustered. And in the third step, the vertices of each subgraph are navigated using the breadth-first traversal algorithm. Finally, these three phases clustered the main graph into several clusters(partitions).

The new topic in parallel computation is to use GPU instead of CPU to increase computation speed, because GPU is capable of computing millions or even billions calculations in only one second. G-DBSCAN is the distributed and GPU based clustering algorithm and it is DBSCAN density based. This algorithm is one of the newest methods presented. G-DBSCAN [29] has two main phases which are both parallelized: (1) Making graphs: each data is a vertex and when the distance between two data is less than a predefined quantity, edge is formed between them. (2) Determining clusters: by the use of Breadth First Search (BFS) algorithm which is made on the graph in previous step. The results show that G-DBSCAN is 112 times faster than its sequential version.

MapReduce clustering algorithms

Although parallel clustering algorithms improve scalability and efficiency. However, complexity and difficulty of storage and processor distribution among different machines

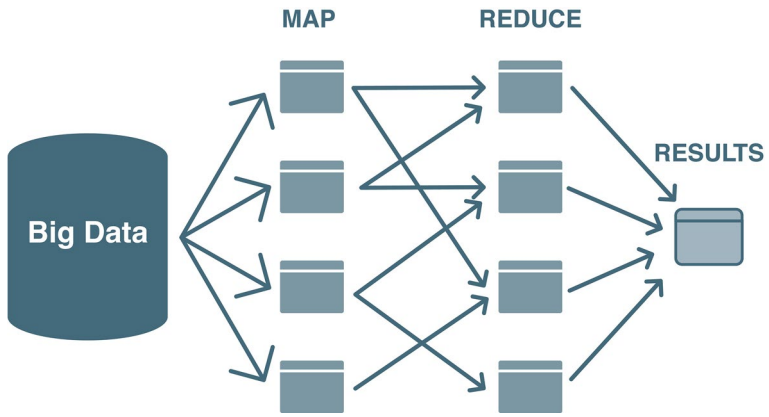

Fig. 4 Architecture of Apache Hadoop framework

still remains as the main challenge. To address this issue, the MapReduce framework was released. In this section, we review algorithms that use MapReduce paradigm and architecture. Figure 4 illustrates the architecture of MapReduce framework. This framework used to solve massive computational problems on large scale data in distributed computing environments.

The algorithms discussed in this section are assests according to the following three aspects: (1) Speed up: It is defined as the ratio of application's execution time on a single processor to the execution time, of the same workload, on a system composed on P processors. (2) Scale up: This is measured as the ratio of a system with X times larger is capable of doing a job with X times larger in the same exact time, and (3) Size up: Is the ratio of the volume of data, on the execution time.

PKMeans [32] is the distributed version of K-Means [33] clustering algorithm. The main goal of the K-Means algorithm is clustering the set of data to K clusters such as data in each cluster have the most similarity with each other, while data in different clusters have the most difference with each other. This algorithm, first selects K data randomly and repeats the following two phases consequently: assigning each data to the nearest cluster, And updating cluster centers with the mean of data in each cluster.

PKMeans distributes computations among different machines to increase speed up and scale up. Initial clustering is done in the map phase and the final clustering in the reduce phase. PKMeans has linear size up and speed up. It has the scale up of 0.75 for four machines. Moreover, PKMeans has the same quality as the sequential version of it.

MR-DBSCAN [34] algorithm is the MapReduce based version of DBSCAN algorithm. The main challenges in the parallelized DBSCAN are failure in load balance among machines, and failure in scale up; because most of its essential functions cannot be parallelized. In MR-DBSCAN, a new mechanism for partitioning and segmentation of computations has been considered, So that most essential functions could be parallelized. Experiments on large datasets have showed its efficiency and scale up.

In [35], an algorithm is presented to cluster big datasets that the volume transferred among machines has been reduced. This algorithm is divided into three phases: (1) In each machine, data clusters are searched by Canopy algorithm [36] to identify the best clusters. Because data shapes may not be spherical, instead of considering one point

as the symbol of the clustering, several points are used to support the shape complexity of data. Mahalanobis distance [37] is used to specify these points. Then, only these points are transferred among machines for final clustering. (2) A weighted clustering is performed over the points specified in the previous phase in different machines. (3) A bayesian classification algorithm [37] is used to calculate the probability that a points belongs to a cluster.

Possibility C-Means algorithm (PCM) [38] is one of the known clustering algorithms in data mining. A weighted PCM algorithm (wkPCM) is presented in [39] for clustering massive data distributedly using MapReduce. Experiments showed this algorithm is capable of clustering big data in an acceptable time. Both time and spatial complexities for this algorithm is $O(n^2)$.

In [40] the iterative clustering K-Means [41] (IKM) is presented that uses MapReduce. In the map phase, IKM runs on the loaded data segmentation, then, in the reduce phase, IKM algorithm again runs on the results taken from map phase. It is claimed that dataset needs just one scan. Therefore, the data transfer volume between map and reduce phases is very low.

