Comparison of blocking and hierarchical ways to find cluster

by

Swapnil Kumar

B.Tech., Jawaharlal Nehru Technological University, Hyderabad, India, 2012

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2017

Approved by:

Major Professor
Prof. William H. Hsu

Copyright

Abstract

Clustering in data mining is a process of discovering groups in a set of data such that the similarity within the group is maximized and the similarity among the groups is minimized.

One way of approaching clustering is to treat it as a blocking problem of minimizing the maximum distance between any two units within the same group. This method is known as Threshold blocking. It works by applying blocking as a graph partition problem.

Chameleon is a hierarchical clustering algorithm, that based on dynamic modelling measures the similarity between two clusters. In the clustering process, to merge two cluster, we check if the inter-connectivity and closeness between two clusters are high relative to the internal inter-connectivity of the clusters and closeness of items within the clusters. This way of merging of cluster using the dynamic model helps in discovery of natural and homogeneous clusters.

The main goal of this project is to implement a local implementation of CHAMELEON and compare the output generated from Chameleon against Threshold blocking algorithm suggested by Higgins et al with its hybridized form and unhybridized form.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude to my advisor and mentor Prof. William H. Hsu of the Department of Computer Science, College of Engineering at Kansas State University. I would like to thank him for his constant input and feedback on this project as well as for my Masters at K-State. I would also like to thanks Prof. Mitchell L. Neilsen and Prof. Daniel Andresen for serving in my committee.

Finally, I would like to express my gratitude to my parents and brother who have always motivated and supported me to give my best. This accomplishment would not have been possible without them. Thank you.

Chapter 1 - Introduction

1.1 Problem definition

In general terms clustering, can be defined as the task of grouping a set of objects in such a way that objects in the similarity in the same group is more when compared to objects in other groups.

Cluster is a general task that can be solved with different approaches. Clusters can be attained by various algorithms that differ significantly in what they mean by a cluster and how they approach to find them.

There are different types of clustering models -

- Connectivity models: Hierarchical clustering builds models based on distance connectivity.
- Centroid models: k-means algorithm represents each cluster by a single mean vector.
- Distribution models: Some clusters are modeled using statistical distributions, such as multivariate normal distributions
- Density models: DBSCAN and OPTICS defines clusters as connected dense regions in the data space.
- Subspace models: Bi-clustering, a method in which clusters are modeled with both cluster members and relevant attributes.
- Group models: Few algorithms give emphasis to grouping information rather than providing the grouping information.
- Graph-based models: Highly Connected Subgraphs provide clusters based on the similarity data present in a similarity graph.

In this report, we are going to focus mainly on two partition based clustering algorithm namely-

- Threshold blocking
- Chameleon

A blocking of an experiment's sample is a partition of its units into disjoint sets, referred to as blocks. The blocking problem is to find a blocking where units assigned to the same block are as similar as possible either to minimize differences on important covariates or to facilitate the study of subgroups of interest.

CHAMELEON is a hierarchical clustering algorithm, that based on dynamic modelling measures the similarity between two clusters. In the clustering process, to merge two cluster, we check if the inter-connectivity and closeness between two clusters are high relative to the internal inter-connectivity of the clusters and closeness of items within the clusters.

1.2 Goal and Technical Objectives

The goal of this report is to show the comparison of the above mentioned two techniques of clustering to find meaningful clusters in the dataset.

1.3 Overview and Synopsis

Blocking problems can be viewed as graph partitioning problems. Each experiment yields a weighted graph where vertices represent units in the sample. Edges connect each pair of units, and edge costs are measured dissimilarity between corresponding units (e.g., the Euclidean distance between their covariate vectors). Minimizing the within-block edge costs when this graph is partitioned subject to a cardinality condition is equivalent to deriving an optimal blocking. In the matched-pair design, the objective is to minimize the sum of all within-block edge costs subject to that each block contains exactly two vertices.

CHAMELEON operates on a sparse graph in which nodes represent data items, and weighted edges represent similarities among the data items. This sparse graph representation of the data set allows CHAMELEON to scale to large data sets and to operate successfully on data sets that are available only in similarity space and not in metric spaces. CHAMELEON finds the clusters in the data set by using a two-phase algorithm. During the first phase, CHAMELEON uses a graph partitioning algorithm to cluster the data items into many relatively small sub-clusters. During the second phase, it uses an agglomerative hierarchical clustering algorithm to find the genuine clusters by repeatedly combining these sub-clusters.

Chapter 2 - Background and Related Work

This chapter discusses the previous work that has been done on clustering and the necessary background information of how clustering methodologies have evolved.

2.1 Literature Survey

There are various methods of finding clusters in a dataset and some of them are mentioned below

2.1.1 Hierarchical methods

The given data set is broken down into smaller clusters in a hierarchical manner. It forms a tree like structure to group the data instances. It builds a hierarchy of clusters. There are two major strategies available under this category –

Agglomerative method, which forms the clusters in a bottom-up fashion until all data instances belong to the same cluster.

Divisive method, in which splits up the data set into smaller cluster in a top-down fashion until each of cluster contains only one instance.

Both the divisive algorithms and agglomerative algorithms can be represented by dendrograms. Based on various constraints, Hierarchical clustering techniques at each step decides which clusters should be joined or left untouched.

In agglomerative hierarchical techniques, a typical idea is to merge the “nearest or closest” pair of clusters, where nearest or closest are measured by a specific similarity of cluster. In most of the methods of hierarchical clustering, the similarity of clusters is measured by a metric and linkage criteria. The metric used help in determining the shape of the clusters using Euclidean distance, Manhattan distance, etc. The linkage criterion determines the distance between sets of observations as a function that defines of the closeness between two clusters i.e. through a single link, complete link or an average link. The single link is used to combine cluster that have two most similar instances. Single link is good for handling non-elliptical shapes. Complete link is also known as “Farthest neighbor clustering” as they combine the cluster based on instances that are farther away from each other. The complete link is less susceptible to noise and outliers and has trouble with convex shapes.

The following are some of the hierarchical clustering algorithms are: Balanced Iterative Reducing and Clustering using Hierarchies – BIRCH, Clustering Using REpresentatives – CURE and CHAMELEON.

2.1.2 Partitioning methods

There are two types of partitioning algoriths-

a) Centroid Algorithm

b) Medoids Algorithm.

Each cluster is represented by Centroid Algorithms by utilizing a central vector in instances. Each cluster is represented by medoid algorithm with the help of those instances which are nearest to the center of gravity. A famous centroid algorithm is the k-means. This method partitions the set of data into k subsets in a way that each and every point in a subset that is already given are nearest to the exact same center. A detailed discussion describes that it selects k instances randomly to represent clusters. Depending on the attributes that are selected, the instances that are remaining are assigned to the centers that are closer to them. K-means method now computes the new centers by taking the average of all the data points that belongs to the same cluster. This is an iterative process and it does not end until there’s some change in the gravity centers. K, unless known beforehand, several values of k is calculated until the most suitable one is found out. The function that measures the distance between instances leads to the effectiveness of this method and other methods. A number of procedures define the distance between the instances. Some important feature of the K-means algorithm are-1)It can process large sets of data efficiently.2)A local optimum marks its termination3)Spherical shaped clusters4)Its noise sensitivity. Different variations of k-means method help to eradicate this problem. The major/key step to the basic k-means process is the making a proper choice of the initial centroids. K-modes is a recent partitioning algorithm that deals with categorical attributes by using a coefficient matching measure. Through the combined dissimilarity measure definition, the K-protypes algorithm integrates k-means and k-modes procedure to allow clustering of instances as described by the mixed attributes. In the recent times another generalization of k-means algorithm is presented. It is applicable to elliptical and ball shaped date clusters without dead-unit problem and also without pre-determining the exact cluster number performs correct clustering. Partitions are formed by traditional clustering approaches. Each and every pattern belongs to a cluster. Hard clustering clusters are disjoint.

This presentation is extended by fuzzy clustering to associate each single pattern with each and every cluster using a membership function. High confidence in assigning pattern to clusters is specified by large membership values. One algorithm that is based on k-means and is widely used is called Fuzzy C-Means(FCM) algorithm. This searches for the most characteristic point in each and every cluster which can be called the center of the cluster and also makes an attempt to find the grade of membership in the cluster for each of its instance. Most soft clustering algorithms are based on the ExpectationMaximization (EM) algorithm. This assumes a probabilistic model with parameters that describes the probability of an instance belonging to a certain cluster. The process of this algorithm starts with initial guesses for the parameters of mixture model. The cluster probabilities for each instance is calculated with these values. Then these probabilities re estimate the parameters and the process cycle again is repeated. A major drawback of this kind of algorithms is that they are not at all computationally inexpensive. One other problem found in the above mentioned approach is over-fitting. This problem is caused by two reasons-1)specifications of a larger number of clusters 2)there are too many parameters in the distribution of probabilities. One possible solution is to adhere to a total Bayesian approach, where each and every parameter has a prior probability distribution.

