ISSN No:-2456 –2165

A Survey of Clustering Algorithm for Very Large Datasets

Tmty. P. Aruna Devi
M.Sc., M.Phil., Assistant Professor,
Department of Computer Science(S),
V.V.Vanniaperumal College for Women.
Virudhunagar, India

Dr.(Tmty.) M. Chamundeeswari
Associate Professor, Department of Computer Science
(Aided), V.V.Vanniaperumal College for Women.
Virudhunagar, India

Abstract:-Clustering in data mining is viewed as unsupervised method of data analysis. Clustering allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Clustering helps to discover groups and identifies interesting distributions in the underlying data. It is one of the most useful technique and used in exploratory analysis of data. It is also used in various areas such as grouping, decision-making, and machine-learning situations, including data mining, document retrieval, image segmentation, classification and image processing. Traditional clustering algorithms both favour clusters with spherical shapes and similar sizes, and are very fragile in the presence of outliers. Clustering plays a major role in analysis of very large data set and it is useful to discover the correlation among attributes both of spherical and non spherical shape which is also robust to outliers. This survey focuses on clustering algorithms that are used on very large data sets which help to find the characteristic of the data. We have taken the best clustering algorithm such as BIRCH, BFR and CURE.

Keywords:-Hierarchical, Centroid, CF Tree, Incremental Algorithm, Representation Points, Classes of Points.

I. INTRODUCTION

We are living in the world where we have lots of data people need some information from those data by just analysing them. The vital means in dealing these data is to group them into set of categories or clusters. Data mining [1] plays important role in the use of automated data analysis techniques to uncover previously undetected relationships among data items. Data mining [1, 2]

often involves the analysis of data stored in a data warehouse. Three of the major data mining techniques are regression, classification and clustering. In this research paper we are working only with the clustering because it is most important process, which is to be applied on a very large database. The goal of this survey is to provide a comprehensive review of different clustering techniques in data mining. Clustering is done to assign set of objects into groups such that the objects within the group is much similar than those objects in other group or cluster. Cluster analysis is the organization of a collection of patterns into cluster based on similarity [3].

Clustering algorithms used in those days can be classified into partitional clustering and hierarchical clustering [4, 5]. Partitional clustering algorithms uses certain criterion function that helps to divide the point space into k clusters. The most commonly used criterion function for metric spaces is

$$E=\sum_{i=1}^{k} \sum_{x\epsilon Ci} d(x, mi)$$

In the above equation, function E attempts to minimize the distance of point from the mean of the cluster to which the point belongs, mi is the centroid of cluster C_i, d is the Euclidean distance between x and m_i.. The way to minimize the criterion function is using an iterative technique. Let us start with k initial partitions, and the data points are moved from one cluster to another which helps to improve the value of the criterion function. The above criterion function yields good result for the numerical data but it is not suitable for the categorical data. Hierarchical clustering algorithm is unsuitable for clustering data sets containing categorical attributes. Once after the action of merge or split step[6] in the Hierarchical methods it can never be undone [7].

These two approaches help in improving the quality of hierarchical clustering: (1) Object linkages at each hierarchical partitioning is performed with careful analysis (2) Integrate hierarchical agglomeration and other approaches. First hierarchical agglomerative algorithm is used to group objects into micro-clusters, and then perform macro-clustering on the micro-clusters using iterative relocation [2]. For instance, consider the centroid based agglomerative hierarchical clustering algorithm [4, 5]. In this algorithm, initially, each point is treated as a separate cluster. Merging of pair of cluster is done for those cluster whose mean is closest. The merging step is done till the desired number of clusters is reached. However for categorical attributes distances between centroids of clusters are a poor estimate of the similarity between them. Clustering plays important role in various fields [8] and it is necessary to learn them.

ISSN No:-2456 –2165

II. CLUSTERING ALGORITHM

The main aim of this survey paper is to provide an emphasis on the study of three different clustering algorithms BIRCH [9], BFR [10], CURE [11] that are implemented and used on large data sets. It focuses on three different algorithms their usage of parameters in those algorithms, their advantages and drawbacks. Finally we would conclude this study by mentioning the best algorithm that could be used on the large data sets.

