Hindawi
Scientific Programming
Volume 2021, Article ID 9988318, 20 pages
https://doi.org/10.1155/2021/9988318

Research Article

Analysis of Simple *K*-Mean and Parallel *K*-Mean Clustering for Software Products and Organizational Performance Using Education Sector Dataset

Rui Shang,[1] Balqees Ara,[2] Islam Zada,[2] Shah Nazir,[3] Zaid Ullah,[4] and Shafi Ullah Khan[5]

[1]*Office of Science and Technology Administration, Heilongjiang Bayi Agricultural University, DaQing 163000, China*
[2]*Department of Computer Science, University of Peshawar, Peshawar, Pakistan*
[3]*Department of Computer Science, University of Swabi, Swabi, Pakistan*
[4]*Department of Computer Science, Bacha Khan University, Charsadda, Pakistan*
[5]*Institute of Computing, Kohat University of Science & Technology, Kohat, Pakistan*

Correspondence should be addressed to Rui Shang; sr197936@byau.edu.cn

Received 24 March 2021; Revised 10 April 2021; Accepted 19 April 2021; Published 17 May 2021

Academic Editor: Zhongguo Yang

Context. Educational Data Mining (EDM) is a new and emerging research area. Data mining techniques are used in the educational field in order to extract useful information on employee or student progress behaviors. Recent increase in the availability of learning data has given importance and momentum to educational data mining to better understand and optimize the learning process and the environments in which it takes place. *Objective*. Data are the most valuable commodity for any organization. It is very difficult to extract useful information from such a large and massive collection of data. Data mining techniques are used to forecast and evaluate academic performance of students based on their academic record and participation in the forum. Although several studies have been carried out to evaluate the academic performance of students worldwide, there is a lack of appropriate studies to assess factors that can boost the academic performance of students. *Methodology*. The current study sought to weigh up factors that contribute to improving student academic performance in Pakistan. In this paper, both the simple and parallel clustering techniques are implemented and analyzed to point out their best features. The Parallel *K*-Mean algorithms overcome the problems of simple algorithm and the outcomes of the parallel algorithms are always the same, which improves the cluster quality, number of iterations, and elapsed time. *Results*. Both the algorithms are tested and compared with each other for a dataset of 10,000 and 5000 integer data items. The datasets are evaluated 10 times for minimum elapse time-varying *K* value from 1 to 10. The proposed study is more useful for scientific research data sorting. Scientific research data statistics are more accurate.

1. Introduction

We are living in a world of data. Every day, people come across a large amount of data and store these data for further exploration or analysis. Such large datasets are growing rapidly so that it is a very challenging task to extract and mine important information, using conventional techniques [1]. Data are the collection of any facts, figures, and numbers that can be processed in order by a computer system. Nowadays, business organizations are accumulating grooving and huge volumes of data in different setups, formats, and databases. These include [2] transactional data also called operational data, i.e., cost, sales, accounting, payroll, and inventory, and nonoperational data, i.e., forecasting of data, industry sales and macroeconomic data, and metadata, such as data dictionary definitions or logical database design. Different efficient and effective data mining methods are used to mine useful and important information from bulky datasets. Clustering and classification are the most popular techniques of data mining for retrieving data from large datasets; classification is known as supervised learning phenomena while that of clustering is the unsupervised learning phenomena.

Classification is also called supervised learning technique [3]. In order to understand a new phenomenon or to learn about a new object, people always try to compare it with other phenomena or objects based on similarities or dissimilarities [4]. After this comparison, a decision can be made. This mechanism is a classification technique of data mining [5]. Clustering also called the unsupervised learning method [6] involves grouping of data items into clusters that have high similarity but are dissimilar to data items of other clusters [7, 8]. The clustering method is used in many areas, for instance, computer sciences, engineering, Earth sciences, life and medical sciences, economics, social sciences, etc. Examples of its application are speech recognition [9], image segmentation, color recognition [10], web mining, text mining, and grouping of patients, students, customers and items [11]. The partitioning method splits the complete dataset into sub-datasets. The whole dataset given is signified by "n," whereas "*K*" signifies the sub-datasets also called clusters. "*N*" represents the total number of data items in a dataset, whereas "*k*" is the sub-datasets (clusters); then, the whole dataset must be greater than the total number of clusters, i.e. $(n > k)$. Data items in a cluster are alike but are different from the items of the other cluster. Partitioning methods are further classified into several different categories, that is, *K*-medoids clustering, *K*-Mean clustering, Relocation algorithms, and Probabilistic clustering [12]. In hierarchical clustering, each cluster node contains parent cluster, child clusters, and sibling clusters. It is further classified into bottom-up (agglomerative) and top-down (divisive clustering). The basic drawbacks of the hierarchical clustering method is that once a split or merge decision is executed, it cannot be readjusted [13]. Hierarchical clustering is a technique that integrates similar objects into different groups called clusters [14]. At the end, the numbers of clusters are ascertained, in which each cluster is different from the other cluster, and the data items in a cluster are similar. This research focuses on the two clustering techniques that are Simple and Parallel *K*-Mean schemes.

Simple *K*-Mean technique is one of the most popular and simple unsupervised learning approaches. Unsupervised learning techniques use input vectors for making inferences from the datasets without referring to labeled outcomes. Initially, Simple *K*-Mean selects the centroids randomly for the processing of data; these centroids are the starting points for these clusters and the optimized positions are calculated using the iterative procedure. Simple *K*-Mean is one of the partitioning methods, which is the simplest clustering technique. Simple *K*-Mean clustering algorithm partitioned the complete datasets into different subclusters, and the subclusters are represented by "*k*". Simple *K*-Mean algorithm is easy and simple to implement but still has some drawbacks. In Simple way *K*-Mean, initial centroids (*k*) are irregularly preferred for each cluster. On each run or execution, Simple *K*-Mean algorithm produces different clusters. Then, different results are produced because of different initial centroids. In Simple *K*-Mean clustering algorithm, "*k*" represents a user-specified parameter. Here, the "*k*" number of different clusters shall be put in advance. Initial centroids should be carefully selected for *K*-Mean clustering algorithm. Because of the random selection of initial centroids, the outcome will be different from the previous one on each run. Due to this drawback, clusters vary from one another; also, data items in clusters may vary from one cluster to another. Simple *K*-Mean algorithm is noise sensitive because the arithmetic mean value can be considerably influenced by noisy data. Median is used to overcome this problem of Simple *K*-Mean clustering. It is applicable to process numeric type of data and cannot be useful for categorical type of data. If a large dataset is given, then more space is required for clustering. Most critical issues for Simple *K*-Mean clustering algorithm are the space and processing speed requirements, when dataset is very large. This algorithm uses Client-Server architecture. To solve the problems of processing speed and memory requirement, Simple Parallel *K*-Mean algorithm is used for huge datasets. In simple parallel *K*-mean clustering algorithm, the dataset is partitioned addicted to subparts, so less space and processing speed will be required to process subparts of the datasets. Initial centroids random selection is a disadvantage of Simple *K*-Mean algorithm. Due to initial centroids random selection, the outcomes of the Parallel *K*-Mean clustering algorithm vary from one run or execution to another. Also, the data items move from one cluster to another, which affects the cluster quality. Just like Simple *K*-Mean, Parallel *K*-Mean also iteratively changes in clusters.

This paper consists of a total of six sections. The Introduction section discussed different general terminologies of clustering and specifically discussion about K-Mean and Parallel K-Mean clustering algorithms using in this study application of the clustering algorithm, different types of clustering. Section 2 presents the existing research about Simple *K*-Mean and Parallel *K*-Mean algorithms, and their analysis based on the research. Section 3 discussed the study implementation and evaluation. Section 4 presents results and discussion of the study. Section 5 is dedicated for Conclusion and future.

2. Literature Review

J. B. MacQueen was among the pioneers to introduce the *K*-Means clustering algorithm (Simple) in 1967 [15]. Many researchers then worked on this Simple *K*-Mean clustering algorithm, and some of the latest research work is discussed here. The procedure is sustained till there is no difference in the clusters. Min-Max distance measure is introduced by Karthikeyani and suguna [16]. The input dataset is normalized first and then within the normalized range (0, 1), and initial centroids are selected on a random basis. Using min-max similarity measure, the distance is calculated. In [3], the whole dataset is partitioned into blocks, called unit blocks (UB), by using the minimum and maximum limits. After the transformation, formulation of the objects in datasets are sorted on the basis of distance and they are also divided into subclusters (*k* sets). Median value is also calculated for each set of values, the calculated median is measured as initial centroids, and then clusters are designed through using the designing of initial clusters [17, 18]. This technique used sorting algorithms, which have extra time complexity. Centroids of each unit block are calculated to form the simplest view of the dataset. For calculating finalized centroids, the reduced dataset is used. Each data point is moved to its appropriate cluster. Fahim et al. [19] and Shi and Xumin [20] have presented a *K*-Mean algorithm, and to store the information of all iterations, they use a data structure. Then, in the forthcoming iteration, the stored information is used. The time of computation is saved by their technique; however, a data structure is required. Also, the initial clusters are based on the random selection of initial centroids. Hongyang and Jia [21] have introduced a dynamic *K*-Mean clustering technique. In the first step at the server side, sub datasets are produced from the given dataset. After the transformation, formulation of the objects in datasets are sorted on the basis of distance and divided into sub clusters (*k* sets). Median value is also computed for each set of values, the calculated medium is measured as initial centroids, and then clusters are designed using the design of initial clusters [22]. This technique used sorting algorithms, which has extra time complexity. All client systems that are connected to the server receive these sub-datasets with "*K*" number of clusters, and also initial centroids. Client system calculates clusters and forwards the results to the server. These processes continue still there is no change occurs in the clusters. The "RBFNN" Radial Basis Function, Neural Network structure theory has been discussed, where a *K*-Means clustering dynamic method is used for the center selection. The experimental results of the study shows that here the approximation of RBFNN has good performance, where dynamic *K*-Mean clustering technique is used for center selection. Clusters formed by all of the client systems are then sent to the server system. At the server side, all of the clusters (received from all the clients) are compiled. Arithmetic means of each of the compiled clusters are calculated. Newly calculated arithmetic means (centroids) are compared with the previous arithmetic means (centroids). If the values of the new means (centroids) are equal to the previous means (centroids), the process is terminated, otherwise it will be continued.