2.1.3 Density based methods

Clusters based on density of regional data points can be evaluated by Density-based clustering algorithms. The principle idea associated with density based clustering is that for every single instance of a cluster the locality of an already known radius(Eps) at the least must contain a minimum number of instances called MinPts. A famous Density based clustering algorithm is DBSCAN. In DBSCAN the data points are divided into Core points, Border points and Noise points i.e. basically into three classes precisely. The data points at the inner part of a cluster are known as Core Points. A data point which is not a Core Point is called a border point. The data points which are neither core points or border points are called Noise Points.

2.1.4 Grid based methods

Quantization of the clustering space into finite number of cells and performing necessary operations on that quantized space is the basic principle of Grid-based clustering algorithms. Dense Cells contain a greater number of data points than a certain number and are joined together to form the clusters. Some well known grid-based clustering algorithms are STatistical INformation Grid-based method – STING , WaveCluster, and CLustering In QUEst – CLIQUE.In Sting,a structural hierarchy is formed by dividing the spatial area into a number of rectangular cells.

2.1.5 Model based methods

Initiating with the random initialization of the parameters Autoclass uses an approach called the Bayesian approach to adjust them incrementally to attempt to find the maximum likelihood estimation. It is also presumed that there are also hidden variables adding up to the predictive attributes. This hidden or unnoticed variable is a reflection of the cluster membership for each and every case in the set of data. The clustering problem is a observed and supervised learning from some data that is incomplete because of the presence of such unobserved and hidden variable. This supervised method of learning is called RBMNs. SOM net is another method that is based on model. It can be imagined as a double layered neural network. Every single neuron is an n-dimesioned weight vector, m = (m1, … , mn),where the dimension of input vector is equal to n. The cluster centers are the neurons of The S.O.M. The map units are connected to form larger clusters to allow interpretation. The S.O.M is iterative in nature. During training, at every step, a vector sample x is chosen randomly from the input data set. The distance between sample vector x and all other S.O.M weight vectors is calculated using a measure of the distance. With the BestMatching Unit, in the input space the BestMatching Unit is moved nearer to the input vector by updating the weight vectors of the S.O.M. The B.M.U topological neighbors are treated similarly. The robust nature of S.O.M stands to be one of its vital property. Map leads to an easy detection of the outlier since it has a larger distance in input space from other units. Values that are missing can be dealt with S.O.M. Some applications requires big amount of high dimensioned data sets to be clustered. But most clustering techniques that are automated do not work efficiently on data set that is high dimensional.

2.2 Limitation of previous work

A major drawback of both Centroid based and Medoid based schemes is that they fail to determine genuine clusters for data in which points in one cluster are closer to the center of another cluster than to the center of their own cluster. This can happen in many natural clusters for example, if there is a large variation in cluster sizes or when cluster shapes are convex.

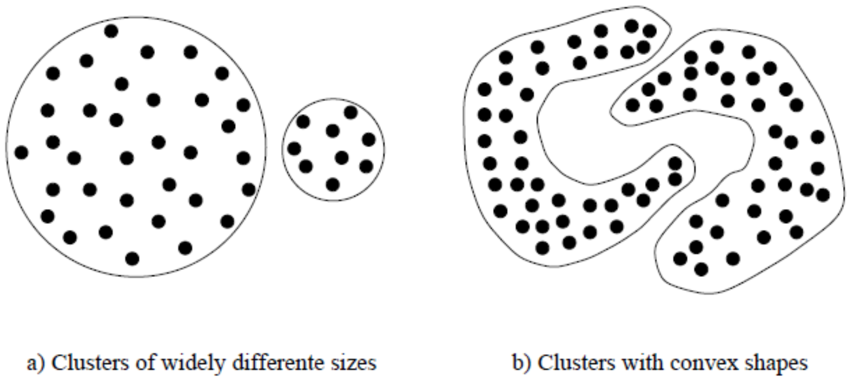
a) Clusters of widely differente sizes
b) Clusters with convex shapes

Figure 1 Cluster with different shapes

A major limitation of agglomerative hierarchical schemes such as the Group Averaging Method, ROCK, and CURE is that the merging decisions are made based upon static modeling of the clusters to be merged. As these schemes do not take the special characteristics of individual clusters into consideration, they perform incorrect merging decisions when the underlying data does not fit in the assumed model, or in case of noise and outliers.

Let us consider the four sub-clusters of points in 2D shown in Figure 2. The selection mechanism of any single link method will prefer to merge clusters (a) with (b) rather than merging clusters (c) and (d), since the minimum distances between the representative points of (a) and (b) will be smaller than those for clusters (c) and (d). But clusters (c) and (d) are better candidates for merging because the minimum distances between the boundary points of (c) and (d) are similar as the average of the minimum distances of any points within these clusters to other points. Hence, merging (c) and (d) will lead to a more homogeneous and natural cluster than merging (a) and (b).

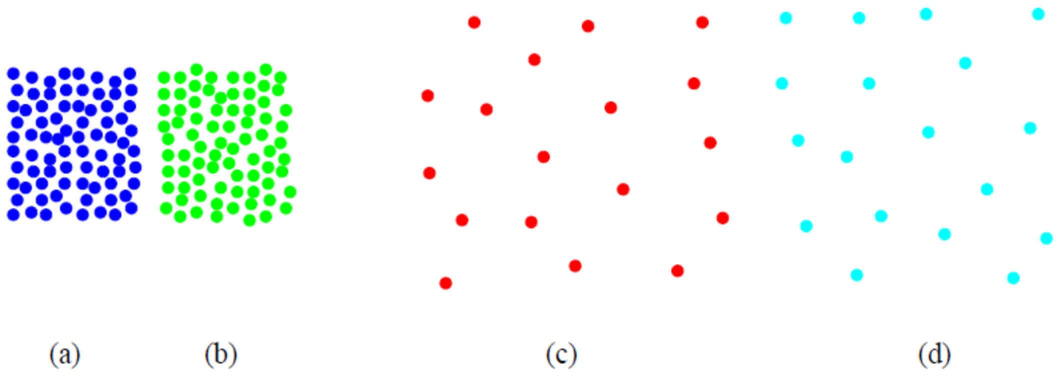

Figure 2 Four clusters of two similar shape

In agglomerative schemes based upon group averaging and related schemes, the way the clusters are to be merged can be scaled with respect to the expected connectivity between these clusters. However, these schemes work by taking a static, user supplied inter-connectivity model to merge the clusters which is the main limitation of all such schemes, which is inflexible and can easily lead to wrong merging decisions when the data set does not follow a certain model or when different clusters exhibit different inter-connectivity characteristics. Although some schemes based on heuristics manage to have connectivity to be different for different problem domains, but it is still the same for all clusters irrespective of their densities and shapes.

Let us consider the two pairs of clusters shown in Figure 3, where each cluster is represented by a sparse graph where vertices indicate data items and edges represent that their two vertices are similar and for simplicity, we consider the level of similarity is same for all edges. The number of items in all four clusters is the same. Then these grouping schemes and the group averaging method will select pair (c) and (d) for merging, whereas the pair (a) and (b) is a better choice.

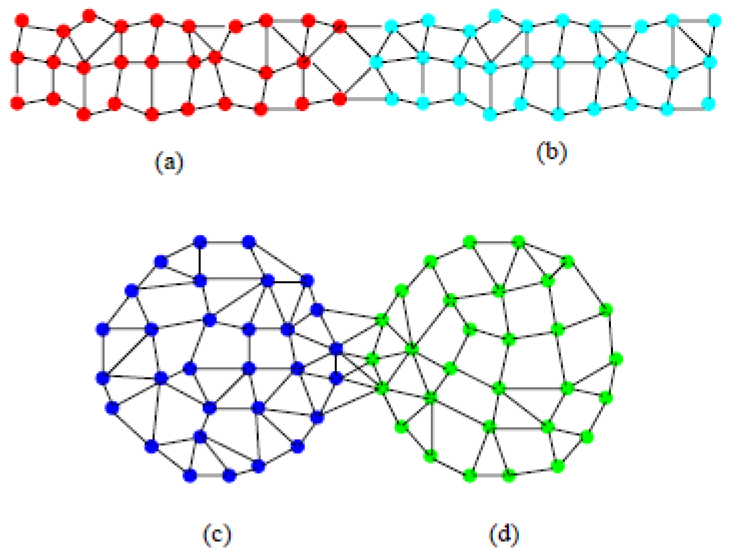

Figure 3 Clusters with different shape

As single link method selection mechanism only considers the minimum distance between the representative points of two clusters, and does not consider the aggregate interconnectivity among the two clusters. Similarly, some of the selection mechanism of algorithms only consider the aggregate inter-connectivity across the pairs of clusters, but ignores the value of the strongest edges across clusters. However, by looking at only one of these two characteristics, these algorithms can easily select to merge the wrong pair of clusters.