In [42], improved map-shuffle-reduce version is used to present a clustering algorithm for spatial big datasets. In the map phase, dataset is divided into very small segments and the closest neighbor to each data (one scan) is specified. In the shuffle phase, the results in the previous phase are transferred to the reduce phase, orderly. Finally, in reduce phase, all points inside each cluster are averaged to calculate new cluster centers.

[43] introduces the design and implementation of a density-based clustering algorithm. It present a parallel Shared Nearest Neighbor (SNN) clustering algorithm using the k-dimensional tree (k-d tree) to reduce search time to improve efficiency.

[44] presents a new meta-heuristic based clustering method to solve the big data clustering. It leverages the searching potential of military dog squad to find the optimal centroids and Map-Reduce architecture to handle the big data sets.

Preliminary

Before the proposed method can be described in detail, some concepts have to be discussed.

In order to optimize clustering, the Bee Colony algorithm has been used to increase the accuracy and quality of the clustering. Therefore, the Bee Colony algorithm is briefly described first.

Artificial Bee Colony Algorithm

Swarm Intelligence (SI) is a subset of artificial intelligence that focuses on the collective behavior of decentralized, self-organized systems, both natural and artificial. This is an emerging field of biologically-inspired artificial intelligence based on the behavioral models of social insects such as ants, fishes, bees, wasps, termites, etc.

An Artificial Bee Colony (ABC) [45] algorithms defined by Dervish Karaboga under the larger umbrella of swarm intelligence, motivated by the intelligent behavior of honey bees, which aims to discover food sources with progressively higher amounts of nectar. Compared to the natural swarms of bees, the main components can be somewhat linked to the model replicating honey bee movements.

Food Source: Represented by profitability, whose value depends on the proximity, richness, and how easily it can be extracted.

Employed Bees: Employed bees go to their food source and come back to hive and dance on this area.

Onlooker Bees: Onlookers watch the dances of employed bees and choose food sources depending on dances.

Scout Bees: The employed bee whose food source has been abandoned becomes a scout and starts to search for finding a new food source.

To compare the real honey bee swarms with ABC algorithm, the swarm size can be considered as the search space, food sources represents the solutions. Employee bees and onlooker bees are components in the model that work in the same way as the bees do, the better the food source, the more fit it is for the bee. The number of employed bees is equal to the number of food sources. Each food source is associated with one and only one

Algorithm 1 Artificial Bee Colony Algorithm

```
Initialize the set of possible solutions;
while termination requirements not met do
    Select sites for local search;
    Recruit bees for the selected sites and to evaluate fitness;
    Select the bee with the best fitness as BestSolution;
    Assign the remaining bees to looking for randomly;
    Evaluate the fitness of remaining bees;
    Update probabilities;
end while
return BestSolution
```

employed bee. The general pseudo code of the ABC algorithm is as Algorithm 1 .

The ABC produces a randomly initial population of SN solutions (food sources). Let $X_i = x_{i,1}, x_{i,2}, ..., x_{i,n}$ represent the i^{th} solution, where n is the dimension size. At each iteration of the algorithm, each employed bee Z_i determines a new neighboring food source V_i of its currently associated food source by Eq. 1, and computes the nectar amount of this new food source:

$$V_{ij} = Z_{ij} + \theta_{ij}(Z_{ij} - Z_{kj}) \tag{1}$$

where X_i is a randomly selected candidate solution, j is a random dimension index selected from the set 1, 2, ... , n, and θ_{ij} is a random number between [− 1,1]. If the nectar amount of this new food source, then this employed bee migrates to this new food source, otherwise it continues with the previous one. After all employed bees complete their search, they share the information they collected about their food sources with onlooker bees. An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source with a probability related to its nectar amount by Eq. 2. The probability P_i of selecting a food source i is computed as:

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{2}$$

where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i and SN is the number of food sources which is equal to the number of employed bees. Once all onlookers have selected their food sources, each of them determines a new neighboring food source of its selected food source and

computes its nectar amount. Providing that this amount is higher than that of the previous one, and then the bee memorizes the new position and forgets the old one. The employed bee becomes a scout bee when the food source which is exhausted by the employed and onlooker bees is assigned as abandoned. In other words, if any solution cannot be improved further through a predetermined number of cycles which is called limit parameter, the food source is assigned as an abandoned source and employed bee of that source becomes a scout bee. In that position, scout generates randomly a new solution by Eq. 3. Assume that Z_i is the abandoned source, the scout discovers a new food source which will be replaced with $Z_{i,j}$:

$$Z_{i,j} = Z_{min,j} + rand(0,1)(Z_{max,j} - Z_{min,j}) \tag{3}$$

where j is a random dimension index selected from the set 1, 2, ... , n different from i.