III. BIRCH

BIRCH algorithm is devised to build an iterative and interactive classification tool and could also be used as a tool for image compression. BIRCH helps to address the problem of processing large data sets with a limited amount of resources (memory and CPU utilization). BIRCH deals with large datasets it produce a more compact summary by retaining the distribution information in the datasets, and then clustering is done on the data summary instead of the original dataset . BIRCH is an efficient and scalable data clustering method, which uses a new in-memory data structure called *C*F-tree, which helps to keep the in-memory summary of the data distribution.

A CF tree is a height -balanced tree which stores the clustering features that is useful in hierarchical clustering. It is same as the B+-Tree or R-Tree. CF tree is balanced tree having maximum number of children per leaf node and with a branch factor B and threshold T. The internal node contains a CF triple (N, LS, SS) where N is the no of data points, LS is the Linear sum and SS is the squared sum for each of its children. Each leaf node contains a CF entry for each sub cluster in it. A sub cluster in a leaf node has a diameter not greater than a threshold value which is maximum diameter of sub-clusters in the leaf node [9].

A. Need for Birch

Hierarchical methods suffer from the fact that once we have performed either merge or split step we cannot undo the previous step [6]. The quality of clustering in hierarchical methods could be preserved by integrating hierarchical clustering with other clustering techniques for multiple phase clustering. The improved hierarchical clustering is represented by our two algorithms BIRCH and CURE hierarchical clustering.

The I/O cost of BIRCH is linear with the size of the dataset. A single scan of the dataset produces a good clustering and we can optionally use one or more additional passes to improve the quality of the cluster. We could evaluate BIRCH's running time, memory usage, clustering quality, stability and scalability and compare it with other existing algorithms. It is considered that the best available clustering method for handling very large datasets could be BIRCH from the evaluation. It is observed that BIRCH actually complements other clustering algorithms by the fact that different clustering algorithms can be applied to the summary produced by BIRCH. BIRCH's architecture could be used in parallel and concurrent clustering, and it tunes the performance by gaining the knowledge on dataset over the course of the execution more interactively and dynamically.

The most important contribution of BIRCH is the formation of the clustering problem that is appropriate for very large datasets. The time and memory constraints are explicit. Next contribution is that BIRCH exploits that the data space is usually not uniformly occupied and so every data point is equally important for clustering purposes. So BIRCH collects a dense region of points (or a sub clusters) and performs a compact *s*ummarization. Thus the problem of clustering the original data points is reduced to cluster the set of summaries which is compared much smaller than the original dataset. The summaries reflect the natural closeness of data which allows for the computation of the distance-based measurements to be maintained efficiently and incrementally. Although we only use them for computing the distances, we note that they are also sufficient for computing the probability-based measurements such as mean, standard deviation and category utility used in Classit.

Compared with the distance-based algorithms, BIRCH is incremental in such a way that the clustering decisions are made without scanning all data points or the currently existing clusters. Compared with prior probability-based algorithms, BIRCH tries to make the best use of the available memory to derive the finest possible sub clusters which helps to minimize I/O costs by using an in-memory balanced tree structure of bounded size. BIRCH plays a significant impact on overall direction of scalability research in clustering [12]. Finally BIRCH does not assume that the probability distributions on separate attributes are independent.

In the paper given by "Tian Zhang ,Raghu Ramakrishnan ,Miron Livny " in "BIRCH: A New Data Clustering Algorithm and Its Applications " the author (propose a birch method for large database) BIRCH creates a height balanced tree with the nodes that has the summary of data by accumulating its zero, first, and second moments. Then Cluster Feature (CF) for the node is constructed which is a tight small cluster of numerical data. The construction of a tree is controlled with some parameters. A new data point is added to the tree with the closest CF leaf if it fits the leaf well and when the leaf is not overcrowded. Then the CF is updated for all nodes from the leaf to the root, otherwise a new CF is constructed. There may be several splits since we have the restriction on the maximum number of child node.

When the tree reaches the size of memory, it should be rebuilt with a new threshold. The outliers are not considered in the CF feature and it is sent to disk, and may be considered during tree rebuilds. The final leaves are provided as the input to any algorithm of choice. The balanced CF-tree allows for the log-efficient search. The parameters that control CF tree construction are branching factor, maximum number of points per leaf, leaf threshold, and also depend on data ordering.

