

Big Data Clustering Techniques: Recent Advances and Survey



Ibrahim Hayatu Hassan, Mohammed Abdullahi,
and Barroon Isma'eel Ahmad

Abstract Clustering as an unsupervised machine learning technique has appeared as a great learning method to examine correctly the huge volume of dataset produced by today's applications. There is a huge amount of information in the field of clustering and a lot of attempts have already been made to identify and evaluate it for a wide number of applications, but the major problems with the application of classical clustering algorithm for big data analysis are its high complicity, massive volume, variety, and generation rate. So, the classical clustering methods are becoming increasingly inept in processing such data. This poses exciting challenges for researchers to develop modern scalable and efficient methods of clustering that can extract useful information from these vast amounts of data produced in different areas of life. Here in this paper we present a classification for the review of big data clustering algorithms by identifying major research subjects. We then present an up-to-date review of the study works within each of the subject.

Keywords Big data · Clustering · MapReduce · Spark · MPI · GPU

1 Introduction

Clustering is an unsupervised machine learning method used to divide unlabeled dataset into groups called clusters, which comprise data elements that are dissimilar from those in other groups and similar to each other in the same cluster [1]. This has been considered as an important technique for extracting knowledge from databases and has been widely applied to a variety of scientific and business areas [2]. In the

H. Ibrahim Hayatu
Institute for Agricultural Research, Ahmadu Bello University, Zaria, Nigeria
e-mail: ibrogo@gmail.com

H. Ibrahim Hayatu · A. Mohammed (✉) · A. Barroon Isma'eel
Department of Computer Science, Ahmadu Bello University, Zaria, Nigeria
e-mail: abdullahilwafu@abu.edu.ng

A. Barroon Isma'eel
e-mail: barroonia@gmail.com

present digital age, based on the enormous advancement and growth of the internet and web technologies, the data produced by machines and devices has reached a huge volume and is expected to increase in the coming years [3]. Data has become the most integral part of the knowledge-driven society and the economy of today. The goal of big data clustering is to summarize, segment, and group the enormous volumes and data diversities generated in groups of similar items at an accelerated rate. In exploratory data analysis, this has turned out to be one of the most important methods. Sadly, the classical clustering methods are becoming increasingly ineffective in the processing of these data due to their high complicity, large volume, variety, and generation rate. This poses exciting opportunities for researchers to develop scalable and efficient methods of clustering that can extract useful information from these vast amounts of data produced in different areas of life [4, 5]. Therefore, the proposed paper presents an up-to-date review of the research works presented to handle big data clustering. The remainder of this paper is structured as follows: Sect. 2 includes a summary of the different approaches to clustering. Section 3 provides an overview of different big data clustering approaches. Section 4 describes numerous applications in the real world that used specific types of the big data clustering. Finally, the paper is summarized in Sect. 5.

2 Clustering Techniques

A number of clustering algorithms have been proposed in the literature. These algorithms are typically grouped according to the basic principle that the clustering algorithm is based on [6]. That leads to the following groups being identified as shown in Fig. 1.

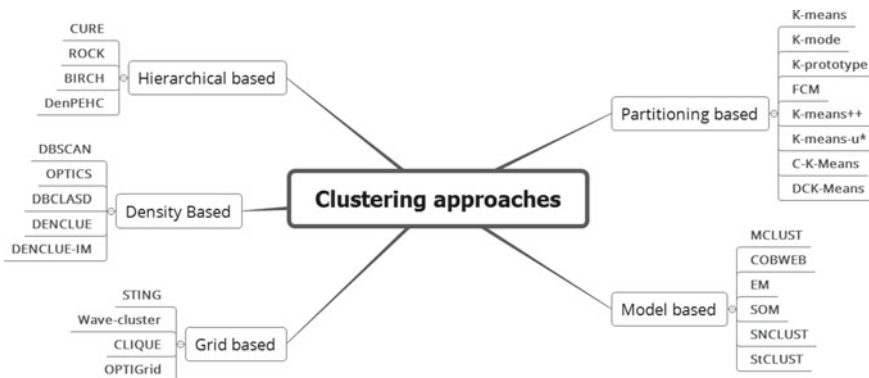


Fig. 1 Grouping of clustering approaches

2.1 Partitioning Methods

Clustering partitioning algorithms organize items into a given number of groups by maximizing a specific objection function capturing the grouping structure. Most partitioning algorithms start via an initial assignment and then use an iterative method which transfers items from cluster to cluster to maximize the objective function. Convergence is therefore local and the best global solution that cannot be guaranteed. These methods required that the number of clusters should be known at the initial stage, and this is normally set by the user [6]. Most of the research in clustering comes from the family of partitioning algorithms. K-means [7], k-mode [8], k-prototype [9], and fuzzy c-means [10] are the examples of the partitioning methods.

2.2 Hierarchical Clustering Methods

Hierarchical clustering techniques build a clusters hierarchy in a top-down approach (Divisive) or using bottom-up approach (Agglomerative) [11]. All these approaches rely on building a similarity matrix between all of the data items; the choice of the metric for the construction of the matrix may affect the shape of the clusters. Examples of common similarity metric that can be employed are: cosine and Jaccard distance metric. Linkage criteria that calculate the distance between groups of objects is a function of the similar distance between objects. But the problem with hierarchical clustering algorithm is that they have large time complexity " $O(n^3)$ " and needs " $O(n^2)$ " space, for which n represents the number of data items. It can be never undone once the merge or the split step is completed. CURE [12], ROCK [13], and BIRCH [14] are some of the well-known hierarchical clustering algorithms.

2.3 Density-Based Methods

Data items are clustered in density-based method on the basis of their boundary, connectivity, and regions of density. A cluster described as a dense component that is connected grows in any direction to which density leads. Such methods are related to neighbours closest to the point. Density-based algorithms are capable of identifying clusters of arbitrary types and providing natural immunity to outliers [15]. Therefore, the total density of a particular point is examined to evaluate the features of the dataset that influence a particular data point. DBSCAN [16], OPTICS [17], DBCLASD [18], and DENCLUE [19] are the most widely used density-based clustering algorithms.

2.4 *Grid-Based Algorithms*

The data item space in grid-based technique is split into grids. The key benefit for this method was its great ability to process, and this is because it runs once over the dataset to measure the statistical grid values. Grid-based clustering is performed on the grid, instead of clustering directly from the database. Quality is heavily dependent on the grid size, which is usually smaller than that of the database. However, the use of a single uniform grid might be insufficient for highly irregular data distributions to find a reliable clustering performance or to meet the time limitation. STING [20], CLIQUE [21], OptiGrid [22], and Wave-Cluster [23] are the common examples of this approach.

2.5 *Model-Based Methods*

Model-based techniques attempt to adjust the fit between some predetermined mathematical formula and data based on the idea that data is created by a combination of simple distributions of probability. This results in the automated detection of number of clusters based on standard statistics, bringing outliers into perception, and thereby building a reliable clustering strategy. Statistical and neural networks are the two major approaches on the model-based techniques. MCLUST by Fraley and Raftery [24] is regarded as the most common model-based algorithm but there are other algorithms like COBWEB [25] (conceptual clustering), EM [26], and neural network methods like self-organizing feature maps [27]. Neural network system uses a combination of linked input/output units with each connection's weight associated. Neural network uses several attributes for clustering which made them famous. The statistical methods employ probability procedures when evaluating clusters. Probabilistic definitions are usually used to describe any idea derived from them [15].

3 **Big Data Clustering Approaches**

Based on the combination of conventional clustering and acceleration techniques, many approaches have been proposed in the literature to manage huge data. This approach focuses on increasing the speed of the clustering process by decreasing the complicity of the computation. On the basis of the acceleration method that is used to enhance the scalability, big data clustering methods can be classified into parallel techniques, data reduction-based techniques, and centre reduction-based techniques as shown in Fig. 2.