Let us consider the example in Figure 4, an algorithm that focuses only on the closeness of two clusters will incorrectly prefer to merge clusters (c) and (d) over clusters (a) and (b). Another scenario would be Figure 5 in which, an algorithm that focuses only on the inter-connectivity of two clusters will incorrectly prefer to merge cluster (a) with cluster (c) rather than with (b). We make an assumption that the aggregate interconnectivity between data items in clusters (a) and (c) is greater than that between data items in clusters (a) and (b). However, the border points of cluster (a) are much closer than those of (b) than to those of (c).)

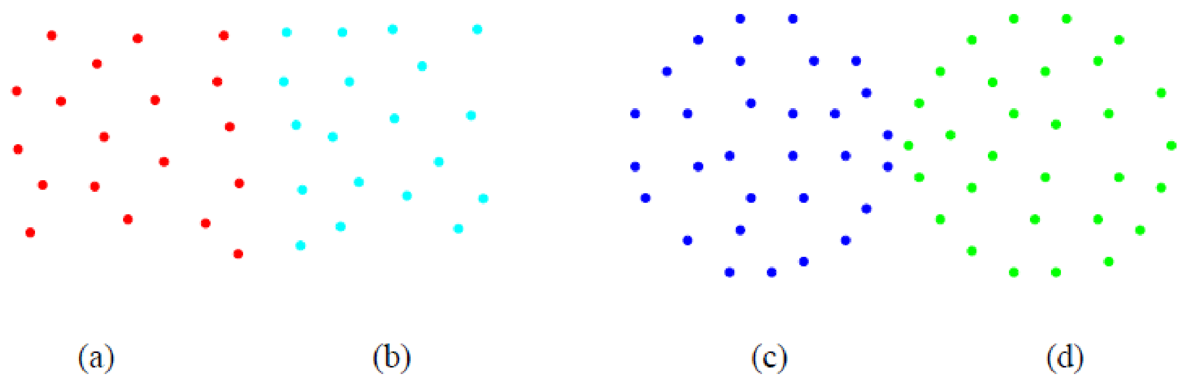

Figure 4 Cluster with different data points

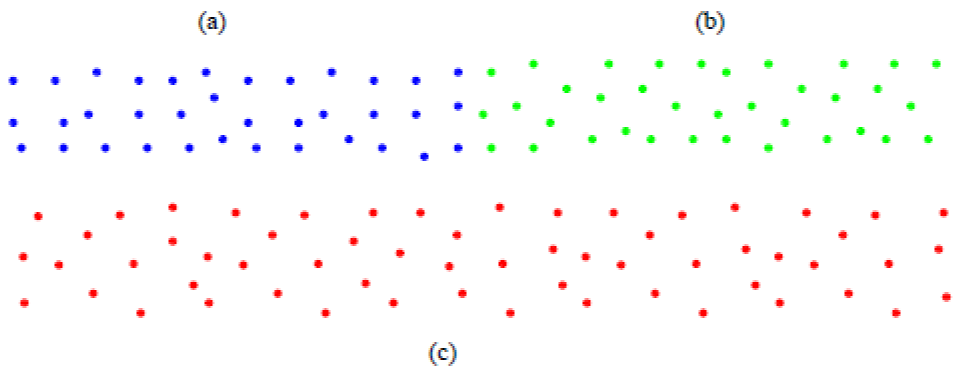

Figure 5 Cluster with different data points

In summary, we find that there are two major limitations of the agglomerative mechanisms used in existing schemes. First, these schemes do not take the features or nature of a cluster into consideration and work based on static modeling. Second, some schemes take only the information about the aggregate interconnectivity of items to merge two clusters, whereas some schemes take information about the closeness of two clusters as defined by the similarity of the closest items across two clusters.

2.3 Useful foundation, framework, components

One of the easiest way to start with clustering is to form a KNN graph out of all the available data points. Then go for partitioning over the graph. This method has been used by various Hierarchical based clustering algorithm.

Hypergraph partitioning is an important problem and has extensive applications in many areas, including VLSI design, efficient storage of large databases on disks, transportation management, and data-mining. The problem is to partition the vertices of a hypergraph in k roughly equal parts, such that the number of hyperedges connecting vertices in different parts is minimized. A hypergraph is a generalization of a graph, where the set of edges is replaced by a set of hyperedges. A hyperedge extends the notion of an edge by allowing more than two vertices to be connected by a hyperedge.

Chapter 3 - Methodology

CHAMELEON uses a two-phase algorithm to find the clusters in the data set. During the first phase, CHAMELEON divides the data items into many relatively small sub-clusters using a graph partitioning algorithm. During the second phase, it repeatedly combining these sub-clusters using an agglomerative hierarchical clustering algorithm to find the genuine clusters in the data set. CHAMELEON operates on a sparse graph in which nodes represent data items, and weighted edges represent similarities among the data items.

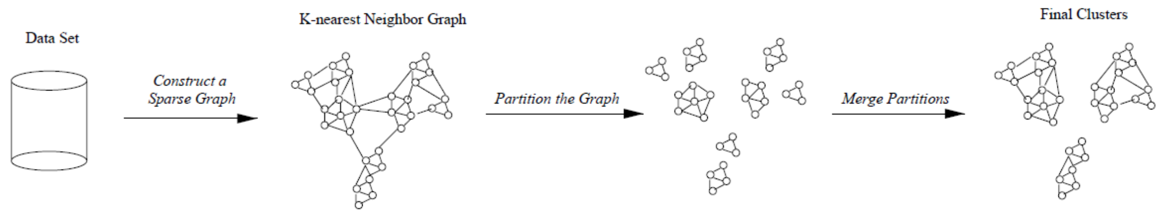

Figure 6 Overall framework of CHAMELEON

The key feature of CHAMELEON’s agglomerative hierarchical clustering algorithm is that it determines the pair of most similar sub-clusters by considering both the inter-connectivity as well as the closeness of the clusters. Using this way of clustering, it can overcome the limitations discussed in previous chapter, that result from using only one of mentioned methods. CHAMELEON uses an approach considers the internal characteristics of the clusters to model the degree of inter-connectivity and closeness between each pair of clusters. So, CHAMELEON can automatically adapt to the internal characteristics of the clusters being merged, as it does not depend on a static user supplied model.

3.1 Modeling the data

In the world of clustering, provided with a similarity matrix, many methods can be used to find a graph representation, which is a very common way of approaching this problem. In CHAMELEON, we represent the sparse graph of the data items using the k-nearest neighbor graph approach. Each vertex of the k-nearest neighbor graph represents a data item. An edge between two data items exists only they are the k-most similar data points of the data point corresponding to the other node.