Fuzzy C-means algorithm

In this section, the Fuzzy C-means clustering algorithm will be introduced. The proposed method in this study is an Fuzzy C-meanse(FCM) based clustering for big data. The Fuzzy C-means (FCM) algorithm is a clustering algorithm developed by Bezdek [46]. FCM does not decide the absolute membership of a data point to a given cluster; instead, it calculates the likelihood (i.e., the degree of membership) that a data point will belong to that cluster. In each iteration of the FCM algorithm, the following objective function J is minimized:

$$J = \sum_{i=1}^{N}\sum_{j=1}^{C} \delta_{ij} \left\| x_i - c_j \right\|^2 \tag{4}$$

Here, N is the number of data points, C is the number of clusters, c_j is the center of cluster j, and δ_{ij} is the degree of membership for the ith data point x_i in cluster j. The norm, $\left\| x_i - c_j \right\|$ measures the similarity (or closeness) of the data point x_i to the center vector c_j of cluster j. In each iteration, the algorithm maintains a center vector for each of the clusters. These data points are calculated as the weighted average of the data points, where the weights are given by the degrees of membership. For a given data point x_i, the degree of its membership to cluster j is calculated as follows:

$$\delta_{ij} = \frac{1}{\sum_{k=1}^{C} (\frac{\left\| x_i - c_j \right\|}{\left\| x_i - c_k \right\|})^{\frac{2}{m-1}}} \tag{5}$$

where, m is the fuzziness coefficient and the center vector c_j is calculated as follows:

$$c_j = \frac{\sum_{i=1}^{N} \delta_{ij}^m . x_i}{\sum_{i=1}^{N} \delta_{ij}^m} \tag{6}$$

In Eq. 6 above, δ_{ij} is the value of the degree of membership calculated in the previous iteration. Note that at the start of the algorithm, the degree of membership for data point i to cluster j is initialized with a random value $\theta_{ij}, \theta \leq \theta_{ij} \leq 1$, such that $\sum_j^C \delta_{ij} = 1$.

In Eqs. 5 and 6 the fuzziness coefficient m, where $1 < m < \infty$, measures the tolerance of the required clustering. This value determines how much the clusters can overlap with one another. The higher the value of m, the larger the overlap between clusters.

The required accuracy of the degree of membership determines the number of iterations completed by the FCM algorithm. This measure of accuracy is calculated using the degree of membership from one iteration to the next, taking the largest of these values across all data points considering all of the clusters. If we represent the measure of accuracy between iteration k and $k+1$ with ϵ, we calculate its value as follows:

$$\epsilon = \delta_i^N \delta_j^C \left| \delta_{ij}^{k+1} - \delta_{ij}^k \right| \tag{7}$$

where, δ_{ij}^k and δ_{ij}^{k+1} are respectively the degree of membership at iteration k and $k+1$.

Apache Hbase

HBase is an open-source non-relational distributed database modeled after Google's Bigtable and written in Java. It is developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of HDFS or Alluxio, providing Bigtable-like capabilities for Hadoop.

Apache Hbase is a column-based NoSQL database that uses a MapReduce programming model [47] for processing and running queries. It means Hbase uses a query engine processor that converts any SQL-like query to some MapReduce jobs. Then, provided jobs run on hadoop HDFS file systems (that stores data) and produce the results of each query. Apache Hbase supports SQL-like query syntax and some aggerations on top of columns in the HDFS file system.

Apache Hbase provide a real-time queries that can run on data that stores in the HDFS file system. In the other words, Hbase allows queries for individual records in the column with billion of records [48]. HBase runs on top of Apache Hadoop(it mostly requires only HDFS where it stores the data) and Apache Zookeeper. Apache Zookeeper cluster is used for failure detection of HBase nodes and stores distributed configuration of HBase cluster.

Master Node is responsible for monitoring RegionServers(detecting failures through Zookeeper), assigning regions to RegionServers, region load balancing between RegionServers and cluster metadata handling. HBase cluster typically consists of multiple Masters, one of which is active and other are backup. When all master instances run, each start leader election(by using Zookeeper) to become an active master. Then some instance wins election, others switch to "observer" state and wait until active master will fail(and start new round of election) [49].

Other component of HBase cluster is RegionServer. It is a worker node which is responsible for serving client requests and managing data regions. Tables in HBase consist of rows which are identified by key. Rows are sorted according it's key in data structures inside of HBase. Region is a group of continuous rows defined by start key and end key of rows which belong to it. RegionServer hosts multiple regions of different tables. It's important to note that regions of the same table may be hosted on different servers, e.g. table data is distributed across cluster. But each region is managed by only one RegionServer at a time.

In the next section, the proposed method is given. The method presented in this study is innovative in several respects: using the ABC algorithm to optimize clustering, implementing the combination of the C-Means and the ABC algorithms by the Map-Reduce architecture, and using a distributed column-based database (Apache Hbase) to store intermediate results in Map-Reduce architecture.

Proposed method

To optimize FCM by using ABC, we consider each individual equivalent to a solution in k dimensional space for the clustering problem, k is the number of clusters. Each component in the individual represents the center of each cluster, which is a p-dimensional vector, p is the number of attributes of the data set. In the initialization phase, each component in the Individual, will randomly be between the two minimum and maximum values, this minimum and maximum values will be for the minimum and maximum values of the corresponding attribute in the data set. Each data vector xi is attributed to the nearest center, and then using the Eq. 4, the fitness of each individual is calculated. In fact, fitness function is the same as the objective function in the FCM algorithm, which as a result of implementing the Bee Colony algorithm, the FCM cost function is optimized and a better quality clustering is performed.