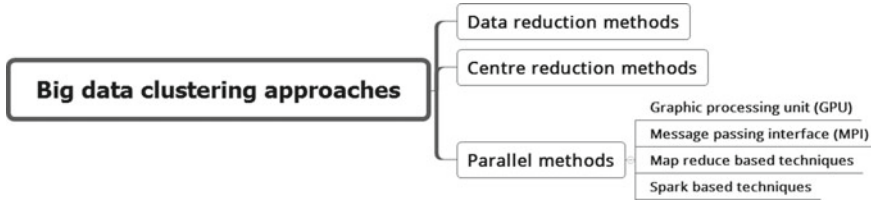


Fig. 2 An overview of big data clustering

3.1 Data Reduction-Based Methods

This technique attempts to reduce the number of data items during cluster development in order to hasten the clustering technique. The MinBatch k-means method (MBKM) was introduced by Sculley [28]. Its basic idea is the use of random small split of fixed size data items which can be placed in memory. This technique is based on the fact that random lots appear to be noisier than single data points. Alternative to k-means for massive-scale data processing named recursive partition k-means (RPKM) has been introduced by Capó et al. [29]. The idea of this technique is to estimate the k-means for the entire dataset by attempting the iterative application of a weighted version of k-means over a small number of data subsets. RPKM’s key steps are defined as follows: first, the data are divided into a number of sets of data in which each set is defined by a member and their corresponding weight. Secondly, over the collection of members, weighted variant of k-means is used.

3.2 Centre-Based Reduction Methods

This technique tries to decrease the amount of evaluations when searching for the nearby cluster centres, which are the most time-consuming step. Kanungo et al. [30] proposed the use of kd-tree structure to speed up k-means (KdtKM). kd-Tree is defined as a binary tree which divides the data space constructed by piercing hyperplanes. The k in kd-tree signify data dimension. Chen et al. [31] proposed a fast density peak clustering for large-scale data based on kNN. Density peak (DPeak) clustering algorithm is not valid for large-scale data, due to two quantities, i.e., both of which are acquired with complexity $O(n^2)$ by brute force algorithm. Hence, a simple but fast DPeak was proposed, namely FastDPeak, which runs in projected time in the intrinsic dimensionality around $O(n \log(n))$. This replaces density with kNN-density, which is determined by fast kNN algorithms such as cover tree, resulting in enormous density calculation improvements. Based on kNN-density, local density peaks and non-local density peaks are defined, and a fast algorithm is also proposed with complexity $O(n)$, which uses two different strategies to calculate for them. Experimental tests show FastDPeak to be successful and to outperform other DPeak variants.

3.3 *Parallel Techniques*

Parallelization technique is among the most frequently used method to reduce the time complexity of traditional clustering techniques. Parallelization is described as a method that splits the task into chunks and executes them in parallel. In the literature, many parallel clustering techniques were proposed. Various frameworks like graphics processing unit [32], message passing interface [33], MapReduce [34], or Spark [35] may be used for parallelization. In this part, we review various research works on parallel clustering techniques with focus on such frameworks.

Graphical Processing Unit-Based Techniques (GPU). GPU is a customized technology built to speed up graphical activities like photo and video formatting. A parallel programming paradigm called compute unified device architecture (CUDA) was used to shorten the development implementations of parallel GPU without going into the details of the technology. Particularly in comparison to a central processing unit (CPU), GPU consists of a huge number of processing cores. It also offers two parallelization stages. GPU will have many multiprocessors on the initial stage; then at the next stage every multiprocessor will also have several streaming processors. Having followed this layout, the GPU application is split into chunks running on streaming processors, and then these chunks are gathered to form chunks blocks running on a multiprocessor [32]. Compared with the few terabytes of RAM now allowed in servers, the comparatively small amount of video random access memory (VRAM) on a GPU card has prompted some to assume that GPU acceleration is restricted to applications with “small data.” But, the assumption lacks two common practices in applications of “big data.” One is that processing a whole dataset at once is rarely needed to achieve the desired effect. GPU VRAM, system RAM, and storage [direct attached storage (DAS), storage area networks (SAN), network-attached storage (NAS), etc.] can provide virtually unlimited large data workload volume. For machine learning, for instance, the training data can be streamed from memory or storage as required. Live streams of data coming from the internet of things (IoT) or other applications such as the Kafka or Spark may also be consumed in a common “incremental” method. The second technique is being able to scale up and out GPU-accelerated configurations. Numerous GPU cards can be cited in a single server, and several servers can be organized in a cluster. This scaling resulted in more cores and more memory, all functioning concurrently and enormously in parallel to process data at unparalleled speed. Therefore, the only real limit to GPU acceleration’s potential computing power is the budget. However, whatever the budget available, it will still be possible for a GPU to speed up configuration and produce more flops per dollar because CPUs are and will remain much more costly than GPU. The GPU database offers a simple and potentially significant price/performance advantage in a single server or cluster [36]. To quicken data clustering via GPU, several works have been proposed, such as the work of Che et al. [37] that introduced a k-means algorithm based on GPU (GPUKM). First, the initial centroid is posted into the GPU’s shared memory and the input data is split and send to each of the multiprocessor. And from there, the distance from each corresponding data item is computed

by each multiprocessor and allocated to the nearest cluster. Depending on the subset of the data items, a local cluster centroid is computed. The central processing unit (CPU) updates the new cluster centroids after each data items is allocated to cluster centroids and submits them to the multiprocessors once more. The above steps will be called multiple times using the same technique until convergence is achieved. The parallel k-means reduces the time complexity of the algorithm. Andrade et al. [38] proposed a GPU-enhanced algorithm for density-based clustering (G-DBSCAN). This is split into two phases: the first phase involves building graph. Each objects represent a node and an edge is created between two objects if the distance is lower than or equal to a predefined thresholds. Once the graph is set, the second phase is the identification of clusters that use breath first search algorithm (BFS) to traverse the graph produced from the first phase. Both phases of the algorithm have been parallelized to achieve better performance. Result indicates that in comparison to the serial implementation, G-DBSCAN is 112 times faster.

A GPU fuzzy c-means technique (GPUFCM) was proposed by Al-Ayyoub et al. [39]. The starting cluster locations are first stored in the shared memory. It then produces from the dataset the starting participation matrix and the starting cluster centroids. After that, by calculating distances, each multiprocessor calculates partial participation. It then calculates the participation values by adding partial participation. The added participation values are transferred from the GPU to the CPU to calculate new cluster centroids from which the algorithm will move to the next iteration. Cuomo et al. [40] suggest an enhanced GPU-based k-means algorithm. The proposed work adopted a parallel processing structure of the graphical processing unit (GPU). The design proposed was optimized to handle CPU space inefficiency and data transmission time for the host device. A parallelize kernel k-means using CPU and GPU was proposed by Baydoun et al. [41]. Kernel k-means involves several computational phases and has additional computational criteria. As a consequence, kernel k-means has not seen the same value and much can still be achieved with respect to its parallelization and stable implementations. Several databases are used, with varying number of features and patterns. The findings show that CUDA typically has the best runtimes with speed-ups ranging from two to more than 200 times over a single-core CPU implementation depending on the dataset used. Shahrezaei and Tavoli [42] studied parallelization of k-means ++ with CUDA. k-means ++ is an algorithm developed to boost the method of locating initial seeds in the algorithm k-means. In this algorithm, the initial seeds are selected consecutively through a probability proportional to the distance from the nearest node. The most critical problem of this algorithm is that it reduces the speed of clustering while running in serial way. This work parallelized the k-means ++ algorithm's most time-consuming steps to increase the runtime of the k-means algorithm while using the k-means ++ seeding technique. In the k-means ++ algorithm, the distance of all points to the selected centre should be determined and the seeds are selected according to a probability formula. Those are the most difficult things for the k-means ++ algorithm being parallelized. This work's parallelization process starts with the partitioning of the points into l different points. Then the measurement of the distance of points to the centres remains parallel until the end of the algorithm. The results of this work

show that the algorithm performs more efficiently over GPU than CPU. It has also been shown that constant memory and texture memory have greater effectiveness in this research than global memory. Even with GPU's certified success in managing massive-scale dataset, it tends to suffer from memory threshold. For instance, it may not handle terabyte data with a bound of 12 GB of memory for every GPU. The efficiency of the GPU-based technique goes way down once the size of the data surpasses the GPU memory capacity. Therefore, developers should first configure the memory before implementing programs via GPU. The drawback to this is the small number of usable GPU tools and algorithms.

