IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-6, Issue-2 E-ISSN: 2347-2693

Determination of Optimal Number of Clusters in Cure Using Representative Points

Khumukcham Robindro[1*], Bisheshwar Khumukcham[2], Ksh. Nilakanta Singh[3]

[1*]Department of Computer Science, Manipur University, Canchipur, Imphal, Manipur, India
[2]Department of Computer Science, Manipur University, Canchipur, Imphal, Manipur, India
[3]Department of Computer Science, Manipur University, Canchipur, Imphal, Manipur, India

Corresponding Author: rbkh@manipuruniv.ac.in

Available online at: www.ijcseonline.org

Received: 21/Jan/2018, Revised: 29/Jan/2018, Accepted: 12/Feb/2018, Published: 28/Feb/2018

***Abstract* -** In most of the clustering algorithms, the number of clusters has to be supplied in as an input. In CURE clustering algorithm also, the same problem exists. In this paper, we try to find the optimal cluster number in the CURE clustering algorithm by calculating an optimality measure corresponding to each cluster number produced by CURE clustering algorithm after it enters a range ,based on the intra cluster measure and the inter cluster measure of the clusters. The clustering along with the optimality check continues as long the optimality measure is increasing and the cluster number doesn't fall below the minimum boundary of our range.

Keywords— Algorithm, Clustering, CURE, Measure

I. INTRODUCTION

Computational geometry is a branch of computer science that studies algorithms for solving geometric problems. In modern engineering and mathematics, it has applications in diverse fields as computer graphics, robotics, VLSI design and molecular modelling etc.[1]. The representation of spatial data such as points, lines, rectangle, regions, surfaces and volumes etc. is becoming important in the applications of computational geometry [2]. Clustering is a fundamental problem in computational geometry and finds numerous applications in the fields of data mining, image processing and pattern classification and recognition [3,4,5,6,7]. The task of assigning a set of objects into groups so that the objects are more similar to other objects in the group than those objects in other groups is known as cluster analysis or clustering. Clustering is useful for discovering groups and identifying interesting distributions in the underlying data. Theoretical or practical approaches have been investigated for solving variant problems on clustering. Also a number of interesting results have been obtained. CURE algorithm is also among these variants and have received considerable attentions. To handle large databases, CURE employs a combination of random sampling and partitioning in which the random sample drawn from the data set is partitioned and each partition is partially clustered [8]. We refer to the groups mentioned above as clusters. In most of the clustering algorithms, the number of clusters are supplied beforehand as input. The clustering, in case of heirarchical clustering, continues until the number of clusters is same as the supplied number of clusters. The output is the supplied k number of clusters. If there is no efficient mechanism to find this best desired number of clusters,the number has to be supplied as a guessed number. In CURE clustering algorithm also, there is no provision for deciding the optimal number of clusters. Here we try to find a method for finding the optimal number of clusters with respect to CURE algorithm using the representative points of the clusters.

This paper is presented in five headings besides the Introduction. The second heading presents a brief explanation on CURE clustering algorithm. Finding an optimal number of clusters has always been a problem in most of the clustering algorithms. In CURE also, there is a need to find the optimal number of clusters to be supplied to the clustering algorithm as an input. In the next heading, for determining whether the considered number of clusters in the CURE algorithm is optimal, we check the compactness of the clusters and the separateness between these clusters formed by the CURE algorithm using the representative points. For a high quality clustering result, the clusters should be more compact in itself i.e. the intra cluster distance should be less and the clusters between themselves should be well separated i.e. the inter cluster distance should be more. The next heading is on the approach methods to find the optimal number of clusters from a range that is to be supplied. The last two headings are on the implementation of algorithm by running on a set of points grouped into three clusters and the conclusion.

II. A BREIF ON CURE CLUSTERING ALGORITHM

Clustering algorithm is an unsupervised machine learning process that creates clusters such that data points inside a cluster are close to each other, and also far apart from data points in other clusters. There are four main categories of clustering algorithms: partitioning, density-based, grid-based and hierarchical algorithms. K-means and PAM are the examples of partitioning algorithm which iteratively refines a set of *k* clusters and do not scale well for larger data sets [9,10]. DBSCAN and DENCLUE under the Density-based algorithms, are able to efficiently produce clusters of arbitrary shape and are also able to handle outliers [11,12]. In these algorithms, if the density of a region is above a specified threshold, those points are assigned to a cluster; otherwise considered to be noise. Grid-based algorithms reduce the clustering space into a grid of cells, enabling efficient clustering of very large datasets [13]. Hierarchical algorithms can be either bottom-up or top-down approach. CURE and Chameleon are examples of hierarchical algorithms [14,15].

