Review of Computer Engineering Research
2017 Vol. 4, No 2, pp. 54-80
ISSN(e): 2410-9142
ISSN(p): 2412-4281
DOI: 10.18488/journal.76.2017.42.54.80
© 2017 Conscientia Beam. All Rights Reserved.

COMPREHENSIVE ANALYSIS & PERFORMANCE COMPARISON OF CLUSTERING ALGORITHMS FOR BIG DATA

Anand Nayyar[1+]
Vikram Puri[2]

[1]Assistant Professor, Department of Computer Applications & IT, KCL Institute of Management and Technology, Jalandhar
[2]Research Scholar, Department of Electronics and Communications Engineering (ECE) Guru Nanak Dev University (Regional Campus), Jalandhar

(+ Corresponding author)

Article History
Received: 29 May 2017
Revised: 6 October 2017
Accepted: 12 October 2017
Published: 17 October 2017

Keywords
Big data
Clustering
Clustering algorithms
Partitioning-based clustering
Grid-Based clustering
Hierarchical clustering
Density-based clustering
Model based clustering
Data mining.

ABSTRACT

21st Century has marked high velocity of data generation not only in terms of size but also in variety. Analyzing large data sets with different forms is also a challenging task. Data Mining is regarded as efficient method to extract meaningful information as per user requirements. But considering the size of modern data, traditional data mining techniques are failing. Clustering can be regarded as one of the most important technique to mine the data by splitting large data sets into clusters. The paper's primary contribution is to provide comprehensive analysis of Big Data Clustering algorithms on basis of: Partitioning, Hierarchical, Density, Grid and Model. In addition to this, performance comparison of algorithms is performed on basis of volume, variety and velocity.

1. INTRODUCTION

In 21st Century, Big Data has become an indispensable part for research and development cum implementation in various areas of industry and academic. With the frequent usage of same words in different aspects poses a serious issue with regard to the structured evolution of its definition. For this reason, it is utmost necessary to invest time and efforts in proper ratio towards the acceptance of standard and refined definition of Big Data. Big Data as a keyword refers to the increase in data volumes which are difficult to store, process and analyze via traditional database techniques and technologies. The nature of Big Data is highly different and involves sophisticated processes to identify process and translate the data into new insights. The word "Big Data" is new word for Information Technology and Business World. Various researchers and industry professionals have coined various definitions to elaborate the word "Big Data" in various literature forms. The term "Big Data" can be defined as a large volume of scientific data for visualization [1]. The word "Big Data" is characterized by three Vs: Volume, Variety and Velocity. The terms- Volume, Variety and Velocity was originally introduced by Gartner to elaborate various elements of Big Data. Gantz and Reinsel [2] defined Big Data technologies as "A new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data, by enabling the high velocity capture, discovery and/or analysis. Big Data is not only defined by three V's

(Velocity, Volume and Variety) but a new V is added to extend the prime definition of Big Data i.e. Value. Actually, the definition of Big Data is completed only via 4V's [1].

"Big Data can be defined as comprehensive collection of tools, techniques and technologies which requires new integration forms to unbox large hidden values from large amounts of data sets which are diverse, complex and of high scale".

In today's world, voluminous amounts of data are generated by people, things and via their technology interactions. Some of the websites like Google, Twitter, Facebook, Wikipedia, YouTube and many more creates voluminous amounts of data in their data centers almost every hour which is more than tera bytes or Peta-bytes of data [3]. The data online comes from different sources and services which are being developed to cater all the needs of the customers. Various services and resources like Cloud Computing, Social Media, Sensor Data etc. produce high volumes of data and proper management is required so that data can be analyzed and utilized as per user needs. Although the data generated from these sources is advantageous to people and organizations, but management and analysis of data is quite cumbersome process. Therefore, even today big data has lots of shortfalls in managing data. Big data requires high volumes data centers, as high voluminous data requires data operations such as analysis, process and retrieval which is quite a time consuming and tough task. There are several ways which are being proposed by different researchers to solve these types of issues. But the most effective solution till date is, Clustering of Data [2, 4]. Clustering of data means creating clusters of data in compact format which remains informative but available in that size which can be managed and operated effectively. Clustering aims of creating effective clusters of data. Clustering is regarded as Unsupervised Learning technique in which each and every data cluster created contains similar data but is different from other groups.

Clustering is regarded as one of the most important problems to be addressed in the area of Data Mining, Big Data, Machine Learning and Deep Learning. Clustering activity is primarily concerned with discovery of homogeneous groups of data objects. Various researchers in area of Data Mining and Big Data have proposed different Clustering Algorithms but the foremost challenge is the nature and capacity of the data is unknown. So, there is an utmost need of design and development of efficient algorithm for big data to mine sparse, incomplete and uncertain data.

Clustering Algorithms can be divided into various basis: Partition-based; Hierarchical-based; Density-based; Grid-Based and Model-Based.

The aim of this research paper is to provide comprehensive analysis and performance based comparison of various Clustering algorithms of Big Data to provide crystal clear understanding to clustering to researchers to enable them to decide which is the most suitable clustering algorithm according to the situation and to design an effective clustering algorithm by considering the pros and cons of each clustering algorithm for better big data manageability.

1.1. Organization of Paper

Section II presents the complete understanding of Big Data- Introduction, 4 V's of Big Data Characteristics-Volume, Variety, Velocity and Value along with Architecture of Big Data. Section III presents comprehensive analysis of various Big Data clustering algorithms. Section IV enlists various parameters for performance comparison of clustering algorithms. Section V concludes the paper with future scope.

2. BIG DATA

a. Overview

Big Data [5, 6] is regarded as evolving term to specify voluminous amounts of structured, semi-structured and unstructured data that has the potential to be minded for information. The term "Big Data" [7] is regarded as usage of predictive analysis, user behavior analytics or other advanced data analytics techniques to extract desired value from data. Analysis of data sets can be used to determine new relations like "Diseases Prevention", "Current Marketing Policies", "Cyber Crime" etc.

Relational Database Management Systems are inefficient to handle big data based queries. Big Data [8] requires ultra-modern data centers, tools, techniques and high scalable servers to cater the needs of analytics, monitoring and security of data.

Big Data can be classified into different categories in order to understand its characteristics. The Table No:1 highlights the categories of Big Data. The classification of data is done on following parameters like: Sources, Category, Format, Processing, Infrastructure, Frequency, Type, Data Type and Storage, Data Science and Consumer & Data Flow.

Table-1. Big Data Classifications

Classification Type	Examples
Sources	• Enterprise Data like CRM, ERP, e-Commerce, M-Commerce • Machine/Sensor Data- Logs, Meters, Camera Monitoring, Manufacturing Sensors. • Social Media Data: Facebook, Twitter, Pinterest
Category	• Movements or Interactions (Travels, Events) • Behavior or Belongings (Habits of Buying) • Social Network • Machine or Sensor
Format	• Structured, Semi-Structured and Unstructured. • Standard Tables, Images, Audio, Video etc. • Document, Text, XML etc.
Processing	• Descriptive (Statistical, Historical) • Predictive (Forecasting, Recommendation) • Prescriptive (Simulation, What-if Analysis) • Reporting & Scorecards
Infrastructure	• Scale out- Cloud • Scale up- Engineered Machines
Frequency	• Real Time • Batch or Stream
Type	• Meta Data • Master Data, Transactional Data • Historical Data, Analytical Data
Data Type	• Key Value • Document • Graph Association
Data Storage	• Traditional OLAP • Traditional OLTP
Data Science	• Prediction- Decision Trees • Classification- Recommendation • Clustering- Matching Similarity • Association- Neural Networks
Consume & Data Flow	• Public, Internal, Monetized • Business Process, Other Enterprise Systems

Source: https://statswiki.unece.org/display/bigdata/Classification+of+Types+of+Big+Data

b. Characteristics of Big Data

The 4 V's of Big Data can be defined as follows [8]:

a. Volume: Volume is defined as the collection of all types of data generated via different sources and expands continuously. The prime advantage of large amounts of data collection is development of hidden information and patterns via data analysis. At present, the data is measured in Petabytes and will be measured in Zettabytes in near future.

b. Variety: It is defined as different types of data collected via varied sources like Social Media, Online Websites, Website Forums, R&D Groups, Sensor data etc. The data type can be Images, Videos, Text, Audio etc. which can either be structured or unstructured.

c. Velocity: Velocity is defined as data transfer speed as data is coming from different sources and is also continuously and constantly changing because of collections, archived data etc.

d. Value: Value is defined as the process of shortlisting hidden values from large data sets with various types. The aspect of value is highly important for user as the user makes use of Big Data analytics tools to query the database for obtaining result, order the data and save the data for future utilization.

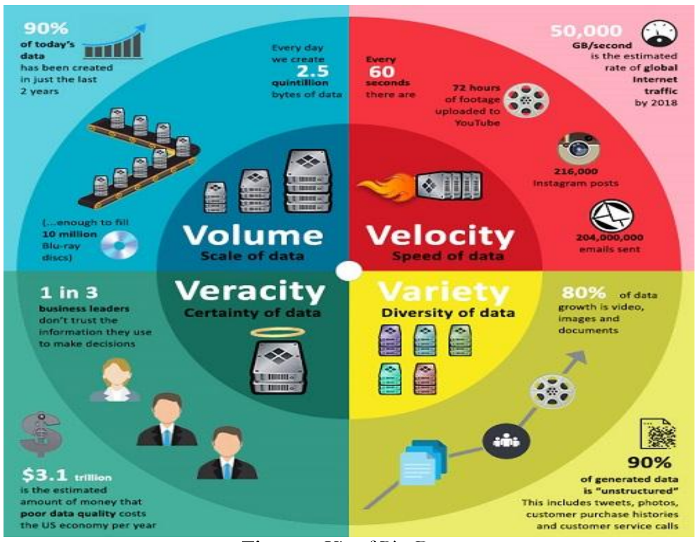

90% of today's data has been created in just the last 2 years
Every day we create 2.5 quintillion bytes of data
(...enough to fill 10 million Blu-ray discs)
Volume
Scale of data
Every 60 seconds there are
72 hours of footage uploaded to YouTube
50,000 GB/second is the estimated rate of global Internet traffic by 2018
216,000 Instagram posts
204,000,000 emails sent
Velocity
Speed of data
Veracity
Certainty of data
1 in 3 business leaders don't trust the information they use to make decisions
$3.1 trillion is the estimated amount of money that poor data quality costs the US economy per year
Variety
Diversity of data
80% of data growth is video, images and documents
90% of generated data is "unstructured"
This includes tweets, photos, customer purchase histories and customer service calls

Fig-1. 4. V's of Big Data
Source: http://www.crewmachine.com/how-ai-and-machine-learning-changing-digital-landscape/

The correspondent criteria of each property of Big Data is enlisted as follows [9]:

- Dataset Type: The traditional clustering algorithms were primarily designed to perform operations on numerical or on categorical data. The data which is collected from the real world contains both numerical and categorical aspects. Traditional clustering algorithms mostly fail on these kinds on data. Clustering algorithms can work efficiently either on numerical data or on categorical data on separate basis; but most of the algorithms fails when data comprise mix elements of both.
- Dataset Size: The dataset size plays a crucial role on clustering quality. Some clustering algorithms perform more efficiently when data is small, and vice versa.
- Input Parameter: The most desired feature for clustering is one that has limited parameters, as large number of parameters effect the quality of cluster because everything depends on parameter values.
- Handling Outliners/Noisy Data: A powerful clustering algorithm is one which handles outliner/noisy data as data collected from almost all sorts of real time applications is not pure. In addition to outliner, noise creates a challenge for the algorithm to cluster the objects into suitable cluster which overall impacts the performance of algorithm.
- Time Complexity: In order to improvise the overall quality of cluster, clustering methods has to be utilized several times. So, in turn, the process of clustering takes long time for operations which makes it unsuitable for applications handling big data.