Author implemented the algorithm that consists of four phases: (1) Loading (2) Condensing (3) Global clustering and (4) Refining. When the tree is constructed with one data

pass, it can be condensed in the 2nd phase to further fit desired input cardinality of post-processing clustering algorithm. Next, in the 3rd phase a global clustering considering an individual point of CF happens. Finally certain irregularities of finding identical points getting to different CFs can be resolved in the 4th phase. Thus one or more passes over data is done by reassigning points to best possible clusters just like the k-means.

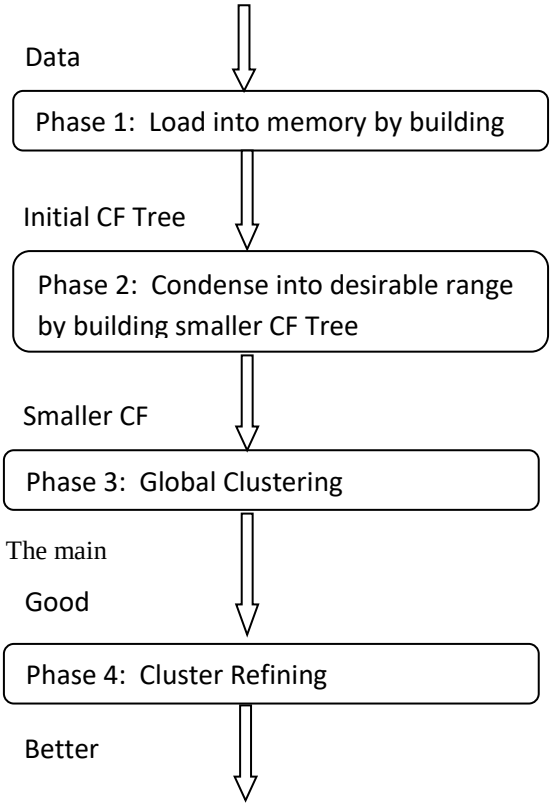

Fig.1

The task of Phase 1 is to build the CF tree by scanning all data using the given amount of memory and recycling space on disk. The constructed CF-tree reflects the clustering information of the dataset in much detail subject to the memory limits. The data points that are crowded are grouped into sub clusters, and sparse data points are removed as outliers. The result of this phase is creating an in-memory summary of the data.

After Phase 1, subsequent computations in later phases will be [9]:

(1) fast because (a) no I/O operations are needed, and (b) the problem is reduced to a smaller problem by clustering the sub clusters in the leaf entries;

(2) accurate because (a) outliers are eliminated and (b) the finest granularity of the remaining data is achieved with given available memory;

(3) less order sensitive because the leaf entries will have an input order with better data locality compared with the arbitrary original data input order.

The result of the algorithm is found to be sensitive to the parameters such as Initial threshold: (1) BIRCH's performance is stable as long as the initial threshold is not excessively high with respect to the dataset. (2) *T*0 = 0:0 works well with a little extra running time. (3) If we do not know a good *T*0, then we can save up to 10% of the time. Page Size *P*: In Phase 1, smaller *P* tends to decrease the running time, requires higher ending threshold, produces less but coarser leaf entries, and hence degrades the quality. However, with the refinement in Phase 4, the experiments suggest that for ***P* = 256 to 4096 [9]**, although the qualities at the end of Phase 3 are different, the final qualities after the refinement are almost the same.

IV. BFR

BFR [Bradley-Fayyad-Reina] is also designed for very large datasets with a slight variance in K-means. *k*-means clustering [13] is a method of cluster analysis that aims at partitioning *n* data points into *k* clusters in which each data point belongs to the cluster with the nearest mean.