Message Passing Interface Techniques. MPI is a language-independent parallel computing communication protocol that supports point-to-point and collective communication [43]. High performance, scalability, and portability are MPI targets. MPI is typically the prevalent model used in high-performance computing [44] and provides portability among parallel programs running on distributed memory systems. But the standard presently does not really support fault tolerance [44] because it primarily addresses problems with high performance computing (HPC). Another MPI downside is that leveraging the parallelism of multicore architectures for shared memory multiprocessing, for example, is not ideal for small grain degree of parallelism. Most researchers seek to combine the MPI with other application programming interface (API), such as the OpenMP, to overcome this issue. OpenMP is an API that enables multiplatform multiprocessing programming for shared memory [45] on most processor architecture and operating systems. OpenMP for its high performance is becoming the standard for parallel shared memory computing but it is not suitable for distributed memory system. The combination of these methods therefore allows for two levels of granularity: a small grain parallel with OpenMP and a large grain parallel with MPI. MPI is implemented in the design of master/slave whereby the master transmits tasks to slaves and collects computed outcomes. However, the job of the slave is to accept tasks, process, and then send the output to the master. Many techniques of clustering that use the MPI paradigm were proposed. For example, Kwok et al. [46] proposed a parallel fuzzy c-means algorithm based on the MPI paradigm (MPIFCM). The master first divides the input data into chunks and sends it to slaves. Each slave gets the associated chunk; it updates the membership matrix based on the distance measurement. The master then receives all the information needed for the slaves to measure the new cluster. These steps are repeated until convergence. A comparative study was made between the MPIFCM and the parallel k-means, which shows a similar structure of parallelism between the two algorithms. A particular implementation of the proposed algorithm is used and tested to cluster a large dataset. It is shown that the MPIFCM algorithm has nearly ideal speed-ups for large dataset and performs equally well when demanding more clusters. Experimentally, the scale-up performance regarding the size of the dataset is also shown to be excellent. Zhang et al. [47] developed a k-means (MPIKM) based on MPI. In this technique, the input data is shared among the slaves. After that, the master chooses k data items as the starting centroid and sends them to the slaves. Each one of the slaves then determines the distance between each data object and

the cluster centroids and assigns it to their nearest cluster. The master then receives all the necessary information from the slaves. The entire process is reiterated until convergence. The experimental results based on the seven datasets used have shown that by increasing the dataset the running time of the proposed MPIKM algorithm is slightly lower than that of the standard k-means. The result also shows that for a small-scale dataset, it is not a good choice to use the proposed algorithm because the time of dividing the dataset and assigning the task into each process occupies a certain proportion. Savvas and Sofianidou [48] proposed a new semi-parallel variant of the k-means algorithm for n-dimensional data objects via MPI. The first step of the proposed strategy is for the master node to discover the resources available and to divide the data into equal or nearly equal parts. The master node passes the data subsets to the workers who in turn start applying the original sequential k-means algorithm on it after obtaining the data. The worker nodes move the local centroids and the number of data points allocated to each centroid to the master node after its termination, and their job ends here. The master node calculates the global centroids which apply the weighted arithmetic mean after collecting this information. The master node at first sorts out the centroids and then divides them into k subgroups and determine the global centroids (by the use of weighted arithmetic mean). The centroids generated from the sequential algorithm by the proposed technique use one-dimensional data points. On increasing the size of n-dimensional data points, the technique generates centroids very similar to the original algorithm with similarities varying from 91 to 100%. Shan et al. [49] proposed a new parallel k-means algorithm for high-dimensional text data using both GPU and MPI. The GPU part design of this work focuses on the measure of similarity between text objects, that is, the computation of the matrix. The original complexity of computation is $O(nd)$. In parallel design, it is designed to start n threads, which runs in parallel. Complexity becomes $O(d)$ which greatly reduces runtime of the algorithm. The portion of the MPI divides nodes into one control node and multiple compute nodes. In order to test the runtime of the proposed technique, the algorithm was tested on CPU, GPU, and GPU + MPI platforms. The results show that the use of MPI and GPU can not only increase the performance of the algorithm, but can also be easily potted to the GPU cluster. However, even with the effectiveness of the MPI paradigm for massive-scale data handling, it is affected by fault intolerance limit. MPI seems to have no fault management tool. A machine downfall in the system can make the entire system to be closed down. In MPI-based techniques developers, therefore, must introduce a fault tolerance method in the design of master/slave to handle machine failures.

MapReduce-Based Techniques. MapReduce has recently become one of the most widely used parallel programming paradigms for handling massive-scale data along clusters systems. It is defined for its strong openness, which enables designers to easily and accurately parallelize algorithms. The MapReduce paradigm consists of two parts, which are Map and Reduce. The map component runs the map functions to create a set of intermediate <key/value> pairs that process each <key/value> in parallel. All intermediate values connected to the same intermediate key are grouped together, while the reduce part executes the reduce function to combine

all intermediate values linked to the same intermediate key. The key factors that make MapReduce a good paradigm for handling massive data are the fault tolerance, linear scalability, and simple programming paradigm [34]. A large number of research works have suggested in the literature that clustering method is suitable via MapReduce. For instance, Zhao et al. [50] implemented k-means based on MapReduce (MRKM) in Hadoop framework. This approach divides the input dataset into blocks, where each block is sent to the mapper and each data element is assigned to its closest cluster based on the distance computation by the map function. By computing the mean data item of each cluster, the reduce function updates the new cluster centroids at the reduce phase. Then the new cluster centroids are taken to the Hadoop distributed file system (HDFS) for the next iteration which is to be used by the map function. In conclusion, until convergence, this whole process will continue. This way the time complexity of the k-means algorithm is reduced. Kim et al. [51] proposed an effective clustering algorithm based on density for massive data using MapReduce (DBCURE-MR). Although the conventional density-based methods discover every cluster one after the other, the proposed algorithm discovers multiple clusters together in parallel. The DBCURE parallelization using MapReduce was performed in four stages. First, the matrix of neighbourhood covariances was calculated in parallel for every point. Similarity joins were performed in the second stage to explore all pairs of points, each of which is inside each of their neighbourhood. In the third stage, core clusters were identified. The core clusters were continuously fusing through the last stage to create the final clusters. Experimental results indicate that DBCURE-MR proficiently discovers specific groups and scales well with MapReduce paradigm. Ludwig [52] proposed a MapReduce fuzzy c-means clustering (MRFCM). Two MapReduce jobs were used in this work. First MapReduce job computes the cluster centre matrix and the second MapReduce calculates the distances that will be used to update the matrix. Using this method the time complexity of the fuzzy c-means was reduced in comparison with the serial fuzzy c-means algorithm. Another MapReduce-based k-prototype (MRKP) for clustering massive-scale mixed data was also proposed by Ben HajKacem et al. [53]. Shahrivari and Jalili [54] implemented a single-pass model based on MapReduce termed as (MRK-means) using the re-clustering strategy. Unlike other MapReduce-based k-means implementations, MRK-means needs only to read the dataset once and is therefore multiple times faster than the original k-means. MRK-means time complicity is linear which is smaller than that of iterative k-means. Because of the use of the seeding method, MRK-means also lead to clusters of greater quality. Dongbo et al. [55] also proposed a canopy k-means (CK-means) clustering algorithm based on MapReduce. Two MapReduce are being used in this work. The first MapReduce estimates the number of clusters and centroids to be used by the second MapReduce using a pre-clustering algorithm. The second MapReduce performed the clustering of k-means by dividing the dataset in which each partition is associated with a map function. The map function assigns every data point of the associated segment to the nearest cluster by measuring the distance between the objects and the cluster centres. In the reduction function the cluster centres are then updated by computing the cumulative average of all data points in each cluster. The new centres are then moved to