- Stability: The most important feature of clustering algorithm is the ability to generate same data partitions irrespective of the order in which patterns are presented to the algorithm. Thus, stability is regarded as important parameter for effective clustering.
- Shape of Cluster: Clustering algorithm should have the capability to handle real data along with varied data types, which in turn develops arbitrary shapes of data clusters.
- High Dimensionality handling: Many real-time applications requires analysis of objects with wide range of dimensions. Example: Text Documents contain thousands of keywords which becomes dimensionality challenge. As the dimensions increases, the data becomes sparse, so that the distance measurement between pairs of points becomes meaningless and average density of points anywhere in data becomes low.

c. Architecture of Big Data

As big data is concerned with voluminous amounts of data, it requires proper management, storage, analysis as well as prediction for effective utilization by end user as per its needs. Considering large data sets containing mix of structured, semi-structured and un-structured data, data warehouses would be inefficient to maintain big data because of their 3-tier centralized architecture. Big data requires distributed processing of data and is dealt with special architecture.

3. BIG DATA CLUSTERING ALGORITHMS [9]

Clustering is regarded as the most essential component or feature of data mining. Without clustering, handling big data could become a cumbersome task for analyzing, reporting and querying data. Till date, various clustering algorithms are designed and proposed.

In this section of research paper, various clustering algorithms will be elaborated. Table No: 2 [10] specifies the broad categories along with types of clustering in comprehensive manner.

Table-2. Overview of Clustering Algorithms Taxonomy

Category of Clustering	Name of Clustering Algorithms
Partitioning-Based [11]	• K-means • K-Medoids • K-modes • PAM • CLARANS • CLARA • FCM
Hierarchical-Based	• BIRCH • CURE • ROCK • Chameleon • Echidna
Density-Based	• DBSCAN • OPTICS • DBCLASD • DENCLUE
Grid-Based	• Wave-Cluster • STING • CLIQUE • OptiGrid
Model-Based	• EM • COBWEB • CLASSIT • SOMs

Source: Sajana, et al. [10].

A. Partitioning-Based Clustering Algorithms [9]

In partitioning-based algorithms, the data is distributed into various data subsets. The reason behind this splitting is lack of feasibility to check every possible subset; there are certain greedy probing schemes which are used in form of iterative inflation. In other terms, the partitioning algorithms, performs the task of dividing data objects into number of partitions, each partition is termed as "Cluster".

Cluster should have the following features:

- Every group must contain atleast one object.
- Each object must belong to exactly one group.

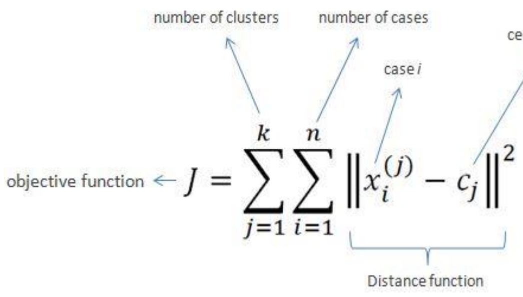
Partitioning based clustering algorithms are: K-means, K-Medoids, K-modes, PAM-Partitioning Around Medoids, CLARA-Clustering Large Applications Methods, CLARANS and FCM-Fuzzy-Cmeans Algorithm.

1. K-Means Algorithm

K-means algorithm is regarded as the most powerful algorithm for discovering structure in data set. K-means clustering method partitions n objects into K clusters in which every object belongs to the cluster with nearest mean [12]. K-means stores k centroids which is used to define cluster. An object is considered to be part of particular cluster, which is closer to cluster's centroid than any other centroid.

In order to determine the best centroid, k-means can make use of any of the following technique:

- Assign data points to cluster based on current centroids.
- Choose centroids (points which are center of a cluster) based on central assignment of data points to cluster.



$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \underbrace{\left\| x_i^{(j)} - c_j \right\|^2}_{\text{Distance function}}$$

Algorithm

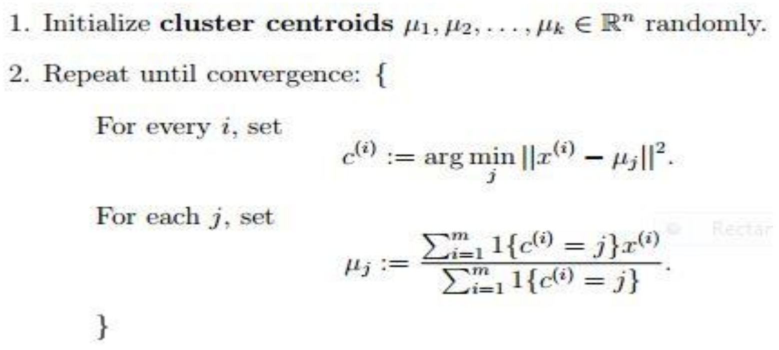
Step 1: Clustering of data into k groups where k is predefined.

Step 2: Select k points at random as cluster centers.

Step 3: Assign Objects to the closest cluster center as per Euclidean Distance function.

Step 4: Determine the Centroid or Mean of all objects in every cluster.

Step 5: Repeat Steps 2, 3 and 4 till the same points are assigned to each cluster in consecutive rounds.



```
1. Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {
       For every $i$, set
           $c^{(i)} := \arg\min_j \|x^{(i)} - \mu_j\|^2.$
       For each $j$, set
           $\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$
   }
```

K-Means Algorithm

The complexity of K-Means algorithm is O(nk) which becomes harder under large nk values. In order to overcome this problem, various sophisticated methodologies have been used in K-means algorithm. Liu, et al. [13] proposed a methodology to reduce the number of clusters at each assignment, a Cluster tree is used for hierarchical k-means algorithm. Silic, et al. [14] proposed Lloyd Algorithm with better simplicity and speed as compared to K-means algorithm.

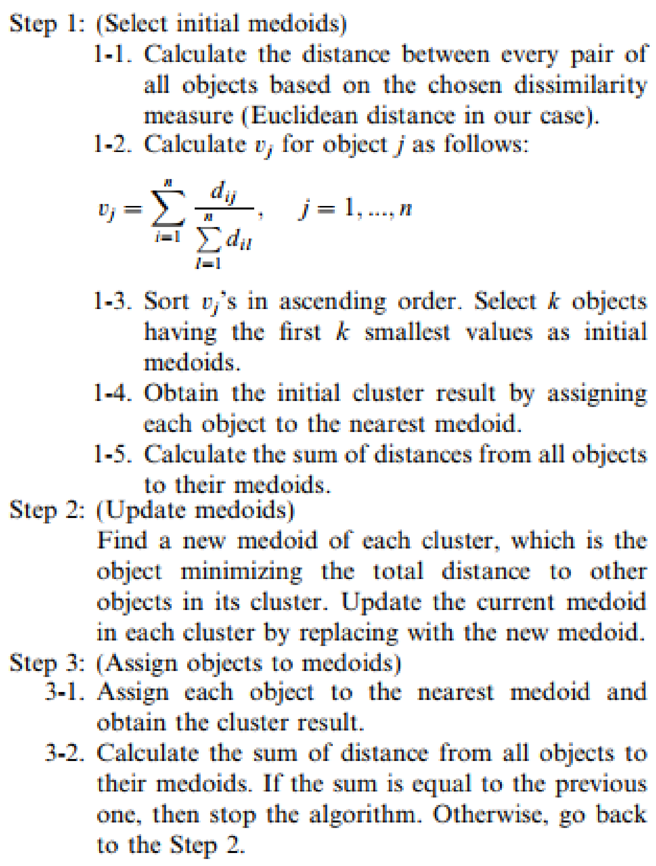
2. K-Medoids Algorithm

The K-Medoids Algorithm [15] a clustering algorithm related to k-means algorithm and medoid shift algorithm. Both the algorithms i.e. k-means and k-medoids are partitional which means breaking the datasets into groups. K-means tries to minimize the total squared error, while k-medoids minimizes the sum of dissimilarities between points labelled to the be in the cluster and a point designated at the center of that cluster. K-medoids algorithm use data points as centres.

It is regarded as powerful partitioning technique which clusters the data set of n objects into k clusters with k known a priori. It is more robust as compared to K-means as it minimizes a sum of general pairwise dissimilarities instead of a sum of squared Euclidean distance.

A medoid is defined as the object of a cluster whose average dissimilarity to all objects in the cluster is minimal i.e. it is regarded as the central located point in the cluster of dataset.

Algorithm



```
Step 1: (Select initial medoids)
    1-1. Calculate the distance between every pair of all objects based on the chosen dissimilarity measure (Euclidean distance in our case).
    1-2. Calculate $v_j$ for object $j$ as follows:
         $v_j = \sum_{i=1}^{n} \frac{d_{ij}}{\sum_{l=1}^{n} d_{il}}, \quad j = 1, \dots, n$
    1-3. Sort $v_j$'s in ascending order. Select $k$ objects having the first $k$ smallest values as initial medoids.
    1-4. Obtain the initial cluster result by assigning each object to the nearest medoid.
    1-5. Calculate the sum of distances from all objects to their medoids.
Step 2: (Update medoids)
    Find a new medoid of each cluster, which is the object minimizing the total distance to other objects in its cluster. Update the current medoid in each cluster by replacing with the new medoid.
Step 3: (Assign objects to medoids)
    3-1. Assign each object to the nearest medoid and obtain the cluster result.
    3-2. Calculate the sum of distance from all objects to their medoids. If the sum is equal to the previous one, then stop the algorithm. Otherwise, go back to the Step 2.
```

K-Medoid Algorithm

3. K-Modes Algorithm

The K-modes Algorithm was proposed by Hartigan and Wong [16].

The primary drawback of k-means clustering algorithm that it is not efficient to cluster categorical data. The K-Mode clustering algorithm is based on K-means algorithm to process the numerical data and is regarded as highly efficient as compared to k-means. K-mode algorithm extends k-means algorithm to cluster categorical data by making the following major modifications [17]:

- Using simple match dissimilar evaluate or hamming distance used for categorical data object.
- Changing means of cluster via modes

Algorithm

Step 1: Generate K clusters via randomly selecting data objects and choosing K as initial cluster center, one for every cluster of data set.

Step 2: Assign data object to the cluster whose cluster is near towards it.

Step 3: Update the k cluster base on data objects allocation and Calculate K latest modes of every one clusters.

Step 4: Repeat Steps 2 to 3 in order to assure that no data object has changed cluster relationship else some additional criterion is fulfilled.