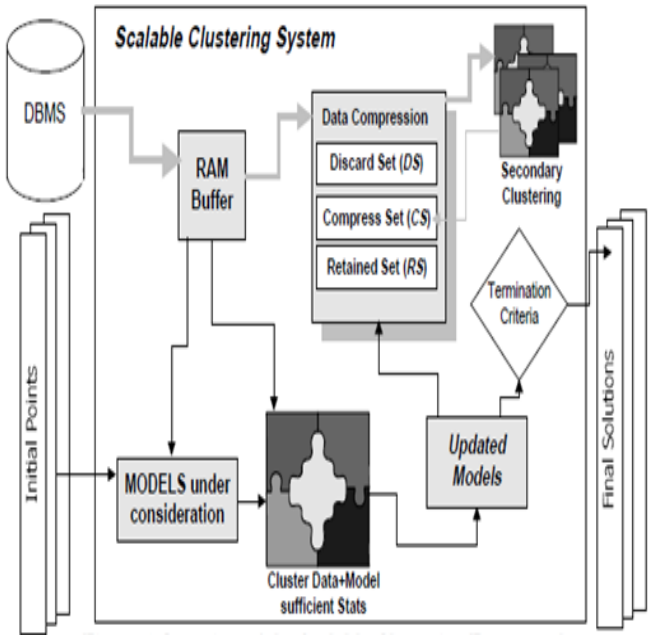

Fig. 2: Overview of the scalable Clustering Framework.

K-means (Macqueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well known clustering problem [14].BFR assumes that clusters are normally distributed around a centroid in a Euclidean space. The Clusters are axis-aligned ellipse. For every point in the data set we can find whether that point belongs to a particular cluster. To do the clustering process the data sets are read one main memory at a time, the points that are previously loaded into memory are summarized by some simple statistics. First take k random points with its centroids then the next loaded points are considered are classified into three classes of points called as Discard Set (DS), Compression Set (CS) and Retained Set (RS). Discard set is called as the points that are close to the centroid.

ISSN No:-2456 –2165

Compression Set is the group of points that are close together but are not close to the centroid, these points are summarized and not assigned as a cluster. Retained Set is the isolated points which are waiting to be added to the compression set.

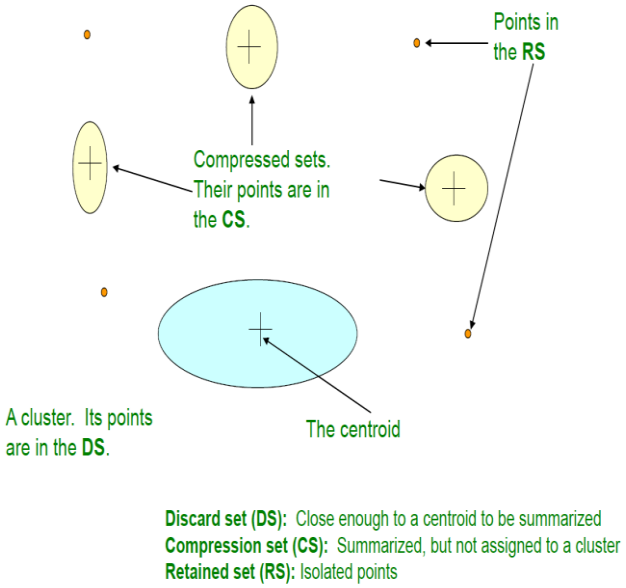

Fig. 3

In the paper given by "P.S.Bradley, Usama Fayyad and Cory Reina" in "Scaling Clustering Algorithms to Large Databases" the authors presents a scalable clustering framework applicable for a wide class of iterative clustering that requires one scan of the entire data sets. The algorithm operates within the confines of a limited memory buffer. They have taken the three classes of points and done the clustering using primary and secondary data compression.

In primary data compression, it is intended to discard the data points that unlikely to change cluster membership in future iterations. With each cluster we associate the list of DS that summarizes the sufficient statistics for the data discarded from the cluster. The method used for primary data compression is the Mahalanobis radius around the cluster center and compressing all data points within that radius. All the data points within that radius are sent to the discard set (DS). Using the past data samples the data points that are discarded by this method need some statistics to merge with the DS of points.

Secondary data compression helps to identify the tight sub-clusters of points amongst the data that we cannot discard in the primary phase. If the sub-cluster change the cluster means then they are updated exactly by the sufficient statistics. The length of the compression set varies depending on the density of points. The process takes dense points which is not compressed during primary compression, then apply the tightness criteria using k- mean algorithm. This compression takes place over all clusters. Then the formed cluster could be merged with other cluster or CS using agglomerative clustering over the clusters and CS. The nearest two sub-clusters are merged which results in new sub-cluster without violating the tolerance condition.