CURE stands for Clustering Using Representatives. It is a hierarchical clustering algorithm. In CURE algorithm, unlike the traditional methods where a single central measure is used as a base for clustering the objects (like the centroid of the cluster), a constant number of well scattered points in a cluster are first chosen. This constant number of scattered points is then shrunk towards the centroid of the cluster by a fraction α. After shrinking, these points are used as the representatives of the cluster. Since we are considering a set of well scattered points as the representatives of the cluster, the shape and extent of the cluster is effectively captured by these points. The clusters with the closest pair of representative points are merged in each step of CURE's hierarchical clustering algorithm. For a pair of point p and q, dist (p, q) denotes the distance between the points. This distance could be L_1 ("Manhattan") or L_2 ("Euclidean") metrics. In CURE, the number of clusters *k* into which the data set is to be clustered is also required to be specified. For best result, the optimum number of clusters, *k*, should be supplied.

A. Representative Points

Since we are using the representative points of a cluster in our method, we are presenting here a detail description of the representative points as they are described in the original CURE paper. As already mentioned above, representative points of a cluster are a set of well scattered points in the cluster shrink by a fraction towards the centroid of the cluster. For selecting the 'c' well scattered points, an iterative procedure is followed. Firstly, the point farthest from the centroid of the cluster is added to the well scattered points set. The next point to be added to the set is the point in the cluster which is farthest from the points in the set. This process continues until the desired number of scattered points have been selected which is given by 'c'.

1) Number of Representative Points: The number of representative points 'c' having a value greater than 10, always finds the right clusters in CURE. If the value of 'c' is less than 10, the representative points fail to capture the real geometry of the cluster and tend to split big clusters.

2) The Shrink Factor α: CURE algorithm always finds the right clusters when the shrink factor α is in the range 0.2 -0.7. A value outside this range doesn't result in a good clustering.

In CURE clustering algorithm, the number of clusters in the data set which are to be clustered have to be provided in advance. Based on the provided number of clusters, the algorithm carries on the clustering. Finding an optimal number of clusters has always been a problem in most of the clustering algorithms. In CURE also, there is a need to find the optimal number of clusters to be supplied to the clustering algorithm as an input.

III. FINDING AN OPTIMAL K-VALUE

In centroid based data mining algorithms like K-means, the quality of the clustering result is checked by checking the compactness within the clusters and the separateness of the clusters based on the centroid of the cluster[]. In CURE, since representative points are used in the clustering, we will used the representative points to check the quality of the cluster for a given number of clusters in order to obtain the optimal value of the number of clusters that we are using in the clustering. For determining whether the considered number of clusters in the CURE algorithm is optimal, we check the compactness of the clusters and the separateness between these clusters formed by the CURE algorithm using the representative points. For determining whether a cluster K_i is compact, we use the intra cluster distance and for determining the separateness of the clusters, we use the inter cluster distance. The above mentioned intra cluster and inter cluster distances will be discussed in the subsections of this section. The intra cluster distance and the inter cluster distance will then be used to find an optimality parameter which will determine the quality of the clustering using that particular number of cluster. For a high quality clustering result, the clusters should be more compact in itself i.e. the intra cluster distance should be less and the clusters between themselves should be well separated i.e. the inter cluster distance should be more.

A. Intra Cluster Distance Calculation

The intra cluster distance is to be calculated for each cluster formed by CURE clustering algorithm. For each point, distance from each representative point is taken into account

and the minimum of these values is taken into consideration. Then the sum of all these minimum distances is calculated. The average of this calculated sum is taken as the intra cluster distance.

For cluster $K_{i,}$ we calculate the intra cluster distance using the equation given below

$$\mathbf{Ita_i} = \sum_{x \in Ki} (\min_{j=1 \text{ to } c} (\text{dist}(x, rp_{ij}))) \quad \text{- Equation 1}$$

where x denotes a point in a cluster, *c* denotes the number of representative points in a cluster and rp_{ij} denotes the j^{th} representative point of the i^{th} cluster.