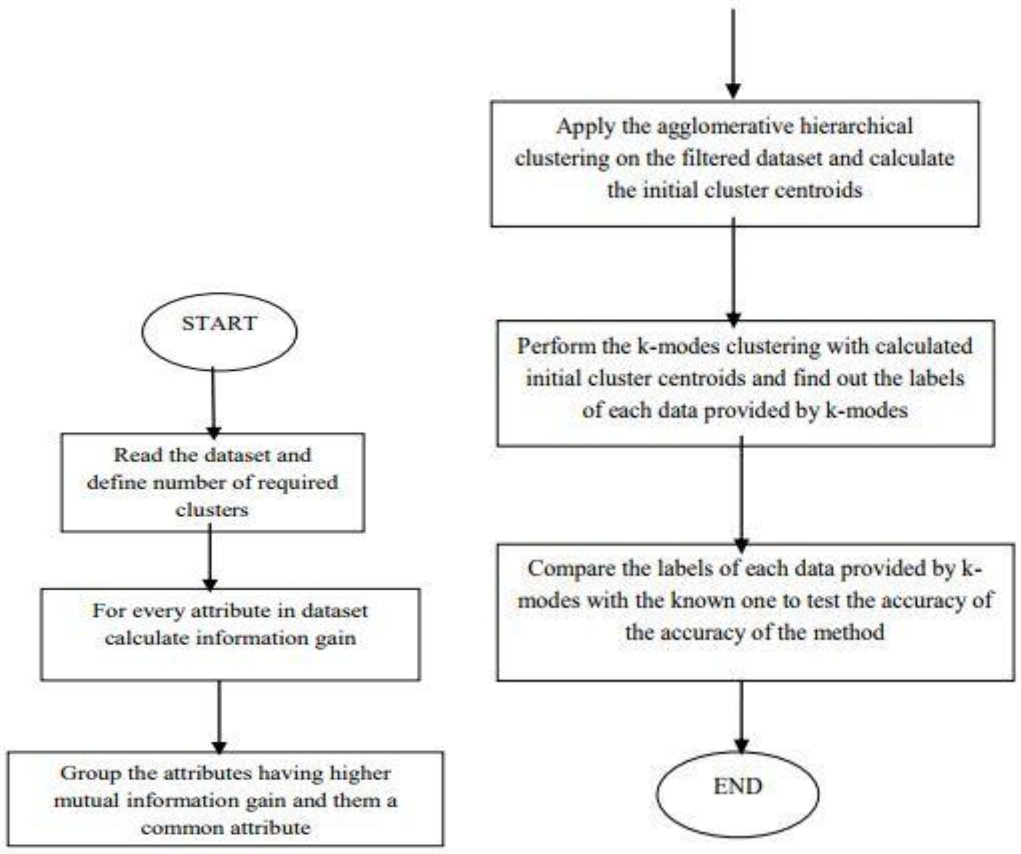

START
Read the dataset and define number of required clusters
For every attribute in dataset calculate information gain
Group the attributes having higher mutual information gain and them a common attribute
Apply the agglomerative hierarchical clustering on the filtered dataset and calculate the initial cluster centroids
Perform the k-modes clustering with calculated initial cluster centroids and find out the labels of each data provided by k-modes
Compare the labels of each data provided by k-modes with the known one to test the accuracy of the accuracy of the method
END

Fig-2. K-modes Algorithm Flowchart [17]

Source: Sharma and Gaud [17].

4. PAM- Partitioning Around Medoids Algorithm

Kaufman and Rousseeuw [18] proposed Partitioning Around Medoids (PAM) algorithm in order to find a sequence of objects called medoids which are located at central point in the clusters. Objects which are called medoids are placed into a set S of selected objects. If O= set of objects then set U= O -S which is regarded as set of unselected objects.

The primary objective behind the development of this algorithm is to minimize the average dissimilarity of objects to their closest selected object.

PAM Algorithm has two phases of operation as follows:

- Phase 1: BUILD: A collection of k objects are selected for an initial set S.
- Phase 2: SWAP: in order to improvise the overall quality of cluster, selected objects are exchanged with unselected objects.

The model utilized by PAM algorithm for solving the problem is:

$$F(x) = \text{minimize} \sum_{i=1}^{n} \sum_{j=1}^{n} d(i,j) z_{ij}$$

Subject to:

1. $\Sigma_{i=1}^{n} z_{ij} = 1$, $j = 1,2,\ldots,n$
2. $z_{ij} \leq y_i$, $i, j = 1,2,\ldots,n$
3. $\Sigma_{i=1}^{n} y_i = k$, k = number of clusters
4. y_i , z_{ij} € $\{0,1\}$, $i, j = 1,2,\ldots,n$

```
PAM Algorithm Input: E = {e1,e2,...en} (dataset to be clustered or matrix of dissimilarity)
        k (number of clusters)
        metric (kind of metric to use on dissimilarity matrix)
        diss (flag indicating that E is the matrix of dissimilarity or not)
Output: M = {m1,m2,...,mk} (vector of clusters medoids)
        L = {l(e) | e = 1,2,...,n} (set of cluster labels of E)
foreach $m_i$ € M do
    $m_i \leftarrow e_j$ € E; (e.g. random selection)
end if diss ≠ true
    Dissimilarity ← CalculateDissimilarityMatrix(E, metric);
else
    Dissimilarity ← E;
end repeat
    foreach $e_i$ € E do
        $l(e_i)$ ← argminDissimilarity($e_i$, Dissimilarity, M);
    end
    changed ← false;
    foreach $m_i$ € M do
        Mtmp ← SelectBestClusterMedoids(E, Dissimilarity, L);
    end
    if Mtmp ≠ M
        M ← Mtmp;
        changed ← true;
    end
until changed = true;
```

PAM Algorithm

5. CLARA-Clustering Large Applications Algorithm

Kaufman and Rousseeuw [19] proposed CLARA-Clustering Large Application Methods, clustering algorithm for handling large data sets and is based on sampling technique. The algorithm extends K-medoids approach to large number of objects by clustering a sample from dataset and assigns all objects in the dataset to these clusters.

The objects are drawn in sufficiently random way, the medoids of the sample would approximate the medoids of the entire data set.

Considering this, CLARA draws multiple samples and is considered as best Partitioning Clustering Algorithm considering the output.

$$Cost(M, D) = \frac{\sum_{i=1}^{n} dissimilarity(O_i, rep(M, O_i))}{n}$$

M= Set of Medoids selected; dissimilarity (Oi, Oj) is the dissimilarity between object Oi and Oj and rep (m, Oi) returns medoid in M which is closest to Oi.

Pseudocode

```
CLARA($X, d, k$)
    $bestDissim \leftarrow \infty$
    for $t \leftarrow 1$ to $S$
    do $X' \leftarrow$ RANDOM-SUBSET($X, s$)
       $D \leftarrow$ BUILD-DISSIM-MATRIX($X', d$)
       $(C', M) \leftarrow$ PAM($X', D, k$)
       $C \leftarrow$ ASSIGN-MEDOIDS($X, M, D$)
       $dissim \leftarrow$ TOTAL-DISSIM($C, M, D$)
       if $dissim < bestDissim$
         then $bestDissim \leftarrow dissim$
              $C_{best} \leftarrow C$
              $M_{best} \leftarrow M$
    return $(C_{best}, M_{best})$
```

Algorithm

Experiments performed by Kaufman and Rousseeuw [19] demonstrate 5 samples of size 40 + 2k yields best results as under:

Step 1: For i= 1 to 5, repeat the following steps as follows:

Step 2: Draw a sample of 40 + 2k objects randomly from entire data set and implement PAM Algorithm to determine k medoid of the sample.

Step 3: For each Object Oj, in entire dataset, determine which k medoid is almost similar to Oj.

Step 4: Calculate the average dissimilarity of the clustering. If the value is less than current minimum, use as current minimum and determine the k medoids found in step: 2 as the best medoid determined.

Step 5: Back to Step for next.

As compared to PAM, CLARA is regarded as best clustering algorithm for 1000 objects in 10 clusters.

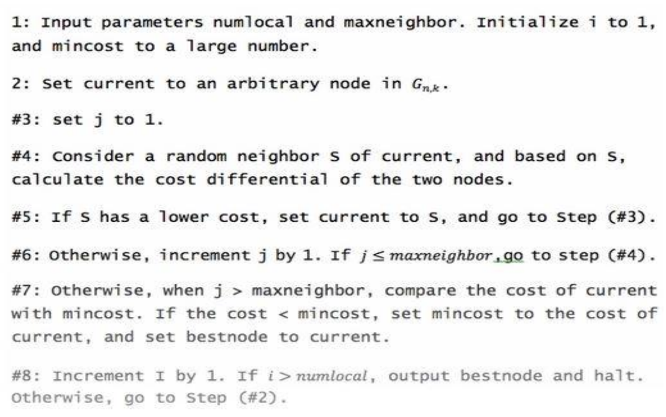
6. CLARANS

Ng and Han [20] proposed CLARANS clustering algorithm primarily designed and proposed to use random search for clustering of large number of objects. It is regarded as highly efficient medoid-based clustering algorithm.

In CLARANS Algorithm, the procedure to determine K medoids from n objects is determined abstractly as searching through certain graph. In the graph, every node is represented by set of k objects as medoids selected.

CLARANS algorithm consider randomly chosen neighboring nodes as candidate of new medoids.

As compared to CLARA, CLARANS gives better clustering results by making few searches.



```
1: Input parameters numlocal and maxneighbor. Initialize i to 1,
and mincost to a large number.
2: Set current to an arbitrary node in $G_{n,k}$.
#3: set j to 1.
#4: Consider a random neighbor S of current, and based on S,
calculate the cost differential of the two nodes.
#5: If S has a lower cost, set current to S, and go to Step (#3).
#6: Otherwise, increment j by 1. If $j \le maxneighbor$, go to step (#4).
#7: Otherwise, when j > maxneighbor, compare the cost of current
with mincost. If the cost < mincost, set mincost to the cost of
current, and set bestnode to current.
#8: Increment I by 1. If $i > numlocal$, output bestnode and halt.
Otherwise, go to Step (#2).
```

CLARANS algorithms makes use of two parameters: The Max number of neighbors examined- maxneighbor, the number of local minima obtained – numlocal.

If the value of maxneighbor is higher, CLARANS is close to PAM.

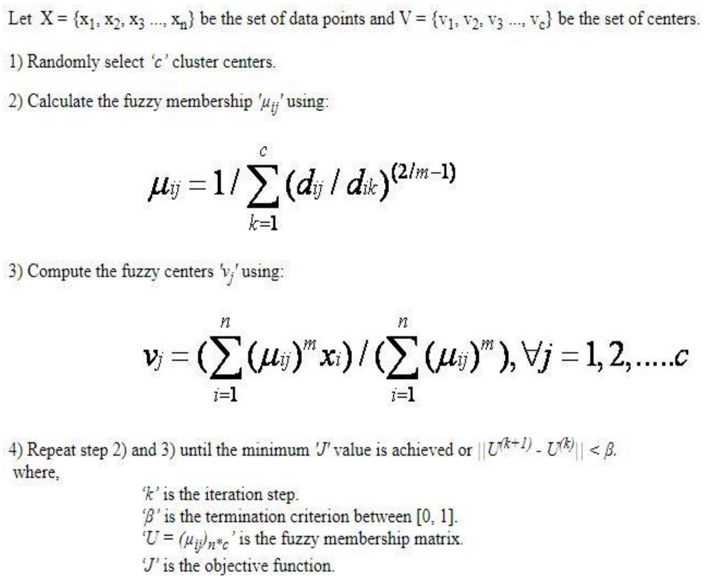
So, CLARANS is regarded as more robust, scalable and efficient as compared to PAM and CLARA.