An algorithm is introduced by Jirong et al. [23], in which some sub-datasets are selected randomly from large datasets. Partitioning clustering algorithms are used to get centroids of the cluster to each of the subset. Partition algorithm is used to get the most popular centroids after gathering the center sets. Clustering and classification are the most popular techniques of data mining for retrieving data from large datasets; classification is known as supervised learning phenomena while that of clustering is the unsupervised learning phenomena. Median value is also calculated for each set of values; the calculated medium is measured as initial centroids and then clusters are designed using the designs of initial clusters [22]. This technique used sorting algorithms, which has extra time complexity. They documented that the Simple *K*-Mean clustering algorithm is explained in [24–26]. From the dataset, "*k*" initial centroids are selected randomly. Each of the data item is placed in its appropriate cluster by calculating distances between the data items and initial centroids. The procedure is sustained till there is no difference in the clusters. Min-Max distance measure is introduced by Karthikeyani and suguna [16]. The input dataset is normalized first and then within the normalized range (0, 1), initial centroids are selected on a random basis. Using min-max similarity measure, the distance is calculated. Santhi et al. has introduced an algorithm, which is able to check the absence or presence of negative numbers in datasets [27]. If a negative number is identified, it is transformed into positive numbers by applying subtraction operation to the lowest values of the dataset from the rest of the dataset's numbers. Median value is calculated for each set of values, the calculated medium is measured as initial centroids, and then clusters are designed using the design of the initial clusters [28]. This technique used sorting algorithms, which has extra time complexity. A brief study on *K*-Mean clustering algorithm is also arranged by Chen and Xu [29], in which they discussed the pros and cons, and also typical requirements of *K*-Mean clustering procedure. Yujun et al. has introduced an algorithm, in which "split" and "merge" are the two major stages of clustering [30]. In the "split" stage, each cluster is divided into small subclusters using *K*-Mean clustering algorithm on selected datasets, while in the "merge" stage, the average distance worked for merging. Fahim et al. [7] and Shi and Xumin [20] have presented a *K*-Mean algorithm, to store the information all iterations they use in a data structure. Then, in the forthcoming iteration, the stored information is used. The time of computation is saved by their technique; however, a data structure is required. Also, the initial clusters are based on the random selection of initial centroids. Chawan et al. have introduced an algorithm, which is able to select the initial centroids randomly in the selected datasets [31], in which the clusters are made by a key which is designed by the ratio of maximal intra and minimal intra cluster distance. If the key distance is lesser than the smallest value, then a data point is assigned to the other cluster in the same dataset. It is investigated and observed that the selection centroids affect the meeting time in their commended algorithm.

An author also introduces an algorithm, which improves the efficiency of *K*-Mean clustering [32], where a dataset is converted into positive values, by subtracting the smallest values from all data items, if a negative value is identified in the dataset. Distance is calculated for data items; from common centroids, origins of a dataset are calculated. Initial centroids are selected in such a way that the first closest object in the dataset to the common centroids is selected as the initial centroid [33]. The objects are removed from the dataset having less correspondence to selected centroids. Chenfei and Zhiyi [15] have introduced a *K*-Mean clustering algorithm, in which any data item is selected in the dataset and the distance between that selected data point and other data points is calculated. Threshold value is calculated by dividing the entire area of data points into 100 parts. If the distance value matches the threshold value, it is selected as the initial centroid. The process is continued until "k" initial centroids are formed and the data points are assigned to their proper clusters. The process is continued to get final clusters. *K*-Mean clustering algorithm for mixed numeric and categorical data are introduced by [34]. Raed and Wasem [13] have proposed an algorithm for the selection of initial centroids in the *K*-Mean clustering method. "*K*" values are selected randomly in the dataset. Distance between the selected data items and other data items are calculated to check whether the guess of the random selection of initial centroids was valid or not? Data points in the given dataset are sorted according to the calculated distance. Closest data items, based on the threshold value, are selected as subset and arithmetic mean of the subset is calculated as the first accepted initial centroid. In this way, remaining initial centroids are computed. Sanpawat and Alva [2] have introduced the parallelized version of *K*-Mean clustering algorithm. Master-slave (Client-Server) method is used in the algorithm. Clustering, also called unsupervised learning method, groups data items into clusters that have high similarity but are dissimilar to data items of other clusters. The clustering method is used in many areas, for instance, computer sciences, engineering, Earth sciences, life and medical sciences, economics, social sciences, etc. Examples of its application are speech recognition image segmentation, color recognition, web mining, text mining, and grouping of patients, students, customers, and items.

Qing et al. [12] have introduced the Parallel *K*-Mean clustering algorithm, which is based on the selection of initial centroids. Distance between each of the data item is calculated. Data items having the farthest distance are removed from the dataset and placed in a new list. A threshold value is selected for this new list. Clustering and classification are the most popular techniques of data mining for retrieving data from large datasets; classification is known as supervised learning phenomena while clustering is called unsupervised learning phenomena. When items in the new list reach the threshold, the values of the new list are returned as initial centroids. ParaMeans software is designed for the Parallel *K*-Mean clustering algorithm by [35]. They implement the parallelized Basic *K*-Mean clustering algorithm for the use of general laboratory. ParaMeans provide an easy and manageable client server application. Technique of dynamic load balance to enhance the Parallel *K*-Mean clustering algorithm is introduced by [36]. Clusters formed by all of the client systems are then sent to the server system. At the server side, all of the clusters (received from all the clients) are compiled. Arithmetic means of each of the compiled clusters are calculated. Newly calculated arithmetic means (centroids) are compared with the previous arithmetic means (centroids). If the value of the new means (centroids) are equal to the previous means (centroids), the process is terminated, otherwise it will be continued. In this technique, same size of the sub-dataset is assigned by the master system to the slave system. The Simple *K*-Mean clustering algorithm is explained in [37]. The distance between the data items and initial centroids is calculated and each of the data item is placed in its appropriate position. Many researchers then worked on this Simple *K*-Mean clustering algorithm, and some of the latest research work is discussed. The procedure is sustained till there is no difference in the clusters. Min-Max distance measure is introduced by Karthikeyani and suguna [38]. The input dataset is normalized first and then within the normalized range (0, 1), initial centroids are selected on a random basis. Using min-max similarity measure, the distance is calculated (0, 1) [39]. Ran Vijay and Bhatia [40] have introduced a *K*-Mean algorithm, items having least frequency. Average of each segment is calculated and considered as the centroid. At the server side, all of the clusters (received from all the clients) are compiled. Arithmetic means of each of the compiled clusters are calculated. Newly calculated arithmetic means (centroids) are compared with the previous arithmetic means for each of the cluster's centroid; a threshold distance is calculated and at last data items are moved to their appropriate clusters.

Chenfei and Zhiyi [41] have documented that any data item is selected in the dataset and distance between that selected data point and other data points is calculated. Threshold value is calculated by dividing the entire area of data points into 100 parts. If the distance value matches to the threshold value, it is selected as the initial centroid. The process is continued to get final clusters. A clustering algorithm is introduced for *K*-Mean for mixed numeric and categorical data by [42]. Raed and Wasem [43] have proposed an algorithm for the selection of initial centroids. Distance between the selected data items and other data items are calculated to check whether the guess of the random selection of initial centroids was valid or not. Closest data items, based on the threshold value, are selected as the subset, and arithmetic mean of the subset is calculated as the first accepted initial centroid. In this way, remaining initial centroids are computed [13]. Sanpawat and Alva [44] have introduced the parallelized version of Simple *K*-Mean clustering algorithm. Master-slave (Server-Client) method is used in the algorithm. This algorithm is based on random selection of the initial centroids. One another researcher applied the Parallel *K*-Mean algorithm using the agricultural data [45]. Client-server architecture is used for the processing of these datasets. They also parallelized Simple *K*-Mean technique also based on random selection of initial centroids [26, 46]. Parallel clustering technique based on message passing interface (MPI) called M-*K*-Means composed of MPI and Sequential *K*-Means is applied on the education datasets [47]. Also, *K*-Mean and Message Passing interface is mutually used in the same experiment, which is also based on random selection of initial centroids, in which a dataset is divided into "*p*" sub-datasets where "*p*" is the number of nodes connected to the main computer. The algorithm is tested on DNA dataset and Simple *K*-Mean algorithm is parallelized, where initial centroids are selected randomly. Qing et al. [48] have investigated the same nature of study where they found that there is a threshold value that is selected for this new list. When items in the new list reach the threshold, the values of the new list are returned as initial centroids. Para Means software is designed for the Parallel *K*-Mean clustering algorithm by [13,49]. They implement the parallelized Simple *K*-Mean clustering algorithm for the use of general laboratory. Para Means provide an easy and manageable client server application, written in C# [39].