After calculating the intra cluster distance for all the clusters, a summation operation is carried out on the values and it is divided by the number of points in our dataset. This resulting value is taken as the intra distance of the clusters. It is given by

$$\mathbf{Intra} = \frac{1}{N} \sum_{1 \le i \le k} (\mathbf{Ita_i}) \quad \text{- Equation 2}$$

where Ita_i is from equation 1, k is the number of clusters we are considering, and N is the number of points in the dataset.

A cluster is compact if the intra cluster measure of the cluster is small. As a whole, a small value of the intra cluster distance that we have calculated shows greater compactness of the clusters. A large value, on the other hand, shows lesser compactness.

B. Inter Cluster Distance Calculation

The inter cluster distance will be calculated based on the representative points of each cluster. This distance is to be calculated for each pair of clusters formed by CURE. The minimum distance between the representative points of cluster pair will be considered as the inter cluster distance between the cluster pair.

Now,

$$\mathbf{Intr_{ij}} = \text{Min1} \le l \le c \, (\text{dist}(rp_{il}, rp_{jl})) \quad \text{- Equation 3}$$

where $rp_j \in K_j$, $rp_i \in K_i$, $rp_j \notin K_i$, $rp_l \notin K_j$., c is the number of representative points, i and j represents cluster I and cluster j.

Using the above calculated minimum distances, we calculate the inter cluster distance of the whole clusters as

$$\mathbf{Inter} = \frac{1}{M} \sum (Intr_{ij}) \quad \text{- Equation 4}$$

where M=kC2 (the number of cluster pairs), $Intr_{ij}$ is the distance between cluster i and cluster j calculated in equation 3.

These inter cluster measure **Inter** gives the separateness of the clusters. A larger inter cluster distance means the clusters are more distant from each other. A smaller value indicates the clusters are closer to each other on the whole. For a quality clustering, we want this measure to be higher.

C. Formulation of an optimality parameter

As stated above, for a good clustering result, the selected number of clusters that we are using should result in small value of the intra cluster distance and a large value of inter cluster distance. Keeping these two points in mind, we now calculate our optimality parameter as

$$\text{Optimality measure} = \frac{Inter}{Intra} \quad \text{- Equation 5}$$

where Inter and Intra are the values calculated in equation 4 and equation 2 respectively.

A larger *Optimality measure* will be produced by a larger inter cluster distance and a smaller intra cluster distance. The number of clusters that we are using is considered to be the optimum number of clusters if it produces the largest value of the *Optimality measure*

IV. APPROACH METHOD

We will be trying to find the optimal number of clusters from a range that is to be supplied. This range will be the range in which the optimal number of clusters is likely to fall. If we are taking the range as k_{max} to $k_{min,}$ then the checking for optimality of the number of clusters form by CURE algorithm will start when the number of clusters reaches the value k_{max}. From this k_{max} onwards, we will check the quality of the clusters formed. The clustering by CURE will continue as long as the optimality measure is on an increasing trend or the minimum possible number of clusters, k_{min} is reached. The modifications made to CURE clustering algorithm is shown in the figures that are to follow. One procedure shows the original clustering algorithm of CURE with the required modifications in it and the other procedure is our optimality measure algorithm for checking the quality of the clusters formed.

A. Modified CURE clustering algorithm


```
Procedure cluster (S,k)

begin

1.      bool test=false
2.      T:= build_kd_tree(S)
3.      Q:= build_heap(S)
4.      while (((size(Q)> k_min) and ((!test and size(Q)>k_max)or(test and size(Q)≤ k_max))) {
5.          u:= extract_mean(Q)
6.          v:=u.closest
7.          delete(Q,v)
8.          oldu=u
9.          oldv=v
10.         w:=merge(u,v)
11.         delete_rep(T,u);
delete_rep(T,v);insert_rep(T,w)
12.         w.closest:=x /* x is an arbitrary cluster in Q */
13.         for each x∈ Q do {
14.             if dist(w,x)< dist(w,w.closest)
15.                 w.closest:=x
16.             if x.closest is either u or v {
17.                 if dist(x,x.closest)< dist(x,w)
18.
        x.closest:=closest_cluster(T,x,dist(x,w))
19.                 else
20.                     x.closest:=w
21.                 relocate(Q,x)
22.             }
23.             else if dist(x,x.closest)> dist(x,w) {
24.                 x.closest:=w
25.                 relocate(Q,x)
26.             }
27.         }
28.         insert(Q,w)
29.         If size(Q)≤ k_max
30.             test= optimality_test(Q)
31.     }
32.     delete(Q,v);insert(Q,oldu);insert(Q,oldv)
33.     delete_rep(T,w);
insert_rep(T,oldu);insert_rep(T,oldv)

end
```