the HDFS, which the map function can use for the next iteration. The whole process is repeated until convergence, which reduces the time complexity of the algorithm compared to the original k-means algorithm. Nonetheless, the algorithm's immunity to noise is low and the threshold values of the canopy algorithm are very difficult to determine, which has some effect on the result. In the work of Valcarce et al. [56], an implementation of posterior likelihood clustering and recommendation-relevance models based on MapReduce was proposed. In the current environment where the number of recommendations hitting the big data scale is rising day after day, high efficiency estimates are not sufficient. This research addresses one urgent and important critical need for recommendation systems, which is scalability of algorithms. The research adapted these highly effective algorithms to the functional MapReduce paradigm, which was earlier proved as an appropriate tool to allow scalability of the recommender. A good scalability behaviour was achieved with respect to the number of nodes in the MapReduce cluster. To evaluate large datasets, the impulse for today's scenario is to boost the conventional methods. Tripathi et al. [57] suggest an efficient clustering tool called an enhanced grey wolf optimizer (MR-EGWO) based on MapReduce for clustering large-scale datasets. The method implemented a new type of grey wolf optimizer, called enhanced grey wolf optimizer (EGWO), in which grey wolf's hunting strategy is hybridized with binomial crossover and levy flight steps are induced to boost the searching capability for prey. The suggested variant is further used to improve the clustering method. The EGWO's clustering performance is evaluated on seven UCI benchmark datasets and compared to the five current clustering techniques, namely k-means, particle swarm optimization (PSO), gravitational search algorithm (GSA), bat algorithm (BA), and grey wolf optimizer (GWO). The EGWO's convergence behaviour and consistency were validated through the convergence graph and boxplots. Therefore, in the Hadoop context the proposed EGWO is parallelized on the MapReduce model and called MR-EGWO to manage the large-scale datasets. Furthermore, the MR-EGWO's clustering accuracy is also validated in terms of F-measure and compared to four state-of-the-art MapReduce-based algorithms, namely, parallel k-means, parallel K-PSO, MapReduce-based artificial bee colony optimization (MR-ABC), and dynamic frequency-based parallel k-bat algorithm (DFBPKBA). Experimental findings confirm that the proposed technique is promising and effective substitute for the efficient and large-scale data clustering. Classical clustering methods generally represent each cluster in all dimensions without any difference and depend only on an individual dimension of projection as the weight of an attribute ignores relevance among attributes. Pang et al. [58] address these two problems with the use of multiattribute weights using a MapReduce-based subspace clustering algorithm called PUMA. The subspace attribute is developed into the PUMA by calculating a value weight attribute dependent on the probability of co-occurrence of attribute values between different dimensions. PUMA obtains parallel sub-clusters of each computing node corresponding to the respective attribute subspaces. Finally, by applying the hierarchical clustering method, PUMA tests different scale clusters to iteratively combine sub-clusters. PUMA was deployed on an Hadoop cluster of 24-nodes. Findings show that using multiattribute weights with subspace clustering can achieve better clustering accuracy in both the artificial

and the real-world large datasets. The results also show that in terms of number of nodes PUMA achieves great success in terms of extensibility, scalability, and the almost linear speed-up. In addition, experimental findings show that PUMA is realistic, efficient, and functional for expert systems such as automatic abstracting, acquiring information, recommending systems, and disambiguating word sense. While MapReduce tends to be ideal for clustering large-scale data, it tends to suffer from the inefficiency of performing iterative operations [59]. The whole data must be read and written back to the HDFS at each iteration of the technique. Consequently, multiple I/O memory operations occur during every iteration, and this significantly reduces the effectiveness of MapReduce-based technique.

Spark-Based Techniques. Spark is a distributed model of data processing developed to resolve the MapReduce limitations. It has been introduced as part of the Hadoop, and is designed to run with Hadoop, specifically by accessing HDFS data. This paradigm is based on resilient distributed datasets (RDDs) which is a particular form of data structure used to coordinate computations transparently. Spark offers a collection of in-memory operators in addition to the standard MapReduce to analyse data faster in distributed settings [35]. A number of research works on big data clustering with Spark paradigm were proposed. For instance, Zayani et al. [60] recommended the parallelization of overlapping k-means clustering technique based on the Spark paradigm (SOKM). The proposed technique can carry out parallel clustering processes that result to pre-disjoint data segmentation. Current DBSCAN parallel algorithms produce data partitions in which the initial data are usually split into many disjoint partitions. But by the increase in data dimensions, dividing and reorganizing huge dimensional space can take much time. In order to solve this problem, Luo et al. [61] propose a Spark-based parallel DBSCAN method (S-DBSCAN) that can rapidly split the data and combine the results of the clustering. The suggested work is broken down into three phases. The first phase is partitioning the set of data based on the strategy of random samples. The aim is to calculate the number of partitions based on the actual computing nodes, and on this basis, the original random data will operate through a custom random function. The original random data would be exporting to each slice. Slice of data points has approximately the same data volume and is similar to a simple random sampling. When the number of samples each slice extracts is large enough, it has a similar distribution with the original raw data. After the data is split, the local DBSCAN is calculated in parallel, so partial clustering results are produced. At the map task, partial cluster is created. The partial cluster is saved to HDFS as a new RDD, and at the reduced task each partial cluster centroid is computed. At the last point, the results of global clustering were created by fusing the partial cluster. They suggested a merger strategy based on the centroids. The idea is to measure the distance in the same partition between each of the two partial clusters, and use a sorting technique to determine the minimum distance. Moreover, it determines the minimum value by sorting the minimum distance. Thirdly, threshold is set to merge partial clusters. Finally, construct a matrix with a centroid distance

and traverse every element in the matrix. If the distance is below the threshold then add it to the merge queue until each element is visited. The experimental results indicate a large speed-up of the S-DBSCAN. For the same number of nodes, the larger the dataset, the greater the speed-up. This completely illustrates the superior efficacy of the S-DBSCAN algorithm compared to other parallel DBSCAN algorithms when dealing with huge data. Ben HajKacem et al. [62] presented a k-prototypes clustering technique for massive-scale data with mixed attributes (KP-S). An RDD object with the input dataset from the m partitions of the dataset was generated to parallelize the k-prototype algorithm using Spark. To accomplish this, a Spark framework operation `testfile()` was used. The map task selects a partition of the dataset, performs the k-prototypes algorithm on that partition, and releases the intermediate centres extracted and their weights as the output. `MapPartition()` transformation was used in this work which runs the k-prototype algorithm separately on each RDD block. Upon completion of the map phase, a set of intermediate weighted centres is obtained as the map phase output and this collection of centres is released to a single reduce step. The reduction step takes the set of intermediate centres and their weights, performs the k-prototype method on them again, and releases the final centres as the output. `ReduceByKey()` transformation from the Spark framework was used to simplify implementation. When the final cluster centres are established, the closest cluster centre is assigned to each data point. The obtained results show that KP-S always completes many times faster than existing methods. For instance, with 100 clusters on the Poker dataset, the KP-S algorithm can decrease runtime by 94.47 and 85.72% in comparison to k-prototypes and k-prototype using MapReduce (KP-MR), respectively. In all of the analysis, over 95% of the running time is spent in the map process, which indicates that KP-S is actually performed in memory. An effective parallel density peak clustering technique using GraphX was implemented based on Spark technology to decrease high time complexity of density peak clustering method by Liu et al. [63]. This implementation is based on construction of graph, computing the truncated distance, computing the local density and computing the distance from higher density points. Construction of graph consists of three phases. First, importing vertex and edge data stored on HDFS or other file systems separately to vertex RDD and edge RDD, and setting each edge's initial value to a constant. Secondly, measuring the distance of each edge based on a distance measure formula updating the value of each edge by the distance. Thirdly, the combination of vertex RDD and edge RDD in GraphX to form a graph. When the vertex set and edge set are updated, the initial value for each edge is set to 1. When measuring the distance of each edge, the distance is changed to the value of each edge. To reduce the computation load, the truncated distance is measured before calculating the local density. After the truncated distance is measured, the local density and the distance from points of higher density is determined. Experimental result shows that the Spark implementation can improve significantly (10x) compared to MapReduce implementation. K-prototype is an iterative algorithm that requires some iteration for generating best results. Conversely, MapReduce has a significant issue with iterative algorithms. As a result, the entire dataset at each iteration must be loaded into the main memory from the file system. Then, the data must be written to the file system