Now we need to design and develop the above algorithm in the form of MapReduce architecture. According to the map-reduction program, for mapping or assigning any job or task, distributed map data and mappings must be defined and designed. Due to the fact that it is read and written in HDFS (Hadoop File System), the speed of access to data is slow. In other words, because the HDFS memory is sequential, the speed of reading and writing is very slow. On the other hand, considering that the clustering algorithm is an iterative algorithm with multiple iterations, at the beginning of each iteration, the data is read from the HDFS, and at the end of the iteration, write again in HDFS, in order to get used in the next iteration.

In this study, instead of storing input and output in HDFS, we used Hbase. In fact, we used the Hbase database to speed up data access at the beginning and end of each Iteration. Therefore, especially for high-volume data (over 100 terabytes), the use of Hbase will have a dramatic effect on the speed and efficiency of the algorithm. Each of the map and decrease phases for the proposed algorithm is explained below.

Map phase

The map phase has the task of performing pre-processing on a block of data. This block of data is available to the mapper by the bulk data infrastructure, the Apache Hadoop tool. The pseudo code of the map phase is shown in Algorithm . Given that individuals are stored in HBase, the input data source for each individual mapper of the individual table (the initial population of bees that are randomly generated and stored in HBase) is in Hbase, which passes the mapper. To be in line 2 of the mapper, the data and clusters are extracted from the individual, and in line 3, the value of the fitness or function for this individual and the probable value in line 4 is computed. Given that the output of the phase of the map should be in the form of a key-value pair, in line 5, this algorithm is

the individual key plus the value of the fitness function and its probability as the mapper

Algorithm 2 Map Phase of FCM Bee Colony Clustering Algorithm

```
1: procedure MAPPER(INDIVIDUAL ID)
2:     points, clusters = translate(ID)
3:     fitness = calculate—fitness(points,clusters)
4:     probability = Calculate the probability values for the individual
5:     submit (ID, (fitness,probability))
6: end procedure
```

output.

Reduce phase

The input of reduce phase is the same as the output of the mapping phase. At this point, from the list of entries, the answer that is the highest value of the utility function is chosen. Then a new answer is generated using Eq. (3), which is best replaced if $best_solution$ is better. Pseudo code of reduce phase is as Algorithm .

Algorithm 3 Reduce Phase of FCM Bee Colony Clustering Algorithm

```
1: procedure REDUCER(LIST<ID,(fitness,probability)>)
2:     Select a solution z depending on fitness and probability
3:     Produce new solution z_new and calculate its fitness
4:     If z_new is better than z, replace z with z_new
5:     Submit best solution
6: end procedure
```

Finally, the best answer is sent as output. Therefore, the above-mapping-down structure runs into the number of Bee Colony algorithm implementations, and each stage outputs the mapping phase along with the individuals in Hbase to be used as inputs for the next iteration. Using Hbase makes the input and output read-write process at a much faster rate, and especially in high-volume data, it increases the efficiency and speed of the algorithm. Also, the overall procedure of each iteration is shown in Fig. 5.

Algorithm complexities

The goal here is to measures the complexity of the proposed Map-Reduce algorithm. As there are many different operations in different phases of the algorithm, measuring the complexity of the algorithm is not easy. So, we only consider the most important parts.

In the map phase, two operations are performed. The fitness calculation is performed first and then the probability is calculated. So the time complexity of this phase is equal to time complexity of fitness function(FF) plus time complexity of probability function(PF):

$$O(map) = O(FF) + O(PF) \tag{8}$$

In this study, the fitness and probability functions are the same as 4 and 5. So, the time complexity of the fitness function is $O(n^2)$ and the time complexity of the probability function is $O(n)$. Therefore, the time complexity of the map phase is $O(n^2 \times s \times (1/p))$, where n is the number of data points, s is the number of data node machines and p is the ping time between physical nodes in Hadoop clusters.