B. Procedure for optimality test

```
Procedure optimality_test(Q)

begin

1.      static oldOM=0;
2.      ita:=0, intra:=intr:=inter:=0
3.      k:=size(Q)
4.      M:=(k^2−k)/2
5.      for each cluster x∈ Q
6.          flag_x:=0
7.      for each cluster x∈ Q {
8.          flag_x :=1
9.          for each point p in cluster x
10.             ita:=ita + min(dist(p,rp): rp ∈ x.rep)
11.         for each cluster (y:y ∈ Q −x and flag_y=0){
12.             intr:=intr + min(dist(rp_x,rp_y): rp_x ∈ x.rep and
rp_y∈ y.rep)
13.     }
14.     intra:=ita/N
15.     inter:= intr/M
16.     newOM:= inter/intra
17.     if newOM > oldOM {
18.         oldOM:=newOM
19.         return true
20.     }
21.     else
22.         return false

end
```

C. Clustering Algorithm

In the modified CURE clustering algorithm (fig. 1) which is only a small modification of the original CURE algorithm (the original algorithm has been reproduced here with added modifications), we have made changes in the condition that the clustering algorithm followed as it goes on merging the clusters until the desired number of clusters is not reached. Instead of just checking whether a particular number of clusters have been reached or not, we are also using the optimality measure indirectly. The clustering will continue only if the number of clusters is still greater than the minimum number of clusters in the range that we considered before and some added conditions. The added condition consist of two parts, one corresponds to those where the number of clusters is still above the maximum number of clusters in our considered range and the other corresponds to those where the number of clusters is equal to or less than k_{max} . In the algorithm, we make use of a Boolean variable test. At the beginning of the algorithm, its value is set to false. This Boolean variable also plays an important part, or we can say the most important part in our modification to the condition of the clustering algorithm. The content of test will remain false until the optimality_test procedure is invoked. We will explain this procedure later. The 'test' stores the value returned by the optimality_test procedure. In the clustering algorithm, the procedure is not invoked until the number of clusters is inside the specified range of clusters

and so its value will be always false. This corresponds to the first part mentioned above. So, the clustering algorithm of CURE will continue clustering unaffected by the changes made by us as long as the number of clusters is above k_{max}. **'(size(Q)> k_{min}) and ((!test and size(Q)>k_{max})'** will take care of this. Once the number of clusters enters the range, then the optimality_test procedure will be invoked and as a result the content of 'test' may change accordingly. And further clustering will take place only if the value of 'test' is true and the number of clusters is less than or equal to k_{max}. The k_{max} mentioned here is the maximum value of the range of number of clusters that we are considering. If the value of 'test' turns out to be false when the number of clusters is within the specified range, then the clustering will not continue further. But instead, we need to go back one step backwards to undo the latest merging of the two nearest clusters. Since we saved the two clusters before they were merged in steps 8 and 9, the merged cluster is removed and the two saved clusters are inserted into Q. This will be the optimal solution of the clustering as further clustering after this yields lesser optimality measure. So the clustering algorithm will stop here.