again after it is processed. Consequently, several I/O disk operations occur during each iteration, and this reduces runtime. To overcome this problem, Ben HajKace et al. [64] proposed a scalable random damping k-prototypes implemented on the Spark paradigm. A reservoir random sampling technique was used for selecting random data sample from the original dataset. The data sampling MapReduce job first generates an RDD object with input data X form by m partitions. In the map phase, each partition is then processed to generate the intermediate data samples. The reduce phase then receives the set of intermediate data samples to produce the final data sample. The map function takes a partition during the map process and applies the random reservoir sampling algorithm to create an intermediate data sample of size r . The map function then releases the intermediate data samples as a single-phase reduce output. The reduce phase gathers the intermediate data samples generated in the map phase to create the final data sample. Upon generating the data sample, the second MapReduce job checks for cluster centres. First, it generates an RDD object with the data sample S form by m partition. Then, each partition is analysed in map phase to generate the intermediate centres. The reduction step afterwards processed the collection of intermediate centres to produce the final clusters. The map function selects a partition during the map process and executes the k-prototypes on that partition to obtain the cluster centres. To achieve good quality, for each obtained centre a weight that signifies the number of assign data points was recorded. The number of allocated data points per cluster centre reflects the centre's importance. At last, the map function releases the intermediate centres extracted and their weights as the output to a single reduce step. In this implementation, `mapPartition(func)` transformation was used separately on every partition of the RDD. The reduce step takes the intermediate centres and their weights, runs the k-prototypes algorithm on them again, and returns the final centres as the output. `ReduceByKey(func)` transformation from the Spark framework was used to simplify the implementation. The experimental results obtained show that the method proposed often finishes many times faster than the classical k-prototypes and the k-prototype based on MapReduce. The method proposed was found to be 14 times faster than the classical k-prototypes and four times faster than the MapReduce-based method. However, it was found that most of the running time was spent on the step of the map which indicates that the proposed method is actually being performed in memory. In the field of agricultural image segmentation, the fuzzy c-means (FCM) algorithm has been commonly used because it provides easy computation and high quality segmentation. However, the sequential FCM is too slow due to its large amount of computation to complete the segmentation task in an appropriate time. Liu et al. [65] suggest a parallel FCM segmentation algorithm based on the Apache Spark for clustering agricultural images. The input image is first converted to the Lab colour space from the RGB colour space and generates point cloud data. Point cloud data is then split and stored in separate computer nodes where the membership degrees of pixel points to separate cluster centres are determined and the cluster centres are modified iteratively in parallel form until the stop condition is met. In the RDD, point cloud data is finally restored after clustering to reconstruct the segmented image. The FCM output is measured on the Spark platform and achieves an average speed of 12.54

on 10 computing nodes. Experimental results show that the Spark-based parallel fuzzy c-means algorithm can achieve substantial speed-up improvements, and the agricultural image test set provides a 128% better performance improvement than the Hadoop-based solution. This work shows that the Spark-based parallel FCM algorithm provides faster segmentation speed for big data on agricultural images and has better scale-up and size-up. Yu et al. [66] presented an effective three-way cluster ensemble approach based on Spark to solve the difficulty of clustering on large-scale data; this methodology has the potential to deal with both soft and hard clustering. A three-way cluster ensemble based on Spark was suggested, inspired by the principle of three-way decisions, and a distributed three-way clustering algorithm for k-means was developed. Also introduced was the concept of cluster unit, which represents the minimum structure of distribution of granularity decided upon by all members of the ensemble. This research also incorporates quantitative tests to determine the relationship between units and between clusters. Eventually, a consensus-based clustering algorithm was suggested, and several three-way decision strategies were developed to assign small cluster units and non-unit objects. The experimental tests using 19 real-world datasets were used with various metrics such as ARI, ACC, NMI and F1-Measure to verify the feasibility of the proposed method. The experimental result shows that the proposed technique can manage large-scale data effectively, and the proposed consensus clustering algorithm has a lower time cost without loss of the consistency of the clusters.

4 Big Data Clustering Applications

Most applications in the real world generate vast amounts of data. Some of these are (but not restricted to) healthcare applications, IoT applications, detection of anomalies. Here we present a list of applications for big data clustering as shown in Fig. 3.



Fig. 3 Taxonomy of big data clustering applications

4.1 Healthcare

Several researchers have used different forms of clustering strategies for big data in occupational medicine [67], acute inflammation [68], and grouping of cancer samples [69]. McParland and Gormley [70] and Mcparland et al. [71] develop a clustering algorithm for studying high-dimensional categorical genotypic and numerical phenotypic data. The research contributes to a greater understanding of metabolic syndrome. Narmadha et al. [72] use a hybrid fuzzy k harmonic means (HFKHM) for the categorization of tumours as benign or malignant. Su et al. [73] also employs a hierarchical clustering to continually group respondents based on similarity structures of 27 different factors of asthma symptoms and 14 different product applications.

4.2 Internet of Things (IoT)

Amini et al. [74] suggested a hybrid clustering algorithm built on density-based clustering method for the IoT streaming data. The proposed method consists of three phases under which the new data point is connected to a grid or embedded into a current mini cluster, the outliers are eliminated and arbitrary shape clusters are formed through the use of an enhanced DBSCAN. A new data clustering framework is used for big sensory data generated by IoT applications by Karyotis et al. [75]. This framework can also be used for community detection and performing more energy-efficient smart-city/building sensing. Clustering algorithms have been widely used in IoT applications such as looking for similar sensing patterns, identifying outliers, and dividing massive real-time behavioural groups [76].

4.3 Anomaly Detection

Fanaee-T and Gama [77] uses an expectation maximization to model an anomaly detection using structural time series for industrial Ethernet congestion into four sections cantered on a method with a specific significance for detection. This model helps to improve the efficacy of identifying irregular and low false alarm levels. A new developed clustering algorithm was also used in the work of Alguliyev et al. [78] for detection of anomalies. Yin et al. [79] used a self-organizing maps to detect anomalous risk in mobile apps

4.4 Social Media

Alsayat and El-Sayed [80] use an improved k-means algorithm to identify communities by clustering posts from huge streams of social data. In the work of Gurusamy et al. [81], k-means algorithm is also used to exploit social network users' attitudes. On the basis of textual similarity, spectral k-means clustering is used to group people in social network by Singh et al. [82]. A varied density-based spatial clustering for twitter data algorithm that extracts clusters from geo tagged twitter posts using heterogeneity in space. The algorithm uses incremental spline interpolation to ascertain various cluster detection search radii [83]. They were successfully tested for event detection in social media using geo tagged twitter post received in the course of a storm in the United States.