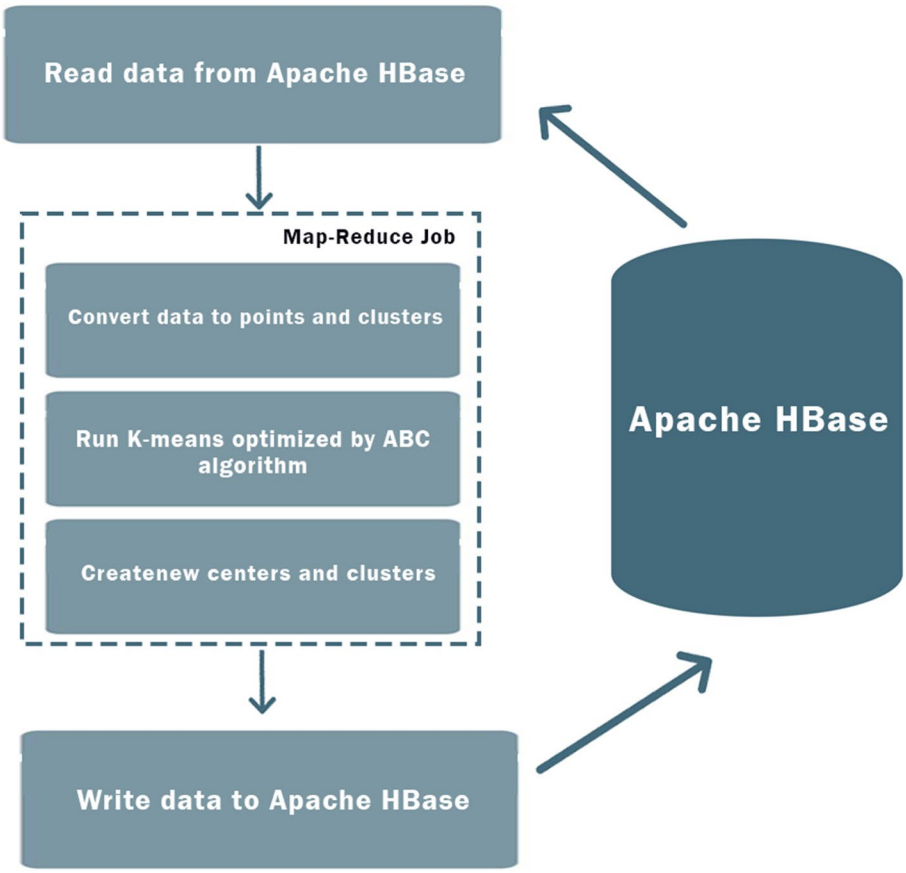

Fig. 5 Overall procedure of each iteration of proposed algorithm

On the other hand, two operations are performed in reduce phase, as well. The select operator is applied first, followed by producing the new solution. So, the time complexity of reduce phase is equal to time complexity of select operator(SO) plus time complexity of produce new solution(PNS):

$$O(reduce) = O(SO) + O(PNS) \tag{9}$$

In this research, the time complexity of the select operator is $O(n \times logn)$; because of sorting fitness values. And, the time complexity of production of the new solution is $O(n^2)$; because of probabilities updating. So, the time complexity of reduce phase is $O(n^2 \times s \times (1/p))$.

The important point is that in addition to the map and reduce phases, other operations are performed in a MapReduce algorithm, such as shuffling. As shuffling only include sorting in our implementation, it is the time complexity is equal to:

$$O(sorting) = O(n \times logn \times s \times (1/p)) \tag{10}$$

So, the total time complexity of our MapReduce algorithm is equal to:

Table 1 Classification of size of data [9]

			Big Data		
Bytes	10^6	10^8	10^{10}	10^{12}	$10^{>12}$
Size	Medium	Large	Huge	Monster	Very Large

Table 2 Cluster physical machine specifications

Operating system	Processor	Main memory	Hard disk	Band width
Ubuntu 14.10 64-bit	Intel Core i7 4.3 GHz	16 GB	1 TB	100 Mbit/s

$$O(MapReduce) = M \times (O(map) + O(sorting) + O(reduce)) \tag{11}$$

where M is the number of FCM iterations. Another parameter that is measured for MapReduce jobs is the calculation of main memory usage. The total memory needed by all reducers is $O(s)$ where s is the number of data node machines, and also the memory needed for each reducer is $O(1)$.

According to the above calculations, as the number of Mappers and Reducers increases, the execution time of the algorithm also decreases linearly. Therefore, the proposed algorithm is designed in full accordance with the scalability structure.It means, the algorithm execution time can be decreased by increasing the number of physical machines. So, the speed up of our algorithm is linear (Table 1).

Results

To implement this method we setup a cluster of machines with 6 nodes. The specifications of each of the physical machines are shown in Table 2. Of the six physical machines, a machine is named NameNode and five other machines are considered as DataNode. Apache Hadoop Version 2.5 and Apache Hbase version 1.2 are installed on this clone, and the programming language used is also python.

Datasets

To evaluate and calculate the accuracy and efficiency of the proposed clustering algorithm we use 3 datasets as follows:

Vehicle_sensIT data set is the one from wireless distributed sensor networks (WDSN). It utilizes two different sensors, that is, acoustic and seismic sensor to record different signals and do classification for three types of vehicle in an intelligent transportation system. We download the processed data from LIBSVM [50] and randomly sample 100 data for each class. Therefore, we have 300 data samples, 2 views and 3 classes (Table 3).

Caltech101 data set is an object recognition data set containing 8677 images, belonging to 101 categories. We chose the widely used 7 classes, i.e. Faces, Motorbikes, Dolla-Bill, Garfield, Snoopy, Stop-Sign and Windsor-Chair [51].