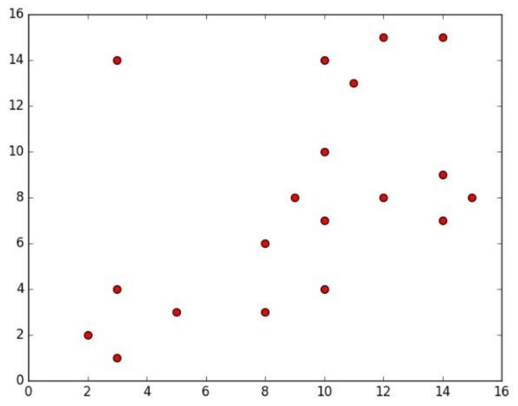

Figure 7 Data points in space

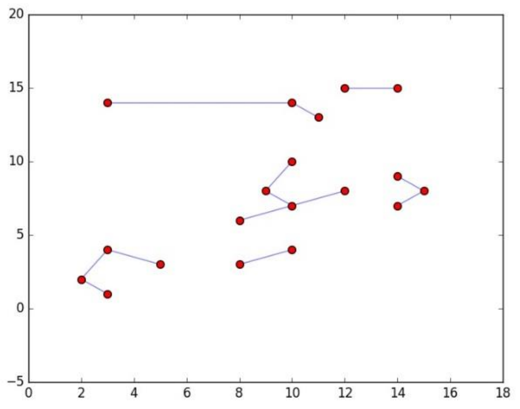

Figure 8 1NN for the above data points

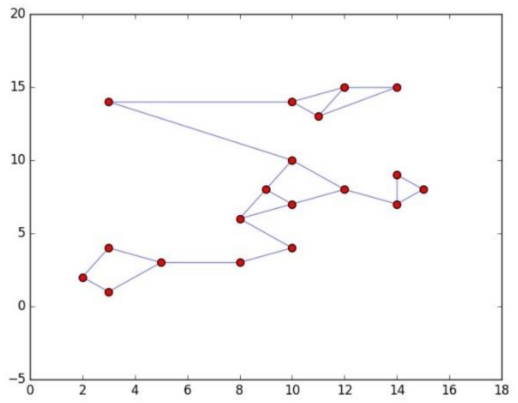

Figure 9 2NN graph for the data points

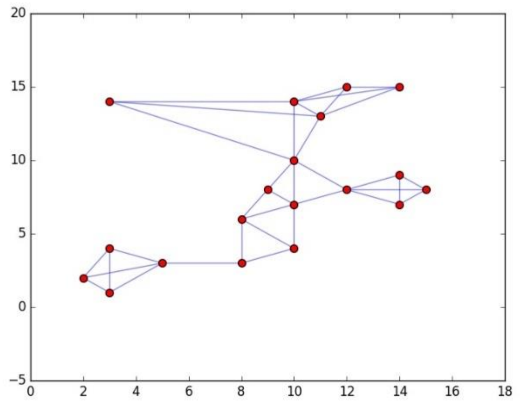

Figure 10 3NN graph of the data points

There are several advantages of representing data using a k-nearest neighbor graph G_k. First, data points that are far apart are completely disconnected in the G_k. Secondly, G_k models its neighborhood dynamically. The neighborhood radius of a data point is determined by the density of the region in which this data point is present. For a dense region, where there are many data points the neighborhood is defined narrowly and in a sparse region, where there are very few data points, the neighborhood is defined more widely. G_k provides a computational advantage over a full graph in many algorithms operating on graphs, including graph partitioning and partitioning refinement algorithms.

3.2 Modeling the Cluster Similarity

The similarity between each pair of clusters C_i and C_j is calculated by taking into account their relative inter-connectivity RI (C_i , C_j) and their relative closeness RC(Ci, C_j). CHAMELEON's hierarchical clustering algorithm selects to merge the pair of clusters for which combination of RI (Ci, C_j) and RC(Ci, C_j) is highest among all the other clusters.

3.2.1 Relative Inter-Connectivity

The relative inter-connectivity between a pair of clusters C_i and C_j is defined as the ration of absolute inter-connectivity between C_i and C_j with average of the internal inter-connectivity of the two clusters C_i and C_j. The absolute inter-connectivity between a pair of clusters C_i and C_j is defined to be as the sum of the weight of the edges that connect vertices in C_i to vertices in C_j . Thus, the relative inter-connectivity between a pair of clusters C_i and C_j is given by

$$RI(C_i, C_j) = \frac{|EC_{\{C_i,C_j\}}|}{\frac{|EC_{C_i}|+|EC_{C_j}|}{2}},$$

which normalizes the absolute inter-connectivity with the average internal inter-connectivity of the two clusters. By focusing on the relative inter-connectivity between clusters, CHAMELEON overcomes the limitations of algorithms that use static inter-connectivity models.

3.2.2 Relative Closeness

The relative closeness between a pair of clusters C_i and C_j is defined as the ratio of absolute closeness between C_i and C_j with respect to the internal closeness of the two clusters C_i and C_j. Some method of finding closeness are based on finding the closest pair of points in between the clusters C_i and C_j. The main drawback of this method is that by relying only on a single pair of points, they are less tolerant to outliers and noise. For this reason, CHAMELEON measures the closeness of two clusters by computing the average similarity between the points in C_i that are connected to points in C_j. As these connections are determined in the graph generated by the k-nearest neighbor algorithm, their average of similarity between the clusters provides a very good measure of the affinity between the data items along the interface layer of the two sub-clusters, and at the same time is tolerant to outliers and noise.

The internal closeness of each cluster C_i can also be measured in several different ways. One possible approach is to find the internal edges and compute the internal closeness of a cluster as the average weight of these edges. In method using hierarchical clustering, the edges used for agglomeration in earlier stages are stronger than those used in later stages. Hence, average weights of the edges on the internal bisection of C_i and C_j will tend to be smaller than the average weight of all the edges in these clusters. But the average weight of these edges is a better indicator of the internal closeness of these clusters.

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}},$$

S_{ECCi} and S_{ECCj} are the average weights of the edges that belong in the min-cut bisector of clusters C_i and C_j, respectively, and $S_{EC\{Ci\ Cj\}}$ is the average weight of the edges that connect vertices in C_i to vertices in C_j . The absolute closeness of the two clusters is normalized by the weighted average of the internal closeness of clusters C_i and C_j, that minimizes the advantage to the cluster with larger number of vertices.

3.2.3 Algorithm

The data items are clustered into many sub-clusters in the first phase such that each sub-cluster contain enough items to allow dynamic modeling. In the second phase of the algorithm, by using the dynamic modeling framework, the sub-clusters are merged in a hierarchical fashion. In the remainder of this section, the two phases of CHAMELEON are explained.

3.2.3.1 Phase I

CHAMELEON finds the initial sub-clusters using a graph partitioning algorithm to partition the k-nearest neighbor graph of the data set into many partitions. The partitions are made such that internal weight (edge-cut) of the cluster calculate by adding the weight of all the edges present in the cluster, is minimized. As each edge in the k-nearest neighbor graph represents the similarity among data points, to minimizes the relationship among data points across the partitions, a partitioning should be made such that it minimizes the edge-cut. The underlying assumption is that links within clusters will be stronger and more plentiful than links across clusters. Hence, the data in the partition are more related to other data items in the same partition when compared to other partitions.

CHAMELEON obtains the initial set of sub-clusters as follows. Initially all the data points are considered to be in the same cluster. The largest sub-cluster among the current set of sub-clusters and is bisect using hMETIS. This process terminates only when the cluster bisected contains fewer data points than a specified number of vertices, that is referred as MINSIZE. The MINSIZE parameter essentially controls the granularity of the initial clustering solution. MINSIZE is set such that it is smaller than the size of most of the clusters that we expect to find in the data set and large enough to evaluate the inter-connectivity and closeness of the items in each sub-cluster in a meaningful fashion.

3.2.3.2 Phase II

After we have the initial sub-clusters from the first phase, we move ahead for merging these sub clusters using a dynamic framework. We employ an agglomerative hierarchical clustering technique that combines these small sub-clusters. The key step in merging of sub clusters in agglomerative hierarchical algorithm is that of find the pair of sub-clusters that are the most similar. For this purpose, we consider a function that takes in relative interconnectivity and relative closeness and then selects to merge the pair of clusters that maximizes this function. Since our goal is to merge together pairs for which both the relative inter-connectivity and the relative closeness are high, a natural way of defining such a function is to take their product. We merge the pair of clusters C_i and C_j that maximizes the function given below -

$$RI(C_i, C_j) * RC(C_i, C_j)^{\alpha},$$

where α is a user specified parameter. If $\alpha > 1$, then CHAMELEON gives a higher importance to the relative closeness, and when $\alpha < 1$, it gives a higher importance on the relative inter-connectivity.

3.2.4 Performance Analysis

The overall computational complexity of CHAMELEON depends on the amount of time it requires to construct the k-nearest neighbor graph and the amount of time it requires to perform the two phases of the clustering algorithm. The amount of time required to find the k-nearest neighbors of a data item is O(n), leading to an overall complexity of $O(n^2)$.

CHAMELEON’s Phase I algorithm generates m clusters by repeatedly partitioning successively smaller graphs, its overall computational complexity is O(nlog(n/m)) which is bounded by O(nlogn).

The amount of time required by the second phase depends on the amount of time needed to compute the internal inter-connectivity and internal closeness for each initial as well as intermediate cluster, and the amount of time needed to select the most similar pair of clusters to merge.

In particular, the amount of time required to bisect each one of the initial m clusters is O(n/m), leading to an overall complexity of O(n). In order to bisect m-1 clusters the complexity rises to O(mn). By using a heap-based priority queue, the overall amount of time required to find the most similar pair of clusters is O(m^2logm). It takes O(m^2logm) time to insert the similarity of the O(m^2) possible pairs of sub-clusters into the priority queue. Now, during each merging step, the pair residing at the top of the priority queue is selected, and the similarity of recently combined cluster to the remaining sub-clusters is updated. Each of these update operations requires O(mlogm) time, leading to an overall complexity of O(m^2logm), for m−1 clusters.

Thus, the overall complexity of CHAMELEON's two-phase clustering algorithm is O(nm + nlogn + m^2logm).