Technique of dynamic load balance to make the Parallel *K*-Mean clustering technique efficient and progressive is presented in [42]. In this technique, same size of the sub-dataset is assigned by the master system to the slave system

[32]. *K*-Mean and Parallel *K*-Mean clustering algorithms are widely known research areas. Many researchers worked on Simple *K*-Mean and Parallel *K*-Mean algorithms individually, and they introduced different algorithms, which are discussed in section 2, But, they have no clear idea or suggestion as to how to use *K*-Mean and Parallel *K*-Mean algorithms, which could be used further in any area relevant to their usage. Clustering and classification are the most popular techniques of data mining for retrieving data from large datasets; classification is known as supervised learning phenomena while that of clustering is the unsupervised learning phenomena. The *K*-mean clustering algorithms are tested and compared with the parallel *K*-mean clustering algorithm for two types of datasets of 10,000 and 5000 data items. These two datasets of 10,000 and 5000 integers may represent the score of 10,000 students in two different subjects and 5000 employee's attendance in two months, respectively.

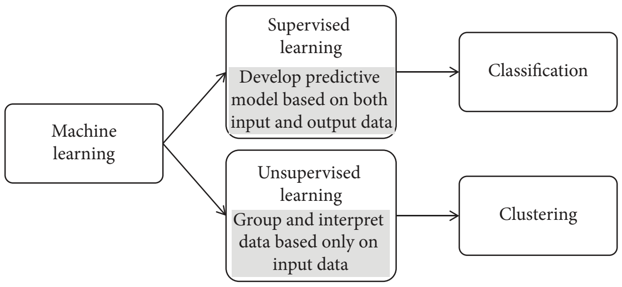
Machine learning
Supervised learning
Develop predictive model based on both input and output data
Unsupervised learning
Group and interpret data based only on input data
Classification
Clustering

Figure 1: Supervised and unsupervised learning.

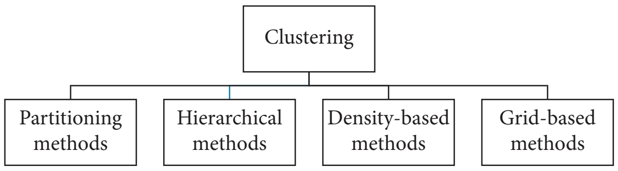

Figure 2: Types of clustering.

3. Research Methodology

Figure 1 depicts the graphical representation of the two approaches, namely, clustering and classification.

Clustering techniques are classified into four basic categories, which are shown in Figure 2.

As discussed, there is no clear idea about *K*-Mean and Parallel *K*-Mean algorithms as to which one is the best and in which domain and situation the particular technique should be used? To overcome these problems, the study implements and analyzes both Simple *K*-Mean and Parallel *K*-Mean clustering to distinguish their performance attributes and cluster quality is applied in general. For this intention, the *K*-Mean clustering algorithms are evaluated, tested, and compared with the Parallel *K*-Mean for two types of datasets of 10,000 and 5000 data items. These two datasets of 10,000 and 5000 integers represent the score of 10,000 students in two different subjects and 5000 employee's attendance in two months, respectively. This chapter discusses the details of the Methodology of this study for Simple *K*-Mean and Parallel *K*-Mean clustering. Figure 3 illustrates the overall methodology of the study as follows. The study implemented and analyzed both Simple and Parallel techniques using java and neat beans as a development environment; the two different parameters, i.e., cluster quality and number of iterations are tested and analyzed for two different datasets of integer-type data of 10000 and 5000, respectively, which shows the improvement and performance of cluster quality. This work overcomes the problem of random selection of initial centroids in *K*-Mean clustering.

This research work implements and analyzes both Simple and Parallel *K*-Mean algorithms, to distinguish their performance attributes and cluster quality in general. For this intention, the *K*-Mean clustering algorithms are tested and compared with the Parallel *K*-Mean clustering algorithm for two types of datasets of 10,000 and 5000 data items. These two datasets of 10,000 and 5000 integers may represent the score of 10,000 students in two different subjects and 5000 employee's attendance in two months, respectively. Typical representation of these students in two different subjects through two techniques is shown in Table 1, while the employee's attendances are shown in Table 2.

The input and output for these two algorithms may be employed as given below.

Here *k* represents the number of clusters obtained from students and employees' score in two different subjects and months, respectively, where the *Di* represents the two datasets containing 10000 students and 5000 employees, while a set of *k* clusters is the output. Simple and Parallel algorithms are applied individually lying on these data items. Standard notations of UML (Unified Modeling Language) are used to form the flowchart of Basic *K*-Mean clustering algorithm. The experimental results of both *K*-Mean and Parallel algorithms are evaluated and analyzed for the difference of these two clustering algorithms. For this purpose, these two algorithms are implemented using Neat beans as an integrated development environment using JAVA and C++platform to run different iterations for specific time and for a specific range of data. Simple *K*-Mean clustering algorithm takes "*k*" initial centroids randomly. In the second step, the Euclidean distance function is used to calculate the distances between centroids and data items.

Different distance functions are mentioned in [24, 50]. Each of the data item is moved to the cluster having less space. In this way, initial clusters are formed. Then the arithmetic mean of each cluster is calculated. Data items having less distance to the arithmetic mean are placed in that particular cluster. Then again, the arithmetic mean of each cluster is designed and data points are move to the desired cluster. The process is sustained until no data point moves from one cluster to another cluster. Improving and creating clusters stop when:

The provided number of iterations has been completed

The clustering has been successful, i.e., cluster has become stable

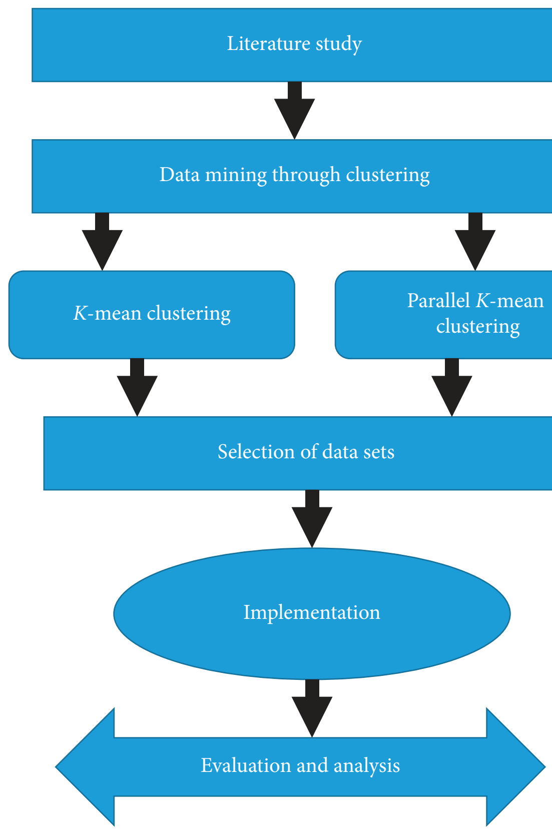

Figure 3: Research flow diagram.

Table 1: Ten thousand student's marks.

Students	Subj-A marks	Subj-B marks
1	83	65
2	75	82
:	:	:
:	:	:
:	:	:
:	85	60
10000	55	75

3.1. Algorithmic Steps of Simple K-Mean Clustering Algorithm. Figure 4 is the pseudo code for Simple K-Mean clustering algorithm [51].

3.1.1. Flow Chart of Simple K-Mean Algorithm. Standard notations of UML are used to form the flowchart of Simple K-Mean algorithm. Figure 5 shows the flow chart of Simple K-Mean algorithm.

3.2. Parallel K-Mean Clustering Algorithm. Most critical issues for Simple K-Mean clustering algorithm are the space and processing speed requirements, when the dataset is very large. To resolve these issues, Simple K-Mean clustering algorithm is parallelized.

3.2.1. Steps of Parallel K-Mean Algorithm. Three main steps of the Simple Parallel K-Mean are

Partition

Computation

Compilation

In the first step at the server side, sub-datasets are produced from a given dataset. All client systems that are connected to the server receive these sub-datasets with the number of clusters, i.e., "k" and initial centroids. The said client systems calculate clusters and forward that results to the server. This process continues till there is no change in the clusters.

Server's activities

Partition dataset and randomly select initial centroids $C=C_1+ C_2+\ldots\ldots\ldots+ C_n$
Send initial centroids C=C1+ C2+. . . . .+ Cn and sub-datasets to all connected clients
Receive clusters and centroids from all the clients
Recalculation of centroids

Client's activities

Receive k, initial centroids C=C1+ C2+. . . . .+ Cn and sub-datasets from the server
Calculate distances from all centroids
Move data items to appropriate clusters
Send cluster elements to the server

3.2.2. Flow Chart of Parallel K-Mean Clustering Algorithm. Steps described above are shown in the flow chart in Figure 6. Standard notations of UML are used to form the flow chart.

3.3. Simple K-Mean Implementation

3.3.1. Calculation of Initial Centroids. Simple K-Mean clustering algorithm depends on the selection of the initial centroids. If different data items are selected in different runs of the same dataset, different results are found in each run. To overcome this issue, initial centroids are calculated by the following steps:

3.3.2. Initial Clusters. In order to find the initial cluster, the whole dataset is partitioned into k number of sub-datasets by using

$$\text{initial clusters} = \text{floor}\left(\frac{n}{k}\right), \quad (1)$$

where "n" is the total number of data items in the given dataset, which is to be clustered and "k" is the total number of clusters.

Floor value of n/k is taken in the parallel technique because all of the sub-datasets will have the same number of data items, and if any of the data items is left, it will be placed in the last cluster.

Table 2: Employees' attendance.

Emp_id	Month_A attendance percentage	Month_B attendance percentage	Total attendance percentage
0001	90	75	$(90+75)/2=\ldots\ldots$
0002	75	90	
0003	85	80	
. . ..	. . .. . .	. . .. . .	
. . ..	. . ..	. . ..	
5000	95	75	

```
Algorithm 3.1-to find the clusters by simple K-mean clustering Algorithm

INPUT : Array {$a_1$, $a_2$, $a_3$.........$a_n$}

        $a$ = data points

        $k$ = number of required clusters

OUTPUT : $a$ set of clusters

STEPS:

    1. Randomly select $k$ data points from dataset $D$ as initial centers.

    2. Calculate the distance between each data point $d_i$ ($1 <= i <= n$) and all $k$ clusters
    $C_j$ ($1 <= j <= k$) and assign data object $d_i$ to the nearest cluster.

    3. For each cluster $j$ ($1 <= j <= k$), recalculate the cluster center by taking
    arithmetic mean of each cluster.

    4. Repeat until no change in the center of cluster.
```

Figure 4: Simple K-Mean clustering algorithm.