MSRC-v1 data set is a scene recognition data set containing 8 classes, 240 images in total. We select 7 classes composed of tree, building, airplane, cow, face, car, bicycle and

Table 3 Summary of the datasets

Dataset	No. of classes	References
Vehicle_sensIT	3	[50]
Caltech101	7	[51]
MSRC-v1	7	[52]

Table 4 Results of clustering quality on the Vehicle_sensIT dataset (ARI Score)

Algorithm	class1	class2	class3	Overall
PCM	0.53	0.51	0.65	0.59
wPCM	0.60	0.64	0.70	0.65
HOPCM-15	0.84	0.75	0.89	0.85
HOPCM	0.84	0.79	0.85	0.84
Proposed method	0.85	0.93	0.91	0.89

Table 5 Results of clustering quality on the Caltech101 dataset (ARI Score)

Algorithm	class1	class2	class3	class4	class5	class6	class7	Overall
PCM	0.63	0.61	0.72	0.73	0.70	0.62	0.71	0.71
wPCM	0.67	0.70	0.75	0.79	0.77	0.77	0.77	0.76
HOPCM-15	0.89	0.85	0.92	0.89	0.87	0.91	0.81	0.89
HOPCM	0.89	0.87	0.93	0.90	0.89	0.92	0.87	0.90
Proposed method	0.94	0.92	0.94	0.93	0.91	0.94	0.93	0.92

each class has 30 images. We also extract the same 6 visual features from each image with Caltech101 dataset [52].

Comparisons

The Adjusted Rand Score (ARI) parameter is used to assess the quality of clustering, which in fact specifies the similarity between the two sets (similarity score between − 1.0 and 1.0. Random labelings have an ARI close to 0.0 and 1.0 stands for perfect match). Therefore, the higher the ARI, the greater the accuracy of clustering. The method presented in this study is compared with PCM, wPCM, HOPCM-15 and HOPCM [53] methods. In Tables 4, 5, and 6, the results of the comparison of the method presented in this study with the above methods are presented on the 3 above datasets. As it is known, the proposed method has improved by 3% at least compared to other methods.

We also evaluated the method presented in this article from another aspect. Because we use the Apache Hbase and MapReduce method, the efficiency of the proposed method will be much higher in the high volume of data than in the normal volume data. Therefore, we repeated the first data set Vehicle_sensIT to the desired number to create more data volume to test the performance of our algorithm in large datasets.

For repeat dataset, the oversampling methods are used [54]. Assume the dataset is 1, 2, 3 and you sample $2N$ values out of it and get: 1, 1, 2, 2, 3, 3. Now if you compute

Table 6 Results of clustering quality on the MSRC-v1 dataset (ARI Score)

Algorithm	class1	class2	class3	class4	class5	class6	class7	Overall
PCM	0.53	0.49	0.62	0.65	0.61	0.54	0.63	0.61
wPCM	0.59	0.61	0.68	0.71	0.65	0.69	0.67	0.67
HOPCM-15	0.81	0.74	0.84	0.81	0.78	0.82	0.72	0.80
HOPCM	0.80	0.77	0.82	0.84	0.79	0.83	0.75	0.80
Proposed method	0.84	0.83	0.81	0.80	0.85	0.84	0.79	0.83

Table 7 Execution time (hour) of proposed method on 5 data nodes with huge volumes of data

Size of data(GB)	Read Phase	Map Phase	Reduce Phase	Export Phase	Import Phase	Overall
200	0.6	13.49	8.58	0.2	0.1	22.97
400	1.4	21.17	13.48	0.69	0.15	36.89
600	2.7	30.05	18.49	1.1	0.23	52.57
800	4.6	36.74	25.27	1.8	0.34	68.75
1000	6.9	43.12	31.24	2.9	0.49	84.65

some statistics on it (for example mean), then you will get the same result for both since they contain the same observations appearing with the same probability.

So we separate the Map-Reduce job of our method into the following parts:

Read Phase: Data read from HDFS that runs one time at the beginning of the algorithm.
Map Phase: Map function applies to the data that repeats K times according to FCM algorithm procedure.
Reduce Phase: Reduce function applies to the output of Map Phase that repeats K times too.
Export Phase: The output of the Reduce phase write to the Hbase database that repeats K times too.
Import Phase: The results of previous iteration reads from Hbase database that repeats $K - 1$ times.

So, We analyze the performance of our method in the above parts and with various of data volume from 200 GBs to 1 TBs. The results of this evaluation are shown in the Table 6.

The execution time of each phase that reports in Table 7 is the total of execution time of each phase in k repeats of the algorithm. In our evaluation the k is equal to 20. As can be seen, the amount of execution time does not increase linearly with increasing data volume. On the other hand, since the intermediate results in the algorithm iterations are stored in the Hbase table, the I/O time of the algorithm is very low.

Conclusion

This research offered a method for clustering large amounts of data, while in addition to maintaining the quality and desirability of data clustering, its execution time is also appropriate to run on a large volume of data. The method described in this research is based on honey Bee Colony algorithm and mapping-reduction architecture pattern. In addition, the use of the Hbase database made this method more efficient than other methods, especially in high data volumes. After that, we proposed a workflow engine processing that supports a custom description language which allows user to define a overall procedure of multiple MapReduce jobs. The proposed method was implemented and evaluated on a cluster to determine its performance in a distributed environment. For future work, other heuristic and clustering algorithms can be implemented in an integrated framework.