A. Observations From the Algorithm

The BFR method is good for the restricted data sets which obey the two assumptions made by authors. The first assumption mentioned by author that the "data set should follow the gaussian distribution", this assumption is reasonable since authors keeping only summary of the DS by using only N, SUM and SUMSQ. The choosing of data sets should be done in such a way that those data sets would become best case scenario for the choosen algorithm [1]. They have taken the data set which follows the gaussian distribution, in fact authors have also mentioned that the data sets which they are referring are not the best case scenario for the real world and there is a high probability that most of the real world data sets will be the worst case input for the algorithm [1].

The experiments are conducted based on the above assumption data sets. And have been shown that the algorithm out performs the naive K-mean algorithm in high dimension data set which obey the assumptions. The algorithm has taken the noise and outliers into their consideration, outliers and noise can easily deform the shape of the clusters. This paper also assume while doing computation for DS, CS, RS that the mean will not move outside of the computed interval, this may not be true in some extreme case when the data point will arrive in increasing sequence. There is a high probability that the mean will not remain into the same interval and the DS, CS and RS may not be valid any more.

In the proposed algorithm authors are keeping the summary of DS into main memory, in the extreme case where number of clusters is very large the summary of DSs could not be kept in main memory. If there are many mini clusters formed between the points then it is difficult to keep CS and RS in main memory. The RS play a vital role in the sense that there may be points that take over the points which neither belongs to big cluster nor into small cluster. So large amount of memory is occupied by summary of DS, CS and RS. This shows that all memory are not available to process the incoming data [5]. Since for DS, we only keep track the summary(N, SUM,SUMSQ) of it.

However the second assumption "The shape of cluster should be axis aligned to the axes of space" however this assumption can be avoided by transforming the dataset points to the axis aligned. This can be done by using one pass of dataset, so the total cost to transform the data set into axis align is linear. So any given data set can be transformed to make the compatible with this algorithm. The space problem due to holding the summary of DS, CS as well keeping the point of RS can be reduce if we apply the divide and conquer technique across the multiple node of the system and finally combing them together. Another method could be to use parallelism to make the computation fast and avoid the memory issue raised by CS, DS and RS. The good summary of DS can be obtained using representative method as well. The axis aligned assumption can be relaxed by using the point transformation. The space complexity can be further reduce by using the divide and conquer approach. A parallel computing approach can be used to

further reduce the time complexity. The application which has data set which follow the gaussian distribution will get benifitted from this algorithm.

V. CURE

CURE uses a novel hierarchical clustering algorithm that acts in between the centroid-based and the all-point extremes. CURE algorithm is more robust to outliers, and identifies clusters having non-spherical shapes and wide difference in size. CURE works by assigning representative points that are scattered in each cluster and then shrinking by specified fraction towards the center of the cluster. More representative point per cluster allows CURE to adjust well to the geometry of non-spherical shapes and the shrinking helps to eliminate the effects of outliers. To handle large databases, CURE uses the combination approach of partitioning and random sampling. First from the data set a random sample is drawn and partitioned and then it is partially clustered. In the second pass the partial clusters are then clustered to yield the desired clusters. Random sampling and partitioning enable CURE to not only outperform existing algorithms but also to scale well for large databases without degrading clustering quality.

In CURE algorithm constant number c of well scattered points in a cluster are first selected. The scattered points help to acquire the shape and extent of the cluster. The selected scattered points are next shrunk towards the centroid of the cluster by a fraction α. Those points after shrinking are used as representative points of the cluster. The clusters with the closest pair of representative points are the clusters that are merged at each step of CURE's hierarchical clustering algorithm. The scattered points approach employed by CURE overcomes the drawbacks of both the all-points and the centroid-based approaches [12]. Because of the shrinking of the scattered points toward the mean CURE is less sensitive to outliers. Outliers are the points that are far away from the mean and are shifted a larger distance due to the shrinking. Multiple scattered representative points on the cluster enables CURE to discover non-spherical shapes like the elongated clusters. For the centroid-based algorithm, the space that constitutes the vicinity of the single centroid for a cluster is spherical. Thus, it favours spherical clusters and splits the elongated clusters. With multiple scattered points as representatives of a cluster, the space that forms can be non-spherical, and this enables CURE to correctly identify the cluster.