3.3.3. Initial Centroids. After getting the initial clusters, initial centroids are calculated by

$$\text{initial centroids} = \text{arithmetic mean of initial clusters}, \tag{2}$$

$$\text{arithmetic mean}\,(A.\text{M})\text{ of initial cluster} = \sum \frac{(\text{sum of all data items in cluster})}{\text{total number of data items}}. \tag{3}$$

3.3.4. Distance Calculation. Distance between initial centroids and each of the data items is calculated by the Euclidean distance equation:

$$\text{Euclidean distance}\,(C_i, D_i) = |C_i - D_i|, \tag{4}$$

where C_i is the initial centroid values for $i \leq k$ (k is the number of clusters) and D_i is the data item for $i \leq n$, where n is the total number of data items in the given dataset.

3.3.5. Assignment of Data Item to Appropriate Cluster. Distances among all data items and initial centroids are compared. Data items having minimum distance from the initial centroid are assigned to that particular cluster. The minimum distance is calculated by the following formula:

$$\text{min distance} = \min\,(i \leq n)|C_i - D_i|, \tag{5}$$

where "i" is a variable and "n" is the number of data items.

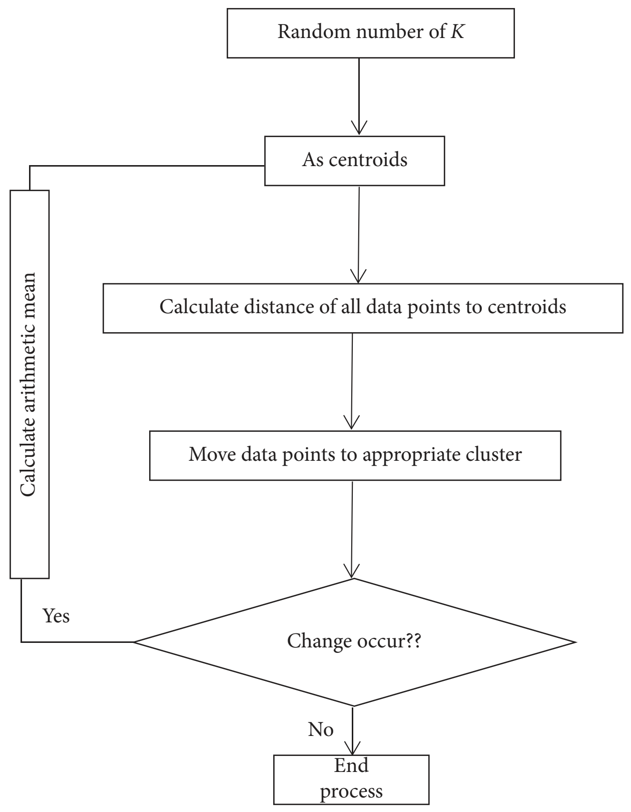
Random number of K
As centroids
Calculate distance of all data points to centroids
Move data points to appropriate cluster
Change occur??
Yes
No
End process
Calculate arithmetic mean

Figure 5: Flow chart of Simple K-Mean algorithm.

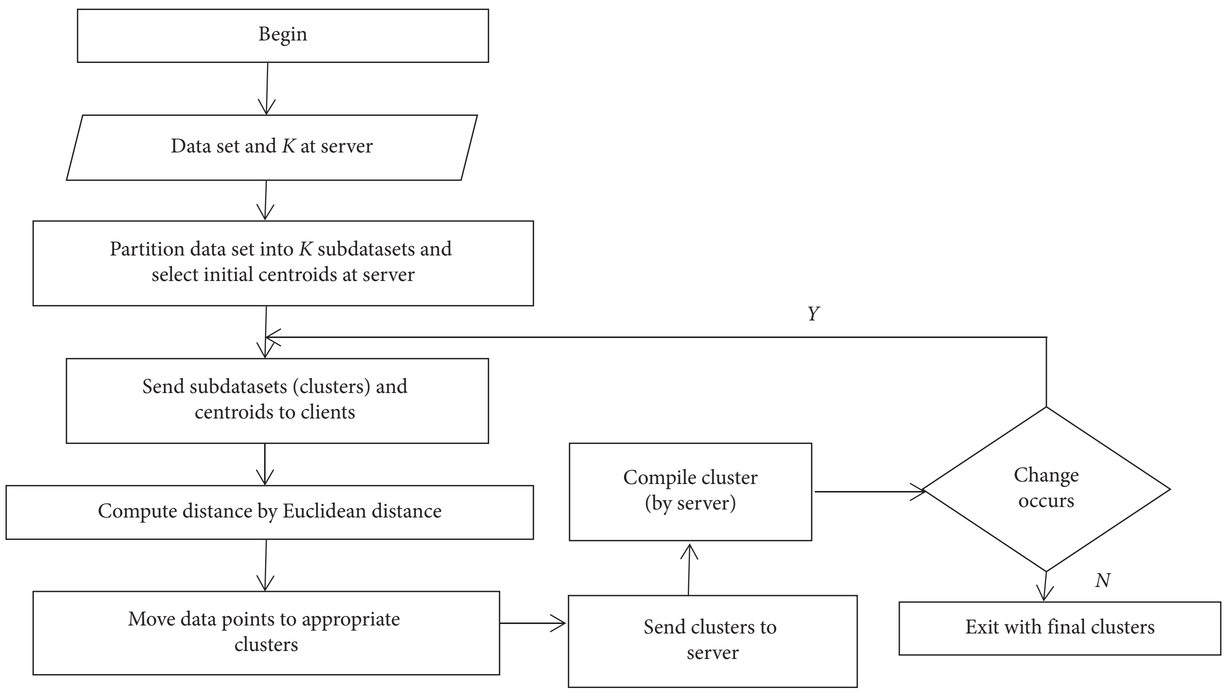
Begin
Data set and K at server
Partition data set into K subdatasets and select initial centroids at server
Y
Send subdatasets (clusters) and centroids to clients
Compute distance by Euclidean distance
Move data points to appropriate clusters
Send clusters to server
Compile cluster (by server)
Change occurs
N
Exit with final clusters

Figure 6: Flow chart of Simple Parallel K-Mean clustering algorithm.

Similar or closest data items are placed in the same cluster.

3.3.6. Process of Termination. Arithmetic mean of each of the calculated clusters is compared with the previous mean value (centroid). If both of the means (centroid values) are the same, then the process will be terminated; otherwise, it will continue until the same centroids (mean values) are found.

3.4. Parallel K-Mean Algorithm Implementation. The basic *K*-Mean clustering algorithm requires maximum space and processing speed for large datasets. For this intention, *K*-Mean is parallelized. The *K*-Mean algorithm selects initial centroids randomly. Problems of the Simple *K*-Mean algorithm also exist in the Simple Parallel *K*-Mean clustering algorithm. To resolve these problems, Parallel algorithm is parallelized. Client server structural design is used in the efficient Parallel *K*-Mean clustering algorithm.

3.4.1. Tasks of the Sever System. Main tasks of the server or the main system are as follows:

Partition Data for all client systems (Client 1, Client 2. . ... Client *n*)

Partition Data and Calculate initial centroids $C=C_1+C_2+\ldots\ldots\ldots+C_n$

Send initial centroids $C=C_1+C_2+\ldots\ldots\ldots+C_n$ to all systems

Receive clusters and centroids from all connected systems

3.4.2. Tasks of Client Systems. Main tasks of the client systems are as follows:

Receive *k*, initial centroids C=C1+ C2+. . . . .+ Cn and sub-dataset from the server system

Calculate distances from all centroids

Move data item to the appropriate cluster

Send cluster elements back to the main system

3.5. Steps of Parallel K-Mean Algorithm. Following are the main steps of the Parallel *K*-Mean algorithm.

3.5.1. Data Partitioning For Client Systems. The first step of the Parallel *K*-Mean clustering algorithm is to divide the input dataset into sub-datasets for all client systems. Dataset is divided into "*x*" number of sub-datasets, if total "*x*" number of client systems are connected to the server system. Value of "*x*" is calculated as

$$\text{subdata sets } (i = x) = \text{floor}\left[\left(\frac{n}{\text{x}}\right)\right], \qquad (6)$$

where "*n*" is the total number of data items in the given dataset and "*x*" is the total number of client systems connected to the server system.

3.5.2. Calculation of Initial Centroids. Initial centroids are calculated in two steps:

Find initial cluster that can be formed by using equation (1)

Calculate initial centroids using equations (2) and (3)

Initial Centroids = Arithmetic Mean of initial clusters such as in (2)

Arithmetic Mean (A.M) of initial cluster = Σ (sum of all data items in cluster)/total number of data items as in (3)

3.5.3. Distance Calculation by Client Systems. Sub-datasets and initial centroids are sent to all clients. Only one sub-dataset is sent to each client. Distance is calculated using equation (4).

3.5.4. Making Clusters by Client Systems. The distance between data items and initial centroids are compared. Data item closest to the centroid value is assigned to that cluster. In this way, data items are placed in their appropriate clusters.

3.5.5. Compiling Results of Client Systems by the Server System and Termination. Clusters formed by all of the client systems are then sent to the server system. At the server side, all of the clusters (received from all the clients) are compiled. Arithmetic means of each of the compiled clusters are calculated. Newly calculated arithmetic means (centroids) are compared with the previous arithmetic means (centroids). If the values of the new means (centroids) are equal to the previous means (centroids), the process is terminated; otherwise, it will be continued.