Acknowledgements
Not applicable.

Authors' contributions
SMR and MK have made equivalent contribution to the conception, analysis of the work. SP has provided professional and scientific advice to improve the quality of work. All authors read, review, and approved the final manuscript.

Funding
The authors declare that they have no funding.

Availability of data and materials
All 3 datasets used are free and everyone can download([50–52]) and use them.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
All of the Authors declare that they do not have any particular competing interest.

Received: 6 January 2021 Accepted: 13 April 2021
Published online: 04 May 2021

References

1. Havens TC, Bezdek JC, Palaniswami M. Scalable single linkage hierarchical clustering for big data. In: 2013 IEEE eighth international conference on intelligent sensors, sensor networks and information processing. IEEE; 2013, pp. 396–401.
2. Cheng X, Dale C, Liu J. Statistics and social network of youtube videos. In: 2008 16th interntional workshop on quality of service. IEEE; 2008, pp. 229–238.
3. Priya V, Vadivel A. User behaviour pattern mining from weblog. Int J Data Warehousing Mining. 2012;8(2):1–22.
4. Taniar D, Rahayu W, Lee V, Daly O. Exception rules in association rule mining. Appl Math Comput. 2008;205(2):735–50.
5. Williams PK, Soares CV, Gilbert JE. A clustering rule based approach for classification problems. Int J Data Warehousing Mining. 2012;8(1):1–23.
6. Meyer FG, Chinrungrueng J. Spatiotemporal clustering of fmri time series in the spectral domain. Med Image Anal. 2005;9(1):51–68.
7. Ernst J, Nau GJ, Bar-Joseph Z. Clustering short time series gene expression data. Bioinformatics. 2005;21(Suppl 1):159–68.
8. Iglesias F, Kastner W. Analysis of similarity measures in times series clustering for the discovery of building energy patterns. Energies. 2013;6(2):579–97.
9. Hathaway RJ, Bezdek JC. Extending fuzzy and probabilistic clustering to very large data sets. Comput Stat Data Anal. 2006;51(1):215–34.
10. McAfee A, Brynjolfsson E, Davenport TH, Patil D, Barton D. Big data: the management revolution. Harvard Bus Rev. 2012;90(10):60–8.
11. Dean J, Ghemawat S. Mapreduce: a flexible data processing tool. Commun ACM. 2010;53(1):72–7.