The below figure shows the working of chameleon-

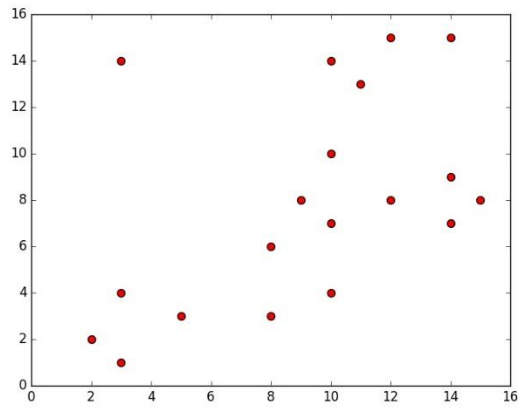

Figure 11 Initial set of data poitns

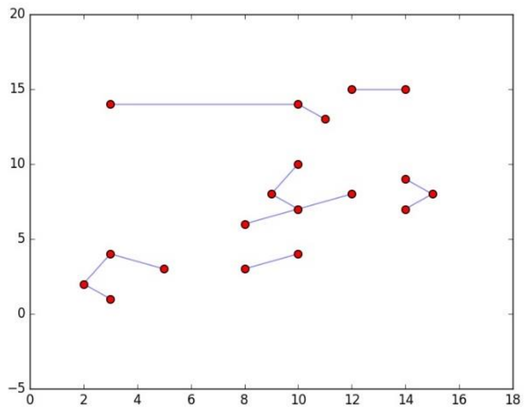

Figure 12 Data points after 1NN

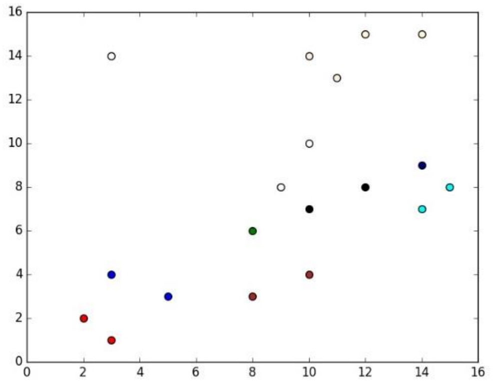

Figure 13 Initial clusters formed after phase I

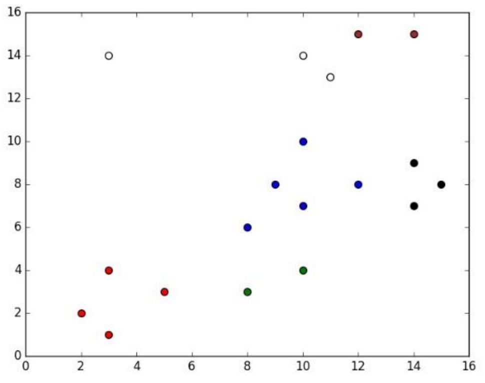

Figure 14 Final clusters formed after phase II

3.3 Threshold blocking

Blocking can be defined as the partition of a set of data points into some disjoint dataset, which are called blocks. The blocking problem finds blocks where data points assigned to the same block are as similar as possible. The main purpose of creating blocks is to minimize differences on covariates and to facilitate the study of subgroups of interest.

3.3.1 The algorithm

Given the graph representation of the data points, G = (V,E), and a pre-specified threshold k, the approximate blocking algorithm proceeds as follows:

1. Construct a (k - 1)-nearest neighbor subgraph of G. Denote this graph G = (V, E).

2. Find a maximal independent set of vertices, S, in the second power of the (k - 1)-nearest neighbor subgraph, G_2. Vertices in S are referred to as the block seeds.

3. For each seed i ∊ S, create a block comprised of its closed neighborhood in G, V = N $_{nn}$[i].

4. For each yet unassigned vertex, assign it to any block that contains one of its adjacent vertices in G.

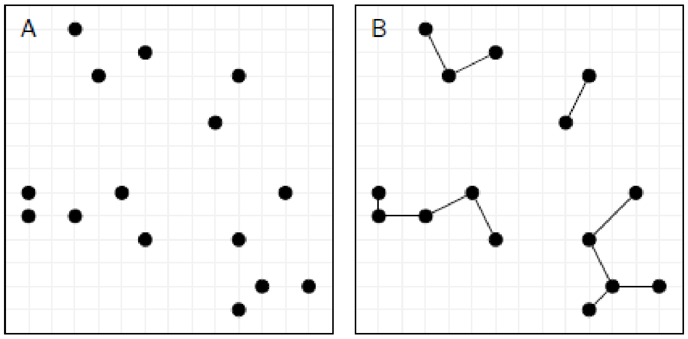

Figure 15 (A) Set of data points and (B) 1NN graph G for the data points

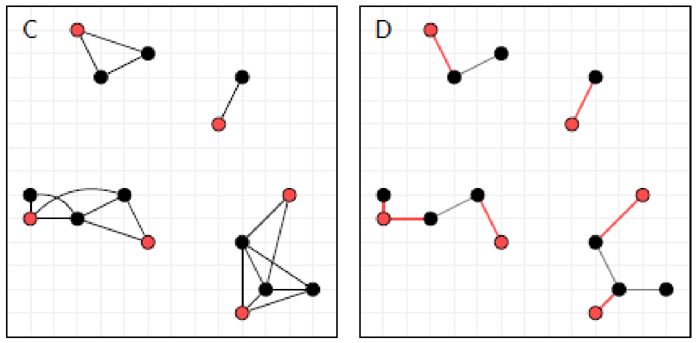

Figure 16 (A) The second power of 1NN on G and generating seeds (B) Connecting vertices to seeds

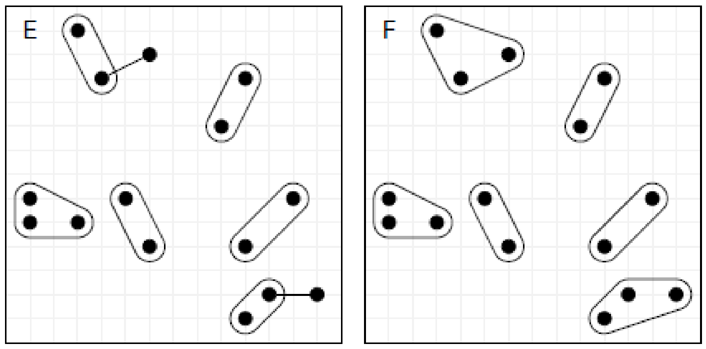

Figure 17 (E) Assign the unassigned vertices to the blocks with nearest vertices (F) The final blocking

Chapter 4 - Experiment design

This section contains the details about the data set used for the experiments and the shape of the data set.

4.1 Test bed Development

Three sets of data have been used for the experiment with clusters of different shapes and sizes. DS3 is the first set of data that have clusters which are six in number and are of various shapes, orientations and sizes. This set of data also have noise points that are random and special articles like clusters that have streaks running across them.

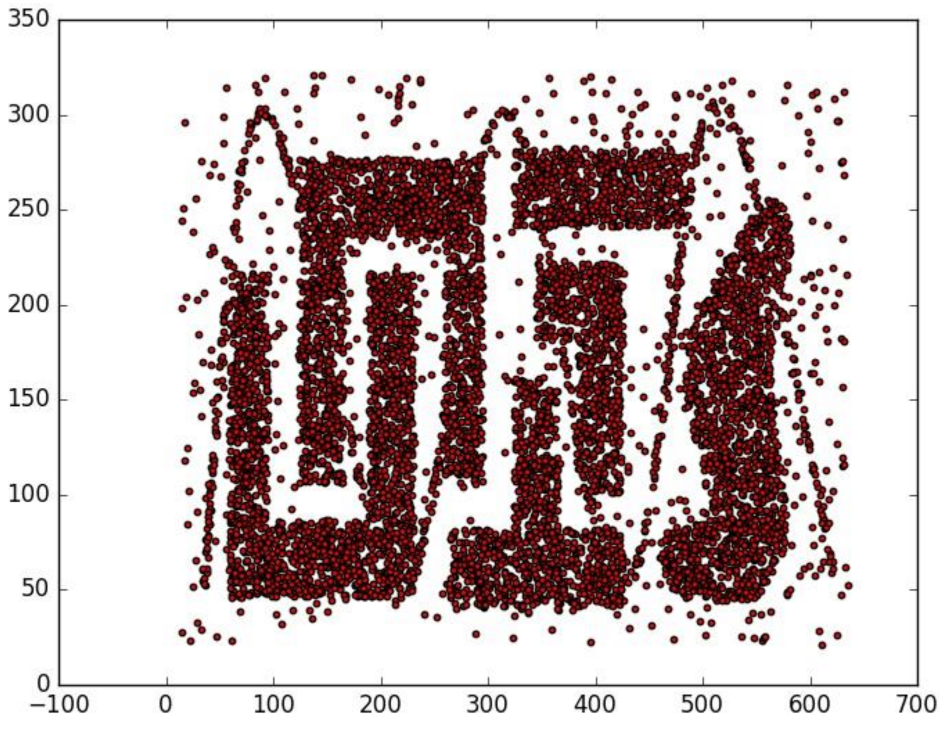

Figure 18 DS3 data set

DS4 is the second set of data that have clusters which are eight in number and are of various shapes, orientations and sizes. Some of the clusters are in the inner space enclosed by other clusters. This set of data also have noise points that are random and special articles like streaks formed vertically by a collection of points.