3.6. Time Complexity of K-Mean Algorithm. *K*-Mean algorithm comprises 2 phases. The 1st phase is to compute the initial clusters by dividing the dataset into "*k*" equal parts and to calculate the arithmetic mean. Time complexity for partitioning dataset into "*k*" equal parts and finding arithmetic mean is O(*n*). Thus, the time complexity of the first phase is O(*n*). In the second phase of the Parallel *K*-Mean clustering algorithm, data items are assigned to the appropriate cluster. This phase takes O(nkt). Where *n* is the number of data items, *k* is the number of clusters, and *t* is the number of iterations. Thus, the overall time complexity of the *K*-Mean clustering algorithm is maximum of O(*n*) and O(nkt). That is, O(*n*) + O(nkt), so the overall time complexity is O(nkt), as we ignore the lowest order and constant terms in the asymptotic notations.

4. Results and Discussion

Both the algorithms are tested and compared with each other for a dataset of 10,000, and 5000 integer data items. The experimental results of both algorithms were satisfactory and overcome the problem of Simple *K*-Mean clustering algorithm. Comparison of Simple *K*-Mean and Parallel *K*-Mean algorithm in terms of number of iterations, elapsed time, and cluster quality is given. The detailed and actual results of the comparison of the Simple and Parallel algorithms are discussed in the following section. Comparison of Simple *K*-Mean and Parallel *K*-Mean algorithm in terms of number of iterations, elapsed time, and cluster quality is given.

4.1. Number of Iterations. For different number of clusters (*K*), performance of the Simple *K*-Mean and Parallel *K*-Mean clustering algorithm is shown in the following tables and graphs. Table 3 represents the iterations for $K = 3$.

In Table 3, Simple *K*-Mean algorithms and Parallel *K*-Mean procedures are compared for the same datasets and ($K = 3$) the number of clusters. For each run, the same dataset, i.e. (10,000 data points), is entered to observe and perceive that each time, the number of iterations is different in Simple K-Mean algorithm. The numbers of iterations are the same (fixed) in the Parallel *K*-Mean algorithm because the initial centroids are not selected randomly. Graph of Table 3 is represented in Figure 7.

Number of iterations are fixed in the case of Parallel *K*-Mean algorithm, i.e., 3 for $k = 3$, but keeps changing from one run to other in the case of Simple *K*-Mean clustering algorithm. Table 4 shows the iterations for $K = 4$.

Table 4 shows that the number of iterations of Parallel *K*-Mean clustering algorithm are less than the number of iterations of Simple *K*-Mean clustering algorithm for $k = 4$. Graph of Table 4 is represented in Figure 8.

Table 5 shows the iterations for $K = 4$.

Table 5 shows that the number of iterations of Parallel *K*-Mean clustering algorithm are fixed and less than the number of iterations of Simple *K*-Mean clustering algorithm for $k = 5$. Graph of Table 5 is represented in Figure 9.

Table 6 shows the iterations for $K = 6$.

Table 6 shows the fixed and less number of iterations of Parallel and Simple *K*-Mean clustering algorithms for $k = 6$. Graph of Table 6 is represented in Figure 10.

Table 7 shows the iterations for $K = 7$.

Table 7 shows the number of iterations of Simple and Parallel algorithms for $k = 7$. Graph of Table 7 is represented in Figure 11.

4.2. Elapsed Time. For different number of clusters (*K*), elapsed time of the Simple and Parallel clustering algorithms is shown in the following tables and graphs. Screenshots are given in Appendix. Table 8 shows the elapsed time for $K = 3$.

In Table 8, Simple and Parallel clustering algorithms are compared for $K = 3$. At each run, elapsed time of the Parallel *K*-Mean clustering algorithm is less than the elapsed time of the Simple *K*-Mean clustering algorithm. Graph of Table 8 is represented in Figure 12.

Elapsed time of the Parallel *K*-Mean algorithm is less than the elapsed time of the Simple *K*-Mean algorithm at different runs or executions. Table 9 shows the elapsed time for $K = 4$.

Table 9 shows the elapsed time of Parallel and Simple *K*-Mean clustering algorithm for $k = 4$. Graph of Table 9 is represented in Figure 13.

Table 10 shows the elapsed time for $K = 5$.

Table 10 shows the comparison of the Parallel clustering algorithm and Simple *K*-Mean algorithm for $k = 5$ in terms of elapsed time. Graph of Table 10 is represented in Figure 14.

Table 11 shows the elapsed time for $K = 6$.

Table 11 shows the similarity of the elapsed time of Parallel and the Simple *K*-Mean algorithms for $k = 6$. Graph of Table 11 is represented in Figure 15.

Table 12 shows the elapsed time for $K = 7$.

Graph of Table 12 is represented in Figure 16.

4.3. Cluster Quality. Comparison of Simple *K*-Mean and Parallel *K*-Mean algorithms in terms of cluster quality is discussed below. Table 13 represents the cluster quality of Simple *K*-Mean clustering for $k = 3$.

In Table 13, different outcomes for the same dataset of 10,000 data items at different runs or executions are shown. Graph of Table 13 is represented in Figure 17.

Table 14 represents the cluster quality of Parallel *K*-Mean clustering for $k = 3$.

In Table 14, same outcomes for the same dataset of 10,000 data items at different runs or executions are shown. Graph of Table 14 is represented in Figure 18.

4.4. Comparison of Simple and Parallel K-Mean Algorithms. Simple and Parallel K-Mean algorithms in terms of the number of iterations and comparison of elapsed time is given in the following.

4.4.1. Number of Iterations. For different number of clusters (K), performance of the Simple K-Mean and Parallel K-Mean algorithms is shown in Table 15.

Table 15 shows that for different "*k*" number of clusters, number of iterations is less in Parallel *K*-Mean algorithm than the Simple *K*-Mean algorithm. Graph of Table 15 is represented in Figure 19.

4.4.2. Elapsed Time. Elapsed time of the Simple *K*-Mean and Parallel *K*-Mean algorithms for $k = 2$ is shown in Table 16.

Table 16 shows that the dataset of 10,000 data items is partitioned into 10 sub-datasets for 10 client systems. Clusters are formed in 2 iterations (elapsed time for iterations by all client systems is mentioned in the table). In the first iteration, client 5 has the largest elapsed time, i.e., 8 ms, while in the second iteration, the largest elapsed time is 6 ms, so the total elapsed time of the client systems is 12 ms for the Parallel *K*-Mean clustering algorithm. Table 17 shows the elapsed time of Simple *K*-Mean for $k = 2$.

This section described the results and their discussion in detail of the proposed research. Experiments were designed to prove that Parallel algorithms are showing significant improvement in the Simple *K*-Mean algorithm. Outcomes of the Parallel algorithms are also same, while outcomes of the Simple *K*-Mean algorithm give different results on every run or execution. Also, total iterations and elapse time is improved in the Parallel algorithms. The next section concludes the completed research work and shows the directions for the future research.

Table 3: Iterations for $K = 3$.

	For $K = 3$	
Runs/executions	Number of iterations by Simple K-Mean clustering	Number of iterations by Parallel K-Mean clustering
01	12	3
02	9	3
03	12	3
04	9	3
05	7	3
06	14	3
07	9	3
08	12	3
09	10	3
10	15	3

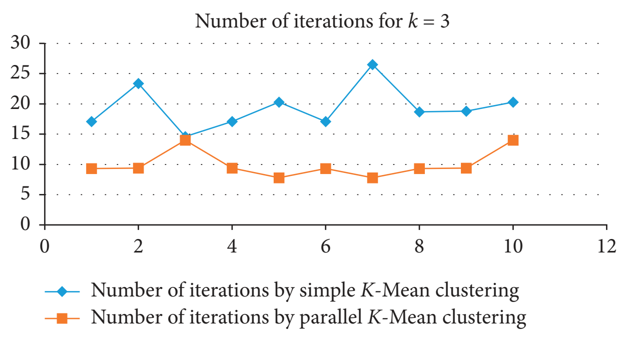

Figure 7: Representation of Table 3.

Table 4: Iterations for $K = 4$.

	For $K = 4$	
Runs/executions	Number of iterations by Simple K-Mean clustering	Number of iterations by Parallel K-Mean clustering
01	15	aph1
02	18	1
03	18	1
04	15	1
05	15	1
06	15	1
07	16	1
08	13	1
09	13	1
10	5	1

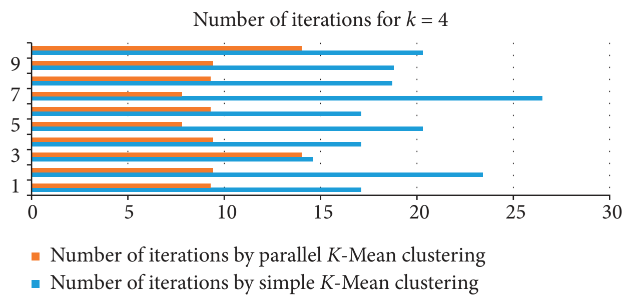

Figure 8: Graph of Table 4.

Table 5: Iterations for $K = 5$.

	For $K = 5$	
Runs/executions	Number of iterations by Simple K-Mean clustering	Number of iterations by Parallel K-Mean clustering
01	14	6
02	10	6
03	7	6
04	8	6
05	13	6
06	18	6
07	16	6
08	26	6
09	28	6
10	26	6

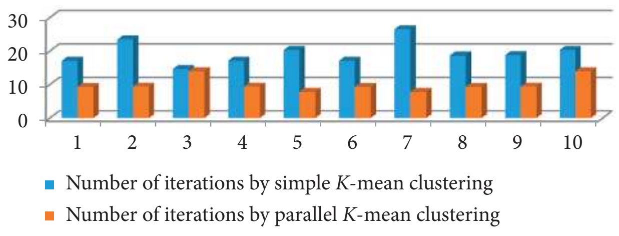

Figure 9: Graph of Table 5.