D. Optimality testing procedure

After the number of clusters enters the range provided, the optimality_test procedure is invoked after the merging of two closest clusters to check the optimality measure of the new sets of clusters formed. As the variable oldOM is set to 0, the optimality_test procedure will return true when it is invoked for the first time. When the procedure is invoked, it calculates the 'intra' parameter of the clusters which is the average of the distance of all the points in the dataset from the nearest representative points of their respective cluster. The procedure also calculates the 'inter' parameter which is the average of the distance of the nearest representative point pair of a pair of cluster. A flag is used to prevent the reconsidering of a cluster pair which has been already considered previously. After both the 'intra' and 'inter' parameters are calculated, the optimality measure of the clusters is then calculated. If the newOM, which is the newly calculated optimality measure, is greater than oldOM, which holds the optimality measure of the previous clustering state, the oldOM value is replaced by the newOM value and the procedure returns true. This is an indication that the newly formed clusters have higher quality than the previous clusters. If this is not the case, the procedure returns false. This is to indicate that the newly formed clusters have lesser quality than the previous clusters. When the procedure returns false, this is when the clustering algorithm stops clustering further. The cluster set from which the new cluster set has been formed will be the optimal clustering solution and the number of clusters will be the optimum cluster number.

E. Complexity of the optimality testing procedure

In the proposed optimality testing algorithm, for calculating the 'intra' parameter, there will be c*N distance calculations and for calculating 'inter' parameter, there will be k*c*c distance calculations.

V. IMPLEMENTATION

As we are pertaining to the modification in CURE clustering algorithm for the determination of the optimal number of clusters in clustering process, it is presented here the working of the optimality measure calculation which are trying to embed in the clustering process of the CURE clustering algorithm. As the clustering by CURE enters a predefined range of cluster numbers, the function determines the optimality measure of the current number of clusters and the clustering is allowed to proceed only if the optimality measure is increasing. The clustering is stopped and the previous clusters are considered to be the best clusters if the current clusters deliver a lower optimality measure as compared to the previous sets of clusters.

Here, the algorithm is implemented by running on a set of points grouped into three clusters (*Note: The points that we are considering here hasn't been really clustered by any clustering algorithm and we have done it manually just to imitate a set of clusters on which to run our implementation*.). From the points in each cluster, we pick out six well scattered points each to act the representative points for each cluster. Then the points of each cluster and the representative points are inputted into our implementation. When we run the program for two different sets of clusters, one set which is supposed to be a good set of clusters, and another set which is supposed to be a bad set of clusters (*the good set of clusters here contain points which are more similar to each other and the bad set of clusters contain points which are all no very similar*) the good set of clusters return a better optimality measure as compare to the bad set of clusters.

Result of the 1st set of clusters considered (the good set of clusters)