7. Fuzzy C-Means Clustering

Fuzzy C-Means Clustering algorithm was proposed by Bezdek, et al. [21]. It is based on k-means for partitioning datasets into clusters. It is regarded as "Soft" clustering algorithm in which objects are assigned to clusters via degree of belief. The algorithm first determines the cluster center, then it computes the membership degree of each object in the cluster.

The algorithm also incorporates some shortcomings of K-means algorithm, as the minimum is just the local one and the final clusters depend on choice of weights.

The Algorithm basically works by a membership to each data point corresponding to each cluster center on basis of distance between cluster center and data point. More the data is near to the cluster center, more is its membership towards particular cluster center. Membership of each data point should be equal to one.



Let $X = \{x_1, x_2, x_3 ..., x_n\}$ be the set of data points and $V = \{v_1, v_2, v_3 ..., v_c\}$ be the set of centers.

1) Randomly select 'c' cluster centers.

2) Calculate the fuzzy membership 'μ_{ij}' using:

$$\mu_{ij} = 1 / \sum_{k=1}^{c} (d_{ij} / d_{ik})^{(2/m-1)}$$

3) Compute the fuzzy centers 'v_j' using:

$$v_j = (\sum_{i=1}^{n} (\mu_{ij})^m x_i) / (\sum_{i=1}^{n} (\mu_{ij})^m), \forall j = 1, 2, ...c$$

4) Repeat step 2) and 3) until the minimum 'J' value is achieved or $||U^{(k+1)} - U^{(k)}|| < \beta$.

where,

'k' is the iteration step.

'β' is the termination criterion between [0, 1].

'$U = (\mu_{ij})_{n*c}$' is the fuzzy membership matrix.

'J' is the objective function.

Fuzzy C-means Clustering Algorithm

The algorithm tries to minimize the Objective Function by:

$$\arg\min_{C} \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^{m} \|\mathbf{x}_i - \mathbf{c}_j\|^2,$$

where:

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|\mathbf{x}_i - \mathbf{c}_j\|}{\|\mathbf{x}_i - \mathbf{c}_k\|} \right)^{\frac{2}{m-1}}}.$$

B. Hierarchical Based Clustering Algorithms

Hierarchical based clustering algorithms basically make data groups to create a tree based structure. They are also popularly known as "Connectivity based Clustering Algorithms". These algorithms can be further classified into two categories: Agglomerative Hierarchical Clustering and Divisive Hierarchical Clustering.

Agglomerative Clustering makes use of Bottom Up Approach, where every data point is considered as separate cluster and on every iteration performed, clusters are merged on basis of criteria.

Under Divisive Approach, all data points are under single cluster and divided into separate clusters and makes use of Top Down Approach.

The major clustering algorithms under Hierarchical Clustering are: BIRCH, CURE, ROCK, Chameleon.

1. BIRCH- Balanced Iterative Reducing and Clustering using Hierarchies

BIRCH, an unsupervised data mining algorithm to carry out hierarchical clustering over large data sets was proposed by [22, 23].

BIRCH algorithm develops a dendrogram know as Clustering Feature Tree (CF Tree). The CF tress can be developed by scanning the dataset in an incremental and dynamic way. So, BIRCH algorithm has no requirement to have the whole dataset well in advance.

3.1. BIRCH Clustering Algorithm

Phase 1: Scan whole of the data and build an initial CF tree using the memory given and recycling space on disk.
Phase 2: Condense the Data: Rebuilt the CF tree with Larger T.
Phase 3: Global Clustering: Make use of some standardized clustering algorithms like k-means, k-modes etc.
Phase 4: Cluster Refining: Making additional passes over the datasets and reassign data points to closest centroids.
Iterate the steps 1 to 4 to form k number of clusters.

BIRCH algorithm is tested as compared to K-Means and CLARANS. CLARANS makes use of graph partitions and searches it locally to get the best one. Random 2-D datasets of K=100 clusters. BIRCH performs exceptionally well in terms of less memory consumption, faster performance, less order-sensitive and highly accurate and even highly scalable as compared to both the algorithms.

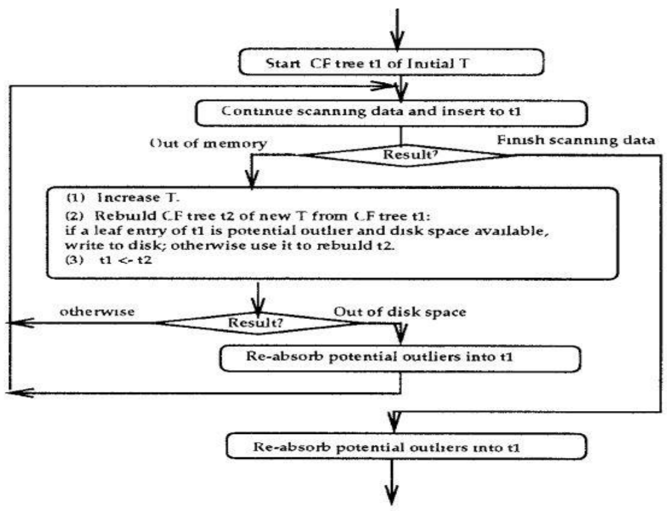

Start CF tree t1 of Initial T
Continue scanning data and insert to t1
Out of memory
Finish scanning data
Result?
(1) Increase T.
(2) Rebuild CF tree t2 of new T from CF tree t1:
if a leaf entry of t1 is potential outlier and disk space available,
write to disk; otherwise use it to rebuild t2.
(3) t1 <- t2
otherwise
Out of disk space
Result?
Re-absorb potential outliers into t1
Re-absorb potential outliers into t1

Fig-3 BIRCH Algorithm
Source: Zhang, et al. [22].

2. CURE-Clustering Using Representatives

Guha, et al. [24] data clustering algorithm for large databases having significant edge in terms of robustness and identifying clusters having non-spherical shapes and variances in size.

CURE makes use of Divisive approach and select all well scattered points from the cluster and then performs the process of shrinking towards the cluster using specified function.

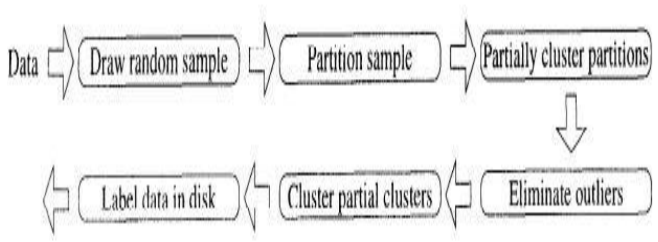
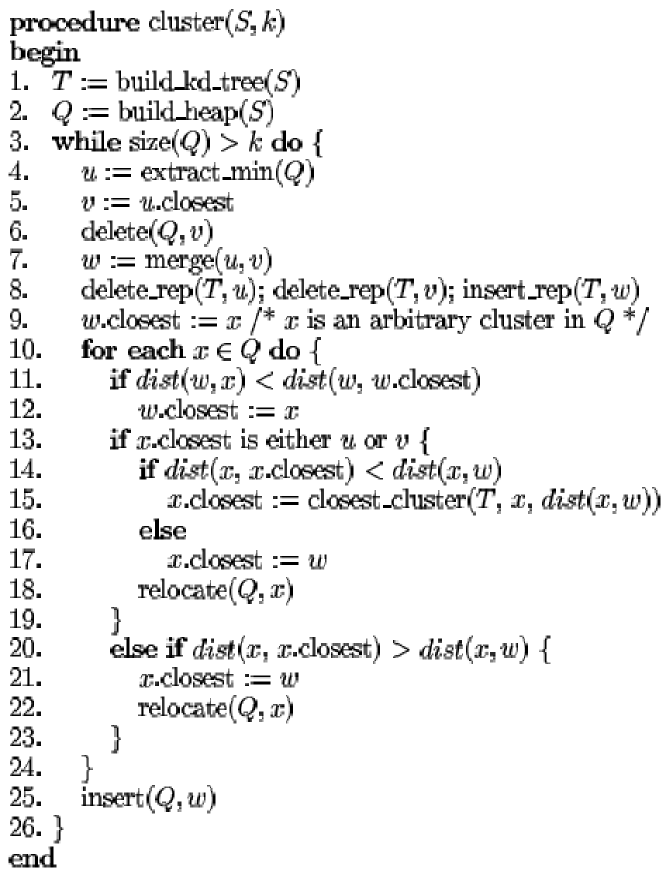


Fig-4. CURE Algorithm Working
Source: Guha, et al. [24].

Algorithm

- All points in every cluster are initialized, and every cluster is identified by specific point.
- The Representative points of a cluster are created by selecting well scattered objects for the cluster and then shrinking is performed towards the cluster by specified factor. This in turn improvises the speed of CURE algorithm working.
- At every step of CURE algorithm, any two clusters with closest representative points are selected and merged together to create a cluster.



```
procedure cluster($S$, $k$)
begin
1.  $T$ := build_kd_tree($S$)
2.  $Q$ := build_heap($S$)
3.  while size($Q$) > $k$ do {
4.      $u$ := extract_min($Q$)
5.      $v$ := $u$.closest
6.      delete($Q$, $v$)
7.      $w$ := merge($u$, $v$)
8.      delete_rep($T$, $u$); delete_rep($T$, $v$); insert_rep($T$, $w$)
9.      $w$.closest := $x$ /* $x$ is an arbitrary cluster in $Q$ */
10.     for each $x \in Q$ do {
11.         if $dist(w, x) < dist(w, w.\text{closest})$
12.             $w$.closest := $x$
13.         if $x$.closest is either $u$ or $v$ {
14.             if $dist(x, x.\text{closest}) < dist(x, w)$
15.                 $x$.closest := closest_cluster($T$, $x$, $dist(x, w)$)
16.             else
17.                 $x$.closest := $w$
18.             relocate($Q$, $x$)
19.         }
20.         else if $dist(x, x.\text{closest}) > dist(x, w)$ {
21.             $x$.closest := $w$
22.             relocate($Q$, $x$)
23.         }
24.     }
25.     insert($Q$, $w$)
26. }
end
```

CURE Clustering Algorithm

CURE Algorithm is compared with BIRCH Algorithm and the results state that CURE employs random sampling and partitioning methodology which can handle large data sets effectively. In addition to this, CURE is equipped with Labeling algorithm which uses multiple random representative points for each cluster to assign data points on disk. CURE is compared with BIRCH and results shows it is highly scalable, robust and fast as compared to BIRCH in data set clustering procedures.

3. ROCK- Robust Clustering Algorithm

ROCK, a hierarchical clustering algorithm was proposed by Guha, et al. [25] which deploys links not distances when performing cluster mergers.

Algorithm

Step 1: A random sample from the data set is selected, the algorithm employs links to the sampled points.

Step 2: Finally, the clusters involving only sampled points are used to assign the remaining data points on disk to appropriate clusters.

A Goodness measure is used to determine criterion function to determine the best pairs of clusters to merge at each step.

Algorithm

```
procedure cluster($S, k$)
begin
1.  $link$ := compute_links($S$)
2.  for each $s \in S$ do
3.      $q[s]$ := build_local_heap($link, s$)
4.  $Q$ := build_global_heap($S, q$)
5.  while size($Q$) > $k$ do {
6.      $u$ := extract_max($Q$)
7.      $v$ := max($q[u]$)
8.      delete($Q, v$)
9.      $w$ := merge($u, v$)
10.     for each $x \in q[u] \cup q[v]$ do {
11.         $link[x, w]$ := $link[x, u] + link[x, v]$
12.         delete($q[x], u$); delete($q[x], v$)
13.         insert($q[x], w, g(x, w)$); insert($q[w], x, g(x, w)$)
14.         update($Q, x, q[x]$)
15.     }
16.     insert($Q, w, q[w]$)
17.     deallocate($q[u]$); deallocate($q[v]$)
18. }
end
```

ROCK Algorithm

ROCK algorithm is tested on real-life as well as data sets of synthetic data and results reveal that link-based approach of ROCK is best for cluster forming, cluster merging and other cluster based operations.