Table 6: Iterations for $K = 6$.

	.For $K = 6$	
Runs/executions	Number of iterations by Simple K-Mean clustering	Number of iterations by Parallel K-Mean clustering
01	13	2
02	17	2
03	8	2
04	11	2
05	13	2
06	11	2
07	20	2
08	18	2
09	17	2
10	20	2

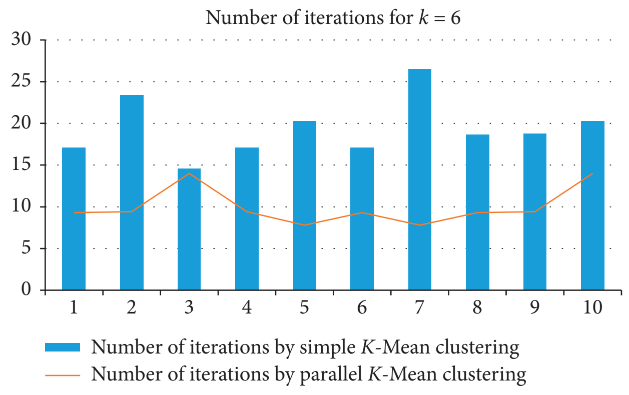

Figure 10: Graph of Table 6. Number of iterations for $K = 6$.

Table 7: Iterations for $K = 7$.

	For $K = 7$	
Runs/executions	Number of iterations by Simple K-Mean clustering	Number of iterations by Parallel K-Mean clustering
01	10	8
02	9	8
03	10	8
04	12	8
05	11	8
06	19	8
07	21	8
08	14	8
09	18	8
10	6	8

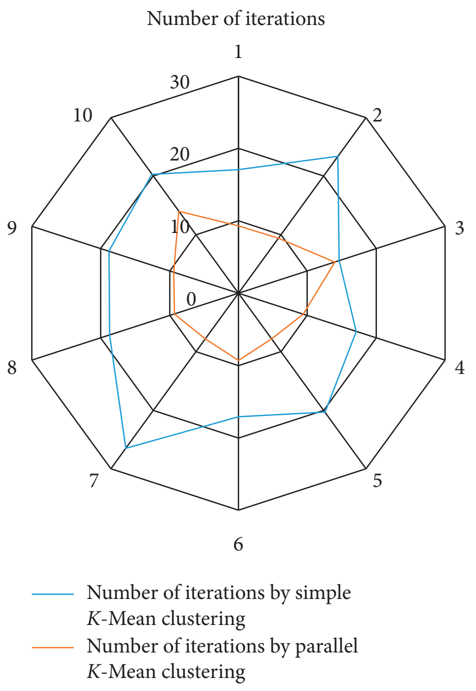

Figure 11: Graph of Table 7. Number of iterations for $K = 7$.

Table 8: Elapsed time for $K = 3$.

	For $K = 3$	
Runs/executions	Elapsed time by Simple K-Mean clustering in ms	Elapsed time by Parallel K-Mean clustering in ms
01	18.7	7.8
02	14.1	7.8
03	20.3	9.3
04	18.7	9.3
05	14.0	9.3
06	20.3	9.4
07	14.6	7.8
08	18.7	9.3
09	14.6	9.3
10	18.7	9.3

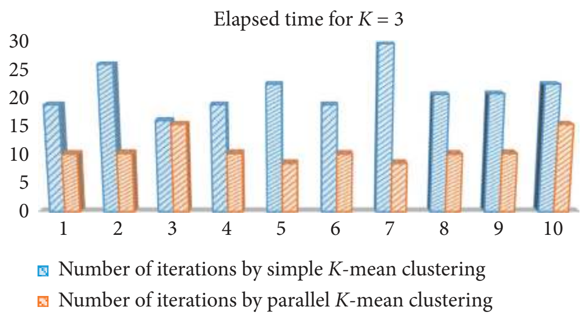

FIGURE 12: Graph of Table 8. Elapsed time for $K = 3$.

TABLE 9: Elapsed time for $K = 4$.

	For $K = 4$	
Runs/executions	Elapsed time by Simple K-Mean clustering in ms	Elapsed time by Parallel K-Mean clustering in ms
01	14.6	7.8
02	18.8	7.8
03	18.7	7.8
04	17.1	6.2
05	18.7	6.2
06	18.8	6.2
07	18.7	6.3
08	14.6	6.2
09	14.6	6.2
10	10.9	6.3

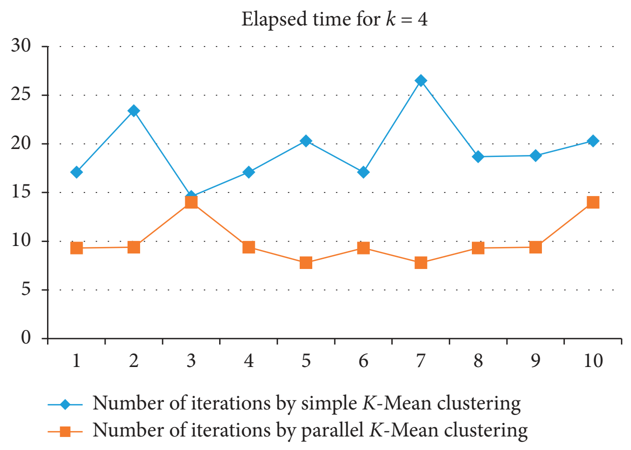

FIGURE 13: Graph of Table 9. Elapsed time for $K = 4$.

TABLE 10: Elapsed time for $K = 5$.

	For $K = 5$	
Runs/executions	Elapsed time by Simple K-Mean clustering in ms	Elapsed time by Parallel K-Mean clustering in ms
01	14.6	10.9
02	14.0	10.9
03	12.5	9.3
04	12.5	10.9
05	14.6	9.3
06	18.7	10.9
07	17.1	10.9
08	23.4	9.4
09	23.4	9.3
10	23.4	9.3

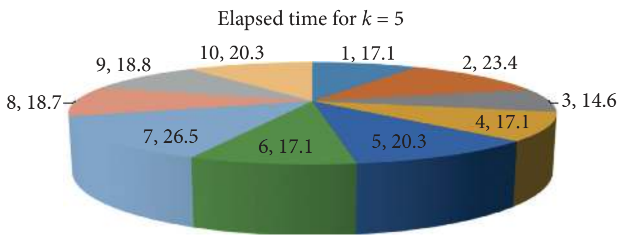

FIGURE 14: Graph of Table 10. Elapsed time for $K = 5$.

TABLE 11: Elapsed time for $K = 6$.

	For $K = 6$	
Runs/executions	Elapsed time by Simple K-Mean clustering in ms	Elapsed time by Parallel K-Mean clustering in ms
01	17.1	9.3
02	23.4	9.4
03	14.6	14.0
04	17.1	9.4
05	20.3	7.8
06	17.1	9.3
07	26.5	7.8
08	18.7	9.3
09	18.8	9.4
10	20.3	14.0

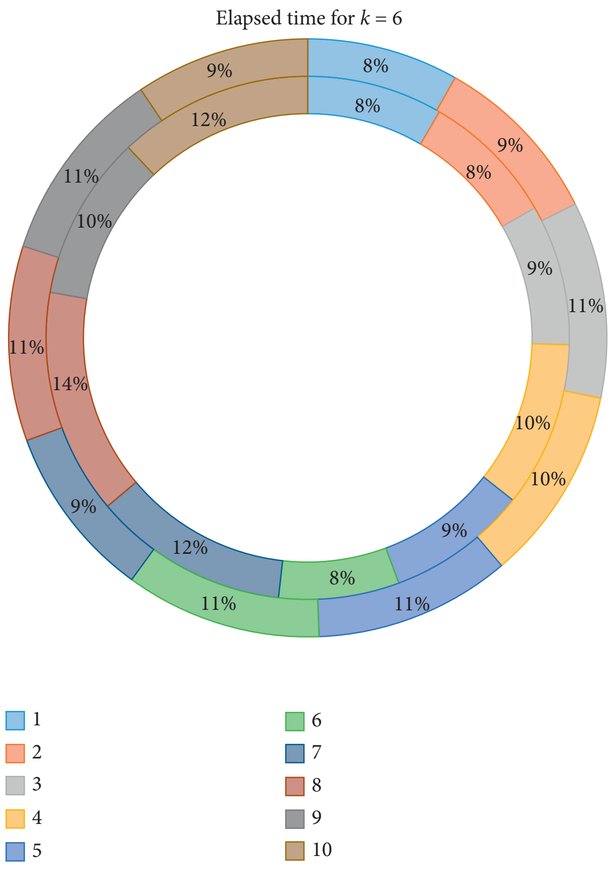

FIGURE 15: Graph of Table 11. Elapsed time for $K = 6$.

Table 12: Elapsed time for $K = 7$.