The main aim of CURE is to find clusters of arbitrary shapes and sizes as shown in Figure 1. The algorithm works as follows to detect the arbitrary clusters. First, a random sample is drawn from the database of points, as shown by the orange nodes in Figure 2. This smaller subset of random points, labelled by node number in Figure 3 is then passed into the hierarchical clustering algorithm. The clustering algorithm first considers each point to be its own cluster, and identifies the nearest neighbour of each cluster, as drawn by the blue lines connecting nearest neighbours in Figure 4 [11]. To keep track of two near clusters Heap is used in this algorithm.

Fig. 4. Example of Clusters of Arbitrary Shapes and Sizes in 2D Detectable by CURE.

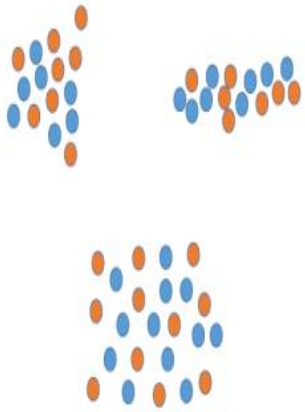

Fig. 5 Example of a Random Sample of Points Drawn From the Entire Dataset – Step 1.

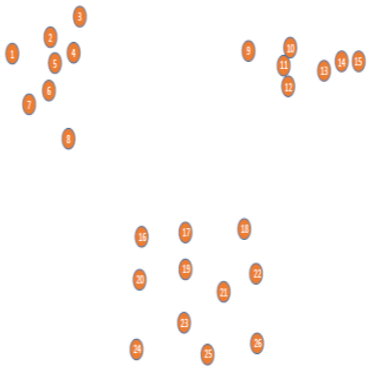

Fig. 6. Labeled Data Points in the Sample Drawn From the Step 1.

ISSN No:-2456 –2165

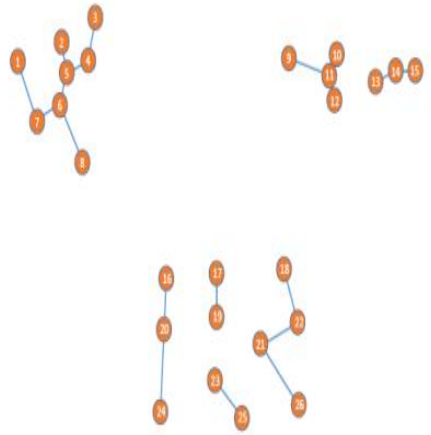

Fig. 7 The Nearest Neighbour of Each Cluster is Found.

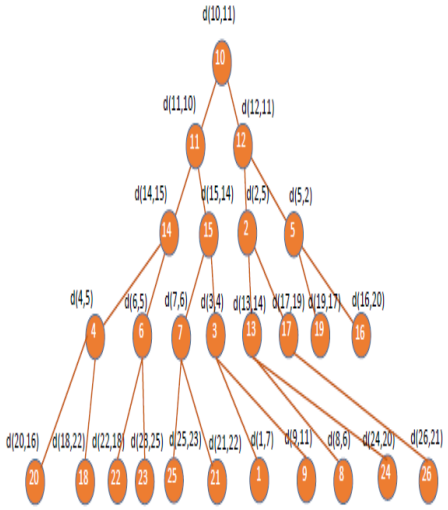

Fig. 8 A Heap Where the Distance of the Neighbour is Used to Sort the Cluster.

The clusters are stored in the heap according to their distance to their nearest neighbour. Cluster u is stored according to its distance d(u; u.nearest neighbour) as shown in Figure 5. In the first step d(u; u.nearest neighbour) is simply the distance between a node and its nearest neighbour.

When a cluster u has more than one node, then d (u; u.nearest neighbour) is determined as the distance between the nearest representative point in u to a representative point in a different cluster v. The number of representative points c is a user-defined input, and the points are chosen according to the following procedure: at each merge between clusters u and v, the weighted mean between the means of these two clusters is chosen as the first representative point of the newly merged cluster w = merge(u; v). The weighting is determined by the number of points in each cluster, so that a cluster with more points assigns more weight to its mean. Next, for each i = 2 c, a node from the merged set u U v becomes a representative point Ri if it is the farthest point from the previous point Ri-1 in u U v. After all c representative points are chosen from

the merge of u and v, and then the points are each “shrunk" by a factor α toward the mean of the merged set [11].