4. CHAMELEON

Karypis, et al. [26] proposed dynamic modeling based hierarchical clustering algorithm to determine the similarity of two clusters.

The CHAMELEON algorithm basically operates on sparse graph in which nodes represent data items and weighted edges highlight similarities between data items. It is only because of sparse graph that the algorithm can scale to large data sets that are available in similarity space not in metric space.

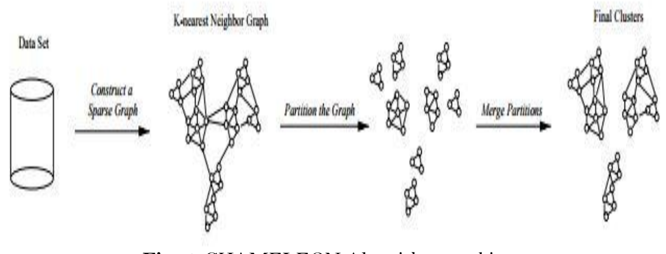


Fig-5. CHAMELEON Algorithm working
Source: Karypis, et al. [26].

Algorithm:

Phase 1: Finding Initial Sub-Clusters: It determines the initial sub-clusters using graph partitioning algorithm to partition k-nearest neighbor graph of the data set into a large number of partitions such as edge-cut using hMETIS method.

Phase 2: Merging Sub-Clusters using Dynamic Framework: It makes use of dynamic modeling framework to select the most similar pairs of clusters by considering relative inter-connectivity and relative closeness.

Performance Comparison: CHAMELEON is compared with DBSCAN and CURE and results state that CHAMELEON outshines in terms of clustering of data sets.

5. Echidna

Mahmood, et al. [27] proposed Echidna, an agglomerative hierarchical clustering algorithm for network traffic data.

The most important limitation for performing clustering in multi-dimensional network traffic data is to deal with wide range of attributes like: Numerical, Categorical etc. So, Echidna algorithm performs suitable clustering all sorts of network traffic parameters.

Algorithm:

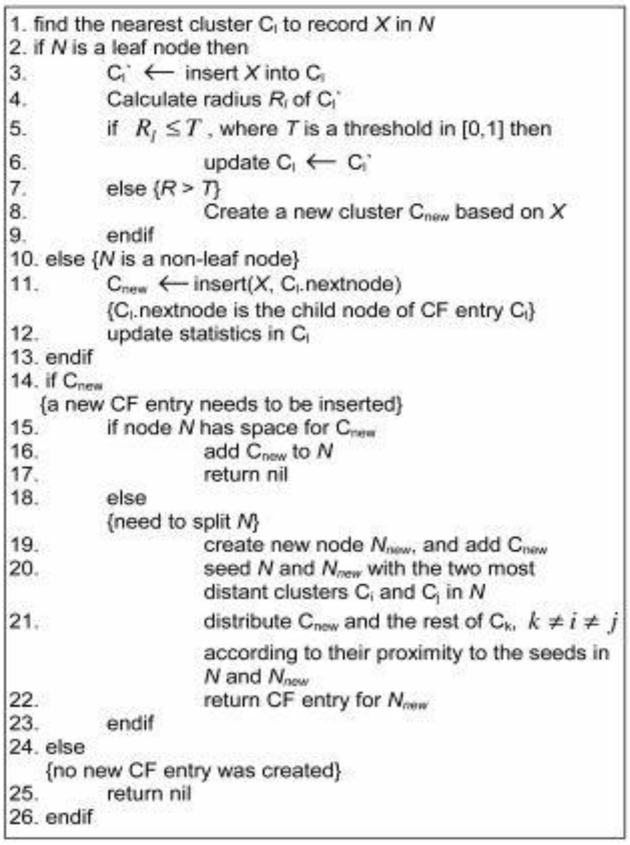
Step 1: Network Traffic data which contains 6-tuple value is extracted on basis of numerical and categorical attributes.

Step 2: Every record creates hierarchical cluster tree called CF-Tree.

Step 3: Insert each record to closest cluster using combined distance function for all attributes into CF-Tree.

Step 4: Considering whether the record should be absorbed or split is done by radius of cluster.

Step 5: Once the cluster is created, all nodes combine to form cluster tree.



```
1. find the nearest cluster $C_i$ to record $X$ in $N$
2. if $N$ is a leaf node then
3.          $C_i' \leftarrow$ insert $X$ into $C_i$
4.          Calculate radius $R_i$ of $C_i'$
5.          if $R_i \le T$, where $T$ is a threshold in [0,1] then
6.                   update $C_i \leftarrow C_i'$
7.          else {$R > T$}
8.                   Create a new cluster $C_{new}$ based on $X$
9.          endif
10. else {$N$ is a non-leaf node}
11.         $C_{new} \leftarrow$ insert($X$, $C_i$.nextnode)
            {$C_i$.nextnode is the child node of CF entry $C_i$}
12.         update statistics in $C_i$
13. endif
14. if $C_{new}$
    {a new CF entry needs to be inserted}
15.         if node $N$ has space for $C_{new}$
16.                  add $C_{new}$ to $N$
17.                  return nil
18.         else
            {need to split $N$}
19.                  create new node $N_{new}$, and add $C_{new}$
20.                  seed $N$ and $N_{new}$ with the two most
                     distant clusters $C_i$ and $C_j$ in $N$
21.                  distribute $C_{new}$ and the rest of $C_k$, $k \ne i \ne j$
                     according to their proximity to the seeds in
                     $N$ and $N_{new}$
22.                  return CF entry for $N_{new}$
23.         endif
24. else
    {no new CF entry was created}
25.         return nil
26. endif
```

Echidna Clustering Algorithm

C. Density-Based Clustering Algorithms

In Density-based Clustering Algorithms, data objects are segregated on basis on density regions, boundary and connectivity. In this approach, clusters are determined in arbitrary manner, where clusters are called as dense regions separated by low density regions.

Density based clustering algorithms are not suitable for larger data sets.

The most important Density-based clustering Algorithms are:

DBSCAN, OPTICS, DBCLASD, DENCLUE.

1. DBSCAN- Density Based Scan Algorithm

DBSCAN was proposed by Ester, et al. [28] is a density-based clustering algorithm which is designed to discover clusters and identify noise in spatial database.

In this algorithm, dense regions are called clusters and low dense regions are called noise.

Algorithm:

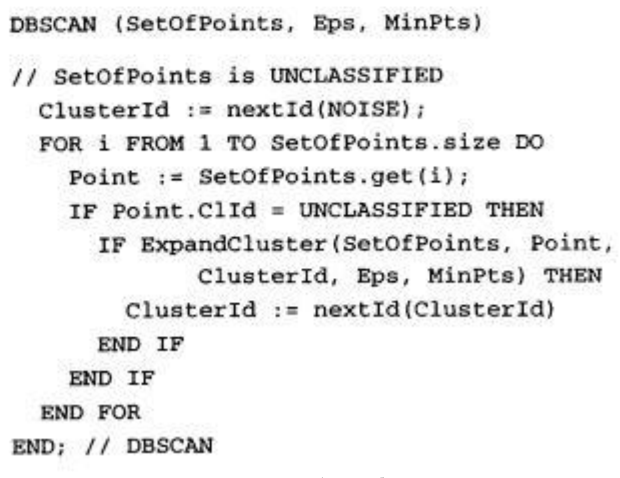
Step 1: Select a point r arbitrarily.

Step 2: Capture all points that are density-reachable from Eps and MinPTS.

Step 3: If r is core point, cluster is created.

Step 4: If r is border point, no points are density reachable from r then DBSCAN traverses to next point of data set.

Step 5: Repeat Steps 1-4 iteratively till all points are processed.



```
DBSCAN (SetOfPoints, Eps, MinPts)

// SetOfPoints is UNCLASSIFIED
  ClusterId := nextId(NOISE);
  FOR i FROM 1 TO SetOfPoints.size DO
    Point := SetOfPoints.get(i);
    IF Point.ClId = UNCLASSIFIED THEN
      IF ExpandCluster(SetOfPoints, Point,
             ClusterId, Eps, MinPts) THEN
        ClusterId := nextId(ClusterId)
      END IF
    END IF
  END FOR
END; // DBSCAN
```

DBSCAN Algorithm

DBSCAN is compared with CLARANS algorithm on basis of SEQUOIA 2000 benchmark data. The results demonstrate that DBSCAN is higher in number of points and DBSCAN outshines CLARANS in terms of speed and robustness.

2. OPTICS-Ordering Points to Identify the Clustering Structure

Optics, a connectivity based clustering algorithm was proposed by Ankerst, et al. [29] based on DBSCAN algorithm and overcomes some shortcomings of DBSCAN i.e. the problem of detecting meaningful clusters in data of varied densities.

Considering DBSCAN, OPTICS required two parameters ϵ, which defines the maximum distance (radius) to be considered and MinPts which describe the number of points to be considered for creating cluster.

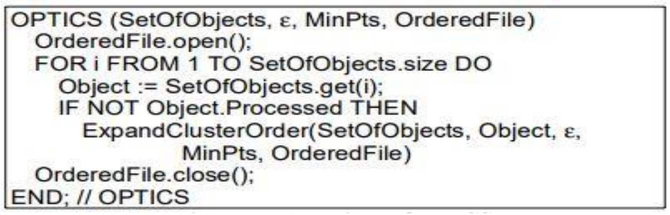
OPTICS algorithm is much more efficient as compared to DBSCAN as it also considers points that are part of more densely packed cluster, so that every point is assigned a core distance that describe the distance to the MinPts closest point.

Terminologies:

- Core Objects: ϵ-Neighborhood of an object contains at least MinPts of objects.
- Directly Density Reachable: An Object q is directly density-reachable from object p if q is within the ϵ-Neighborhood of p and p is core object.

- Density Reachable: An object p is density reachable from q w.r.t. ϵ and MinPts if there is a chain of objects p1 to pn.

Algorithm:



```
OPTICS (SetOfObjects, ε, MinPts, OrderedFile)
  OrderedFile.open();
  FOR i FROM 1 TO SetOfObjects.size DO
    Object := SetOfObjects.get(i);
    IF NOT Object.Processed THEN
      ExpandClusterOrder(SetOfObjects, Object, ε,
                MinPts, OrderedFile)
  OrderedFile.close();
END; // OPTICS
```

OPTICS Algorithm

Step 1: Start with an Arbitrary object from the input database as the current object p.

Step 2: It retrieves the ϵ-Neighborhood of p, determines the core-distance and sets the reachability-distance to undefined.

The current object, p, is written to output.

Step 3: If p is not a core Object, OPTICS simply moves on to the next object in the OrderSeeds list.

Step 4: If p is core object, then for each object, q, in the ϵ-Neighborhood of p, Optics updates its reachability-distance from p and inserts q into OrderSeeds if q has not yet been processed.

Step 5: Iteration continues from step 1 to 4 until the input is finally consumed and OrderSeeds is empty.