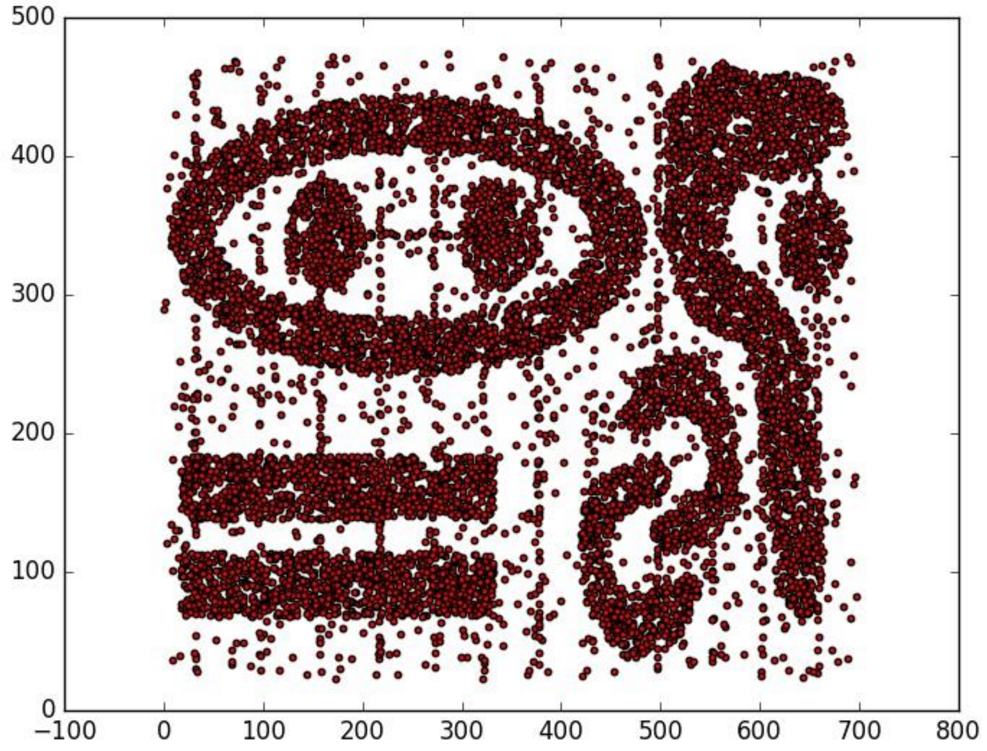

Figure 19 DS4 data set

DS5 is the third set of data that have clusters which are eight in number and are of various shapes, orientations, densities and size. This set of data also have noise points that are random. A notable feature of this set of data is that the clusters are very near to each other and have variant densities. This acts as a challenge to handle such a data set. The size range of such data set is from 6,000 to 10,000 points.

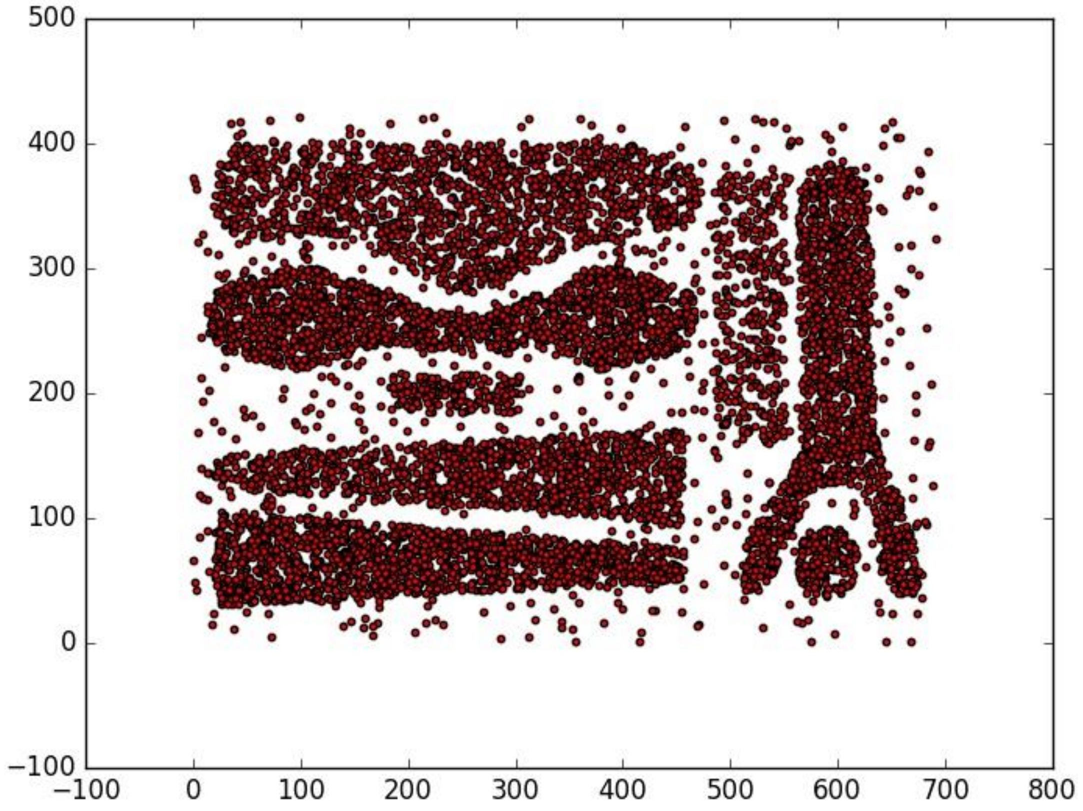

Figure 20 DS6 data set

These datasets contain all the examples that we have discussed in the limitation section to show how CHAMELEON overcomes the drawback of the previous schemes used to detect cluster. The data set was obtained from http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download

Chapter 5 - Results

This section shows the result of the clustering algorithm in both the phases.

Result obtained for DS3 with the local implementation for K = 5 and MINSIZE = 2.5% passed to the algorithm -

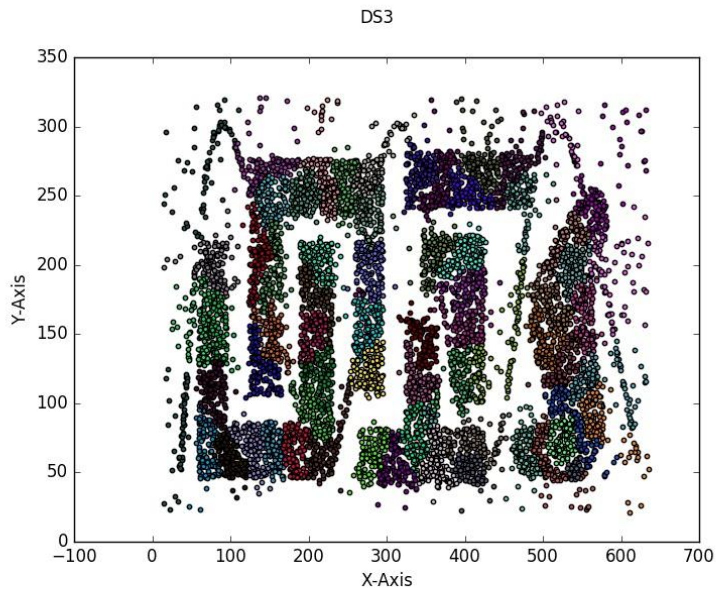

Figure 21 Result on DS3 after phase I

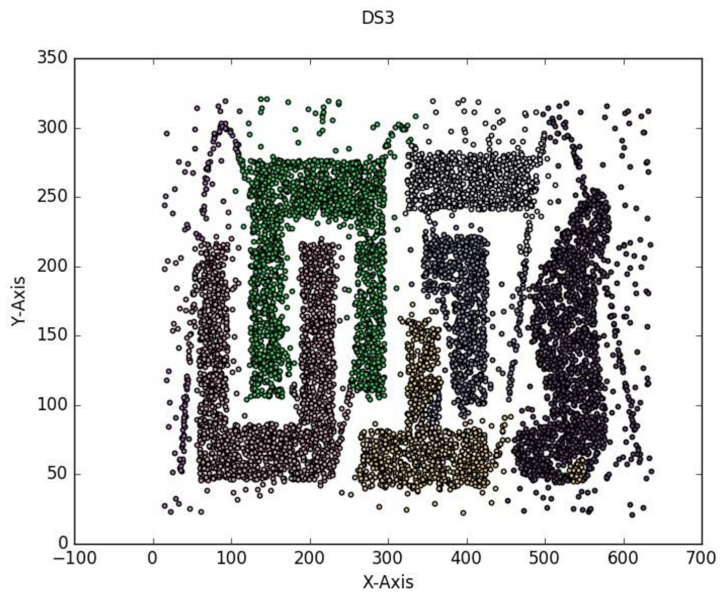

Figure 22 Result on DS3 after phase II

Result obtained for DS4 with the local implementation for K = 10 and MINSIZE = 7.5% passed to the algorithm -