Snapshots of the input:


```
Enter the NUMBER OF POINTS that we are considering:     150

Enter the NUMBER OF CLUSTERS to be evaluated:   3

Enter the NUMBER OF REPRESENTATIVE POINTS in each cluster:      6

               Cluster-1

Enter the POINTS in the cluster:<ENTER (999,0) WHEN YOU ARE FINISHED ENTERING>
 2.5922    0.1608
   2.6613    0.1023
   2.7535    0.0439
   2.8687    0.0146
   2.9608   0.0439
   3.0069    0.1023
   3.1452    0.1608
   3.2604    0.2193
   3.3295    0.3363
   3.3295    0.5409
   3.2834    0.5409
   2.9608    0.5409
   3.0760    0.5409
   2.8456    0.5409
   2.7535    0.5409
   2.2696    0.5409
   2.5461    0.5702
   2.3387    0.5702
   2.1774    0.5409
   2.0853    0.5409
   1.6935    0.5409
   1.8318    0.5409
   1.9240    0.5409
   1.9009    0.4532
   3.1682    1.8275
   2.9608    1.8567
   2.8226    1.8567
   2.6843    1.8567
   2.6152    1.8275
   2.7995    1.7398
   3.0069    1.9444
   3.0300    2.0322
   3.1682    1.9737
   2.6613    1.6520
   2.4078    1.5936
   2.3387    1.6520
   2.4078    1.6813
   2.2465    1.7690
   2.1083    1.8275
   2.0392    1.8275
   1.8548    1.9444
   2.3157    1.3304
   2.1544    1.4766
   2.1313    1.4474
   2.1083    0.9795
   2.0161    1.0965
```

```
   2.4309    1.4474
   2.5000    2.0906
999 0

Enter the REPRESENTATIVE POINTS:
1.8318   0.5409
2.2696   0.5409
2.8456   0.5409
2.5922   0.1608
3.0069   0.1023
3.3295   0.5409

               Cluster-2

Enter the POINTS in the cluster:<ENTER (999,0) WHEN YOU ARE FINISHED ENTERING>
1.8548    2.2076
   1.8318    2.1199
   1.8779    1.7690
   1.8779    1.4181
   1.8779    1.2427
   1.8779    0.9795
   1.8779    0.8041
   1.9009    0.7164
   1.9009    0.4825
   1.9009    0.2485
   1.8779    0.0146
   1.8318    0.1901
   1.8318    0.2778
   1.9700    2.1199
   2.1774    2.0906
   2.2926    2.0322
   2.4309    1.9737
   2.6382    1.9737
   3.2834    2.1491
   2.8917    2.1491
   2.7304    2.1491
   2.3848    2.2076
   2.2005    2.2076
   2.4447    1.9737
   3.5829    2.0614
   3.6290    1.9152
   3.4908    1.7690
   3.3986    1.7398
   3.2143    1.7105
   3.0300    1.5643
   2.9839    1.5643
   2.8226    1.4766
   2.7765    1.4766
   2.7074    1.4181
   2.5922    1.3596
   2.3157    1.3012
   2.3848    1.1842
   2.2465    1.1842
   2.1544    1.1257
   2.1544    0.9795
   2.1313    0.9211
   2.1313    0.8333
```

```
   2.5461    0.1901
   2.4770    0.1901
999 0

Enter the REPRESENTATIVE POINTS:
0.2834    2.1491
3.6290    1.9152
2.3848    2.2076
2.8226    1.4766
2.1313    0.9211
2.5922    1.3596

               Cluster-3

Enter the POINTS in the cluster:<ENTER (999,0) WHEN YOU ARE FINISHED ENTERING>
0.0346    2.0029
    0.1959    2.0614
    0.4493    2.0614
    0.3111    2.0614
    0.6106    2.0322
    0.2650    1.9737
    0.2650    1.9152
    0.3802    1.9444
    0.2650    1.7690
    0.1959    1.6813
    0.0115    1.6813
    0.0576    1.6520
    0.2880    1.5643
    0.3571    1.5351
    0.4954    1.5058
    0.2650    1.3889
    0.1498    1.4181
    0.0806    1.3889
    0.2880    1.2135
    0.4032    1.1842
    0.5184    1.1842
    0.5184    1.2719
    0.3571    1.3304
    0.5645    0.7749
    0.7028    0.8041
    0.4724    1.0380
    0.3571    1.0088
    0.0115    1.4474
    0.2419    1.5936
    0.5415    1.7105
    0.4493    1.7690
    0.4032    2.0322
    0.0806    2.0322
    0.0576    1.8275
    0.4493    2.0029
    0.5184    2.0906
    0.4724    0.7456
    0.3571    0.7456
    0.4032    0.8333
    0.4954    0.8626
    0.5184    0.5702
    0.4263    0.5702
```

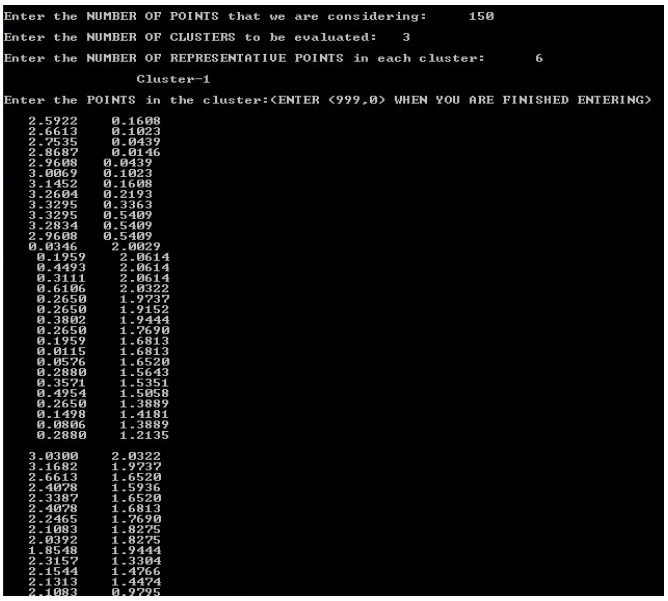
```
    0.5184    0.5702
    0.4263    0.5702
    0.4263    0.3947
    0.4954    0.3363
    0.5184    0.1901
    0.4724    0.1608
    0.3341    0.1316
    0.3341    0.1316
    0.3111    0.3070
    0.2189    0.2193
    0.1959    0.1901
    0.3341    0.0146
    0.4493    0.0731
    0.4954    0.2193
    0.4954    0.2485
    0.4954    0.3363
    0.4493    0.4240
    0.3341    0.4240

999 0

Enter the REPRESENTATIVE POINTS:
0.5415    1.7105
0.3571    1.3304
0.4493    0.0731
0.0115    1.6813
0.2650    1.9152
0.1498    1.4181
```