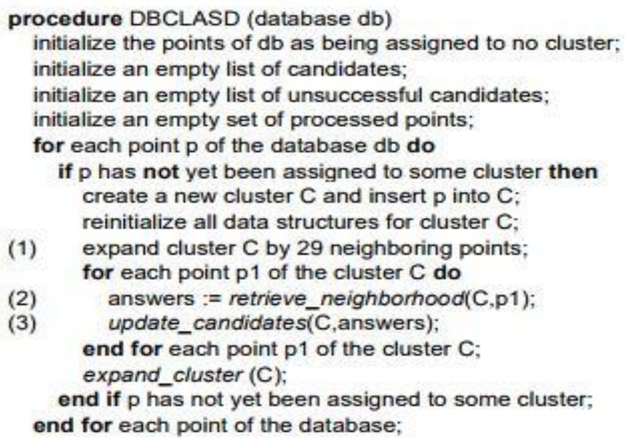
3. DBCLASD- Distribution based Clustering of Large Spatial Databases

Xu, et al. [30] a distribution based clustering algorithm for mining in large spatial databases. It is basically regarded as incremental algorithm i.e. assignment of a point on cluster will be based on points processed without considering the whole cluster or complete database.

In DBCLASD, cluster is defined by following three major properties:

- Expected Distribution Condition NNDistSet (C) which is regarded as set of nearest neighbors of cluster C and has expected distribution with required confidence level.
- Maximality Condition: Every points that comes into neighboring of C doesn't fulfill condition above.
- Connectivity Condition: Each pair (a,b) are connected via grid cell structure.

Algorithm



```
    procedure DBCLASD (database db)
      initialize the points of db as being assigned to no cluster;
      initialize an empty list of candidates;
      initialize an empty list of unsuccessful candidates;
      initialize an empty set of processed points;
      for each point p of the database db do
        if p has not yet been assigned to some cluster then
          create a new cluster C and insert p into C;
          reinitialize all data structures for cluster C;
(1)       expand cluster C by 29 neighboring points;
          for each point p1 of the cluster C do
(2)         answers := retrieve_neighborhood(C,p1);
(3)         update_candidates(C,answers);
          end for each point p1 of the cluster C;
          expand_cluster (C);
        end if p has not yet been assigned to some cluster;
      end for each point of the database;
```

DBSCLAD Algorithm

Performance Comparison: DBSCLAD Algorithm is compared with CLARANS and DBSCAN in terms of effectiveness and efficiency on databases ranging from 5000 to 25000 and the results figures state that run time of DBSCLAD is twice than run time of DBSCAN and DBSCLAD outshines CLARANS by factor of 60. DBSCLAD also outnumbers in terms of scalability as compared to DBSCAN and CLARANS algorithm.

4. DENCLUE- Density based Clustering

Hinneburg and Keim [31] proposed DENCLUE- Density Based Clustering algorithm which models the cluster formation according to the sum of influence function of all of the data points. Main two concepts are utilized in the algorithm: Influence and Density Functions.

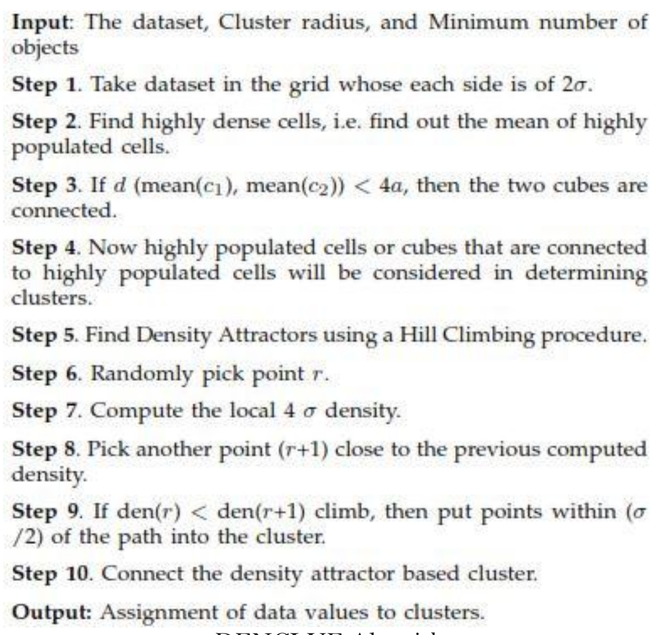
Influence of each data point is regarded as mathematical function and is called Influence function and stresses on the impact of data point within its neighborhood.

Density function is sum of influence of all data points.

Under DENCLUE Algorithm, two types of clusters are formed: Center defined and Multi center defined.

As compared to other algorithms, DENCLUE Algorithm has edge in these important areas: it has a strong mathematical foundation; good clustering properties in data sets with large noise; allows compact mathematical description of arbitrarily shaped clusters in high-dimensional data sets; and faster as compared to DBSCAN and CLARANS.

Algorithm:



Input: The dataset, Cluster radius, and Minimum number of objects

Step 1. Take dataset in the grid whose each side is of 2σ.

Step 2. Find highly dense cells, i.e. find out the mean of highly populated cells.

Step 3. If d (mean(c_1), mean(c_2)) $< 4a$, then the two cubes are connected.

Step 4. Now highly populated cells or cubes that are connected to highly populated cells will be considered in determining clusters.

Step 5. Find Density Attractors using a Hill Climbing procedure.

Step 6. Randomly pick point r.

Step 7. Compute the local 4 σ density.

Step 8. Pick another point (r+1) close to the previous computed density.

Step 9. If den(r) $<$ den(r+1) climb, then put points within (σ /2) of the path into the cluster.

Step 10. Connect the density attractor based cluster.

Output: Assignment of data values to clusters.

DENCLUE Algorithm

D. Grid-Based Clustering Algorithms

Grid-based clustering algorithms are the most popular clustering algorithms for mining clusters in large multi-dimensional space where clusters are regarded as denser regions as compared to their surroundings.

The main advantage of Grid based clustering algorithms is reduction in computations especially when large data set is required to be processed. The grid based clustering approach is different from conventional clustering approach as it is not concerned with data points but with the value space that surrounds the data points.

Grid Based Algorithms have three main stages:

Stage 1: Division of space into rectangular cells to obtain grid of cells of equal size.

Stage 2: Delete the low density of cells.

Stage 3: Combine adjacent cells having high density to form clusters.

Some of the most popular Grid based clustering algorithms are: Wave-Cluster, STING, CLIQUE and OptiGrid

1. Wave-Cluster Algorithm

Sheikholeslami, et al. [32] proposed Wave-Cluster algorithm based on wavelet transformations for efficient detection of clusters of arbitrary shape. Wave-Cluster algorithm is also regarded as most efficient in terms of time-complexity trade-off.

Algorithm:

Step 1: Quantization: Arrange all the data points into a cell. Implement wavelet transform for filtering data points.

Step 2: Apply wavelet transform on feature space.

Step 3: Locate all the connected clusters in subbands of feature space being transformed at different levels.

Step 4: Assign labelling to the units. Develop lookup table.

Step 5: Object mapping to the clusters to be performed.

Performance Comparison: Wavecluster algorithm is tested on varied data distributions and compared with BIRCH and CLARANS clustering algorithms. Every data set consists of 100,000 points. Based on operations, Wavecluster outshines in performance almost 8 to 10 times as compared to BIRCH and 20 to 30 times faster as compared to CLARANS.

Wavecluster is faster and stable clustering algorithm as compared to others and highly scalable and robust in data estimations and cluster formations.

2. STING- Statistical Information Grid Based Method

Wang, et al. [33] proposed STING, grid based hierarchical clustering algorithm, for spatial data mining to reduce cost and is almost similar to BIRCH hierarchical algorithm.

The main objective behind its development is to capture statistical information with regard to spatial cells in such a manner that whole queries and clustering based problems can be answered without the resource of individual objects.

Hierarchical Structure for STING Clustering

The area is divided into rectangular cells and a hierarchical structure is deployed. The root of Hierarchy is at Level 1; Children at Level 2 etc. A cell in Level I directly corresponds to the union of areas of its children at level I + 1. Each cell has 4 children and each child corresponds to 1 quadrant of the parent cell. The root cell located at Level 1 corresponds to whole spatial area. The size of leaf cells is dependent on density of objects. The number of layers can be obtained by changing the number of cells that form higher level cell. Assuming the 2-D space, the generalization of hierarchy becomes easy.

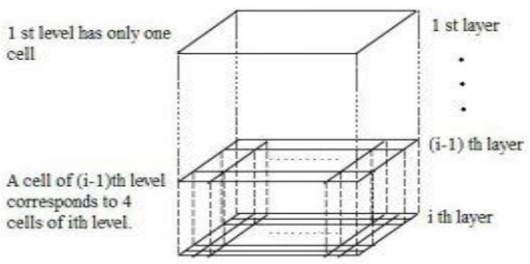
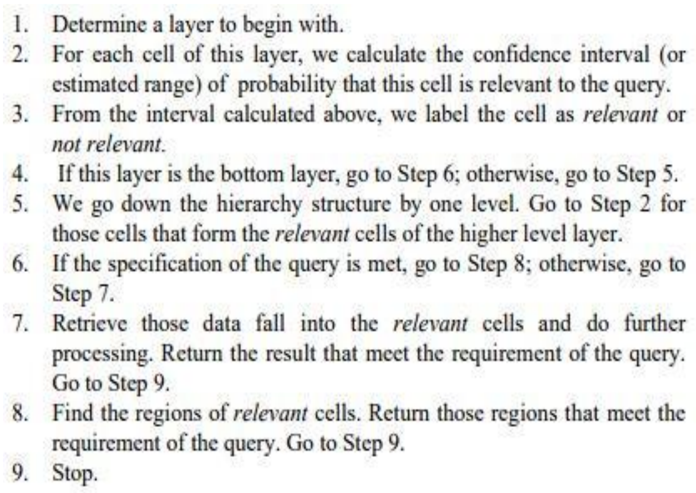

1 st level has only one cell
1 st layer
(i-1) th layer
A cell of (i-1)th level corresponds to 4 cells of ith level.
i th layer

Fig-6. STING Algorithm- Hierarchical Structure
Source: Wang, et al. [33].

Algorithm:



1. Determine a layer to begin with.
2. For each cell of this layer, we calculate the confidence interval (or estimated range) of probability that this cell is relevant to the query.
3. From the interval calculated above, we label the cell as *relevant* or *not relevant*.
4. If this layer is the bottom layer, go to Step 6; otherwise, go to Step 5.
5. We go down the hierarchy structure by one level. Go to Step 2 for those cells that form the *relevant* cells of the higher level layer.
6. If the specification of the query is met, go to Step 8; otherwise, go to Step 7.
7. Retrieve those data fall into the *relevant* cells and do further processing. Return the result that meet the requirement of the query. Go to Step 9.
8. Find the regions of *relevant* cells. Return those regions that meet the requirement of the query. Go to Step 9.
9. Stop.

STING Algorithm

Performance Comparison: The performance of STING algorithm is compared with DBSCAN and CLARANS algorithm using SEQUOIA 2000 and ran the query on data size between 1252 and 12512. The results stated that BIRCH is better than CLARANS almost 20 to 30 times so STING is somewhat better with margin as compared to BIRCH. STING algorithm is somewhat slow in run time as compared to DBSCAN.