5 Discussion

Parallel methods for big data clustering have been categorized in this work into four GPU-based, MPI-based, MapReduce-based, and Spark-based methods groups. But before applying clustering to any of the parallel frameworks, consideration of certain challenges for each of the framework is necessary. GPU suffers from limited memory. If the data size exceeds the GPU memory size, the performance reduces the GPU-based approach significantly. For example, it is not acceptable to handle terabyte data with a maximum of 12 GB of memory per GPU. Moreover, MPI has no fault control function at all. One system failure in the network will cause the entire network to shut down. Practitioners also need to incorporate some kind of fault tolerance system inside the software to resolve mistakes. The MapReduce architecture looks better than MPI as it is defined by a straightforward structure for programming, linear scalability, and tolerance of faults. It is, however, inappropriate to run iterative algorithms, because the entire dataset must be read and written to disks at each iteration and this results in high (I/O) operations. This significantly degrades the performance of MapReduce-based method. Finally, Spark framework is an alternative to MapReduce which is designed to overcome the disk I/O limitations and improve the performance of MapReduce framework. A recent survey outlined the various frameworks for big data analytics and presents the advantages and drawbacks of each of these frameworks based on different metrics such as scalability, data I/O rate, fault tolerance, and iterative task support [84, 85].

While most of the big data partition clustering methods described in this work provide an efficient analysis for large-scale data for users, some parameters have to be calculated before the learning is finished. The methods include configuring the number of clusters in advance, which is not a trivial task in real-life applications where the number of clusters predicted is typically unknown. One could use various model heuristics as a solution to determine the optimal number [86, 87]. For instance, with an increased number of clusters, the user may check various clusters and then take the

clustering that has the best balance between minimizing the objective function and number of clusters. In addition, the partitioning methods must initialize the centres of the clusters. However, initialized centres of high quality are critical, both for the accuracy and performance of traditional clustering methods. In order to overcome this problem, users may use random sampling methods to obtain cluster centres or initialization techniques that exploit the fairly widespread use of good clustering [88–90]. The outcome of the described methods, using centre initialization techniques, often converges to a local optimum of the objective criterion, rather than the global optimum. To resolve this problem, users should combine traditional methods with heuristic techniques to avoid optimum local clustering results [3, 21, 29, 91–93].

6 Challenges and Future Research Work

Despite the growth in these technologies and algorithms to handle big data, there are still challenges such as:

- Scalability and storage issues: Data rises are much higher than existing computing systems. Storage devices are not adequately capable of processing these data [94–97]. A processing system needs to be built which not only addresses the needs of today but also future needs.
- Analytical timeliness: Data value decreases over time. Many applications, such as telecom, insurance, and banking fraud detection, require real-time or near-real-time transactional data processing [94, 95].
- Heterogeneous data representation: Data derived from different sources are of heterogeneous type. It is difficult to store and process unstructured data such as images, videos, and social media data using conventional methods such as SQL. Smartphones are now capturing and exchanging images, audios, and videos at an exponentially higher pace, pushing our brains to further process. However, there is a lack of effective storage and processing of the medium for representing pictures, audios, and videos [94, 95, 98].
- Privacy and security: New devices and technologies like cloud computing provide a gateway to access and to store information for analysis. This integration of IT architectures will pose greater risks to data security and intellectual property. Access to personal information like buying preferences and call detail records will lead to increase in privacy concerns [96, 99]. Researchers have technological infrastructure to access data from any source of data, including social networking sites, for potential use, although users are unaware of the benefits that can be made from the information they post [100]. The disparity between privacy and convenience is unknown to big data researchers.

Based on this review, the following future works have been formulated.

- Hybrid approaches that combine multiple acceleration techniques showed great promise for big data clustering. For the development of a reliable hybrid approach for big data clustering, further work is needed, however.
- Due to the effectiveness of the big data parallel methods, a method for the parallelization of the hybrid methods is highly needed.

7 Conclusion

This paper offers an in-depth analysis of the big data clustering algorithms to guide the selection of the big data algorithms. We also include categorization of various big data clustering research works. At the end, we describe various applications where big data clustering was used. In sum up, though conventional sampling and dimension reduction algorithms are still useful, but they don't have enough power in managing massive quantities of data because even after sampling a petabyte of data, it is still very large and clustering algorithms cannot cluster it, so the future of clustering is related to distributed computing. Although parallel clustering is potentially very useful for clustering, but the complexity of implementing such algorithms is still a challenge.

References

1. Bakr, A.M., Ghanem, N.M., Ismail, M.A.: Efficient incremental density-based algorithm for clustering large datasets. *Alexandria Eng. J.* 1147–1152 (2015)
2. Jia, H., Cheung, Y.-M.: Subspace clustering of categorical and numerical data with an unknown number of clusters. *IEEE Trans. Neural Netw. Learn. Syst.* 1–17 (2017)
3. Benabdellah, A.C., Benghabrit, A., Bouhaddou, I.: A survey of clustering algorithms for an industrial contex. In: *Second International Conference on Intelligent Computing in Data Sciences (ICDS 2018)*, pp. 291–302 (2019)
4. Oussous, A., Benjelloun, F.-Z., Lahcen, A. A., Belfkih, S.: Big data technologies: a survey. *J. King Saud Univ. Comput. Inform. Sci.* 432–448 (2018)
5. Nasraoui, O., Ben N'Cir, C.-E.: *Clustering Methods for Big Data Analytics Techniques, Toolboxes and Applications*. Springer, Switzerland (2019)
6. Sanse, K., Sharma, M.: Clustering methods for big data analysis. *Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET)* 642–648 (2015)
7. MacQueen, J.: Some methods for classification and analysis of mult-variate observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297. University California Press, USA (1967)
8. Khan, S.S., Kant, D.: Computation of initial modes for k-modes clustering algorithm using evidence accumulation. In: *Proceedings 20th International Joint Conference, Artificial Intelligent*, pp. 2784–2789. Kaufmann, USA (2007)
9. Zhaxue, H.: Clustering large data sets with mixed numeric and categorical values. In: *Proceedings of the First Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 21–34 (1997)
10. Bezdek, J.C., Ehrlich, R., Full, W.: FCM: the fuzzy c-means clustering algorithm. *J. Comput. Geosci.* 191–203 (1984)