Next, using a kd-tree data structure the nearest neighbours are re-computed for all clusters. The kd-tree stores all the representative points and is used to find the closest point to a given node. The heap is then reordered according to the new nearest neighbours of each cluster.

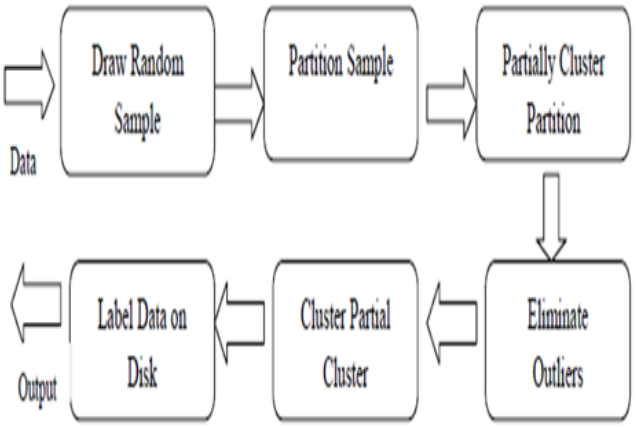

Fig. 9

Salient features of CURE [11] are: (1) the clustering algorithm can recognize arbitrarily shaped clusters (2) Robust even though there is the presence of outliers, and (3) the algorithm needs only linear storage and time complexity of O(n) for low-dimensional data. The n data points which are given as input to the algorithm may be either a sample drawn randomly from the original data points, or a subset if partitioning on data points are done.

In the paper given by “Sudipto Guha, Rajeev Rastogi, Kyuseok Shim” in “CURE: An Efficient Clustering Algorithm for Large Databases **“**the author has given the clustering steps and the sensitivity parameters for that algorithm.

Clustering procedure: Initially, for each cluster u, the set of representative points u.rep contains only the point in the cluster. Then all input data points are inserted into the k-d tree. Next a heap is constructed with each input point as a separate cluster then computes u.closest for each cluster u and then inserts each cluster into the heap in the increasing order. Once the heap Q and tree T are initialized the closest pair of clusters is merged until K cluster is obtained. At the top of the heap Q there is the cluster u for which u and u.closest are the closest pair of clusters. Next merge step is done on the closest pair of clusters u and v, new representative points for the new merged cluster w is computed which are subsequently inserted into T. The points in cluster w is the union of two cluster u and v.

ISSN No:-2456 –2165

- **Sensitivity to Parameters for CURE:** we present here the sensitivity analysis for CURE with respect to the parameters (α, c, s and p).
- **Shrink Factor α**: α varies from 0.1 to 0.9. When α = 1 and α = 0, they are similar to BIRCH and MST, respectively. CURE behaves same as centroid-based hierarchical algorithms for values of α between 0.8 and 1 since the representative points end up close to the center of the cluster. CURE always finds the right clusters for α values from 0.2 to 0.7. Thus, 0.2- 0.7 is a good range of values for α to identify non-spherical clusters ignoring the effects of outliers.
- **Number of representative Points c**: The number of representative points is varied from 1 to 100. The quality of clustering suffered when value of c is smaller. The big cluster is split when c = 5. This is because a small number of representative points do not adequately capture the geometry of clusters. CURE always found right clusters for values of c greater than 10.
- **Number of Partitions p**: CURE always discovered the desired clusters for varied number of partitions from 1 to 100. However, when we divide the sample into 100 partitions, the quality of clustering suffers due to the fact that each partition does not contain enough points for each cluster. Thus, the relative distances between points in a cluster become large compared to the distances between points in different clusters - the result is that points belonging to different clusters are merged.
- **Random Sample Size s**: Random sample sizes could range from 500 – 5000. Clusters would be of poor quality for sample sizes up to 2000. However, from 2500 sample points and above CURE always correctly identified the clusters.