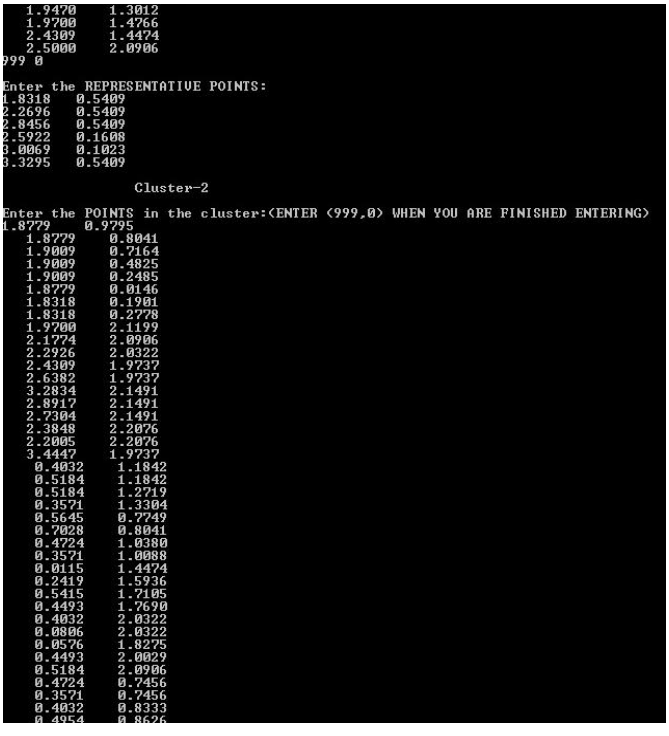
Snapshot of output:

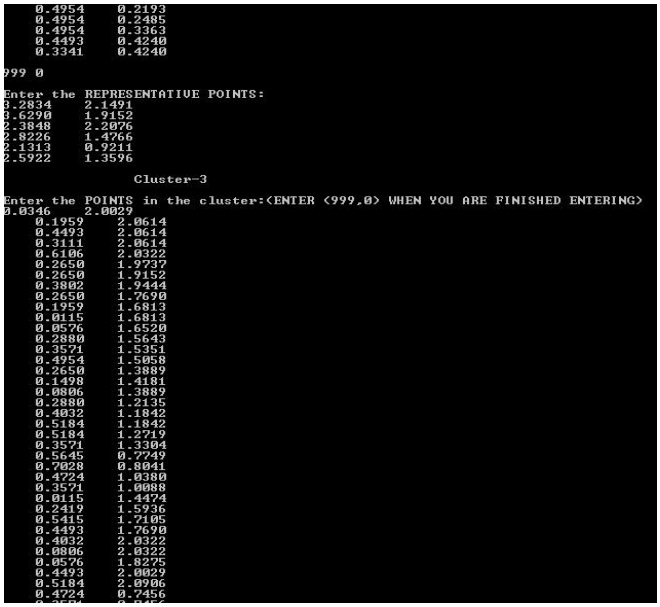
```
      Intra Measure    Inter Measure    Optimality Measure(Inter/Intra)
          0.415508      1.213024         2.919376
```