11. Ahmad, A., Khan, S.S.: Survey of state-of-the-art mixed data clustering algorithms. *Inst. Electr. Electron. Eng. Access* 31883–31902 (2019)
12. Guha, S., Rastogi, R., Shim, K.: CURE: An efficient clustering algorithm for large databases. In: *Proceedings of the ACM SIGMOID Conference on Management of Data*, pp. 73–84. ACM Press (1998)
13. Guha, S., Rastogi, R., Shim, K.: Rock: a robust clustering algorithm for categorical attributes. *Inform. Syst.* **25**(5), 345–366 (2000)
14. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: A New Data Clustering Algorithm and Its Applications, pp. 141–182. Kluwer Academic Publishers. Manufactured in The Netherlands
15. Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Kkalil, I., Zomaya, Y.A.,... Bouras, A. (2014). A survey of clustering algorithms for big data: taxonomy and empirical analysis. *IEEE Trans. Emerg. Top. Comput.* 267–279
16. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: *KDD' 96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231. AAAI Press, Portland (1996)
17. Ankerst, M., Breunig, M.M., Krie, H.-P., Sander, J.: OPTICS: ordering points to identify the clustering structure. In: *ACM SIGMOID International Conference on Management of Data*, pp. 49–60. ACM press
18. Xu, X., Ester, M., Kriegel, H.-P., Sander, J.: A distribution-based clustering algorithm for mining in large spatial databases. In: *Published in the Proceedings of 14th International Conference on Data Engineering (ICDE'98)*. IEEE, Orlando, USA (1998)
19. Hinneburg, A., Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise. *J. Am. Assoc. Artif. Intell.* 58–65 (1998)
20. Wang, W., Yang, J., Muntz, R.R.: STING: a statistical information grid approach to spatial data mining. In: *Proceedings of the 23rd International Conference on Very Large Databases*, pp. 186–195. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997)
21. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, R.: Automatic subspace clustering of high dimensional data for data mining applications. In: *ACM SIGMOID International Conference on Management of Data*, pp. 94–105. ACM Press, Sanfrancisco (1998)
22. Hinneburg, A., Keim, D.A.: Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering. In: *Proceeding of the 25th International Conference on Very Large Databases*, pp. 506–517. Morgan Kaufman, San Francisco, CA, USA (1999)
23. Sheikholeslami, G., Chatterjee, S., Zhang, A.: WaveCluster: a wavelet-based clustering approach for spatial data in very large databases. *VLDB J.* 289–304 (2000)
24. Fraley, C., Raftery, E.: MCLUST: software for model-based cluster and discriminant analysis. Department of Statistics University of Washington, Seattle, WA, USA (1998)
25. Fisher, D.H.: *Knowledge Acquisition Via Incremental Conceptual Clustering*. Machine Learning, pp. 139–172. Kluwer Academic Publishers, Netherlands (1987)
26. Moon, T.K.: The expectation-maximization algorithm. *IEEE Signal Process. Mag.* 47–60 (1996)
27. Kohonen, T.: The self-organizing map. In: *Proceedings of the IEEE*, pp. 1464–1480, Finland (1990)
28. Sculley, D.: Web-scale k-means clustering. In: *Proceedings of the 19th International Conference on World Wide Web*, pp. 1177–1178. ACM, Raleigh, NC, USA (2010)
29. Capó, M., Pérez, A., Lozano, J.A.: An efficient approximation to the K-means clustering for massive data. *Knowl.-Based Syst.* 56–69 (2017)
30. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.* 881–892 (2002)
31. Chen, Y., Hu, X., Fan, W., Shen, L., Zhang, Z., Liu, X.,... Li, H.: Fast density peak clustering for large scale data based on kNN. *Knowl.-Based Syst.* 104824 (2020)
32. Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C.: GPU computing. *Proc. IEEE* 879–889 (2008)

33. Snir, M.: *MPI—The Complete Reference: The MPI Core*. MIT Press, Cambridge (1998)
34. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: 6th Symposium on Operating Systems Design and Implementation, pp. 107–113. ACM Press (2008)
35. Zaharia, M., Chowdhury, M., Franklin, M., Shenker, S., Stoica, I.: Spark: cluster computing with working set. *HotCloud 10* (2010)
36. Eric, M., Roger, B.: *Introduction to GPUs for Data Analytics Advances and Applications for Accelerated Computing*. O'Reilly, USA (2017)
37. Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J.W., Skadron, K.: A performance study of general-purpose applications on graphics processors using CUDA. *J. Parallel Distrib. Comput.* 1370–1380 (2008)
38. 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.* 369–378 (2013)
39. Al-Ayyoub, M., Abu-Dalo, A.M., Jararweh, Y., Jarrah, M., Al Sa'd, M.: A GPU based implementations of the fuzzy C-means algorithms for medical image segmentation. *J. Supercond.* 3149–3162 (2015)
40. Cuomo, S., De Angelis, V., Farina, G., Marcellino, L., Toraldo, G.: A GPU-accelerated parallel K-means algorithm. *Comput. Electr. Eng.* 1–13 (2017)
41. Baydoun, M., Ghaziri, H., Al-Husseini, M.: CPU and GPU parallelized kernel K-means. *J Supercomput.* 3975–3998 (2018)
42. Shahrezaei, M.H., Tavoli, R.: Parallelization of K-means ++ using CUDA. *arXiv* (2019)
43. Sato, K., Moody, A., Mohror, K., Gambelin, T., de Supinski, B.R., Maruyama, N., Matsuoka, S.: Fmi: fault tolerant messaging interface for fast and transparent recovery. In: *IEEE International Parallel and Distributed Processing Symposium*, pp. 1225–1234 (2014)
44. Sur, S., Koop, M.J., Panda, D.K.: High-performance and scalable MPI over infiniband with reduced memory usage: an in-depth performance analysis. In: *CM/IEEE Conference on Super Computing*, p. 105. ACM (2006)
45. D'Urso, P.: Exponential distance-based fuzzy clustering for interval valued data. *Fuzzy Optim. Decis. Making* 51–70 (2017)
46. Kwok, T., Smith, K., Lozano, S., Taniar, D.: Parallel fuzzy c-means clustering for large data sets. In: *Euro-Par 2002 Parallel Processing*, pp. 27–58. Springer, Berlin, Heidelberg (2002)
47. Zhang, J., Wu, G., Hu, X., Li, S., Hao, S.: A parallel K-means Clustering Algorithm with MPI. In: *Fourth International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 60–64. IEEE (2011). <https://doi.org/10.1109/paap.2011.17>
48. Savvas, I.K., Sofianidou, G.N.: A novel near-parallel version of k-means algorithm for n-dimensional data objects using MPI. *Int. J. Grid Utility Comput.* 80–91 (2016)
49. Shan, X., Shen, Y., Wang, Y.: A parallel K-means algorithm for high dimensional text Data. In: *IEEE International Conference on Consumer Electronics, Taiwan* (2018)
50. Zhao, W., Ma, H., He, Q.: Parallel K-means clustering based on MapReduce. In: *Proceedings of Cloud Computing*, pp. 674–679. Springer, Berlin, Heidelberg (2009)
51. Kim, Y., Shim, K., Kim, M.-S., Lee, J.S.: DBCURE-MR: an efficient density-based clustering algorithm for large data using MapReduce. *Inform. Syst.* 15–35 (2014)
52. Ludwig, S.A.: MapReduce-based fuzzy c-means clustering algorithm: implementation and scalability. *Int. J. Mach. Learn. Cybern.* 923–934 (2015). <https://doi.org/10.1007/s13042-015-0367-0>
53. Ben HajKacem, M.A., Ben N'cir, C.-E., Essouss, N.: Parallel K-prototypes for clustering big data. In: *Proceedings of Data Science and Advanced Analytics*, pp. 628–637. Springer (2015)
54. Shahrivari, S., Jalili, S.: Single-pass and linear-time k-means clustering based on MapReduce. *Inform. Syst.* 1–12 (2016)
55. Dongbo, Z., Shou, Y., Xu, J.: An improved parallel K-means algorithm based on MapReduce. *Int. J. Embedded Syst.* 275–282 (2017)
56. Valcarce, D., Parapar, J., Barreiro, Á.: A MapReduce implementation of posterior probability clustering and relevance models for recommendation. *Eng. Appl. Artif. Intell.* 114–124 (2018)