From our study, we establish that BIRCH fails to find out the clusters with non-spherical shapes or clusters with variances in size. MST is much better at clustering arbitrary shape and it is very sensitive to outliers. CURE could discover clusters with interesting shapes and it is less sensitive to outliers. Sampling and partitioning are the techniques that helps in pre clustering to reduce the input size of large data sets without sacrificing the quality of clustering. The execution time of CURE is low in practice.

VI. CONCLUSION

Birch helps to cluster on large data set using CF Tree that scans the data set only once and the second scan helps to improve the quality of the cluster. This algorithm can be used when limited memory is available and running time of the algorithm is to be minimized. BFR uses the three classes of points which makes the summary of data sets but when compared to BIRCH the total run time will be less due to the fact that we require one or less scans and there is no need to maintain the large CF Tree. CURE employs a combination of random sampling and partitioning that allows it to handle large data sets efficiently, it also adjust well to the geometry of cluster having non spherical shapes and wide variances in sizes. It also scales well for large databases without sacrificing cluster quality. Thus the quality of the cluster will be good when we are using CURE.

The concepts and characteristics of the BIRCH, BFR and CURE algorithms is summarized in the table given below.

S.No	Property	BIRCH	BFR	CURE
1.	Input Data Set	Dataset is loaded in memory by building CF Tree	Dataset is loaded using three classes of points	Random sampling is done on the dataset.
2.	Handle large dataset	Yes	Yes	Yes
3.	Geometry	Shape of the cluster is almost spherical and uniform in size.	Shape of the cluster is spherical and axis aligned.	Shape of the cluster are of any arbitrary shape and in varying sizes.
4.	Input order sensitivity	Yes	Yes	No
5.	Complexity	$O(n^2)$	O(n)	$O(n^2)$
6.	Working on Data	Indirectly on the summarization	Directly	Directly
7.	Distance Measure	Euclidean	Mahalanobis	Manhattan 0r Euclidean
8.	Type of Data	Numeric	Numeric	Numeric and Nominal

Table 1. Result From Observations

REFERENCES

[1]. Rajaraman, J. Leskovec, and J. D. Ullman. “Chapter 7. Mining of Massive Datasets”.
[2]. Jiawei Han and Micheline Kamber (2006), “Data Mining: Concepts andTechniques”, The MorganKaufmann/Elsevier India.
[3]. A.K. Jain, M.N.Murty and P.J.Flynn “Data Clustering: A Review”.
[4]. Anil K. Jain and Richard C. Dubes. “Algorithms for Clustering Data”. Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[5]. Richard O. Duda and Peter E. Hard. "Pattern Classification and Scene Analysis". A Wiley-Interscience Publication, New York, 1973.
[6]. Chris ding and Xiaofeng He (2002), "Cluster Merging And Splitting In Hierarchical Clustering Algorithms".
[7]. Yogita Rani and Dr.Harish Rohil " A Study of Hierarchical clustering algorithm" International Journal of Information and Computation Technology.ISSN 0974-2239 Volume 3, Number 11 (2013), pp. 1225-1232.
[8]. L.V Bijuraj "Clustering and its applications" Proceedings of National Conference on New Horizons in IT – NCNHIT 2013.
[9]. Tian Zhang, Raghu Ramakrishnan, and Miron Livny" Birch: An efficient data clustering method for very large databases". In Proceedings of the ACM SIGMOD Conference on Management of Data, pages 103-114, Montreal, Canada, June 1996.
[10]. S. Bradley, U.M. Fayyad, and C.Reina,"Scaling clustering algorithms to large databases," Proc. Knowledge Discovery and Data Mining, pp.9-15,1988.
[11]. S. Guha, R. Rastogi, And K.Shim"Cure: An Efficie nt Clustering Algorithmfor Large databases," Proc. Acm Sigmod Intl. Conf. On Management Of Data, Pp. 73-84, 1998.
[12]. Pavel Berkhin "Survey of Clustering Data Mining Techniques" , Accrue Software, Inc.
[13]. Jinxin Gao, David B. Hitchcock —James-Stein "Shrinkage to Improve K-meansCluster Analysis" University of South Carolina,Department of Statistics November 30, 2009.
[14]. Narendra Sharma, Mr. Ratnesh Litoriya " Comparision of the various clustering algorithm of weka tools". International Journal of Emerging Technology and Advanced Engineering (ISSN 2250-2459, Volume 2, Issue 5, May 2012).