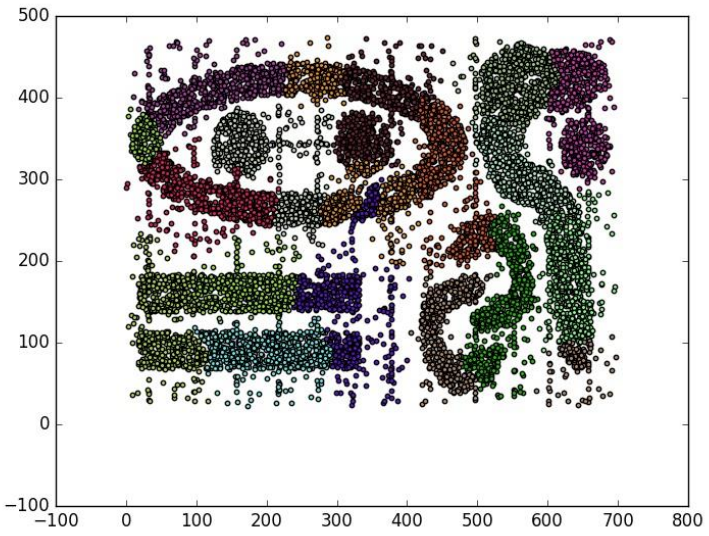

Figure 23 Result on DS4 after phase I

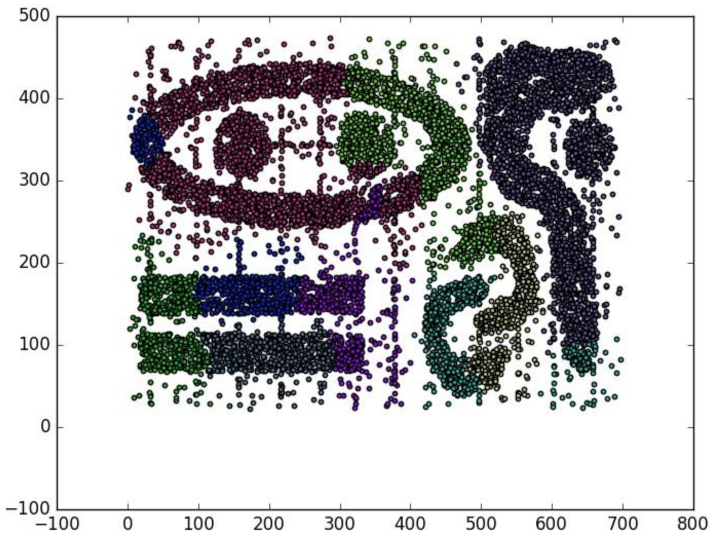

Figure 24 Result on DS4 after phase II

Result obtained for DS5 with the local implementation for K = 5 and MINSIZE = 2.5% passed to the algorithm -

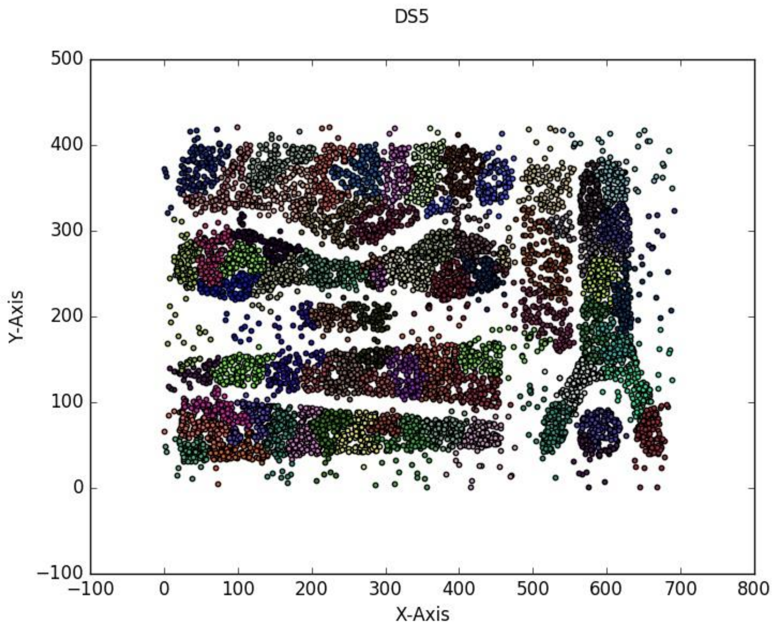

Figure 25 Result obtained on DS5 after phase I

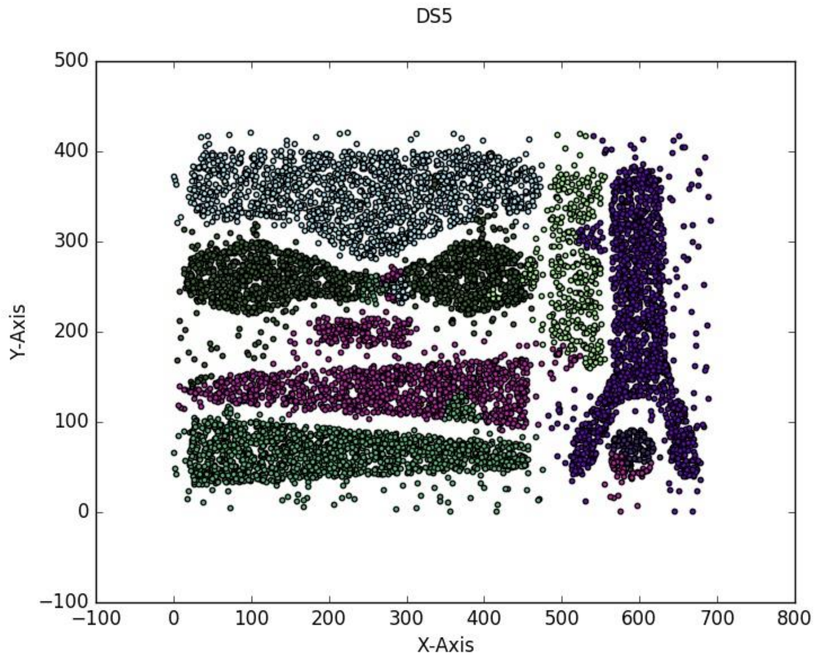

Figure 26 Result obtained on DS5 after phase II

The inter-cluster distance between any two clusters can be defined as the distance between the centroids of the clusters.

The intra-cluster distance is defined as the maximum distance that can be measured between any pair of elements in each cluster.

The internal closeness of cluster is defined as the average of the weights of the edges that are present in the cluster from the KNN graph.