3. CLIQUE- Clustering in QUEst

In order to estimate dense regions from multi-dimensional data, CLIQUE Jain and Dubes [34] clustering technique is utilized. Data extracted from data warehouse or large database can have multiple dimensions known as attributes. Various clustering algorithms can handle upto 3 dimensions. But fails at above levels. So, in that situations CLIQUE clustering algorithm comes to rescue. CLIQUE algorithm is highly efficient in finding dense units and partitions m-dimensional data space into non-overlapping rectangular unit.

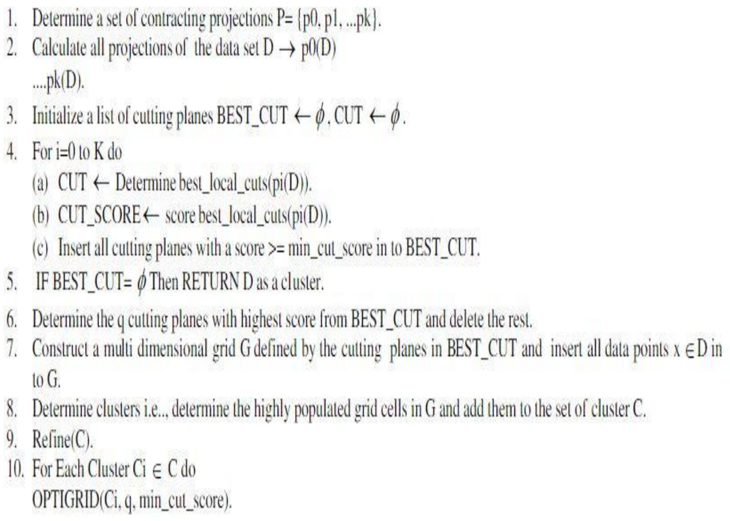
Algorithm:
Step 1: Data points are considered from data set, at one pass apply equal width to set of points to create grid cells.
Step 2: Rectangular cells whose density exceed beyond dimensional limited are placed into equal grids.
Step 3: Repeat Step 1 and 2 to form q-1 dimensional units to q dimensional units.
Step 4: In order to form clusters with equal width-size, the subspaces are connected to each other.

4. OptiGrid- Optimal Grid

OptiGrid Algorithm was proposed by Hinneburg and Keim [35] is based on development of optimal grid partitioning of data being estimated by the calculation of the best partitioning hyper planes for each dimension using certain data projections. Every plane is selected to have minimal point density to separate the dense region into equal two halves. After every step of a multi-dimensional grid construction defined by best cutting planes, OptiGrid algorithm is used to estimate the clusters.

In every round of iteration, OptiGrid maintains data objects in these dense grids which makes it the highly efficient clustering algorithm for large high-dimensional databases. OptiGrid requires a through selection of projections, density estimation and determination of what forms the best optimal cutting plane from users.

Algorithm



```
1. Determine a set of contracting projections P= {p0, p1, ...pk}.
2. Calculate all projections of the data set D $\rightarrow$ p0(D)
   ....pk(D).
3. Initialize a list of cutting planes BEST_CUT $\leftarrow \phi$, CUT $\leftarrow \phi$.
4. For i=0 to K do
   (a) CUT $\leftarrow$ Determine best_local_cuts(pi(D)).
   (b) CUT_SCORE$\leftarrow$ score best_local_cuts(pi(D)).
   (c) Insert all cutting planes with a score >= min_cut_score in to BEST_CUT.
5. IF BEST_CUT= $\phi$ Then RETURN D as a cluster.
6. Determine the q cutting planes with highest score from BEST_CUT and delete the rest.
7. Construct a multi dimensional grid G defined by the cutting planes in BEST_CUT and insert all data points x $\in$ D in
   to G.
8. Determine clusters i.e.., determine the highly populated grid cells in G and add them to the set of cluster C.
9. Refine(C).
10. For Each Cluster Ci $\in$ C do
    OPTIGRID(Ci, q, min_cut_score).
```

Opti-Grid Algorithm

E. Model Based Clustering Algorithms

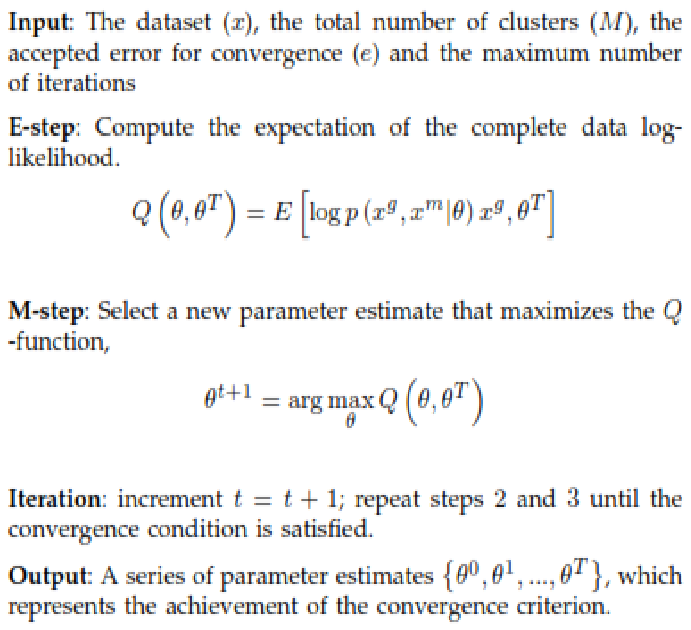
Model Based Clustering Algorithms are one of the major approaches to clustering analysis. Model based clustering techniques makes use of certain models for clusters and optimize it to fix efficiently between data and models.

The following are the most important Model based Clustering algorithms: EM, COBWEB, CLASSIT and SOMs.

1. EM- Expectation and Maximization

Expectation-Maximization Algorithm Bradley, et al. [36] is regarded as recursive method to estimate maximum likelihood or maximum a posteriori (MAP) estimates of parameters in different statistical models where whole model depends on unobserved latent variables.

EM algorithms approximates the unknown models with two steps: E- Expectation and M- Maximization. In E Step, the current model parameter values are used to estimate the posterior distribution of latent variables. In M Step, the fractional assignment is given by re-estimating the model parameters with maximum likelihood rule.



Input: The dataset (x), the total number of clusters (M), the accepted error for convergence (e) and the maximum number of iterations

E-step: Compute the expectation of the complete data log-likelihood.

$$Q\left(\theta,\theta^{T}\right)=E\left[\log p\left(x^{g},x^{m}|\theta\right)x^{g},\theta^{T}\right]$$

M-step: Select a new parameter estimate that maximizes the Q -function,

$$\theta^{t+1}=\arg\max_{\theta}Q\left(\theta,\theta^{T}\right)$$

Iteration: increment $t=t+1$; repeat steps 2 and 3 until the convergence condition is satisfied.

Output: A series of parameter estimates $\{\theta^{0},\theta^{1},...,\theta^{T}\}$, which represents the achievement of the convergence criterion.

EM Algorithm

2. COBWEB Algorithm

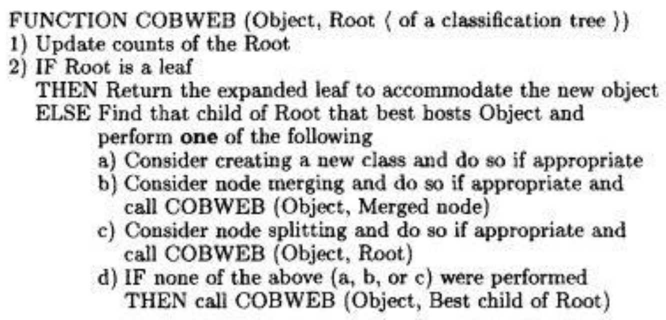
Fisher [37] and is regarded as incremental system for hierarchical conceptual clustering. The algorithm creates a hierarchical clustering in form of classification tree.

Each node in the tree represents a class and is defined as Probabilistic concept which summarizes the attribute-value distributions of objects.

Cobweb Operations

Cobweb Algorithm Primarily Performs Four Main Functions

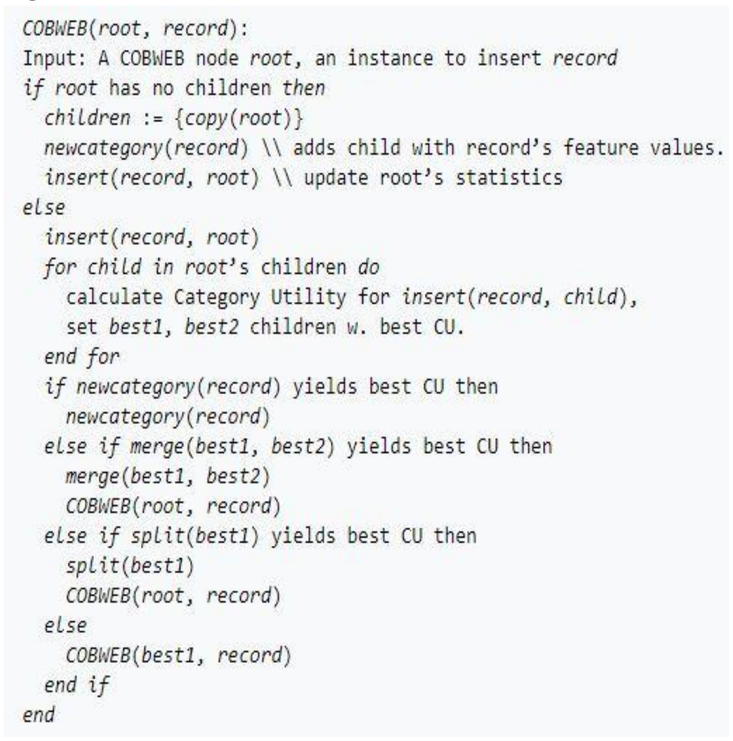
1. Merging Two Nodes: It means the node replacement whose children are the union of original nodes set of children and summarize the attribute-value distributions of all objects classified under them.
2. Splitting a Node: Node is split by its children replacement.
3. Node Insertion: A node is created and inserted into tree.
4. Passing down the Object to the Hierarchy: Effective call of COBWEB Algorithm on the object.



```
FUNCTION COBWEB (Object, Root ( of a classification tree ))
1) Update counts of the Root
2) IF Root is a leaf
   THEN Return the expanded leaf to accommodate the new object
   ELSE Find that child of Root that best hosts Object and
        perform one of the following
        a) Consider creating a new class and do so if appropriate
        b) Consider node merging and do so if appropriate and
           call COBWEB (Object, Merged node)
        c) Consider node splitting and do so if appropriate and
           call COBWEB (Object, Root)
        d) IF none of the above (a, b, or c) were performed
           THEN call COBWEB (Object, Best child of Root)
```

COBWEB Structure

Pseudocode- COBWEB Algorithm



```
COBWEB(root, record):
Input: A COBWEB node root, an instance to insert record
if root has no children then
  children := {copy(root)}
  newcategory(record) \\ adds child with record's feature values.
  insert(record, root) \\ update root's statistics
else
  insert(record, root)
  for child in root's children do
    calculate Category Utility for insert(record, child),
    set best1, best2 children w. best CU.
  end for
  if newcategory(record) yields best CU then
    newcategory(record)
  else if merge(best1, best2) yields best CU then
    merge(best1, best2)
    COBWEB(root, record)
  else if split(best1) yields best CU then
    split(best1)
    COBWEB(root, record)
  else
    COBWEB(best1, record)
  end if
end
```

3. CLASSIT Algorithm

Gennari, et al. [38] proposed CLASSIT algorithm which uses a similar approach of clustering like COBWEB, but cannot store probability counts for continuous data. It assumes normal distribution around an attribute and thus can just store a mean and variance.