	For $K = 7$	
Runs/executions	Elapsed time by Simple K-Mean clustering in ms	Elapsed time by Parallel K-Mean clustering in ms
01	14.1	10.9
02	14.0	12.5
03	14.6	14.1
04	17.2	14.0
05	14.6	14.1
06	12.5	14.1
07	20.3	12.5
08	23.4	14.0
09	17.1	14.1
10	20.2	12.5

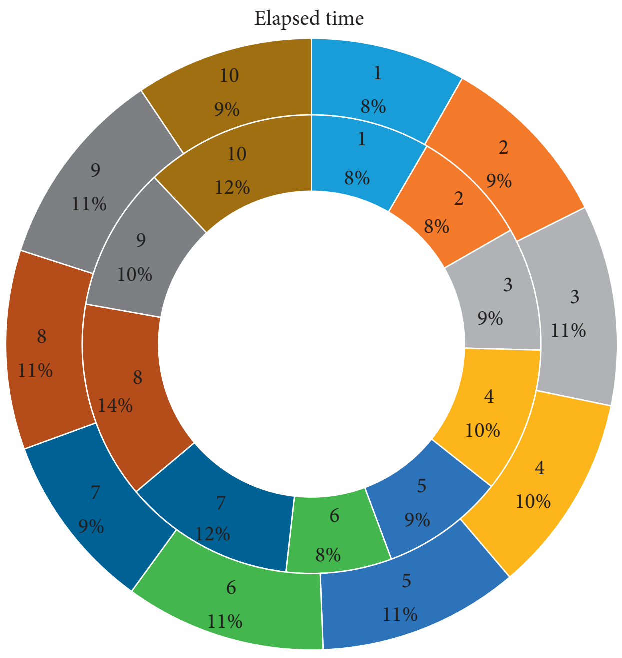

Figure 16: Graph of Table 12. Elapsed time for $K = 7$.

Table 13: Cluster quality of Simple K-Mean clustering for $K = 3$.

S.#	No. of iterations	Elapsed time in ms	# of data items in cluster 1	# of data items in cluster 2	# of data items in cluster 3	Total number of data items
01	12	18.7	3312	2764	3924	10000
02	9	14.1	2838	3456	3706	10000
03	12	20.3	2838	3456	3706	10000
04	9	18.7	3706	2838	3456	10000
05	7	14.0	2764	3312	3924	10000
06	14	20.3	3706	2838	3456	10000
07	9	14.6	3456	2838	3706	10000
08	12	18.7	2764	3312	3924	10000
09	10	14.6	3456	2838	3706	10000
10	15	18.7	3924	3312	2764	10000

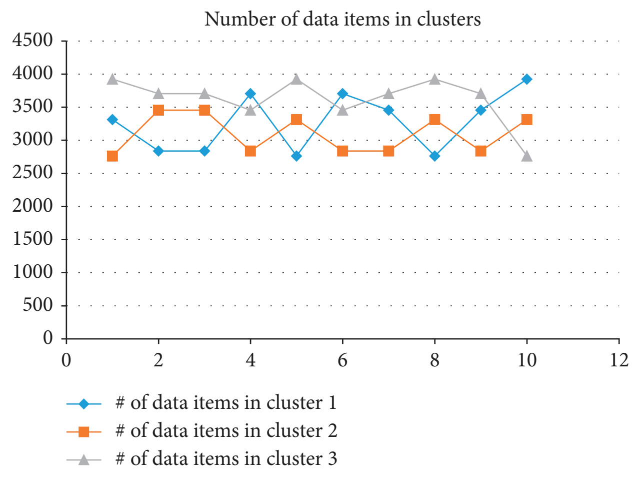

FIGURE 17: Graph of the Table 13. Cluster quality of Simple K-Mean for $K = 3$.

TABLE 14: Cluster quality of Parallel K-Mean clustering for $K = 3$.

S.#	No. of iterations	Elapsed time in ms	# of data items in cluster 1	# of data items in cluster 2	# of data items in cluster 3	Total number of data items
01	3	7.8	3822	2722	3456	10000
02	3	7.8	3822	2722	3456	10000
03	3	9.3	3822	2722	3456	10000
04	3	9.3	3822	2722	3456	10000
05	3	9.3	3822	2722	3456	10000
06	3	9.4	3822	2722	3456	10000
07	3	7.8	3822	2722	3456	10000
08	3	9.3	3822	2722	3456	10000
09	3	9.3	3822	2722	3456	10000
10	3	9.3	3822	2722	3456	10000

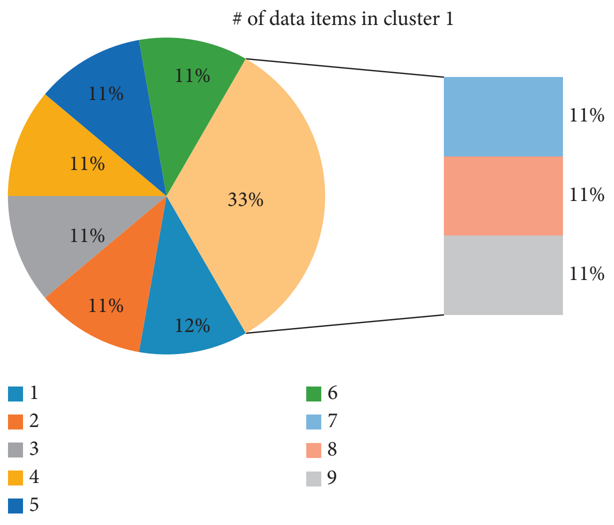

FIGURE 18: Graph of the Table 14. Cluster quality of Parallel K-Mean clustering for $K = 3$.

Table 15: Number of iterations for Simple and Parallel K-Means.

K	Number of iterations	
	Simple K-Mean clustering	Parallel K-Mean clustering
2	7	2
3	15	3
4	9	5

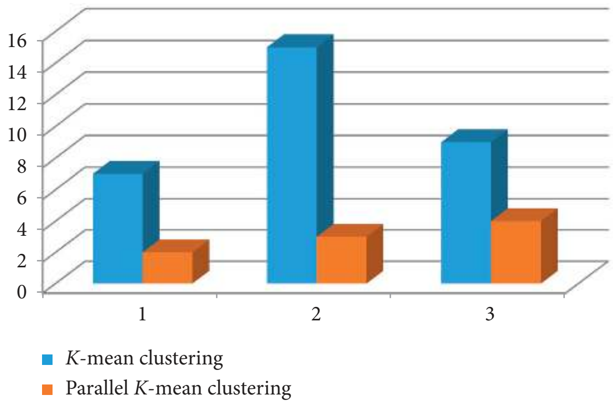

Figure 19: Graph of Table 14. Number of iterations for Simple and Parallel K-Mean algorithms.

Table 16: Elapsed time of Parallel K-Mean clustering for $k = 2$.

$K = 2$					
S#	Client systems	Elapsed time in ms (1^{st}itr + 2^{nd}itr)	# of data items in cluster 1	# of data items in cluster 2	Total data items
1	Client 1	6.0 + 6.0	534	466	1000
2	Client 2	4.0 + 4.0	1000	0	1000
3	Client 3	4.0 + 4.0	1000	0	1000
4	Client 4	4.0 + 4.0	1000	0	1000
5	Client 5	8.0 + 4.0	1000	0	1000
6	Client 6	6.0 + 6.0	552	448	1000
7	Client 7	4.0 + 4.0	0	1000	1000
8	Client 8	4.0 + 4.0	0	1000	1000
9	Client 9	4.0 + 4.0	0	1000	1000
10	Client 10	4.0 + 4.0	0	1000	1000
Total		**12.0**	**5086**	**4914**	**10000**

Table 17: Elapsed time of Simple K-Mean for $k = 2$.

$K = 2$				
S#	Elapsed time	# of data items in cluster 1	# of data items in cluster 2	Total data items
1	14.0	5392	4608	10000

5. Conclusion and Future Work

The main problem in the existing technique is the different outcomes for the same data. In this research, both the Simple and Parallel clustering techniques are implemented and analyzed to point out their best features. The Parallel K-Mean algorithms overcome the problems of Simple algorithm and the outcomes of the parallel algorithms are always same, which improves the cluster quality, number of iterations, and elapsed time. Also, for different runs or execution, the outcomes of the Simple K-Mean are different, so the number of iterations are also different for each run or execution. Experiments were designed to prove that Parallel algorithms are showing significant improvement in the Simple K-Mean algorithm. Outcomes of the Parallel algorithms are also same, while outcomes of the Simple K-Mean algorithm give different results on every run or execution. Also, the number of iterations and elapsed time is improved in the Parallel algorithms. Experiments were designed to prove that Parallel algorithms are showing significant improvement in the Simple algorithm. Some other areas in which research can be extended in our future study as mentioned bellow:

A method for the *K*-Mean clustering should be developed, which works for different nature of data. For example, a method should work better for categorical type of data. Selection of "k" number of clusters is an open research area. The Parallel algorithm is just to reduce elapsed time, number of iterations, and also to develop the quality of elapsed time. In the enhanced framework, number of clusters should be input by the user. Other enhanced algorithms can be extended for the selection of "k" number of clusters. The Parallel algorithm has been worked and tested just for the integer's type of data, which can also be extended for text type of data like words of English language. When a dataset contains different words and all those words are clustered, the same words are placed in the same groups or clusters. A search engine can be introduced using enhanced *K*-Mean clustering algorithm to search for some specific words in a document.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This study was funded by the university patent development project of Heilongjiang Intellectual Property Office in 2019.

References

[1] Q. Cai, H. Zhang, W. Guo et al., "MemepiC: towards a unified in-memory big data management system," *IEEE Transactions on Big Data*, vol. 26, no. 1, 2018.

[2] L. Belcastro, F. Marozzo, D. Talia, and P. Trunfio, "Big data analysis on clouds," in *Handbook of Big Data Technologies*-Springer, Berlin, Germany, 2017.

[3] C. Coronel and S. Morris, "Database systems: design, implementation, & management: nelson education," 2016.

[4] M. K. Bhetwal, "Data warehouse and business intelligence: comparative analysis of olap tools," 2011.

[5] R. Agarwal and M. V. Joshi, "PNrule: a new framework for learning classifier models in data mining (a case-study in network intrusion detection)," in *Proceedings of the 2001 SIAM International Conference on Data Mining*, pp. 1–17, Chicago, IL, USA, April 2001.