57. Tripathi, A.K., Sharma, K., Bala, M.: A novel clustering method using enhanced grey wolf optimizer and MapReduce. *Big Data Res.* 93–100 (2018)
58. Pang, N., Zhang, J., Zhang, C., Qin, X., Cai, J.: PUMA: parallel subspace clustering of categorical data using multi-attribute weights. *Exp. Syst. Appl.* 233–245 (2019)
59. Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.H., Qiu, J., Fox, G.: Twister: a runtime for iterative MapReduce. In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pp. 810–818. ACM, New York (2010)
60. Zayani, A., Ben N’Cir, C.E., Essoussi, N.: Parallel clustering method for non-disjoint partitioning of large-scale data based on spark framework. In: *Proceedings of IEEE International Conference on Big Data*, pp. 1064–1069. IEEE, Piscataway (2016)
61. Luo, G., Luo, G., Gooch, T.F., Tian, L., Qin, K.: A parallel DBSCAN algorithm based on spark. In: *IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)*, pp. 548–553. IEEE, Atlanta, GA, USA (2016). <https://doi.org/10.1109/bdcloud-socialcom-sustaincom.2016.85>
62. Ben HajKacem, M.A., Ben N’cir, C.E., Essoussi, N.: KP-S: a spark-based design of the K-prototypes clustering for big data. In: *Proceedings of ACS/IEEE International Conference on Computer Systems and Applications*, pp. 1–7. IEEE, Hammamet, Tunisia (2017)
63. Liu, R., Li, X., Du, L., Zhi, S., Wei, M.: Parallel implementation of density peaks clustering algorithm based on spark. *Procedia Comput. Sci.* 442–447 (2017)
64. Ben HajKacem, M.A., Ben N’cir, C.-E., Essoussi, N.: Scalable random sampling K-prototypes using spark. In: *International Conference, DaWaK 2018*, pp. 317–326. Springer, Regensburg, Germany (2018)
65. Liu, B., He, S., He, D., Zhang, Y., Guizani, M.: A spark-based parallel fuzzy c-means segmentation algorithm for agricultural image big data. *IEEE Access* 42169–42180 (2019)
66. Yu, H., Chen, Y., Lingras, P., Wang, G.: A three-way cluster ensemble approach for large-scale data. *Int. J. Approx. Reason.* 32–49 (2019)
67. Saadaoui, F., Bertrand, P. R., Boudet, G., Rouffiac, K., Dutheil, F., Chamoux, A.: A dimensionality reduce clustering methodology for heterogeneous occupational medicine data mining. *IEEE Trans. Nanobiosci.* 707–715 (2015)
68. Pathak, A., Pal, N.R.: Clustering of mixed data by integrating fuzzy, probabilistic and collaborative clustering framework. *Int. J. Fuzzy Syst.* 339–348 (2016)
69. Zainul Abidin, F.N., Westhead, D.R.: Flexible model-based clustering of mixed binary and continuous data: application to genetic regulation and cancer. *Nucl. Acids Res.* 1–11 (2016)
70. McParland, D., Gormley, I.C.: Model based clustering for mixed data: clustMD. *Adv. Data Anal. Classif.* 155–169 (2016). <https://doi.org/10.1007/s11634-016-0238-x>
71. Mcparland, D., Philips, C.M., Brennan, L., Roche, H.M., Gormley, I.C.: Clustering high-dimensional mixed data to uncover sub-phenotypes: joint analysis of phenotypic and genotypic data. *J. Stat. Med.* 4548–4569 (2017)
72. Narmadha, D., Balamurugan, A.A., Sundar, G.N., Priya, S.J.: Survey of clustering algorithms for categorization of patient records in healthcare. *Indian J. Sci. Technol.* 1–5 (2016)
73. Su, F.-C., Friesen, M.C., Humann, M., Stefaniak, A.B., Stanton, M.L., Liang, X.,... Virji, M.A.: Clustering asthma symptoms and cleaning and disinfecting activities and evaluating their associations among healthcare workers. *Int. J. Hygiene Environ. Health.* 1–11 (2019)
74. Amini, A., Saboohi, H., YingWah, T., Herawan, T.: A fast density-based clustering algorithm for real-time Internet of Things stream. *Sci. World J.* 1–11 (2014)
75. Karyotis, V., Tsitsekis, K., Sotiropoulos, K.: Big data clustering via community detection and hyperbolic network embedding in IoT applications. *MDPI/Sens.* 1–21 (2018)
76. Rui, T., Fong, S.: Clustering big IoT data by metaheuristic optimized mini-batch and parallel partition-based DGC in Hadoop. *Fut. Gen. Comput. Syst.* 1–33 (2018)
77. Fanaee-T, H., Gama, J.: Tensor-based anomaly detection: an interdisciplinary survey. *Knowl.-Based Syst.* 1–28 (2016)

78. Alguliyev, R., Aliguliyev, R., Sukhostat, L.: Anomaly detection in big data based on clustering. *Stat. Optim. Inform. Comput.* 325–340 (2017)
79. Yin, C., Zhang, S., Kim, K.-J.: Mobile anomaly detection based on improved self-organizing maps. *Mob. Inform. Syst.* 1–9 (2017)
80. Alsayat, A., El-Sayed, H.: Social media analysis using optimized K-means clustering. In: 2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA), pp. 1–10. IEEE, Towson, MD, USA (2016)
81. Gurusamy, V., Kannan, S., Prabhu, J.R.: Mining the attitude of social network users using K-means. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* 226–230 (2017)
82. Singh, K., Kumar Shakya, H., Biswas, B.: Clustering of people in social network based on textual similarity. *Perspect. Sci.* 570–573 (2016)
83. Ghaemi, Z., Farnaghi, M.: A varied density-based clustering approach for event detection from heterogeneous Twitter data. *Int. J. Geo-Inform.* 1–18 (2019)
84. Mohebi, A., Aghabozorgi, S., Wah, T.Y., Herawan, T., Yahyapour, R.: Iterative big data clustering algorithms: a review. *Softw. Pract. Exp.* 107–129 (2016)
85. Singh, D., Reddy, C.K.: A survey on platforms for big data analytics. *J. Big Data* 8 (2015)
86. Likas, A., Vlassis, N., Verbeek, J.J.: The global k-means clustering algorithm. *Pattern Recogn.* 451–461 (2003)
87. Pelleg, D., Moore, A.: X-means: extending k-means with efficient estimation of the number of clusters. In: *Proceedings of the 17th International Conference on Machine Learning*, pp. 27–734 (2000)
88. Bradley, P.S., Fayyad, U.M.: Refining initial points for K-means clustering. In: *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 1–99. ICML (1998)
89. Emre Celebi, M., Kingravi, H.A., Vela, P.A.: A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Syst. Appl.* 200–210 (2013)
90. Li, Q., Wang, P., Wang, W., Hu, H., Li, Z., Li, J.: An efficient k-means clustering algorithm on MapReduce. In: *Proceedings of Database Systems for Advanced Applications*, pp. 357–371 (2014)
91. Bandyopadhyay, S., Maulik, U.: An evolutionary technique based on K-means algorithm for optimal clustering in RN. *Inform. Sci.* 221–237 (2002)
92. Esmín, A., Coelho, R.A., Matwin, S.: A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data. *Artif. Intell. Rev.* 23–45 (2015)
93. Krishna, K., Narasimha Murty, M.: Genetic K-means algorithm. *IEEE Trans. Syst. Man Cybern.* 433–439 (1999)
94. Chen, M., Mao, S., Liu, Y.: Big data: a survey. *Mob. Netw. Appl.* 19(2), 171–209 (2014)
95. Li, H., Lu, X.: Challenges and trends of big data analytics. In: *Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, Guangzhou, China. pp. 566–567 (2014)
96. Kaisler, S., Armour, F., Espinosa, J., Money, W.: Big data: issues and challenges moving forward. In: *46th Hawaii International Conference on System Sciences (HICSS)*, Hawaii, pp. 995–1004 (2013)
97. Assunção, M.D., Calheiros, R.N., Bianchi, S., Netto, M.A.S., Buyya, R.: Big data computing and clouds: trends and future directions. *J. Parallel Distrib. Comput.* 79–80, 3–15 (2015)
98. Cuzzocrea, A., Song, I.-Y., Davis, K.C.: Analytics over large scale multidimensional data: the big data revolution! In: *Proceedings of the ACM 14th International workshop on Data Warehousing and OLAP (DOLAP'11)*, pp. 101–104. ACM, New York, NY, USA (2011)
99. Benjamins, V.R.: Big data: from hype to reality? In: *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, pp. 2:1–2:2. ACM, New York, NY, USA (2014)
100. Boyd, D., Crawford, K.: Critical questions for big data. *Commun. Soc.* 15(5), 662–679 (2012)