CLASSIT uses a formal cut-off mechanism to support better generalization and noise handling. But the algorithm is not complete in itself and require further research for better clustering and stable outcomes.

4. SOM- Self Organized Map Algorithm

Kohonen [39] is based on unsupervised learning and grid structure.

Algorithm:
Step 1: The grid comprising nodes are placed where data points are distributed.
Step 2: Data point is sampled on the basis of closest and neighboring node. And sampling procedure moves on and on.
Step 3: Iterate Step 1 and 2 until all data points are sampled several times.
Step 4: Every Cluster is defined with reference to a node specifically comprise of those data points which represents the closest node.

4. BIG DATA CLUSTERING ALGORITHMS-PERFORMANCE COMPARISON

In this section of Research paper, the performance comparison of all Clustering Algorithms of Big Data will be highlighted. Table No: 3 highlights the performance comparison of Big Data Clustering Algorithms on basis of varied parameters like: Dataset Size, Efficiency in handling High Dimensionality, Efficiency in handling noisy data, Dataset type, Cluster Shape and Algorithm Complexity.

5. CONCLUSION AND FUTURE SCOPE

Considering the present scenario, the data size is huge and expanding day by day along with its variety. The velocity of data generation is also growing at rapid pace because of increase in mobile devices and adaptability of Internet of Things (IoT). The data generated in different forms is utilized by different businesses for profit purposes and with integration of cloud services, data can be stored, processed and analyzed any time, every time cum anywhere and everywhere.

The paper presents comprehensive overview of Big Data clustering algorithms which are being used to manage large data sets. Considering the working, pros and cons and even the testing performed by developers on varied data sets, it can be analyzed that almost every algorithm is not sufficient to face all challenges of data operations and not every mix of data can be analyzed by any single algorithm. Lots of research is required to propose efficient clustering algorithm to solve issues of Big Data.

Some algorithms are efficient but pose certain challenges during implementation. But still for analyzing large data sets algorithms like BIRCH, CLIQUE and CLARANS can be utilized.

In order to effectively manage and utilize large data sets for efficient results, clustering algorithms needs to be improvised sophisticatedly in terms of memory and time consumption.

5.1. Future Scope

Considering future work, research can be directed towards improvising exiting clustering algorithms in terms of time and memory space trade off and making them effective for analyzing large data sets with varied forms of data. The approach would also be to propose efficient clustering algorithm to perform better as compared to existing algorithms.

Funding: This study received no specific financial support.
Competing Interests: The authors declare that they have no competing interests.
Contributors/Acknowledgement: Both authors contributed equally to the conception and design of the study.

REFERENCES

[1] M. Cox and D. Ellsworth, "Managing big data for scientific visualization," *In ACM Siggraph*, vol. 97, pp. 21-38, 1997. *View at Google Scholar*

[2] J. Gantz and D. Reinsel, "Extracting value from chaos," *IDC iview*, vol. 1142, pp. 1-12, 2011. *View at Google Scholar*

[3] P. Zikopoulos, K. Parasuraman, T. Deutsch, J. Giles, and D. Corrigan, *Harness the power of big data the IBM big data platform*: McGraw Hill Professional, 2012.

[4] J. J. Berman, *Principles of big data: Preparing*, 1st ed.: Sharing, and Analyzing Complex Information Morgan Kaufmann, 2013.

[5] S. Sagiroglu and D. Sinanc, "Big data: A review. In collaboration technologies and systems (CTS)," presented at the International Conference on. IEEE, 2013.

[6] A. McAfee and E. Brynjolfsson, "Big data: The management revolution," *Harvard Business Review*, vol. 90, pp. 60-68, 2012. *View at Google Scholar*

[7] O. Tene and J. Polonetsky, "Big data for all: Privacy and user control in the age of analytics," *Northwestern Journal of Technology and Intellectual Property*, vol. 11, p. 27, 2012. *View at Google Scholar*

[8] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98-115, 2015. *View at Google Scholar* | *View at Publisher*

[9] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, and A. Bouras, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, pp. 267-279, 2014. *View at Google Scholar* | *View at Publisher*

[10] T. Sajana, C. S. Rani, and K. V. Narayana, "A survey on clustering techniques for big data mining," *Indian Journal of Science and Technology*, vol. 9, pp. 1-12, 2016. *View at Google Scholar* | *View at Publisher*

[11] S. Äyrämö and T. Kärkkäinen, "Introduction to partitioning-based clustering methods with a robust example," Reports of the Department of Mathematical Information Technology. Series C, Software Engineering and Computational Intelligence 1/20062006.

[12] J. A. Hartigan and J. A. Hartigan, *Clustering algorithms* vol. 209. New York: Wiley, 1975.

[13] S. Liu, X. Liu, H. Zhao, and W. Fu, "Composite service execution petri net and service composition optimization," presented at the In Service Operations and Logistics, and Informatics (SOLI). IEEE International Conference on. IEEE, 2012.

[14] M. Silic, G. Delac, and S. Srbljic, "Prediction of atomic web services reliability for qos-aware recommendation," *IEEE Transactions on Services Computing*, vol. 8, pp. 425-438, 2015. *View at Google Scholar* | *View at Publisher*

[15] H. S. Park and C. H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Systems with Applications*, vol. 36, pp. 3336-3341, 2009. *View at Google Scholar* | *View at Publisher*

[16] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, pp. 100-108, 1979. *View at Google Scholar* | *View at Publisher*

[17] N. Sharma and N. Gaud, "K-modes clustering algorithm for categorical data," *International Journal of Computer Applications*, vol. 127, pp. 1-6, 2015. *View at Google Scholar* | *View at Publisher*

[18] L. Kaufman and P. J. Rousseeuw, "Partitioning around medoids (Program Pam)," *Finding Groups in Data: An Introduction to Cluster Analysis*, vol. 344, pp. 68-125, 1990.

[19] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: An introduction to cluster analysis* vol. 344: John Wiley & Sons, 2009.

[20] R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, pp. 1003-1016, 2002. *View at Google Scholar* | *View at Publisher*

[21] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, pp. 191-203, 1984. *View at Google Scholar* | *View at Publisher*

[22] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *In ACM Sigmod Record*, vol. 25, pp. 103-114, 1996. *View at Google Scholar* | *View at Publisher*

[23] S. Virpioja, "BIRCH: Balanced iterative reducing and clustering using hierarchies," 2008.

[24] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," *In ACM Sigmod Record*, vol. 27, pp. 73-84, 1998. *View at Google Scholar* | *View at Publisher*

[25] S. Guha, R. Rastogi, and K. Shim, "ROCK: A robust clustering algorithm for categorical attributes," *Information Systems*, vol. 25, pp. 345-366, 2000. *View at Publisher*

[26] G. Karypis, E. H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computers*, vol. 32, pp. 68-75, 1999. *View at Google Scholar* | *View at Publisher*

[27] A. N. Mahmood, C. Leckie, and P. Udaya, "An efficient clustering scheme to exploit hierarchical data in network traffic analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, pp. 752-767, 2008. *View at Google Scholar* | *View at Publisher*

[28] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *In Kdd*, vol. 96, pp. 226-231, 1996. *View at Google Scholar*

[29] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *In ACM Sigmod Record*, vol. 28, pp. 49-60, 1999. *View at Google Scholar* | *View at Publisher*

[30] X. Xu, M. Ester, H. P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases. In data engineering, 1998," in *Proceedings 14th International Conference on. IEEE*, 1998, pp. 324-331.

[31] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," *In KDD*, vol. 98, pp. 58-65, 1998. *View at Google Scholar*

[32] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases," *In VLDB*, vol. 98, pp. 428-439, 1998. *View at Google Scholar*

[33] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," *In VLDB*, vol. 97, pp. 186-195, 1997. *View at Google Scholar*

[34] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*: Prentice-Hall, Inc, 1988.

[35] A. Hinneburg and D. A. Keim, "Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering," presented at the Proceedings of the 25th International Conference on Very Large Databases, 1999.

[36] P. S. Bradley, U. Fayyad, and C. Reina, "Scaling EM (Expectation-Maximization) clustering to large databases," Redmond: Technical Report MSR-TR-98-35, Microsoft Research1998.

[37] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning*, vol. 2, pp. 139-172, 1987. *View at Google Scholar* | *View at Publisher*

[38] J. H. Gennari, P. Langley, and D. Fisher, "Models of incremental concept formation," *Artificial Intelligence*, vol. 40, pp. 11-61, 1989. *View at Google Scholar* | *View at Publisher*

[39] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, pp. 1-6, 1998. *View at Google Scholar*

Table-3. Performance Comparison of Big Data Clustering Algorithms

Algorithm Type	Name of Algorithm	Dataset Size	Whether Efficient in Handling High Dimensionality (Yes/No)	Whether Efficient in Handling Noisy Data (Yes/No)	Dataset Type	Shape of Cluster	Algorithm Complexity
Partitioning Algorithm	K-Means	Large	No	No	Numerical	Non-Convex	O(nkd)
	K-Modes	Large	Yes	No	Categorical	Non-Convex	O(n)
	K-medoids	Small	Yes	Yes	Categorical	Non-Convex	O(n^2dt)
	PAM	Small	No	No	Numerical	Non-Convex	O(k$(n-k)^2$)
	CLARA	Large	No	No	Numerical	Non-Convex	O(k(40+k) ^{2}+k(n-k))
	CLARANS	Large	No	No	Numerical	Non-Convex	O(kn^2)
	FCM	Large	No	No	Numerical	Non-Convex	O(n)
Hierarchical Algorithms	BIRCH	Large	No	No	Numerical	Non-Convex	O(n)
	CURE	Large	Yes	Yes	Numerical	Arbitrary	O(n^2log n)
	ROCK	Large	No	No	Categorical and Numerical	Arbitrary	O(n^2+nmmma+ n^2logn)
	Chameleon	Large	Yes	No	Data- All Types	Arbitrary	O(n^2)
	ECHIDNA	Large	No	No	Multivariate Data	Non-Convex	O(N * B(1+ $\log_B$m))
Density-Based Algorithms	DBSCAN	Large	No	No	Numerical	Arbitrary	O(nlogn)
	OPTICS	Large	No	Yes	Numerical	Arbitrary	O(nlogn)
	DBCLASD	Large	No	Yes	Numerical	Arbitrary	O(3n^2)
	DENCLUE	Large	Yes	Yes	Numerical	Arbitrary	O(log|D|)
Grid Based Algorithms	Wave-Cluster	Large	No	Yes	Special Data	Arbitrary	O(n)
	STING	Large	No	Yes	Special Data	Arbitrary	O(k)
	CLIQUE	Large	Yes	No	Numerical	Arbitrary	O(Ck + mk)
	OptiGrid	Large	Yes	Yes	Special Data	Arbitrary	Between O(nd) and O(nd log n)
Model-Based Algorithms	EM	Large	Yes	No	Special Data	Non-Convex	O(knp)
	COBWEB	Small	No	No	Numerical	Non-Convex	O(n^2)
	CLASSIT	Small	No	No	Numerical	Non-Convex	O(n^2)
	SOMs	Small	Yes	No	Multivariate Data	Non-Convex	O(n^2m)

Source: Sajana, et al. [10].