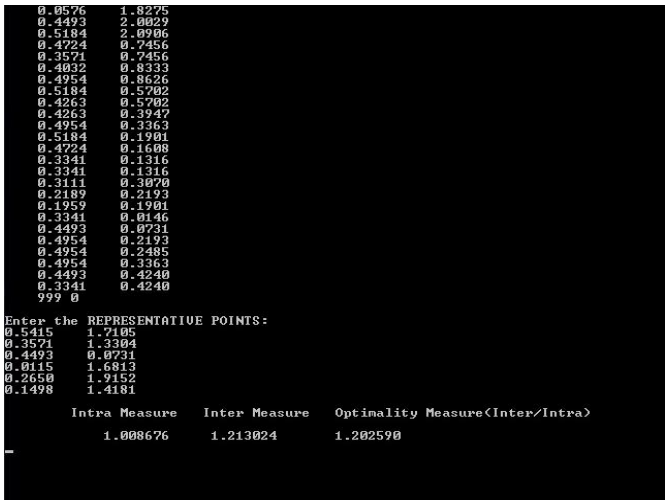

Result of the second set of clusters considered (the bad set of clusters)

Snapshots of the input:

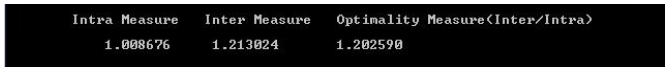








Snapshot of output:



Inference from the two results:

Cluster set	Intra measure	Inter Measure	Optimality Measure
Set 1	0.415508	1.213024	2.919376
Set 2	1.008676	1.213024	1.22590

From the result, it is seen that a better set of clusters has a greater value optimality measure. Set 1 which is supposed to have better clusters have an optimality measure of 2.919376 while set 2 has an optimality measure of 1.22590.

Hence, the optimality measure can be used to find the best set of clusters. The best set of clusters is supposed to have the best optimality measure.

VI. CONCLUSION

CURE clustering algorithm which is used for clustering large databases is a better algorithm than the traditional clustering algorithm in many respects. CURE is effective in finding arbitrary shaped clusters, and effectively handles the problem of outliers. CURE handles large databases efficiently by employing a combination of random sampling as well as partitioning. But CURE doesn't have a provision for checking for the optimum number of clusters. A desire number of cluster is taken as input to the algorithm. In this work, it is tried to improve CURE by adding a mechanism in CURE clustering algorithm itself which checks the quality of the clusters formed by the algorithm and thus determine the best number of clusters based on their quality.

REFERENCES

[1] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*, The MIT Press, Massachusetts, 1990.

[2] Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Publishing Company Inc., New York, 1990.

[3] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley-Interscience, New York, 2001.

[4] J.Han, M. Kamber, Data Mining, Morgan Kaufmann Publishers, 2000.

[5] M. Grotschel, L. Lovasz, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, second ed., Algorithm and Combinatorics, vol. 2, Springer-Verlag, Berlin, Heidelberg, 1994.

[6] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: A review, ACM Computing Surveys 31 (1999) 264–323.

[7] V.V. Vazirani, Approximation Algorithms, Springer, 2001.

[8] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim (1998*) CURE: An Efficient Clustering Algorithm for Large Databases*

[9] Siddheswar Ray and Rose H. *Turi Determination of Number of Clusters in K-Means Clustering and Application in Colour Image Segmentation*.

[10] Ng R., Han J. (1994) Efficient and effective clustering method for spatial data mining. In Proc. Conf. on Very Large Data Bases, pp 144-155.

[11] Ester M., Kriegel H., Sander J., and Xu X. (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining*, AAAI Press.

[12] Hinneburg A., Keim D. (1998) An Efficient Approach to Clustering in Large Multimedia Databases with Noise. Proc AAAI.

[13] Seikholeslami G., Chatterjee S., and Zhang A. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. Proceedings of the 24th VLDB Conference, 1998

[14] Guha, S., R. Rastogi & K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In *Proc. Of ACM SIGMOD Intl. Conf. on Management of Data*, pp. 73-82, 1998.

[15] Karypis, G., E. Han & V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8) pp. 68-75, 1999.

Authors Profile

Khumukcham Robindro is currently working as as Assistant Professor in the Department of Computer Science, Manipur University. He joined the Department on 5th November, 2014. Currently, he is Principal Investigator at e-Varaha Project, in collaboration with IIT Guwahati and Tezpur University under ITRA, Media Lab Asia, MeiTy, Govt. of India, in the Department of Computer Science, Manipur University. His area of interests in research includes Machine Learning, Intelligent Systems and Knowledge Discovery.