Table 1 Quantitative results for DS5

Cluster Name	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
Intra cluster distance	68.6	459.35	613.55	475.36	474.14	261.5	435.52
Intra cluster closeness	22.22	169.22	157.84	153.67	145.65	84.84	128.6

Inter cluster distance	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
Cluster 1	0	410.19	312.44	379.14	444.62	213.61	134.94
Cluster 2	410.19	0	129.85	188.59	94.68	279.52	374.33
Cluster 3	312.44	129.85	0	105.18	211.37	250.6	315.74
Cluster 4	379.14	188.59	105.18	0	282.72	353.29	408.03
Cluster 5	444.62	94.68	211.37	282.72	0	273.18	379.94
Cluster 6	213.61	279.52	250.6	353.29	273.18	0	109.71
Cluster 7	134.94	374.33	315.74	408.03	379.94	109.71	0

Table 2 Quantitative results for DS3

Cluster Name	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
Intra cluster distance	332.91	298.34	265.66	164.95	277.93	297.38	304.45
Intra cluster closeness	88.95	74.92	72.22	58.22	102.62	102.68	116.05

Inter cluster distance	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
Cluster 1	0	191.64	171.53	138.11	388.67	332.77	481.38
Cluster 2	191.64	0	183.07	99.93	209.78	194.31	316.82
Cluster 3	171.53	183.07	0	86.83	293.82	202.17	356.86
Cluster 4	138.11	99.93	86.83	0	257.66	194.67	344.22
Cluster 5	388.67	209.78	293.82	257.66	0	111.95	115.76
Cluster 6	332.77	194.31	202.17	194.67	111.95	0	154.99
Cluster 7	481.38	316.82	356.86	344.22	115.76	154.99	0

Table 3 Quantitative results for DS4

Cluster Name	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9
Intra cluster distance	200.32	279.88	447.89	225.98	332.38	400.35	317.12	234.2	297.1
Intra cluster closeness	68.6	87.07	146.37	63.7	111.36	135.18	108.82	76.27	102.56

Inter cluster distance	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9
Cluster 1	0	129.21	245.36	146.78	126.92	461.04	325.82	339.18	293.27
Cluster 2	129.21	0	217.08	255.49	177.56	331.83	209.84	222.92	187
Cluster 3	245.36	217.08	0	247.47	142.2	407.07	215.49	392.19	377.9
Cluster 4	146.78	255.49	247.47	0	112.13	574.6	409.78	477.07	435.3
Cluster 5	126.92	177.56	142.2	112.13	0	472.88	300.3	397.86	364.54
Cluster 6	461.04	331.83	407.07	574.6	472.88	0	192.47	187.93	239.54
Cluster 7	325.82	209.84	215.49	409.78	300.3	192.47	0	234.3	249.85
Cluster 8	339.18	222.92	392.19	477.07	397.86	187.93	234.3	0	56.69
Cluster 9	293.27	187	377.9	435.3	364.54	239.54	249.85	56.69	0

All the result obtained through the algorithm were close to the actual result obtained with the authors implementation. Later, we have tested the data set DS4 with pure Threshold blocking and Hybridized threshold blocking with DBSCAN to make a comparison of the clustering algorithms.

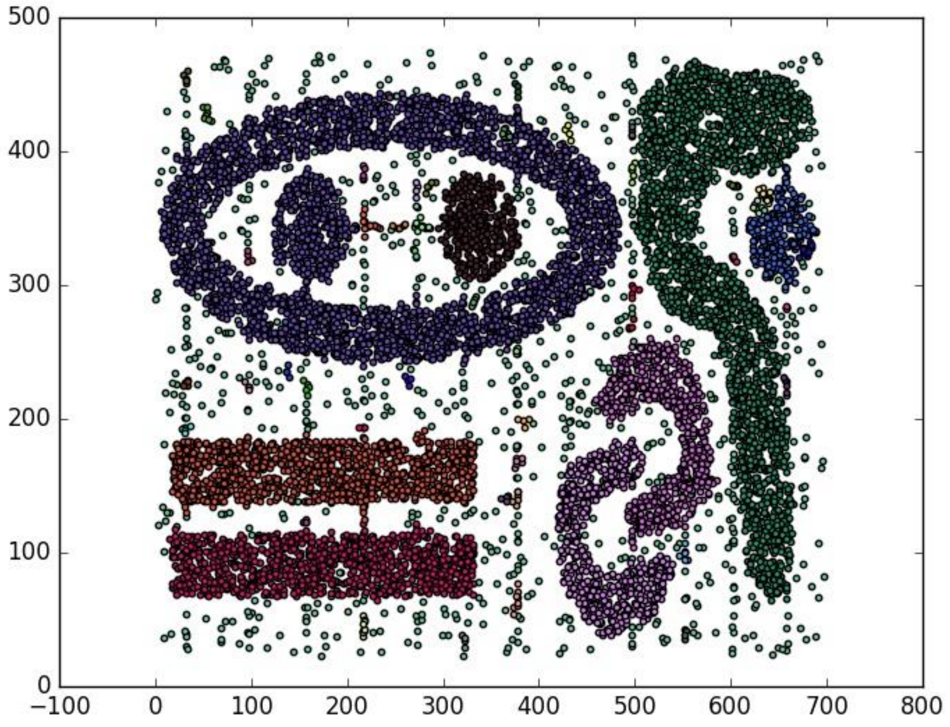

Figure 27 Result of DBSCAN on DS4

We observe that DBSCAN was able to find the genuine clusters in the data set correctly, but the R program showed the number of clusters to be 60, of which most of the less prominent clusters are of very small size and lie around the corner or in middle of prominent clusters in the above figure. It was obtained with Eps of 6.2 and Minpt of 4.

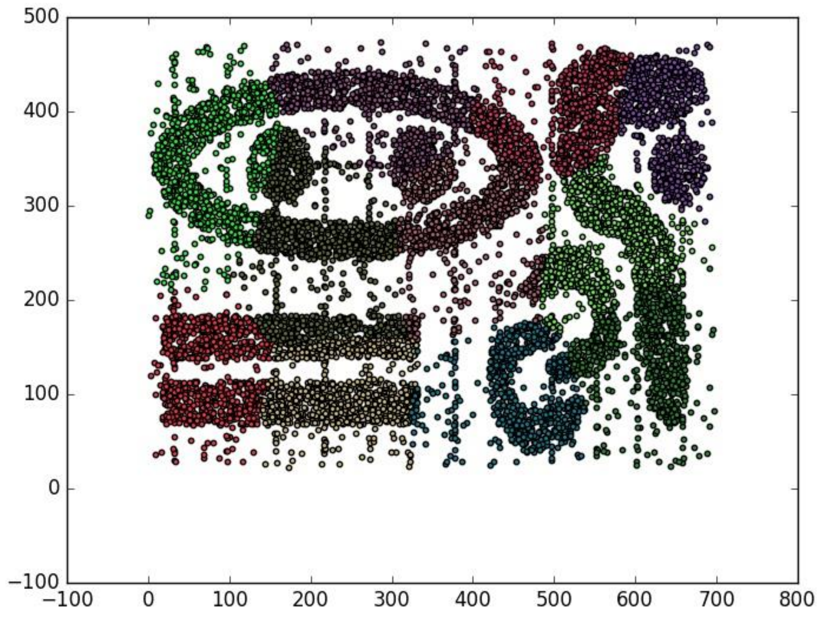

Figure 28 Result on DS4 based on pure threshold blocking

The above result was obtained by running Threshold blocking algorithm on the data set without any hybridization with k = 750. The above generated result is close to the phase I result of CHAMELEON but not the correct result. Later, I have tested the same the same dataset with hybridized threshold blocking but by taking the centroid of the sub-clusters generated by threshold blocking we again ran into the limitation discussed in chapter 2 and so it did not provide us with significant result.

Table 4 Runtime comparison of algorithm

Data set	Chameleon (Time in secs)	Threshold Blocking (Time in secs)
DS3	28878	2
DS4	72865	3
DS5	28904	2

Chapter 6 - Conclusions and future work

This section discusses the conclusion and the future work that can be worked upon.

6.1 Conclusion

- The local implementation of CHAMELEON was successful though the complexity of the algorithm is not at its best and has scope of improvement. The results obtained were close to the result from the original research paper.
- k-NN is one of the most effective algorithm to generate a sparse graph and go ahead with the partitioning of the graph to generate sub-clusters.
- Threshold blocking is way faster when compared to phase I of CHAMELEON and generates similar sub clusters. Again, this could be only the dataset that I have used it for. A better scheme is to be implemented to go ahead from partitioning to merging the clusters in Hybridized threshold blocking.

6.2 Future work

- One of the problems that I faced while running the algorithm was running it on a single core was taking a lot of time. So, to overcome this drawback I would like to parallelize the algorithm to run on multiple cores and utilizing the full capability of the CPU to give better result.
- Another improvement would be to use better data structure to improve the complexity of the program.
- Based on the result from threshold blocking, the partitions generated could be hybridized with phase II of CHAMELEON to give faster and better result in generating the clusters.

- Release the code under GPL as there are no current implementation of CHAMELEON in any clustering libraries.

References

G. Karypis, E. Han, V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling", *IEEE Computer*, vol. 32, no. 8, pp. 68-75, Aug. 1999.

G. Karypis and V. Kumar. METIS 4.0: Unstructured graph partitioning and sparse matrix ordering system. Technical report, Department of Computer Science, University of Minnesota, 1998.

G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. In *Proc. of 1998 ACM-SIGMOD Int. Conf. on Management of Data*, 1998.

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: a robust clustering algorithm for categorical attributes. In *Proc. of the 15th Int'l Conf. on Data Eng.*, 1999.

G. Karypis and V. Kumar. hMETIS 1.5: A hypergraph partitioning package. Technical report, Department of Computer Science, University of Minnesota, 1998.

Michael J. Higgins, Fredrik Savje, Jasjeet S. Sekhon: Improving massive experiments with threshold Blocking, 2016.