12. Ng RT, Han J. Clarans: a method for clustering objects for spatial data mining. IEEE Trans Knowl Data Eng. 2002;14(5):1003–16.
13. Zhao G-F, Qu G-Q. Analysis and implementation of clara algorithm on clustering. J Shandong Univ Technol. 2006;2:45–8.
14. Kaufman L, Rousseeuw PJ. Partitioning around medoids (program pam). Finding groups in data: an introduction to cluster analysis, vol. 344. Hoboken: Wiley; 1990. p. 68–125.
15. Zhang T, Ramakrishnan R, Livny M. Birch: an efficient data clustering method for very large databases. ACM Sigmod Rec. 1996;25(2):103–14.
16. Guha S, Rastogi R, Shim K. Cure: an efficient clustering algorithm for large databases. ACM Sigmod Rec. 1998;27(2):73–84.
17. Foley T, Sugerman J. Kd-tree acceleration structures for a gpu raytracer. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on graphics hardware; 2005, pp. 15–22.
18. Alguliyev RM, Aliguliyev RM, Sukhostat LV. Efficient algorithm for big data clustering on single machine. CAAI Trans Intell Technol. 2020;5(1):9–14.
19. Fern XZ, Brodley CE. Random projection for high dimensional data clustering: a cluster ensemble approach. In: Proceedings of the 20th international conference on machine learning (ICML-03); 2003, pp. 186–193.
20. Boutilier C, Goldszmidt M. Proceedings of the sixteenth conference on uncertainty in artificial intelligence (2000); 2013. arXiv preprint arXiv:1304.3842.
21. Boutsidis C, Zouzias A, Drineas P. Random projections for *k*-means clustering. In: Advances in neural information processing systems; 2010, pp. 298–306.
22. Golub GH, Reinsch C. Singular value decomposition and least squares solutions. In: Linear algebra. Berlin: Springer; 1971, pp. 134–151.
23. Drineas P, Kannan R, Mahoney MW. Fast monte carlo algorithms for matrices iii: computing a compressed approximate matrix decomposition. SIAM J Comput. 2006;36(1):184–206.
24. Sun J, Xie Y, Zhang H, Faloutsos C. Less is more: Compact matrix decomposition for large sparse graphs. In: Proceedings of the 2007 SIAM international conference on data mining. SIAM; 2007, pp. 366–77.
25. Olman V, Mao F, Wu H, Xu Y. Parallel clustering algorithm for large data sets with applications in bioinformatics. IEEE/ACM Trans Comput Biol Bioinform. 2008;6(2):344–52.
26. Januzaj E. Kriegel H-P, Pfeifle M. Dbdc: Density based distributed clustering. In: International conference on extending database technology. Springer; 2004, pp. 88–105.
27. Aggarwal CC, Reddy C. An introduction to cluster analysis; 2013.
28. Kriegel H-P, Kröger P, Sander J, Zimek A. Density-based clustering. Wiley Interdiscipl Rev Data Mining Knowl Discov. 2011;1(3):231–40.
29. Andrade G, Ramos G, Madeira D, Sachetto R, Ferreira R, Rocha L. G-dbscan: a gpu accelerated algorithm for density-based clustering. Procedia Comput Sci. 2013;18:369–78.
30. Karypis G, Kumar V. Parallel multilevel series k-way partitioning scheme for irregular graphs. Siam Rev. 1999;41(2):278–300.
31. Karypis G, Kumar V. Multilevelk-way partitioning scheme for irregular graphs. J Parallel Distrib Comput. 1998;48(1):96–129.
32. Zhao W, Ma H, He Q. Parallel k-means clustering based on mapreduce. In: IEEE international conference on cloud computing. Springer; 2009, pp. 674-9.
33. Mirkin B. Clustering: a data recovery approach. Boca Raton: CRC Press; 2012.
34. He Y, Tan H, Luo W, Feng S, Fan J. Mr-dbscan: a scalable mapreduce-based dbscan algorithm for heavily skewed data. Front Comput Sci. 2014;8(1):83–99.
35. Ma C, Liang X, Ma Y. A succinct distributive big data clustering algorithm based on local-remote coordination. In: 2015 IEEE international conference on systems, man, and cybernetics. IEEE; 2015, pp. 1839–844.
36. Zhang G, Zhang C, Zhang H. Improved k-means algorithm based on density canopy. Knowledge-Based Syst. 2018;145:289–97.
37. De Maesschalck R, Jouan-Rimbaud D, Massart DL. The mahalanobis distance. Chemometri Intell Lab Syst. 2000;50(1):1–18.
38. Pal NR, Pal K, Keller JM, Bezdek JC. A possibilistic fuzzy c-means clustering algorithm. IEEE Trans Fuzzy Syst. 2005;13(4):517–30.
39. Zhang Q, Chen Z. A weighted kernel possibilistic c-means algorithm based on cloud computing for clustering big data. Int J Commun Syst. 2014;27(9):1378–91.
40. Nguyen CD, Nguyen DT, Pham V-H. Parallel two-phase k-means. In: International conference on computational science and its applications. Springer; 2013, pp. 224–31.
41. Pham DT, Dimov SS, Nguyen C. An incremental k-means algorithm. Proc Inst Mech Eng Part C J Mech Eng Sci. 2004;218(7):783–95.
42. Hu C, Kang X, Luo N, Zhao Q. Parallel clustering of big data of spatio-temporal trajectory. In: 2015 11th international conference on natural computation (ICNC). IEEE; 2015, pp. 769–774.
43. Pokhrel AR, Wang S. Design of fast and scalable clustering algorithm on spark. In: Proceedings of the 2020 4th international conference on cloud and big data computing; 2020, pp. 43–7.
44. Tripathi AK, Sharma K, Bala M, Kumar A, Menon VG, Bashir AK. A parallel military-dog-based algorithm for clustering big data in cognitive industrial internet of things. IEEE Trans Ind Inform. 2020;17(3):2134–42.
45. Karaboga D. Artificial bee colony algorithm. Scholarpedia. 2010;5(3):6915.
46. Bezdek JC, Ehrlich R, Full W. Fcm: The fuzzy c-means clustering algorithm. Comput Geosci. 1984;10(2–3):191–203.
47. Lämmel R. Google mapreduce programming model revisited. Sci Comput Program. 2008;70(1):1–30.
48. Eadline D. Hadoop fundamentals livelessons (video training). Boston: Addison-Wesley Professional; 2013.
49. Uzunkaya C, Ensari T, Kavurucu Y. Hadoop ecosystem and its analysis on tweets. Procedia-Soc Behav Sci. 2015;195:1890–7.
50. Chang C-C, Lin C-J. Libsvm: a library for support vector machines. ACM Trans Intell Syst Technol. 2011;2(3):1–27.

51. Griffin G, Holub A, Perona P. Caltech-256 object category dataset; 2007.
52. Winn J, Jojic N. Locus: learning object classes with unsupervised segmentation. In: Tenth IEEE international conference on computer vision (ICCV'05) Volume 1, vol. 1. IEEE; 2005, pp. 756–63.
53. Zhang Q, Yang LT, Chen Z, Li P. Pphopcm: privacy-preserving high-order possibilistic c-means algorithm for big data clustering with cloud computing. IEEE Transactions on Big Data; 2017.
54. Gosain A, Sardana S. Handling class imbalance problem using oversampling techniques: a review. In: 2017 international conference on advances in computing, communications and informatics (ICACCI). IEEE; 2017, pp. 79– 85.
55. Shirkhorshidi AS, Aghabozorgi S, Wah TY, Herawan T. Big data clustering: a review. In: International conference on computational science and its applications. Springer; 2014, pp. 707–20.

Publisher's Note