[6] B. Banerjee, F. Bovolo, A. Bhattacharya, L. Bruzzone, S. Chaudhuri, and B. K. Mohan, "A new self-training-based unsupervised satellite image classification technique using cluster ensemble strategy," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 4, pp. 741–745, 2015.

[7] P. S. Gamare and G. A. Patil, "Efficient clustering of web documents using hybrid approach in data mining," 2015.

[8] A. Onan, S. Korukoğlu, and H. Bulut, "A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification," *Information Processing & Management*, vol. 53, no. 4, pp. 814–833, 2017.

[9] C. Dugast, P. Beyerlein, and R. Haeb-Umbach, "Application of clustering techniques to mixture density modelling for continuous-speech recognition," in *Proceedings of the Acoustics, Speech, and Signal Processing*, pp. 524–527, Detroit, MI, USA, May 1995.

[10] S. Saha and S. Bandyopadhyay, "Application of a multiseed-based clustering technique for automatic satellite image segmentation," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 2, pp. 306–308, 2010.

[11] X. Zheng and N. Liu, "Color recognition of clothes based on *K*-means and mean shift," in *Proceedings of the Intelligent Control, Automatic Detection and High-End Equipment (ICADE)*, pp. 49–53, Beijing, China, July 2012.

[12] I. A. Maraziotis, S. Perantonis, A. Dragomir, and D. Thanos, "*K*-Nets: clustering through nearest neighbors networks," *Pattern Recognition*, vol. 88, pp. 470–481, 2019.

[13] Y. Jeong, J. Lee, J. Moon, J. H. Shin, and W. D. Lu, "*K*-means data clustering with memristor networks," *Nano Letters*, vol. 18, no. 7, pp. 4447–4453, 2018.

[14] I. A. Pagnuco, J. I. Pastore, G. Abras, M. Brun, and V. L. Ballarin, "Analysis of genetic association using hierarchical clustering and cluster validation indices," *Genomics*, vol. 109, no. 5-6, pp. 438–445, 2017.

[15] A. Sinha and P. K. Jana, "A hybrid MapReduce-based *K*-means clustering using genetic algorithm for distributed datasets," *The Journal of Supercomputing*, vol. 74, no. 4, pp. 1562–1579, 2018.

[16] N. K. Visalakshi and J. Suguna, "*K*-means clustering using Max-min distance measure," in *Proceedings of the NAFIPS 2009-2009 Annual Meeting of the North American Fuzzy Information Processing Society*, pp. 1–6, Lafayette, LI, USA, June 2009.

[17] P. Dahiya and D. K. Srivastava, "A comparative evolution of unsupervised techniques for effective network intrusion detection in hadoop," in *Proceedings of the International Conference on Advances in Computing and Data Sciences*, pp. 279–287, Dehradun, India, April 2018.

[18] A. Onan, "Two-stage topic extraction model for bibliometric data analysis based on word embeddings and clustering," *IEEE Access*, vol. 7, pp. 145614–145633, 2019.

[19] A. Yadav and S. K. Singh, "An improved *K*-means clustering algorithm," *International Journal of Computing*, vol. 5, pp. 88–103, 2016.

[20] S. Ren and A. Fan, "*K*-means clustering algorithm based on coefficient of variation," in *Proceedings of the 2011 4th International Congress on Image and Signal Processing*, pp. 2076–2079, Shanghai, China, October 2011.

[21] T.-A. Hoang and E.-P. Lim, "Modeling topics and behavior of microbloggers," *ACM Transactions on Intelligent Systems and Technology*, vol. 8, no. 3, pp. 1–37, 2017.

[22] A. R. Barakbah and Y. Kiyoki, "A pillar algorithm for *K*-means optimization by distance maximization for initial centroid designation," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Data Mining*, pp. 61–68, Beijing, China, April 2009.

[23] Y. Watanabe, M. Asahara, and Y. Matsumoto, "A graph-based approach to named entity categorization in Wikipedia using

conditional random fields," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 649–657, Prague, Czech Republic, December 2007.

[24] S. Mehrotra and S. Kohli, "Comparative analysis of *K*-means with other clustering algorithms to improve search result," in *Proceedings of the 2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 309–313, Delhi, India, October 2015.

[25] A. K. Jain, "Data clustering: 50 years beyond *K*-means," in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 3-4, Antwerp, Belgium, September 2008.

[26] M. Kumar, P. Chhabra, and N. K. Garg, "An efficient content based image retrieval system using BayesNet and *K*-NN," *Multimedia Tools and Applications*, vol. 77, no. 16, pp. 21557–21570, 2018.

[27] S. Kamal, S. H. Ripon, N. Dey, A. S. Ashour, and V. Santhi, "A MapReduce approach to diminish imbalance parameters for big deoxyribonucleic acid dataset," *Computer Methods and Programs in Biomedicine*, vol. 131, pp. 191–206, 2016.

[28] S. Zahra, M. A. Ghazanfar, A. Khalid, M. A. Azam, U. Naeem, and A. Prugel-Bennett, "Novel centroid selection approaches for KMeans-clustering based recommender systems," *Information Sciences*, vol. 320, pp. 156–189, 2015.

[29] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: a survey on Big Data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[30] Y. Lin, T. Luo, S. Yao, K. Mo, T. Xu, and C. Zhong, "An improved clustering method based on *K*-means," in *Proceedings of the 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 734–737, Sichuan, China, May 2012.

[31] O. A. Folorunso and S. S. Mohd, "Visualising pipeline sensor datasets with modified incremental orthogonal centroid algorithm," *International Journal of Computer Science Issues*, vol. 8, no. 5, 2011.

[32] Y. Wei, X. Zhang, Y. Shi et al., "A review of data-driven approaches for prediction and classification of building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 82, pp. 1027–1047, 2018.

[33] A. Onan, H. Bulut, and S. Korukoglu, "An improved ant algorithm with LDA-based representation for text document clustering," *Journal of Information Science*, vol. 43, no. 2, pp. 275–292, 2017.

[34] S. Saha and S. Bandyopadhyay, "Application of a multiseed-based clustering technique for automatic satellite image segmentation," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, pp. 306–308, 2009.

[35] M. J. Reddy and B. Kavitha, "Clustering the mixed numerical and categorical dataset using similarity weight and filter method," *International Journal of Database Theory and Application*, vol. 5, pp. 121–134, 2012.

[36] C. Dugast, P. Beyerlein, and R. Haeb-Umbach, "Application of clustering techniques to mixture density modelling for continuous-speech recognition," in *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing*, pp. 524–527, Detroit, MI, USA, May 1995.

[37] C. Mary and S. K. Raja, "Refinement OF clusters from *K*-means with ant colony optimization," *Journal of Theoretical & Applied Information Technology*, vol. 6, 2009.

[38] A. Sharma and R. Dhir, "A wordsets based document clustering algorithm for large datasets," in *Proceedings of the 2009 International Conference on Methods and Models in Computer Science (ICM2CS)*, pp. 1–7, Delhi, India, December 2009.

[39] M. Capó, A. Pérez, and J. A. Lozano, "An efficient K-Means clustering algorithm for massive data," 2018, https://arxiv.org/abs/1801.02949.

[40] R. V. Singh and M. S. Bhatia, "Data clustering with modified *K*-means algorithm," in *Proceedings of the 2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, pp. 717–721, Chennai, India, April 2011.

[41] J. Wang and X. Su, "An improved *K*-Means clustering algorithm," in *Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks*, pp. 44–46, Xi'an, China, January 2011.

[42] Y. Zhang, Z. Xiong, J. Mao, and L. Ou, "The study of parallel *K*-means algorithm," *The 6th World Congress on Intelligent Control and Automation*, vol. 26, no. 6, pp. 5868–5871, 2006.

[43] R. T. Aldahdooh and W. M. Ashour, "DIM*K*-means distance-based initialization method for *K*-means clustering algorithm," *DIMK-means Distance-based Initialization Method for K-means Clustering Algorithm*, vol. 5, 2013.

[44] A. E. Top, F. Ş. Torun, and H. Kaya, "Parallel and distributed image segmentation based on colors using *K*-means clustering algorithm," in *Proceedings of the ICES 2019: 5th International Conference on Engineering Sciences*, Ankara, Turkey, September 2019.

[45] V. Ramesh, K. Ramar, and S. Babu, "Parallel *K*-means algorithm on agricultural databases," *International Journal of Computer Science Issues (IJCSI)*, vol. 10, p. 710, 2013.

[46] A. Onan, "A *K*-medoids based clustering scheme with an application to document clustering," in *Proceedings of the 2017 International Conference on Computer Science and Engineering (UBMK)*, pp. 354–359, Antalya, Turkey, October 2017.

[47] B. Bozdemir, S. Canard, O. Ermis, H. Möllering, M. Önen, and T. Schneider, "Privacy-preserving density-based clustering," 2021.

[48] Q. Liao, F. Yang, and J. Zhao, "An improved parallel *K*-means clustering algorithm with MapReduce," in *Proceedings of the 2013 15th IEEE International Conference on Communication Technology*, pp. 764–768, Phoenix Park, South Korea, January 2013.

[49] L. Wang, H. Wang, W. Zhou, and X. Han, "A novel adaptive density-based spatial clustering of application with noise based on bird swarm optimization algorithm," *Computer Communications*, vol. 6, 2021.

[50] E. Y. Cheu, C. Keongg, and Z. Zhou, "On the two-level hybrid clustering algorithm," in *Proceedings of the International Conference on Artificial Intelligence in Science and Technology*, pp. 138–142, Cairns, Australia, June 2004.

[51] S. Shukla and S. Naganna, "A review on *K*-means data clustering approach," *International Journal of Information and Computation Technology*, vol. 4, pp. 1847–1860, 2014.